

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
імені ВАДИМА ГЕТЬМАНА
КОНСАЛТИНГОВА КОМПАНІЯ «СІДКОН»

Ірада ДЖАЛЛАДОВА
Олег КАМІНСЬКИЙ
Анатолій КОЛОТ
Юрій КОГУТ
Дмитро ЛУК'ЯНЕНКО

СМАРТ-АНАЛІТИКА ТА БЕЗПЕКА

Навчальний посібник



Рецензенти

В. В. Собчук, д.т.н., проф.
(Київський національний університет імені Тараса Шевченка)

Д. І. Черній, д.т.н., проф.
(Київський національний університет імені Тараса Шевченка)

Л. М. Колєчкіна, д.ф.-м.н., проф.
(Київський національний економічний університет
імені Вадима Гетьмана)

*Рекомендовано до друку Науково-методичною радою КНЕУ
Протокол № 10 від 30 травня 2024 р.*

Смарт-аналітика та безпека: [Електронний ресурс] навч. посібник /
C50 Рада Джалладова, Олег Камінський, Анатолій Колот, Юрій Когут, Дмитро
Лук'яненко. — Київ : КНЕУ, 2024. — 317, [3] с.
ISBN 978-966-926-495-4

Цільова аудиторія посібника включає IT-професіоналів, менеджерів із безпеки, аналітиків, студентів, викладачів та навіть будь-кого з мінімальними знаннями. Смарт-аналітика та безпека є важливими аспектами сучасних технологій. Смарт-аналітика використовується для збору, аналізу та застосування даних з різних джерел при прийнятті ефективних та обґрунтованих рішень. Це дозволяє підприємствам та організаціям оптимізувати процеси, підвищувати продуктивність і удосконалювати стратегію розвитку. Однак разом з розвитком смарт-аналітики зростає і важливість забезпечення безпеки цих даних. Збільшення обсягів інформації, її розподіл по різних платформах та пов'язані з цим ризики зловживань і кібератак роблять безпеку даних надзвичайно актуальною проблемою. Для забезпечення безпеки смарт-аналітики важливо використовувати сучасні методи шифрування, аутентифікації користувачів, моніторингу доступу до даних та захисту від кіберзагроз. Також важливо вибудувати чітку стратегію реагування на можливі інциденти безпеки і регулярно проводити аудит безпеки. Забезпечення безпеки даних у смарт-аналітиці є значущим фактором успішного впровадження цієї технології, що допомагає зберегти довіру користувачів і клієнтів. Посібник гарно структурований. Привабливий вигляд і сучасний дизайн робить посібник зручним у користуванні. Автори вважають, що посібник «Смарт-аналітика та безпека» буде корисним для цільової аудиторії та сприятиме їхньому успішному навчанню та розвитку.

УДК 004.056(075.8)

*Розповсюджувати та тиражувати
без офіційного дозволу КНЕУ заборонено*

© І. Джалладова, О. Камінський, А. Колот,
Ю. Когут, Д. Лук'яненко, 2024
© КНЕУ, 2024

ISBN 978-966-926-495-4

Зміст

Передмова	4
1. Вступ	6
1.1. Смарт-аналітика та безпека: минуле, сьогодення, майбутнє.....	6
1.2. Природа даних. Мистецтво і наука попередньої обробки даних	15
1.3. Великі дані.....	31
1.4. Штучний інтелект: концепції, драйвери, основні технології	40
1.5. Машинне навчання	52
1.6. Інтернет речей: платформи та основні переваги	63
Мозковий штурм	75
LAB 1. Попередня обробка даних різної природи	76
Підсумки: чотири ери аналітики	78
2. Смарт-аналітика	79
2.1. Візуалізація даних	79
2.2. Статистичні методи	98
2.3. Аналіз часових рядів	112
2.4. Аналіз соціальних мереж (SNA).....	123
2.5. Аналітика природної мови.....	138
2.6. Смарт-аналітика в проблемах міжнародної безпеки	154
Мозковий штурм	173
LAB 2. Розвідувальний аналіз цін на Amazon.com	174
Підсумки: чотири цілі смарт-аналітика	176
3. Смарт-безпека	177
3.1. Ідентифікація систем кібербезпеки.....	177
3.2. Аналітика даних щодо кіберзлочинів	189
3.3. ІІІ у кіберпросторі	208
3.4. Машинне навчання у захисті інформації	219
3.5. Аналітика етичного хакерства.....	229
3.6. Смарт безпека у HR.....	239
Мозковий штурм	255
LAB 3. Аналіз даних в кібербезпеці бізнесу	256
4. Кейси	259
4.1. Data-driven підхід до прийняття рішень у бізнес-розвідці.....	259
4.2. Аналіз та сегментація осіб за лояльністю у кіберрозвідці.....	264
4.3. Ідентифікація осіб методом «K-means».....	269
4.4. Кібербезпека у сфері торгівлі дорогоцінними металами	281
4.5. Використання LLM для виявлення інцидентів кібертероризму	293
Термінологічний словник	303
Список використаних джерел.....	314

Передмова

1. **Цільова аудиторія посібника** — від IT-професіоналів, менеджерів із безпеки, аналітиків даних, студентів, викладачів, навіть будь-хто з мінімальними знаннями.

2. **Основні цілі та завдання посібника.** Ознайомлення з основами смарт-аналітики, вивчення методів захисту даних, розгляд конкретних прикладів використання смарт-аналітики для підвищення безпеки.

3. **Основні теми, які охоплені у посібнику.** Ролі смарт-аналітики у сучасному бізнесі, кібербезпеці, методи аналізу даних, важливість безпеки даних у смарт-аналітиці, бізнес-розвідці, застосування.

Смарт-аналітика та безпека — важливі аспекти сучасних технологій. Смарт-аналітика використовується для збору, аналізу і застосування даних із різних джерел у прийнятті ефективних та обґрунтованих рішень. Це дозволяє підприємствам та організаціям оптимізувати процеси, підвищувати продуктивність і удосконалювати стратегію розвитку. Однак разом з розвитком смарт-аналітики зростає і важливість забезпечення безпеки цих даних. Збільшення обсягів інформації, її розподіл по різних платформах та пов'язані з цим ризики зловживань і кібератак роблять безпеку даних надзвичайно актуальною проблемою.

Для забезпечення безпеки смарт-аналітики важливо використовувати сучасні методи шифрування, аутентифікації користувачів, моніторингу доступу до даних і захисту від кіберзагроз. Також важливо вибудувати чітку стратегію реагування на можливі інциденти безпеки і регулярно проводити аудит безпеки.

Забезпечення безпеки даних у смарт-аналітиці є значущим фактором успішного впровадження цієї технології, що допомагає зберегти довіру користувачів і клієнтів.

4. **Гарно структурований.** Розподіл матеріалів по певних розділах та підрозділах, включення практичних завдань, кейсів і прикладів для кращого засвоєння матеріалу.

5. Визначено **найтефективніші методи навчання**, такі як текстові пояснення, графіки, відеоуроки, вправи і тести.

6. **Оформлення та дизайн на сучасному рівні.** Привабливий вигляд і сучасний дизайн робить посібник зручним у користуванні.

7. Автори будуть раді оцінкам і відгукам.

Автори вважають, що посібник «Смарт-аналітика та безпека» буде корисним для цільової аудиторії та сприятиме їхньому успішному навчанню та розвитку.

Деякі позначення



Позначення нових термінів, понять



Цікава інформація для «чайників»



Підсумки, важливі коментарі і думки



Запитання для перевірки знань

Розділ 1. ВСТУП

1.1. Смарт-аналітика та безпека: минуле, сьогодення, майбутнє



Найпростіше визначення **аналітики** — це наука про аналіз. Проте найкращим у практичній площині було б твердження про опис процесу, наприклад, як суб'єкта чи бізнесу, що рухається до оптимального чи реалістичного рішення з використанням наявних даних. Протягом десятиліть аналітика ставала все більш важливим компонентом управління — і це ще до того, як з'явилися технології, які дозволили аналізувати інформацію на рівні сьогодення.

1.1.1. Минуле

Наприкінці 1800-х років *Фредерік В. Тейлор* представив першу формалізовану систему аналітики в Сполучених Штатах. Система наукового менеджменту Тейлора розпочалася з досліджень часу виконуваної роботи, аналізуючи виробничі технології, рухи тіла робітників, щоб віднайти найбільшу ефективність, що в кінцевому підсумку прискорили б розвиток промислового виробництва. Ф. Тейлор був консультантом відомого Генрі Форда і безпосередньо вплинув на вимірювання часу роботи конвеєра автомобілів на заводах Форда. *Генрі Форд* вимірював швидкість конвеєра, зробивши революцію у виробництві.

У 1880 р., задовго до появи сучасних програмно-керованих пристроїв, Бюро перепису населення США знадобилося майже сім років, щоб обробити зібрану інформацію та скласти остаточний звіт. У відповідь винахідник *Герман Холлеріт* створив «машину для складання таблиць», яку використовували під час перепису 1890 р. Пристрій для табуляції міг систематично обробляти записані на перфокарти дані, що скоротило час обробки результатів цього статистичного обліку населення до 18 місяців.

Фразу «BUSINESS INTELLIGENCE» вперше було використано у 1864 р. у книзі про лаштунки підприємницької діяльності за авторством *Річарда Девенса*. Відтоді це стало ключовим компонентом в ухваленні рішень для усіх лідерів, а це відбулося завдяки аналітиці вже як галузі знань.

Втім найбільше увагу почали приділяти аналітиці наприкінці 1960-х років, коли комп'ютери набули повсюдного використання у системах підтримки прийняття рішень.

Реляційні бази даних винайшов *Едгар Ф. Кодд* у 1970-х, що набули широкої популярності у 1980-х роках. Реляційні бази даних RDBM дозволили користувачам писати послідовні запити SQL і отримувати дані зі своєї бази даних.

Протягом 1970-х років основна увага була прикута до надання структурованих періодичних звітів, які менеджер міг використовувати в ухваленні рішень чи ігнорувати їх. Компанії складали регулярні звіти, щоб інформувати осіб, які приймають рішення (менеджерів) про те, що відбулося за попередній період, наприклад, день, тиждень, місяць, квартал. У середині 1990-х років Інтернет став надзвичайно популярним і реляційні бази даних вже не встигали за його швидким розвитком. Величезний потік інформації в поєднанні з різноманітністю типів даних, що надходять із багатьох різних джерел, сприяли появі нереляційних баз даних — **NoSQL**. Бази даних NoSQL можуть швидко перекладати дані, використовуючи різні мови і формати, та уникають структурованості SQL, замінюючи власне організоване зберігання більшою гнучкістю.

Систему **Google Analytics** вперше представлено 2005 р. Світ на той час вже виявився переповненим даними, доступними ще від світанку інформаційної ери, а Google надав користувачам безкоштовний спосіб почати аналізувати принаймні частину з них. Незважаючи на те що Google Analytics не була досконалим пакетом і найбільш зручним у користуванні, втім якщо розглядати еволюцію аналітики, система **Google Analytics** дозволяла користувачам глибше зануритися в конкретні показники, про які Тейлор і Форд могли лише мріяти.

З усіх нових джерел даних, наприклад, мітки радіочастотної ідентифікації **RFID**, цифрові лічильники енергії, вебжурнали кліків, розумні домашні пристрої, переносне обладнання для моніторингу стану власного здоров'я, мабуть, найцікавішим і найскладнішим виявилися соціальні мережі / соцмережі. Ці неструктуровані дані характеризуються багатим інформаційним вмістом, але аналіз таких джерел даних створює значні проблеми для обчислювальних систем як з програмного, так і з апаратного забезпечення. Багато удосконалень як в апаратному забезпеченні (наприклад, масова паралельна обробка з дуже великою обчислювальною пам'яттю та високопаралельними багатопроцесорними обчислювальними системами), так і у програмному забезпеченні / алгоритмах, наприклад, **Hadoop** і **Spark**, були розроблені для вирішення питань такого обсягу даних.

У 2005 р. *Роджер Магулас* впровадив новий термін «**BigDate**», що описував велику кількість даних, з якими неможливо впоратися за допомогою доступних на той час інструментів аналітики. У тому ж році був запущений сервіс **Hadoop**, який міг обробляти такі BigDate. Основою Hadoop послуговував інший фреймворк програмного забезпечення з відкритим вихідним кодом під назвою **Nutch**, який згодом об'єднали із Google MapReduce.

Apache Hadoop — це платформа програмного забезпечення з відкритим кодом, яка може обробляти як структуровані, так і неструктуровані дані, що надходять майже з усіх цифрових джерел. Ця гнучкість дозволяла Hadoop обробляти великі дані. Наприкінці 2000-х кілька проєктів з відкритим кодом, таких як **Apache Spark** і **Apache Cassandra**, були розроблені, щоб впоратися з цим викликом.

За останнє десятиліття набір технологій аналітики зазнав карколомних перетворень. Так, наприклад, фахівці з аналізу відійшли від використання EXCEL, PPT і SPSS ще 2005 р., а від Tableau, SAS, SQL, платформи Hadoop відмовилися ще у 2010 р.

Нещодавно сталося чергове вибухове зростання технологічного ландшафту, тому більшість фахівців-аналітиків використовує відкритий код Python, Spark та інструменти візуалізації, зокрема, Tableau, Qlik і Power BI.

Отже, теперішні обчислювальні спроможності сягнули такого рівня, коли генерація та обробка даних навколо нас зростає безпрецедентними темпами, і це вміщує усі види даних — структуровані, неструктуровані, соціальні, веб і та ін.

Математичні методи аналізу та алгоритми також еволюціонували. Наприклад, для статистичного аналізу зараз переважно використовується **машинне навчання і алгоритми штучного інтелекту**. Попри те, що компанії досі ще хочуть зрозуміти, які змінні впливають на ключові показники ефективності, використовуючи такі методи, як аналіз настроїв, втім також хочуть і зосередитися на автоматизації аналітики, щоб заохочувати процеси самообслуговування. Так і виникла новітня категорія аналітики — смарт-аналітика.

1.1.2. Сьогодення



Смарт-аналітика схожа на приготування кави — добір якісного продукту з правильним методом екстракції гарантує, що кінцевий продукт найкраще задовольнить конкретні потреби. В аналітиці, як і у разі з приготуванням запашного напою, бракує універсального рішення на всі випадки.



SMART — це мнемонічна абревіатура, компоненти якої є критеріями постановки завдань: *конкретний, вимірюваний, призначений, реалістичний, пов'язаний із часом*: **S** / specific, **M** / measurable, **A** / assignable, **R** / realistic, **T** / time-related. Вважається, що відповідність завдань наведеним критеріям значно підвищує ймовірність їх виконання та досягнення поставленої мети.



Смарт-аналітика — це практика використання передових аналітичних інструментів і методів для виявлення прихованих закономірностей, тенденцій і кореляцій у величезних обсягах даних. Це передбачає інтеграцію можливостей штучного інтелекту та машинного навчання для автоматизації обробки даних і прийняття інтелектуальних рішень.



Смарт-аналітика — це інтелектуальне застосування аналітичних технологій для перетворення необроблених даних у значущу інформацію. Смарт-аналітика використовує алгоритми на базі штучного інтелекту в автоматичному визначенні та встановленні пріоритетів відповідних точок даних, вилучення ключової інформації та надання практичних рекомендацій користувачам.



Смарт-аналітика — це загальний термін, який охоплює різні методи аналітики, включаючи прогнозовану аналітику, прескриптивну аналітику та когнітивну аналітику. Це передбачає використання моделей і алгоритмів, керованих штучним інтелектом, для аналізу даних у режимі реального часу, що дозволяє компаніям приймати проактивні рішення, оптимізувати процеси та покращувати загальну продуктивність.



Смарт-аналітика передбачає поєднання традиційних аналітичних підходів із передовими технологіями, такими як ШІ, ML, обробка природної мови NLP і глибоке навчання. Це дасть змогу організаціям вилучати цінну інформацію зі складних і різноманітних джерел інформації, зокрема структурованих і неструктурованих даних, щоб отримати повне розуміння своїх операцій і поведінки клієнтів.

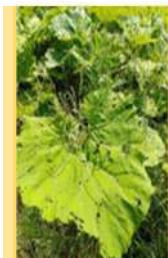


Смарт-аналітика — це конвергенція аналітики, автоматизації та інтелектуальних технологій для отримання інформації в реальному часі та прийняття рішень на основі даних. Смарт-аналітика передбачає використання систем на базі штучного інтелекту, які постійно навчаються на основі даних, адаптуються до мінливого середовища і надають контекстну інформацію на підтримку стратегічних бізнес-ініціатив.

Маючи у розпорядженні передові технологічні інструменти, такі як **ІоТ**, **Cloud Computing**, математичні моделі, людина може дістати доступ до найбільш точнішої та актуальної інформації. Використання цієї точності — одна з найважливіших переваг смарт-аналітики.

Усі інші *переваги смарт аналітики* впливають зі згаданої вище своєчасної утилізації інформації або практичних застосувань. Наприклад:

- миттєва реакція на ситуації та зміни з аналізом у реальному часі;
- моніторинг та аналіз з місць і речей поза межами досяжності людини за допомогою таких інструментів, як ІоТ;
- виявлення унікальних споживчих проблем і надання рішень з позицій нових продуктів і послуг;
- прийняття рішень на основі даних, а не інтуїції, особливо у складних бізнес-ситуаціях;
- персоналізація даних для кращого споживчого досвіду;
- комплексні прогнози на майбутнє на основі даних минулого і сьогодення.



Загалом кількість даних у світі подвоюється кожні два роки. Комп'ютеризовані системи використовуються в складних сферах управління, починаючи від проєктування й управління автоматизованими заводами й закінчуючи застосуванням аналітичних методів в оцінці пропонування об'єднань і поглинань. Інструменти аналітики і бізнес-аналітики, такі як сховища даних, машинне навчання, онлайн-аналітична обробка OLAP, інформаційні панелі та використання хмарних систем є наріжним камінням сучасного менеджменту.

Основні технологічні чинники, які вплинули на розвиток аналітики, подано на рис. 1.1.



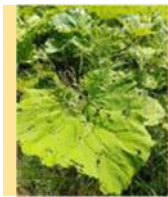
Рис. 1.1. Фактори впливу технологій на розвиток аналітики

1. **Групове спілкування та співпраця.** Програмні додатки, такі як GitHub, можуть покращити процес співпраці у групі та дозволити її членам перебувати у різних місцях (заощаджуючи витрати на дорогу). Така співпраця дозволяє виробникам знати про зміни в моделях попиту майже в режимі реального часу і у такий спосіб швидше реагувати на зміни ринку.
2. **Покращене керування даними.** Аналіз економічних чи безпекових процесів потребує складних обчислень, дані для яких можуть зберігатися в різних базах даних будь-де в організації і за її межами. Дані можуть включати текст, звук, графіку та відео, і вони можуть бути різними мовами. Сучасні системи дозволяють шукати, зберігати та передавати необхідні дані швидко, економічно, безпечно і прозоро.

3. **Управління хмарними сховищами даних і Big Data** Спеціальні методи, включаючи паралельні обчислення та програмні платформи Hadoop / Spark, використовуються для впорядкування, пошуку та видобутку даних у великих сховищах даних DW. Витрати на зберігання та видобуток даних стрімко знижуються. Технології та математичні моделі, які підпадають під широку категорію технологій Big Data, дозволяють отримувати величезні обсяги даних, що надходять із різноманітних джерел і в різних формах, та дають змогу підвищувати ефективність організації.
4. **Аналітична підтримка.** Завдяки більшій кількості даних і технологій аналізу можна оцінити більше альтернатив, покращити прогнози, швидко виконати аналіз ризиків, а думки експертів (деякі з яких можуть перебувати у віддалених місцях) можна зібрати швидко та з меншими витратами.
5. **Подолання когнітивних обмежень в обробці та зберіганні інформації.** Людський розум має лише обмежену здатність обробляти та зберігати інформацію. Термін «когнітивні межі» вказує на те, що здатність людини розв'язувати проблеми обмежена, коли потрібен широкий спектр різноманітної інформації та знань. Комп'ютеризовані системи, зокрема штучний інтелект, дозволяють людям долати свої когнітивні межі шляхом швидкого доступу та обробки величезних обсягів збереженої інформації.
6. **Управління знаннями.** Організації зібрали величезні запаси інформації про власні операції, клієнтів, внутрішні процедури, взаємодію співробітників за допомогою неструктурованих і структурованих мереж комунікацій. Завдяки таким технологіям, як текстова аналітика та IBM Watson можна створювати цінність з таких сховищ знань.
7. **Розвиток засобів віддаленої роботи.** Розвиток мобільних технологій, платформ соціальних медіа та аналітичних інструментів уможливило створення інформаційних систем ІС іншого рівня для підтримки бізнесу. Це зростання надання підтримки на основі даних поширюється не лише на менеджерів, а й на споживачів.
8. **Штучний інтелект.** Через складність процесу прийняття рішень часто потрібен більш інноваційний підхід. Штучний інтелект забезпечує суттєвий розвиток таких інновацій. ШІ може впливати майже на кожен крок, створюючи синергію в у процесі прийнятті рішень.

Аналітика та її методи відповідають на більшість запитань про достатність розв'язання проблем. Додавання приписної аналітики збільшить витрати, але не обов'язково покращить продуктивність. Тому наступне покоління платформ аналітики використовуватиме конвергенцію інтелектуальних систем і штучний інтелект для автоматичного пошуку, візуалізації та опису важливих речей. ШІ також можна використовувати для створення автоматичних повідомлень і текстів. Крім того, машинне та глибоке навчання можуть підтримувати аналітику, виконуючи завдання розпізнавання образів і створення точних прогнозів. Штучний інтелект допоможе порівняти фактичну продуктивність із прогнозованою. Машинне навчання та інші технології штучного інтелекту також можуть забезпечити стратегію постійного удосконалення.

1.1.3. Майбутнє



Серед експертів з ШІ існує загальна згода, що штучний інтелект змінить усе в нашому світі на краще. Однак існують розбіжності в тому, коли відбудуться такі зміни та яким буде їхній вплив.

Дослідження штучного інтелекту прискорюються завдяки удосконаленню різних пов'язаних комп'ютерних технологій (наприклад, чипів, Інтернету речей), удосконаленням інтелектуальних методологій та інструментів, активізації діяльності високотехнологічних компаній, які прагнуть здобути лідерство у певних сферах інтелектуальних систем, і фірм, які інвестували мільярди доларів у штучний інтелект, розробку інструментів і методологій ШІ та багато іншого.

Схема на рис. 1.2 ілюструє довгострокову проєкцію штучного інтелекту. Майбутнє ділиться на три частини: сьогодні, приблизно через двадцять років і у віддаленому майбутньому.

	Сьогодні: Вузький штучний інтелект	Протягом 20-25 років Загальний штучний інтелект	Далеке майбутнє Супер штучний інтелект
	<i>З часом збільшуються можливості, але не зрівняються з людським інтелектом</i>	<i>Автономні системи повсюди; адаптація до змін у навколишньому середовищі; може самостійно розгортати завдання; може міркувати, впроваджувати інновації</i>	<i>Такий же розумний, як людина, а в деяких випадках навіть більше. Підтримка досліджень, інновацій та навчання</i>
Можливі впливи	Все частіше виконують рутинні завдання, деякі разом з людьми. Забезпечують швидкість, якість та консультування. Зменшують вартість.	Автономні транспортні засоби повсюди. Роботи беруть на себе більше завдань. У людей більше часу. Змагання з людьми	Може бути небезпечним, якщо не контролювати. Може значно покращити якість нашого життя

Рис. 1.2. Прогноз майбутнього штучного інтелекту

Майбутнє аналітики показано на рис. 1.3. Деякі додаткові прогнози — це смарт-аналітика, аналіз як послуга та каталогізація даних.

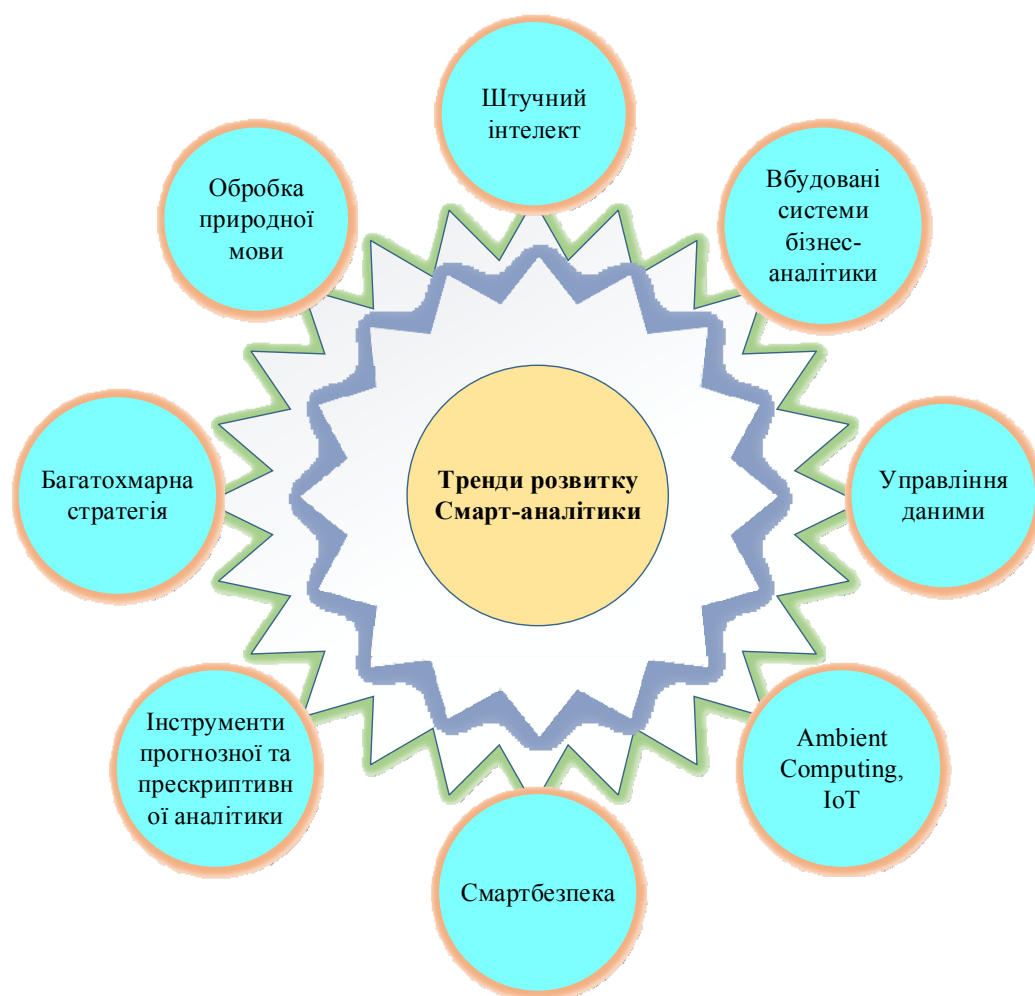


Рис. 1.3. Мандала майбутнього аналітики

Навколишні обчислення (Ambient Computing). Тісно пов'язані з IoT, чат-боти, розумні будинки, аналітика, датчики та «речі» входять до концепції навколишніх обчислень чи парадигмальних обчислень. Концепція має кілька визначень, але за змістом стосується електронних компонентів (наприклад, мережевих пристроїв, таких як датчики), які достатньо чутливі і реагують на людей та зміни їхнього середовища. Отже, розумні пристрої можуть підтримувати людей у виконанні будь-якого завдання, яке виконують. Після з'ясування оточення пристрої

надають різні методи введення / виведення, які залежать від конфігурації ситуацій, наприклад, того, що люди роблять у певний час.

Підсумовуючи, можемо стверджувати, що усе у нашому житті буде комп'ютеризованим і розумним. Концепція базується на попередніх дослідженнях у сферах поширених обчислень, взаємодії людини та машини, усвідомлення контексту, профілювання, персоналізації та дизайну взаємодії.



Модельна проблема. Ця історія стала майже класичним прикладом пояснення необхідності аналітики. Відомий учений Рассел Акофф, піонер дослідження теорії організацій і систем, описав проблему з чисельними скаргами на повільні ліфти у високій вежі готелю. Перепробувавши багато рішень, щоб знизити невдоволення клієнтів і кількість скарг від них (зокрема, розподіл ліфтів на різні поверхи, додавання операторів та ін.), керівництво визначило, що справжня проблема полягає не в фактичному часі очікування, а в уявному. Тому було вирішено встановити дзеркала в повний зріст на дверях ліфтів на кожному поверсі. Відповідно: «Жінки дивилися на себе в дзеркало й поправляли зовнішність, тоді як чоловіки дивилися на жінок, і ніхто не помічав, як з'являвся ліфт». Зменшуючи передбачуваний час очікування, питання зникло. Якщо аналіз визначив, що справжню складність становить уявний час очікування, це може значно змінити запропоновані рішення та їх вартість. І, звісно, дзеркала у повний зріст коштують набагато дешевше за додавання ще одного ліфта!



Запитання для самоперевірки



1. Дайте визначення аналітики.
2. Назвіть три терміни, які були попередниками аналітики.
3. Дайте визначення терміна «SMART».
4. Перелічіть деякі з факторів впливу на розвиток аналітики.
5. Назвіть основні типи інтелекту.
6. Назвіть деякі з факторів успіху ІІІ.
7. Що таке Ambient Computing?

1.2. Природа даних. Мистецтво і наука попередньої обробки даних



Дані є основними складовими для будь-якої бізнес-розвідки та бізнес-аналітики. Без даних жодна з технологій не могла б існувати та бути популяризованою, хоча традиційно ми створювали аналітичні моделі, використовуючи експертні знання та досвід у поєднанні з дуже невеликою кількістю даних або взагалі без них. Однак це були старі часи, а тепер дані мають суттєве значення. Дані, які колись сприймалися як серйозна проблема щодо збору, зберігання та керування, сьогодні вважаються одними з найцінніших активів організації, які можуть створити безцінний базис для кращого розуміння клієнтів, конкурентів і бізнес-процесів.

1.2.1. Дані



Термін «дані» (традиційно вживається в однині) стосується набору фактів, зазвичай отриманих у результаті експериментів, спостережень, транзакцій чи з досвіду. Дані можуть складатися з цифр, літер, слів, зображень, записів голосу тощо як вимірювання набору змінних (характеристик предмета чи події, які ми вивчаємо).

Дані можуть бути малими чи дуже великими. Вони можуть бути структурованими — добре організованими для комп'ютерної обробки; або неструктурованими — наприклад, текст, створений для людей і, отже, нелегкий для розуміння / перегляду комп'ютерами. Дані можуть надходити невеликими партіями безперервно чи відразу великою партією. Це деякі характеристики, які визначають природу сучасних даних, які ми часто називаємо *великими даними*. Незважаючи на те що ці характеристики даних ускладнюють їхню обробку і використання, це також підвищує їхню цінність, оскільки характеристики збагачують їх понад звичайні межі, дозволяючи відкривати нові та нові знання. Традиційні способи збору даних вручну через опитування чи бізнес-транзакції, введені людиною, здебільшого поступилися своїм місцем новітнім механізмам збору даних, які використовують Інтернет та/або комп'ютеризовані мережі на основі датчиків / радіочастотної ідентифікації **RFID**. Ці автоматизовані системи збору даних не лише дозволяють нам збирати більші обсяги інформації, а й підвищують якість і цілісність таких даних. На рис. 1.4 показано типовий аналітичний континуум — дані для аналітики — дієва інформація.

Іноді репрезентація даних залежить від типу використовуваної аналітики. Для прогнозованих алгоритмів, як правило, потрібен плоский файл із цільовою змінною. Звідси підготовка аналітичних даних до прогнозування означатиме, що

набори інформації мають бути перетворені на формат плоского файлу та готові для прийому в ці прогнозовані алгоритми. Важливо також узгодити дані з потребами конкретного прогнозованого алгоритму та/або програмного засобу. Наприклад, алгоритми нейронних мереж вимагають, щоб усі вхідні змінні були представлені в числовому вигляді — навіть номінальні змінні мають бути перетворені в псевдодвійкові числові змінні, тоді як алгоритми дерева рішень не потребують такого числового перетворення — вони можуть легко й оригінально обробляти поєднані номінальні та числові змінні.

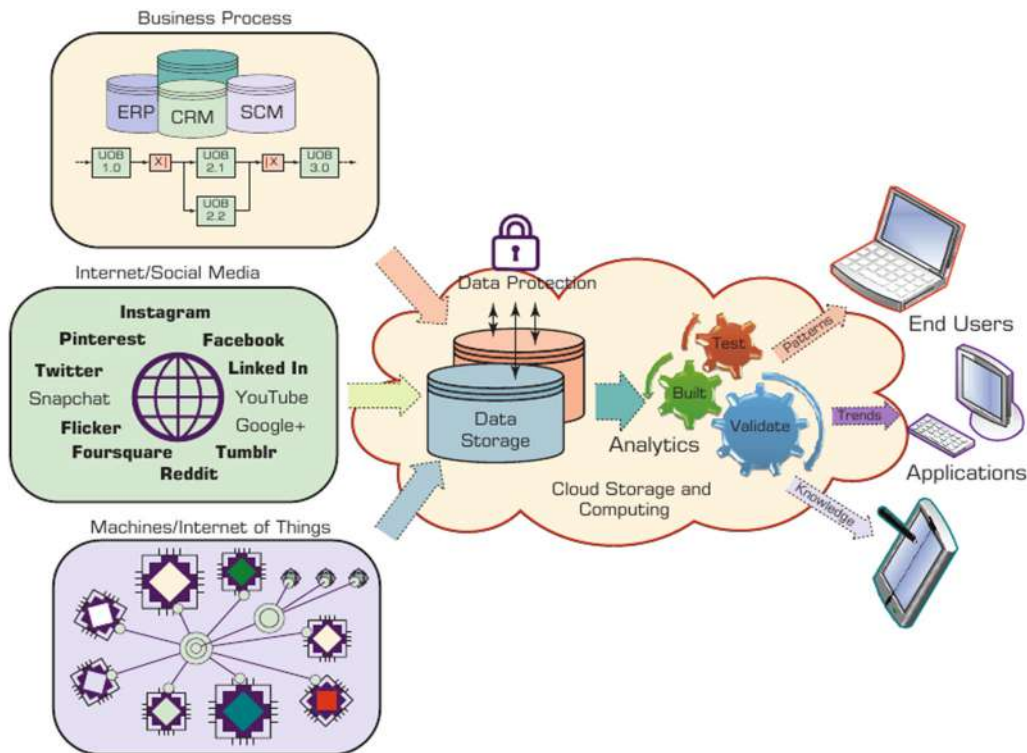


Рис. 1.4. *Континуум від даних до знань*

Аналітичні проекти, у яких не враховуються завдання, пов'язані з даними, часто закінчуються неправильною відповіддю на правильну проблему, а ці ненавмисно створені, хороші, відповіді можуть призвести до неточних і несвоєчасних рішень.

1.2.2. Характеристики даних

- **Надійність джерела даних.** Цей термін стосується оригінальності та відповідності носія, на якому отримані дані, відповідаючи на запитання «Чи маємо ми належну впевненість у цьому джерелі даних?» Якщо це взагалі можливо,

завжди слід шукати вихідне джерело / творця даних, щоб усунути / пом'якшити можливості спотворення і перетворення даних, спричинені неправильним поводженням з інформацією, коли вони переміщуються від джерела до місця призначення через один чи кілька кроків. Кожне переміщення даних створює ймовірність навмисного видалення або переформатування елементів даних, що обмежує цілісність і, можливо, справжню точність набору даних.

- **Точність вмісту даних.** Це означає, що дані правильні та добре відповідають проблемі аналітики, тобто дають відповідь на запитання «Чи є у нас правильні дані для роботи?» Інформація має відображати те, що було призначено чи визначено вихідним джерелом даних. Наприклад, контактні відомості про клієнта, записана в базі даних, повинні збігатися з тими, які вказав клієнт.

- **Доступність даних.** Цей термін означає, що інформацію легко отримати, відповідаючи на запитання «Чи можемо ми легко мати доступ до даних, коли нам це потрібно?» Такий доступ може виявитися складним, особливо якщо інформація зберігається в кількох місцях і на різних носіях, і їх потрібно об'єднувати / перетворювати під час доступу до них. Традиційні системи управління реляційними базами даних співіснують з новим поколінням носіїв, таких як озера даних та інфраструктура Hadoop, відповідно важливість / критичність доступності зростає.

- **Безпека та конфіденційність даних.** Це означає, що дані захищені у такий спосіб, щоб дозволити їх використання лише тим людям, які мають на те повноваження і щоб запобігти сторонньому доступу. Зростаюча популярність освітніх рівнів і програм сертифікації з забезпечення захисту інформації свідчить про критичність і зростаючу актуальність цього показника якості даних. Будь-яка організація, яка веде медичні записи для окремих пацієнтів, повинна мати системи, які не тільки захищені від несанкціонованого доступу, а й точно ідентифікують кожного пацієнта, щоб надати належний і своєчасний доступ до записів авторизованих користувачів.

- **Насиченість даними.** Це означає, що всі потрібні елементи даних мають бути включені до набору даних. За змістом насиченість (або повнота) вказує на те, що доступні змінні відображають досить багату вимірність основного предмета для точного аналітичного дослідження. Це також вказує на повний (або майже повний) інформаційний вміст для побудови прогнозованої та/або прескриптивної аналітичної моделі.

- **Послідовність даних.** Це свідчить про те, що дані точно збираються і комбінуються / об'єднуються. Послідовні дані — це інформація про виміри (змінні, що становлять інтерес), які надходять із потенційно різних джерел, проте стосуються однієї теми. Якщо інтеграція / об'єднання даних належно не виконується, деякі змінні різних суб'єктів можуть з'являтися в одному записі, переплутуючи два різні записи пацієнтів. Наприклад, це може статися під час об'єднання записів демографічних відомостей і клінічних результатів тестування ліків.

- **Актуальність даних / своєчасність даних.** Інформація має зберігати актуальність (або відомості мають бути настільки свіжими / новими, наскільки це можливо) для певної аналітичної моделі. Це також означає, що дані записуються в момент події чи спостереження або ж близько до нього, щоб уникнути спотворення інформації, пов'язаного з затримкою часу (неправильне запам'ятовування і кодування). Оскільки точна аналітика базується на чітких і своєчасних даних, важливою характеристикою інформації, готової до аналітики, є її новизна даних і доступ.

- **Деталізація даних.** Це вимагає, щоб змінні та значення даних були визначені на найнижчому (або настільки низькому, як потрібно) рівні деталізації для передбачуваного використання. Якщо дані агреговані, вони можуть не містити рівня деталізації, потрібного для того, щоб алгоритм аналітики навчився розрізняти різні записи / випадки. Наприклад, у медичному закладі числові значення результатів лабораторних досліджень слід записувати з точністю до відповідного десяткового знаку, як це потрібно для повної інтерпретації результатів тестування та належного використання цих значень в алгоритмах аналітики. Слід пам'ятати, що зібрану інформацію не можна деагрегувати (без доступу до вихідного джерела), але можна легко агрегувати на основі їх детального представлення.

- **Достовірність даних.** Це термін використовується для опису відповідності / невідповідності між фактичними та очікуваними значеннями даних. Необхідно визначити прийнятні значення або діапазони значень для кожного елемента даних. Наприклад, дійсне визначення даних, пов'язаних із статтю, включатиме три значення: чоловік, жінка та невідомий.

- **Релевантність даних.** Це означає, що всі змінні в наборі даних стосуються дослідження, яке проводиться. Релевантність не є дихотомічним показником, можливо, вона має спектр релевантності від найменш релевантного до найбільш релевантного. Виходячи з використовуваних алгоритмів аналітики, можна вибрати включення лише найбільш релевантної інформації (тобто змінних) або, якщо алгоритм достатньо спроможний їх відсортувати, можна вибрати включення всіх релевантних змінних незалежно від їх рівнів. Єдине, чого варто уникати в аналітичних дослідженнях, — це включення абсолютно нерелевантних даних у побудову моделі, оскільки це може забруднити інформацію для алгоритму, що призведе до неточних і оманливих результатів.

Дані часто розглядають як найнижчий рівень абстракції, з якого витягується інформація, а потім і знання. На найвищому рівні абстракції можна класифікувати дані як структуровані та неструктуровані чи напівструктуровані.

Перераховані вище характеристики є, мабуть, найпоширенішими показниками, з якими слід йти в ногу; справжня якість даних і чудова готовність до аналітики для певної сфери застосування вимагали б приділяти різні рівні уваги цим метричним параметрам і, можливо, додати більш конкретні до цієї колекції.

1.2.3. Класифікація даних



Неструктуровані дані складаються з будь-якої комбінації тексту, зображень, голосу і вебвмісту. **Структуровані дані** — це те, що використовують алгоритми аналізу даних, і їх можна класифікувати як категоріальні або числові. **Категоричні дані** можна розділити на номінальні, або порядкові, дані, тоді як **числові дані** можна розділити на інтервали або співвідношення.

На рис. 1.5 показано просту класифікацію даних.

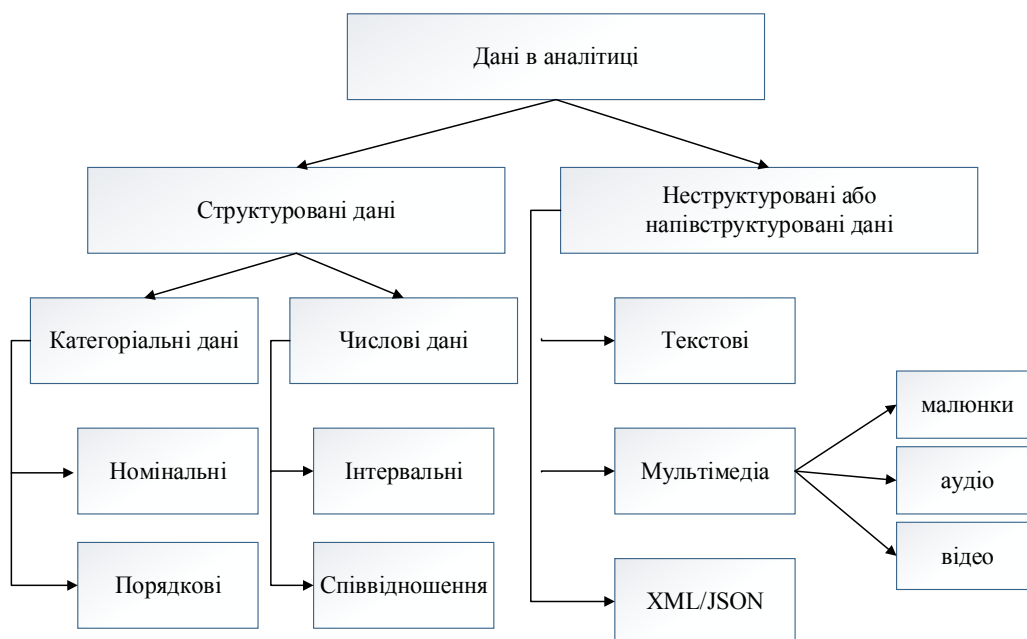


Рис. 1.5. Проста таксономія даних

- **Категоріальні дані.** Це класи, які використовуються для поділу змінних на певні групи. Приклади категоріальних змінних включають національність, стать, вікову групу і рівень освіти. Категоріальні також можна назвати дискретними даними, що передбачає кінцеву кількість значень без континууму між ними. Навіть якщо значення, що використовуються для категоріальних (або дискретних) змінних, числові, ці числа є символами без можливості розрахунку дробових значень.
- **Номінальні дані** містять прості коди, призначені об'єктам як невимірювані мітки. Наприклад, змінну «сімейний стан» можна узагальнено класифікувати як (1) неодружений, (2) одружений і (3) розлучений. Номінальні дані можуть бути представлені двома можливими значеннями (наприклад, так /

ні, істина / неправда, добре / погано) або мультиномінальними значеннями, що мають три чи більше можливі значення (наприклад, коричневий / зелений / синій).

- **Порядкові дані** містять коди, призначені об'єктам чи подіям як мітки, які також представляють порядок рангу серед них. Наприклад, змінний кредитний бал можна загалом класифікувати як (1) низький, (2) середній або (3) високий. Подібні впорядковані зв'язки можна побачити в таких змінних, як «вікова група» (дитина, молодь, середнього віку, похилого віку) або «рівень освіти» (середня, коледж, аспірантура). Деякі прогнозовані аналітичні алгоритми, такі як множинна логістична регресія, враховують цю додаткову інформацію, щоб побудувати кращу модель класифікації.
- **Числові дані** являють собою числові значення конкретних змінних, наприклад, вік, кількість дітей, загальний дохід, відстань подорожі (у кілометрах) чи температуру (у градусах). Числові значення, що представляють змінну, можуть бути цілими або дійсними (дробовими). Числові також можна назвати безперервними даними, маючи на увазі, що змінна містить безперервні показники в певній шкалі, яка дозволяє вставляти проміжні значення. На відміну від дискретної змінної, яка представляє обмежені обчислювані дані, безперервна змінна представляє масштабоване вимірювання, і дані можуть містити нескінченну кількість дробових значень.
- **Інтервальні дані** — це змінні, які можна виміряти за інтервальними шкалами. Типовим прикладом вимірювання інтервальної шкали є температура за шкалою Цельсія. У цій конкретній шкалі одиницею вимірювання — 1/100 різниці між температурою плавлення і температурою кипіння води при атмосферному тиску, тобто не існує абсолютного нульового значення.
- **Дані співвідношення**. До них належать вимірювальні змінні, які зазвичай зустрічаються у фізичних науках та техніці. Маса, довжина, час, плоский кут, енергія та електричний заряд слугують прикладами фізичних мір, які є шкалами співвідношення. Тип шкали дістав назву через те, що вимірювання — це оцінка співвідношення між обсягом постійної величини та одиничною величиною того ж виду. Відмінною рисою шкали відношення, неформально, є наявність невинного нульового значення. Наприклад, температурна шкала Кельвіна має невиннову нульову точку абсолютного нуля, яка дорівнює $-273,15$ градусів за Цельсієм. Ця нульова точка не довільна, оскільки частинки, які складають речовину при цій температурі, мають нульову кінетичну енергію.

Інші типи даних, включаючи текстові, просторові, зображення, відео та голос, потрібно перетворити на певну форму категоріального або числового представлення, перш ніж їх можна буде обробити методами аналітики. Дані також можна класифікувати як статичні або динамічні часові ряди.

Якщо розглянути поширені набори даних, то їх можна класифікувати так:



- електронні листи (неструктуровані, дискретні);
- цифрові зображення (неструктуровані, дискретні);
- журнали фондового ринку (структуровані, безперервні);
- історичні ціни на золото (структуровані, безперервні);
- записи про затвердження кредиту (структуровані, дискретні);
- друзі та стосунки в соціальних мережах (неструктуровані, дискретні);
- твіти та популярні теми (неструктуровані, безперервні);
- записи про продажі (структуровані, безперервні).

1.2.4. Джерела та набори даних



Деякі методи смарт-аналітики потребують подання усіх змінних у вигляді змінних із числовими значеннями, наприклад, нейронні мережі, опорні векторні машини, логістична регресія. Номінальні, або порядкові, змінні перетворюються на числові представлення за допомогою певного типу псевдозмінних 1 із N , наприклад, категоріальна змінна з трьома унікальними значеннями може бути перетворена в три псевдозмінні з двійковими значеннями 1 або 0 . Оскільки цей процес може збільшити кількість змінних, слід бути обережним щодо впливу таких представлень, особливо для категоріальних змінних, які мають велику кількість унікальних значень.

Дані в їхній оригінальній формі (тобто дані реального світу) зазвичай не готові до використання в аналітичних завданнях (вони часто брудні, неправильно вирівняні, надто складні та неточні). Для перетворення необроблених даних реального світу в добре вдосконалену форму для аналітичних алгоритмів необхідний виснажливий і трудомісткий процес (так звана *попередня обробка даних*). Їх потрібно отримати, очистити, нормалізувати та перетворити необроблені дані в стандартні формати, такі як файли значень, розділені комами / CSV або опис об'єктів JavaScript / JSON, методів очищення та перетворення даних.



Джерело даних — це термін, який використовується для всіх технологій, пов'язаних із вилученням і зберіганням даних. Джерелом даних може бути будь-що — від простого текстового файлу до великої бази даних. Необроблені дані можуть надходити з журналів спостережень, датчиків, транзакцій або навіть з поведінки користувача.

1. Джерела даних: відкрити дані, текстові файли, файли **Excel**, бази даних **SQL**, бази даних **NoSQL**, мультимедіа, вебскрапінг.
2. Методи очищення даних: статистичні методи, розбір тексту, перетворення даних.
3. Формати даних **CSV**, **JSON**, **XML**, **YAML**.



Набір даних — це сукупність даних, як правило, подані у формі таблиці. Кожен стовпець являє собою певну змінну, а кожен рядок відповідає певному елементу даних, як показано на рис. 1.6.

		Columns					
		id	outlook	temperature	humidity	windy	play
Rows	1	sunny	85	85	FALSE	no	
	2	sunny	80	90	TRUE	no	
	3	overcast	83	86	FALSE	yes	
	4	rainy	70	96	FALSE	yes	
	5	rainy	68	80	FALSE	yes	
	6	rainy	65	70	TRUE	no	
	7	overcast	64	65	TRUE	yes	
	8	sunny	72	95	FALSE	no	
	9	sunny	69	70	FALSE	yes	
	10	rainy	75	80	FALSE	yes	
	11	sunny	75	70	TRUE	yes	
	12	overcast	72	90	TRUE	yes	
	13	overcast	81	75	FALSE	yes	
	14	rainy	71	91	TRUE	no	

Values

Рис. 1.6. Набір даних

Набір даних становить фізичну реалізацію джерела даних. Загальні характеристики набору даних такі:

- тип набору даних (наприклад, багатовимірний або одновимірний);
- кількість екземплярів;
- сфера (наприклад, життя, бізнес і та ін.);
- характеристики атрибутів (а саме реальні, категоріальні, номінальні);
- кількість атрибутів;
- пов'язані завдання, такі як класифікація або кластеризація;
- відсутні значення.

1.2.5. Етапи попередньої обробки даних



Багато аналітиків свідчать, що час, витрачений на попередню обробку даних (яка, мабуть, є найменш приємною фазою в усьому процесі), значно довший за час, витрачений на решту аналітичних завдань — від побудови до оцінки аналітичної моделі.

Попередня обробка даних — це початковий крок, який окреслює перевірку вмісту набору даних, що передбачає прийняття рішень про відсутні або неправильні значення даних:

- робота з відсутніми значеннями даних;
- очищення зашумлених текстових даних;
- видалення тегів HTML;
- видалення смайлів;
- робота з емодзі / смайликами;
- фільтрація даних;
- групування даних;
- обробка форматів валюти та дати.



Очищення даних — це підмножина обробки даних, яка передбачає видалення непотрібних даних і обробку відсутніх даних. Для текстових даних може знадобитися видалити HTML-теги, знаки пунктуації тощо. Для числових даних можливо, що символи алфавіту змішані з числовими даними. Однак набір даних із числовими функціями може мати неправильні значення або відсутні значення. Крім того, обчислення мінімального, максимального, середнього, медіани та стандартного відхилення значень ознаки, стосується лише числових значень, що зображено на рис. 1.7.

На першому етапі попередньої обробки даних збирається відповідна інформація з ідентифікованих джерел, вибираються потрібні записи і змінні (на основі глибокого розуміння даних відфільтровується непотрібна інформація), а записи, що надходять із кількох інтегрованих / об'єднаних джерел (знову ж таки, використовуючи глибоке розуміння даних, синоніми та омоніми можуть оброблятися належним чином).

На другому етапі попередньої обробки даних інформація очищається (цей крок відомий як очищення даних). Дані в їх вихідному / необробленому / реальному вигляді ідентифікуються й обробляються.

У деяких випадках відсутні значення можна вважати аномалією в наборі даних, і в цьому разі їх потрібно приписувати (заповнювати найбільш імовірним значенням) чи ігнорувати; в інших випадках відсутні значення є природною частиною набору даних (наприклад, люди з найвищим рівнем доходу часто залишають без відповіді питання про доходи сім'ї). На цьому етапі аналітик також повинен визначити шумові значення в даних (тобто незвичайні значення в межах змінної) і згладити їх.

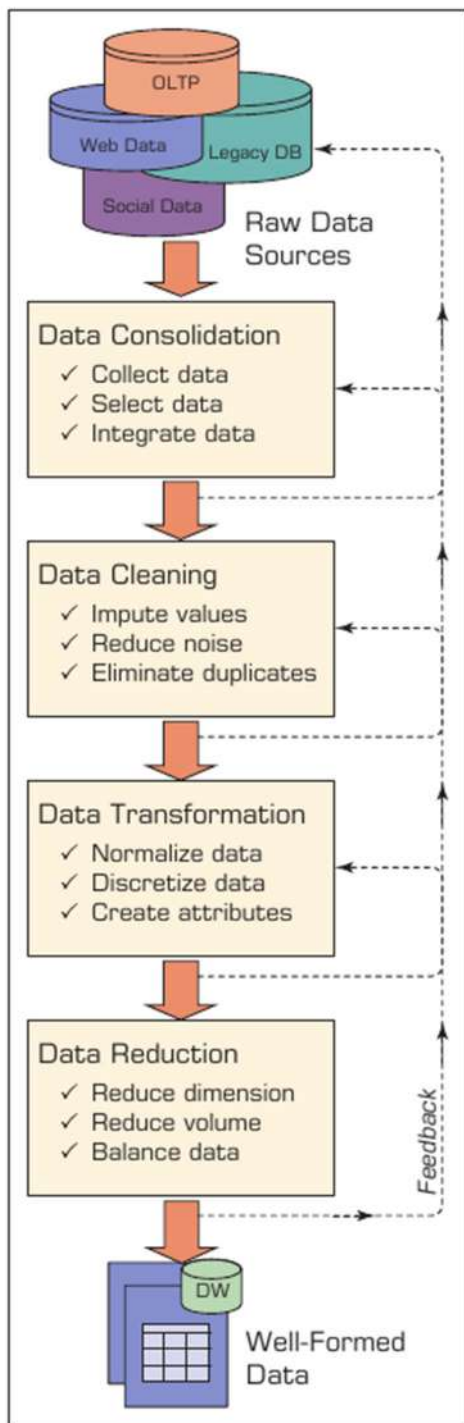


Рис. 1.7. Етапи попередньої обробки даних

На третьому етапі попередньої обробки даних інформація трансформуються для кращої обробки. Наприклад, у багатьох випадках дані нормалізуються між певним мінімумом і максимумом для всіх змінних, щоб пом'якшити потенційне зміщення однієї змінної, що має великі числові значення (наприклад дохід сім'ї), домінуючи над іншими змінними (такими, як кількість дітей чи роки вислуги, яка може бути важливішою). Інша трансформація, яка має місце, — це дискретизація та/або агрегація. Інколи числові змінні перетворюються на категоричні значення (наприклад, низький, середній, високий); в інших випадках унікальний діапазон значень номінальної змінної скорочується до меншого набору із використанням ієрархій понять (наприклад, на відміну від використання окремих 50 різних значень, можна вибрати використання кількох регіонів для змінної, яка показує розташування), щоб мати набір даних, який краще піддається комп'ютерній обробці. Проте подекуди можна вибрати створення нових змінних на основі існуючих, щоб розширити інформацію, знайдену у наборі даних. Наприклад, у наборі даних про переливання крові можна вибрати одну змінну, що показує збіг групи крові (1: збіг, 0: не збіг) на відміну від окремих мультиномінальних значень для групи крові як донора, так і одержувача. Таке спрощення могло б збільшити інформаційний вміст, одночасно знижуючи складність зв'язків у даних.

Останньою фазою попередньої обробки даних є їхня редукція. Незважаючи на те що спеціалісти з аналітики віддають перевагу великим наборам

даних, забагато інформації також може становить проблему. У найпростішому розумінні можна візуалізувати дані, які зазвичай використовуються в проєктах прогнозованої аналітики, як плоский файл, що складається з двох вимірів: змінних (кількість стовпців) і справ / записів (кількість рядків). У деяких випадках (наприклад, обробка зображень або проєкти геному зі складними даними мікроматриць) кількість змінних може бути достатньо великою, і аналітик повинен зменшити їхню кількість до контрольованого розміру. Оскільки змінні розглядаються як різні виміри, які описують явище з різних точок зору, у прогностичній аналітиці та смарт-аналітиці цей процес називають зменшенням розмірів. Навіть попри те, що немає єдиного найкращого способу виконання цього завдання, можна скористатися висновками з раніше опублікованих літературних джерел, провести відповідні статистичні тести (аналіз головних компонентів або аналіз незалежних компонентів).

1.2.6. Завдання, задачі, методи попередньої обробки даних

Варто також розглянути короткий опис завдань попередньої обробки даних, який наведено в табл. 1.1.

Таблиця 1.1

Основне завдання	Задачі	Методи
Консолідація даних	Отримайте доступ і зберіть інформацію.	SQL запити, програмні агенти, вебсервіси.
	Виберіть і відфільтруйте дані.	Доменна експертиза, SQL запити, статистичні тести.
	Інтегруйте та уніфікуйте дані	Запити SQL, експертиза домену, відображення даних на основі онтології
Очистка даних	Обробіть відсутні значення в даних.	Заповніть відсутні значення (імпутації) найбільш відповідними значеннями (середнє значення, медіана, мінімальне / максимальне значення, мода тощо); перекодуйте відсутні значення за допомогою константи, наприклад ML; видаліть запис пропущеного значення; нічого не робити.
	Виявіть і зменшіть шум у даних.	Визначайте викиди в даних за допомогою простих статистичних методів (таких як середні значення та стандартні відхилення) або за допомогою кластерного аналізу; після виявлення видаліть викиди чи згладьте їх за допомогою групування, регресії або простих середніх.
	Знайдіть і видаліть помилкові дані	Визначайте помилкові значення в даних, такі як непарні значення, непослідовні мітки класів, непарні розподіли; після ідентифікації скористайтеся знаннями домену, щоб виправити значення або видалити записи, що містять помилкові значення

Основне завдання	Задачі	Методи
Трансформація даних	Нормалізуйте дані.	Зменшіть діапазон значень у кожній змінній з числовим значенням до стандартного діапазону (наприклад, від 0 до 1 або від -1 до +1), використовуючи різні методи нормалізації чи масштабування.
	Дискретизуйте або агрегуйте дані	Якщо потрібно, перетворите числові змінні у дискретні представлення за допомогою методів групування на основі діапазону або частоти; для категоріальних змінних знизьте кількість значень, застосовуючи відповідні до ієрархії поняття
	Створіть нові атрибути	Отримайте нові та більш інформативні змінні з існуючих, використовуючи широкий спектр математичних функцій (простих, як додавання та множення, або складних, як гібридна комбінація логарифмічних перетворень)
Скорочення даних	Зменшити кількість атрибутів.	Використовуйте аналіз головних компонентів, аналіз незалежних компонентів, тестування χ^2 -квадрат, кореляційний аналіз та індукцію дерева рішень.
	Знизити кількість записів.	Виконуйте випадкову вибірку, стратифіковану вибірку, цілеспрямовану вибірку на основі експертних знань.
	Баланс спотворених даних	Перевибірка менш представлених класів або недостатня вибірка більш представлених класів



Попри експоненціальне зростання обчислювальної потужності, обробка такої великої кількості записів не вбачається практичною чи здійсненою. У таких випадках може знадобитися вибірка підмножини даних для аналізу. Припущення, що лежить в основі вибірки, полягає в тому, що підмножина даних міститиме всі релевантні моделі повного набору даних. У однорідному наборі даних таке припущення може мати місце, але дані реального світу майже ніколи не є однорідними. Аналітик повинен бути обережним у виборі підмножини даних, яка відображає суть повного набору даних і не є специфічною для підгрупи. Майже неможливо недооцінити цінність попередньої обробки даних. Це одна з тих видів діяльності, які потребують багато часу, інвестиції в яку окупаються без помітних обмежень для зменшення віддачі.

1.2.7. Модельна проблема



Припустимо, у вас є набір даних, що містить інформацію про продаж компанії за останні кілька років. Але дані не дуже чисті, і ви помітили, що є деякі проблеми із форматуванням і деякі рядки містять помилки. Необхідно очистити дані про продаж компанії за останні кілька років за допомогою бібліотеки Pandas. У світі більшість завдань аналітики пов'язані з обробкою великих обсягів даних, одержуваних від різних джерел. Часто ці дані містять помилки, дублікати і пропуски, що може призвести до неправильних висновків і рішень. Одним із інструментів, які дозволяють очистити та перетворити дані, є бібліотека Pandas для мови програмування Python.

Вихідні дані: файл CSV, що містить інформацію про продаж компанії за останні кілька років. Файл містить такі стовпці: дата продажу, назва продукту, кількість проданого товару, ціна за одиницю, загальна вартість продажу, ім'я продавця і регіон продажу. У деяких рядках є помилки, наприклад, неправильний формат дати або відсутність ціни за одиницю.

Завдання, які потрібно виконати: Завантажити вихідні дані з файлу Pandas DataFrame; видалити рядки, які містять помилки у даних; привести стовпець із датами до формату `datetime`; привести стовпці з числами до числового формату (`float` або `int`); визначити та видалити дублікати рядків; зберегти очищені дані до нового файлу CSV.

Опис стовпців:

`date` — дата продажу у форматі "YYYY-MM-DD";

`product_name` — назва продукту

`quantity` — кількість продукту

`unit_price` — ціна за одиницю продукту

`total_price` — загальна вартість продукту, що дорівнює добутку кількості та ціни за одиницю

`seller_name` — ім'я продавця

`region` — регіон продажу

Завантаження даних. Щоб завантажити дані в Pandas, можна використовувати метод `read_csv()` для завантаження даних із файлу CSV або `read_excel()` для завантаження даних із файлу Excel. У нашому випадку ми маємо csv файл. Імпортуємо потрібні бібліотеки та завантажуюмо дані.

```
import pandas as pd
import re
df = pd.read_csv('data_with_errors.csv')
```

Виводимо наш DataFrame. `df.head(5)`, як на рис. 1.8.

	date	product_name	quantity	unit_price	total_price	seller_name	region
0	2022-10-13	Product E	27	4.36	117.72	Bob	New York
1	2022-02-25	Product A	46	3.36	154.56	Bob	Miami
2	2022-10-01	Product E	41	-9.16	NaN	Bob	Miami
3	2022-12-20	Product E	46	-2.21	NaN	Bob	Miami
4	2022-09-15	Product D	44	7.12	313.28	Jane	Chicago
...	...	...	...	...	...	...	...
10095	2022-07-28	Product C	37	7.52	278.24	Kate	Miami
10096	2022-04-08	Product B	24	-9.39	NaN	Jane	Miami
10097	2022-10-03	Product B	-3	-5.66	NaN	Kate	Chicago
10098	2022-08-04	Product D	40	-2.98	NaN	John	New York
10099	2022-05-12	Product B	46	-7.42	NaN	Jane	Chicago

10100 rows x 7 columns

Рис. 1.8. Датафрейм у Pandas

Спочатку ми перевіримо типи стовпців, і якщо виявиться, що якісь стовпці не відповідають даним, які в них є, ми змінимо тип на відповідний. У разі виникнення проблем під час виконання цього завдання ми виправлятимемо те, що буде необхідно.

Спочатку поглянемо на те, які типи мають наші стовпці. Для цього допоможе команда `info`.

```
df.info()
```

Ми отримуємо інформацію про наш `DataFrame`, яка вказує нам на наявність пропусків у стовпці `total_price`. Також ми бачимо, що стовпець із датами має рядковий тип так само, як і стовпці `quantity` та `unit_price`, які містять числові дані. Нам потрібно це виправити.

Так як дати в нашому стовпці `date` вказані у форматі "YYYY-MM-DD", ми можемо використати регулярний вираз для пошуку всіх значень стовпця, які не відповідають цьому формату. Звідси ми створимо лямбда-функцію, яка буде використана до стовпця методом `map()`. Регулярний вираз виглядатиме так: ``\d{4}-\d{2}-\d{2}``.

```
search = lambda x: x if re.search(r"\d{4}-\d{2}-\d{2}", x) else 'not found'
```

Застосовуємо лямбда-функцію до стовпця `date`:

```
df['date'] = df['date'].map(search)
```

Перевіримо результат. `df.query('date == "not found").count()`

На рис. 1.9 бачимо, що у 53 рядках дані не відповідають формату.

```
date          53
product_name  53
quantity      53
unit_price    53
total_price   22
seller_name   53
region        53
dtype: int64
```

Рис. 1.9. Датафрейм (Pandas) / Рядки, що не відповідають формату

Погляньмо на ці рядки, щоб зрозуміти, з чим ми маємо справу:

```
df.query('date == "not found"').head(5)
```

Ми помічаємо відсутність дати, а також латинські літери замість чисел у стовпці кількості quantity. Крім того, стовпець total_price містить безліч пропусків, а unit_price має багато значень, що повторюються. У такому вигляді дані не становлять цінності для аналізу, і ми повинні видалити рядки, що містять помилки, як зазначено у завданні:

```
df = df.drop(df.query('date == "not found"').index)
df.query('date == "not found"').count()
```

Приведемо стовпець date до потрібного типу даних:

```
df['date'] = pd.to_datetime(df['date'], format='%Y-%m-%d')
df.info()
```

Бачимо, що стовпець date тепер має тип datetime.

Для того щоб привести стовпець quantity до числового формату, ми можемо використати метод to_numeric() з параметром errors='coerce', який перетворює значення в числа, а нечислові значення замінює на NaN:

```
df['quantity'] = pd.to_numeric(df['quantity'], errors='coerce')
```

Те саме ми робимо з unit_price:

```
df['unit_price'] = pd.to_numeric(df['unit_price'],
errors='coerce')
```

Для продовження ми сфокусуємось на негативних значеннях у вказаних стовпцях і виведемо їх на екран:

```
df.query('quantity < 0')
```

Ці значення буде перетворено за допомогою функції abs(), щоб змінити негативні значення у стовпцях. Ця операція дозволить отримати абсолютні значення та позбутися знаку мінус:

```
df.loc[df['quantity'] < 0, 'quantity'] = abs(df['quantity'])
```

Так само ми зробимо з `unit_price`:

```
df.query('unit_price < 0')
df.loc[df['unit_price'] < 0, 'unit_price'] = abs(df['unit_price'])
```

Обробимо стовпець `total_price`. У задачі вказано, що `total_price` є загальною вартістю продукту, яка дорівнює добутку кількості та ціни за одиницю. Отже, ми можемо заповнити порожні значення у цьому стовпці, помножуючи значення `quantity` на значення `unit_price`:

```
df.loc[df['total_price'].isna(), 'total_price'] =
df['quantity'] * df['unit_price']
```

Дані також містять кілька дублікатів, які треба видалити відповідно до завдання:

```
df[df.duplicated()]
```

Видаляємо дублікати:

```
df = df.drop(df[df.duplicated()].index)
```

Перевіримо категоріальні змінні:

```
df.product_name.unique()
df.seller_name.unique()
df.region.unique()
```

Усі значення категоріальних змінних у порядку. Залишилося тільки зберегти дані в CSV. У цьому допоможе функція `to_csv()`:

```
df.to_csv('processed_data.csv', index=False, header=True)
```

У результаті ми матимемо файл `processed_data.csv`, який вже можна використати для аналізу.

Запитання для самоперевірки



1. Чому вихідні / необроблені дані не можна легко використовувати в аналітичних завданнях?
2. Які основні етапи попередньої обробки даних?
3. Що означає очистити дані? Які дії виконуються на цьому етапі?
4. Навіщо нам потрібна трансформація даних? Які задачі перетворення даних найчастіше використовуються?
5. Скорочення даних можна застосувати до рядків (вибірка) та/або стовпців (відбір змінних). Що є більш складним?

1.3. Великі дані



Використання даних для розуміння поведінки клієнтів, бізнес-операцій, на підтримку зростання прибутковості стає дедалі складнішим завданням для сучасних підприємств. Оскільки все більше і більше даних стають доступними у різних формах і модах, своєчасна обробка даних традиційними засобами стає непрактичною. Це явище називається **Big Data** й активно висвітлюється в пресі, що викликає дедалі більший інтерес як у бізнес-користувачів, так і у IT-фахівців.

Великі дані перевищують потенціал загальноновживаних апаратних середовищ та/або можливості програмних інструментів для збору, керування, обробки у межах прийнятеного для користувачів проміжку часу.

1.3.1. Характеристика великих даних



Великі дані — це дані, які у нас є. Хороші дані — це дані, що відповідають поставленим завданням. Великі дані — це погані дані, якщо не можуть дійсно відповісти на запитання, що нас цікавлять.

- **Необ'єктивне представництво.** Будь-якому зовнішньому джерелу даних властива необ'єктивність вибірки. Дані з конкретного сайту соціальної мережі не відображають людей, які ними не користуються, тому з узагальненнями слід бути обережним.
- **Спам і вміст, створений машиною.** Великі джерела даних ще гірші за необ'єктивні. Нерідко вони були спроектовані так, щоб навмисно діяти нечесно. Будь-яка мережева платформа — досить обширна, щоб створювати величезні обсяги даних, досить велика, щоб існували економічні стимули для їх спотворення. Безліч проплачених рецензентів щодня публікують фальшиву інформацію, чим вводять в оману огляди продуктів. Роботи випускають механічно написаний словесний матеріал і ще більш довгі тексти у величезному обсязі: значна частина повідомлень на будь-якому сайті проводиться механічними ботами, а не людьми. Майже 90 % всієї електронної пошти, що надсилається мережами, є спамом: ефективність фільтрів спаму на декількох етапах передачі і є єдиною причиною, з якої ви його не бачите.
- **Надмірність.** Багато видів людської діяльності підпорядковуються розподілу за статичним законом, а отже, дуже мала частка предметів становить великий відсоток загальної активності. Новини та соціальні мережі значною мірою концентруються на останніх діях різних знаменитостей, що

присвячують їм тисячі статей. Багато з цих повідомлень буде майже точними копіями інших статей. Цей закон нерівномірного охоплення передбачає, що більшість даних з навколишніх джерел — це те, що ми вже переглядали і раніше. Усунення такого дублювання — важливий етап очищення інформації для багатьох програм.

- **Сприйнятливість до зміщення часу.** Товари і продукти змінюються у відповідь на конкуренцію та зміни споживчого попиту. Найчастіше інформація про покращення змінює спосіб використання цих продуктів. Тимчасовий ряд, отриманий в результаті збору даних, може кодувати кілька змін продукту / інтерфейсу, що ускладнює різницю між артефактом і сигналом. Наприклад, «Google Flu Trends», який протягом кількох років успішно прогнозував спалахи захворювань на основі запитів пошукових систем, втім згодом модель почала погано працювати. Одним із факторів було те, що Google додав механізм автозаповнення у полі, де він пропонує ввести пошуковий запит. Це змінило розподіл пошукових запитів достатньою мірою, щоб дані тимчасового ряду були незіставні з новими даними.

1.3.2. Ознаки великих даних



Big Data традиційно визначають за трьома V: обсяг (Volume), різноманітність (Variety), швидкість (Velocity). На додаток до трьох ознак деякі з провідних постачальників рішень для обробки великих даних додають інші V, такі як правдивість (IBM), варіативність (SAS) і ціннісні пропозиції.

• **Обсяг (Volume).** Обсяг, очевидно, є найпоширенішою рисою великих даних. Багато факторів сприяли експоненціальному збільшенню обсягу даних, наприклад, дані на основі транзакцій, що зберігаються роками, текстові дані, які постійно надходять із соціальних мереж, збільшення кількості даних датчиків, що збираються, автоматично генеровані дані RFID та GPS тощо. У минулому надмірний обсяг даних створював проблеми зі зберіганням, як технічні, так і фінансові. Але завдяки сучасним передовим технологіям у поєднанні зі зниженням витрат на зберігання, ці проблеми більше не є суттєвими; натомість виникли інші проблеми, зокрема, як визначити релевантність серед великих обсягів даних і як створити цінність із даних, які вважаються релевантними. Великі — поняття відносне, яке змінюється з часом і сприймається по-різному різними організаціями.

З приголомшливим збільшенням обсягу даних навіть визначення назви наступного ешелону великих даних стало проблемою. Найбільша маса даних, яку раніше називали петабайтами (PB), поступила місцем зетабайтам (ZB), що становить трильйон гігабайт (GB) або мільярд терабайт (TB).

Міра розміру даних важко встигає за новими назвами. Усі ми знаємо кілобайт (КБ дорівнює 1 тис. байтам), мегабайт (МБ = 1 млн байт), гігабайт (ГБ = 1 млрд байт) і терабайт (ТБ = 1 трлн байт). Щодня в Інтернеті створюється ексабайт даних (Exabyte), що дорівнює 250 млн DVD з інформацією. І ідея ще більших обсягів даних — зетабайт — не надто далека, коли йдеться про обсяг інформації, що передається Інтернетом за один рік. Очікується, що інтернет-трафік досягне 300 ГБ на душу населення на рік.

З огляду на коротку історичну перспективу, у 2009 р. у світі було близько 0,8 ZB даних; у 2010 р. обсяг перевищив позначку 1 ZB; на кінець 2011 р. — 1,8 ZB. Зі зростанням сенсорів та Інтернету речей (IoT — буде представлено в наступному розділі) усі ці прогнози можуть бути помилковими. Хоча ці цифри вражають розміром, можливості, які з ними пов'язані, набагато вищі.

- **Різноманітність (Variety).** Інформація сьогодні надходить в усіх типах форматів — від традиційних баз даних до ієрархічних сховищ даних, створених кінцевими користувачами та системами OLAP, текстових документів, електронної пошти, XML, показників лічильників і датчиків, відео, аудіо та відомостей з біржового тикеру. За деякими оцінками, від 80 до 85% даних усіх організацій перебувають в якомусь неструктурованому чи напівструктурованому форматі (це формат, який не підходить для традиційних схем баз даних).

- **Швидкість (Velocity).** Термін «швидкість» вказує як на швидкість створення даних, так і на швидкість їх обробки (тобто збору, зберігання та аналізу), щоб задовольнити потреби чи попит користувачів. RFID-мітки, автоматизовані датчики, GPS-пристрої та інтелектуальні лічильники викликають дедалі більшу потребу працювати з потоками даних майже в реальному часі. Швидкість — це, мабуть, найбільш критична характеристика для великих даних. З плином часу ціннісна пропозиція даних погіршується і зрештою стає низькою. Незалежно від того, чи йдеться про здоров'я пацієнтів, благополуччя системи транспортного трафіку, чи ціни інвестиційного портфеля, доступ до даних і швидша реакція на обставини завжди створюють кращі результати для бізнесу.

- **Достовірність (Veracity).** «Достовірність» — це термін, розроблений IBM, який використовується як четверта літера V в описі великих даних. Це означає відповідність фактам: точність, якість, правдивість даних. Інструменти та методи часто використовуються для обробки правдивості великих даних шляхом перетворення їх на якісну та надійну інформацію.

- **Мінливість (Variability).** На додаток до зростаючих швидкостей і різноманітності даних, їхні потоки можуть бути дуже несумісними з періодичними піками. Щоденні, сезонні та спричинені подіями пікові навантаження даних можуть великі відмінності, і тому їх складно контролювати, особливо з використанням соціальних мереж.

- **Цінність (Value proposition).** Популярність Big Data є її ціннісною пропозицією. Упереджене ставлення до великих даних полягає в тому, що вони містять (або мають містити) вищий потенціал, більше шаблонів і цікавих аномалій, ніж

невеликі за обсягом дані. Звідси, аналізуючи великі та багатофункціональні дані, організації можуть мати більшу цінність для бізнесу, якої могли б не мати в інший спосіб. Хоча користувачі можуть виявити закономірності в невеликих наборах даних за допомогою простих статистичних методів і методів машинного навчання чи спеціальних запитів і інструментів звітності, наявність Big Data вказуватиме на застосування методів Big Data аналітики.

Оскільки точне визначення великих даних (або термінів, які його замінюють), все ще лишається предметом обговорення в академічних і бізнесових колах, цілком імовірно, що до цього переліку буде додано ще характеристики (можливо, більше V).

На рис. 1.10 показано концептуальну архітектуру, де великі дані перетворюються на бізнес-аналітику за допомогою комбінації розширеної аналітики і надаються різним користувачам для швидшого прийняття рішень.

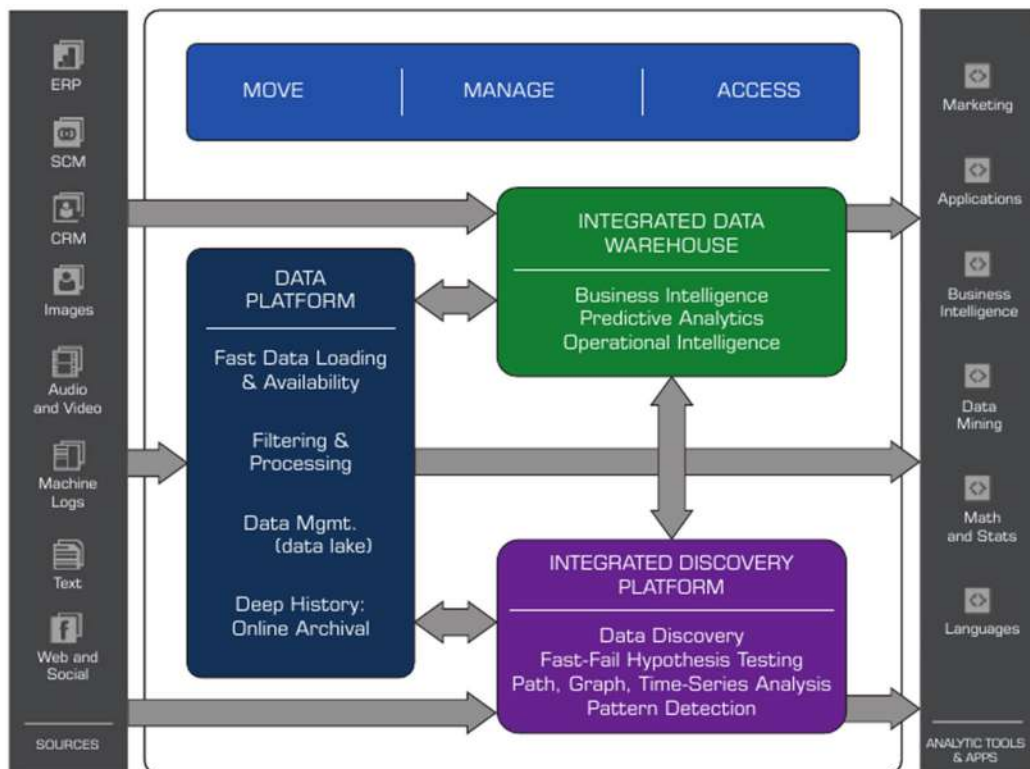


Рис. 1.10. Концептуальна архітектура високого рівня для організації великих даних

1.3.3. Високопродуктивні обчислення



Великі дані складаються з величезної кількості рядків у відносно невеликій кількості стовпців. Отже, великі дані часто надмірні, щоб точно відповідати одній моделі для певної проблеми. Цінність зазвичай полягає у підборі множини різних моделей, наприклад, у навчанні, та окремій моделі, що індивідуалізована для кожного окремого користувача.

Зі збільшенням розміру і складності підвищується потреба у більш ефективних аналітичних системах. Щоб не відставати від обчислювальних потреб Big Data, було розроблено низку інноваційних обчислювальних методів і платформ. Ці методи разом називаються *високопродуктивними обчисленнями*, і містять таке:

- **Аналітика у пам'яті:** вирішує складні питання майже в режимі реального часу з високоточними даними, дозволяючи аналітичним обчисленням і великим даним оброблятися в пам'яті та розподілятися між спеціальним набором вузлів.
- **Аналітика у базі даних:** пришвидшує час аналізу та покращує керування даними, виконуючи їхню інтеграцію та аналітичні функції всередині бази даних. Отже, вам не доведеться постійно переміщувати чи перетворювати дані.
- **Грид-обчислення:** сприяють ефективності, нижчій від вартості та кращій за продуктивність завдяки обробці завдань у спільному централізованому керованому пулі ІТ-ресурсів.
- **Прилади:** об'єднують апаратне і програмне забезпечення у фізичному блоці, який є не лише швидким, а й масштабованим у разі потреби.

Вимоги до обчислень — це лише частина переліку викликів, які великі дані ставлять перед сучасними підприємствами. Розглянемо перелік проблем, які, на думку керівників компаній, мають значний вплив на успішне впровадження аналітики великих даних:

Інтеграція даних: здатність об'єднувати дані, які різні за структурою чи джерелом, і робити це швидко і за розумну ціну.

Можливості обробки: можливість швидко обробляти дані, коли вони збираються. Традиційний спосіб збору й обробки даних може не спрацювати. У багатьох ситуаціях дані потрібно аналізувати відразу після їх збору, щоб отримати найбільшу цінність. (Це називається потоковою аналітикою.)

Управління даними: здатність не відставати від питань безпеки, конфіденційності, власності та якості великих даних. Зі зміною обсягу, різноманітності (формат і джерело) і швидкості даних мають змінюватися і можливості практики управління.

Наявність навичок: Великі дані використовуються за допомогою нових інструментів і розглядаються по-різному. Не вистачає людей (їх часто називають спеціалістами з обробки даних), які мають навички роботи з великими даними.

Вартість рішення: Щоб забезпечити позитивну віддачу від інвестицій у проєкт великих даних, вкрай важливо зменшити вартість рішень, які використовуються для визначення цієї цінності.

Такі хмарні платформи, як Amazon AWS, Google Cloud і Microsoft Azure, дозволяють легко орендувати велику (або невелику) кількість комп'ютерів для коротко- або довгострокових завдань.



MapReduce — це технологія, популяризована Google, яка розподіляє обробку дуже великих багатоструктурованих файлів даних між великим кластером машин. Висока продуктивність досягається шляхом розбиття обробки на невеликі одиниці роботи, які можуть виконуватися паралельно на сотнях, можливо, тисячах вузлів у кластері.



MapReduce — це модель програмування (а не мова програмування), призначена для використання програмістами, а не бізнескористувачами. Непрограмісти можуть використовувати переваги готових додатків MapReduce і бібліотек функцій. Доступні як комерційні, так і відкриті бібліотеки MapReduce, які надають широкий спектр аналітичних можливостей.

Наприклад, **Apache Mahout** — це бібліотека машинного навчання з відкритим вихідним кодом «алгоритмів для кластеризації, класифікації та пакетної спільної фільтрації», які реалізовані за допомогою MapReduce.

Більша частина великомасштабних завдань аналітики має таку базову структуру:

- Перебір великої кількості елементів, записи даних, текстових рядків чи каталогів файлів.
- Вилучення чогось цікавого з кожного елементу; чи це значення певного поля, підрахунок частоти кожного слова або наявність / відсутність програничних шаблонів в кожному файлі.
- Об'єднання проміжних результатів по всіх елементах та отримання відповідного комбінованого результату.



Великі колекції комп'ютерів можуть використовувати під час виконання завдання власний простір пам'яті (RAM) і локальне сховище, а не лише свої процесори. Розподілена файлова система, така як Hadoop, може бути реалізована у вигляді розподіленої хеш-таблиці. Після того як набір машин зареєструє свою доступну пам'ять у системі координації часу виконання, кожному з них може бути призначений певний діапазон хеш-таблиць, за який він буде відповідати. Кожен процес, що виконує трансформацію, може гарантувати, що передані елементи будуть відправлені у відповідне сховище на відповідному комп'ютері.

Повна екосистема великих даних складається з безлічі різних платформ і технологій. Важливим класом є бази даних **NoSQL**, які здатні розподіляти структуровані дані по розподіленій мережі машин, дозволяючи об'єднувати RAM і диски кількох машин. Крім того, ці системи зазвичай розробляються так, щоб у разі потреби могли і приєднувати додаткові машини і залучати нові ресурси. Ціна такої гнучкості полягає в тому, що бази традиційно підтримують більш прості мови запитів, ніж повноцінний SQL, але все ж таки вони досить обширні для багатьох додатків.



Великі дані можуть дати багато чого для світу, проте вони також можуть завдати і шкоди людям та загалом суспільству. Поведінка, яка нешкідлива у невеликих обсягах, наприклад, плагіат, у значних масштабах стає крадіжкою інтелектуальної власності. Опис точності вашої математичної моделі у надмірно сприятливому світлі вважається звичайним явищем для презентацій PowerPoint, втім матиме реальні наслідки, коли ваша модель регулює дозвіл на одержання кредиту чи для доступ до медичного обслуговування. Втрата доступу до вашого облікового запису електронної пошти можна вважати неприємністю, проте неправильний захист особистих даних ста мільйонів клієнтів — це потенційно злочинно.

1.3.4. Модельна проблема



Представники цього класу завдань включають підрахунок частоти слів, кластеризацію методом k-середніх і обчислення Page Rank. Усіх їх можна вирішити за допомогою простих ітеративних алгоритмів, чий час роботи лінійно залежить від розміру вхідних даних. Але цього може бути замало для вхідних даних великого обсягу, коли файли не розміщуються у пам'яті однієї машини. Цього може бути недостатньо для вхідних даних великого розміру, коли файли не містяться у пам'яті однієї машини, наприклад, під час підрахунку частоти вживаних слів у мільярдах повідомлень соцмереж, або кластеризація методом k-середніх у сотнях мільйонів профілів Facebook.

Ключем до розподілу таких обчислень є налаштування розподіленої хеш-таблиці сегментів, в якій всі елементи з однаковим ключем співвідносяться з одним і тим же сегментом.

Підрахунок слів. Для розрахунку загальної частоти конкретного слова w у наборі файлів нам потрібно зібрати значення частоти для всіх файлів в одному сегменті, пов'язаному з w , де їх можна буде скласти, щоб отримати підсумкову суму.

Кластеризація методом k-середніх. Критичним кроком у кластеризації методом k-середніх є оновлення нового центроїду c' з точок, найближчих до поточного центроїду c . Після хешування всіх точок p , найближчих до c , в одному сегменті, пов'язаному з c , можна обчислити c' за один прохід через цей сегмент.

PageRank. Новий PageRank вершини v є сумою старого Page Rank для всіх сусідніх вершин x , де (x, v) — це направлене ребро у графі. Хешування PageRank x у сегмент для всіх з граничних вершин v збирає всю необхідну інформацію в потрібному місці, тому можна оновити PageRank за один прохід через нього.

Ці алгоритми можуть бути визначені за допомогою двох нанесених програмістом функцій *map* і *reduce*.



Функція *map*. Здійснює трансформацію кожного вхідного файлу, хешує, або випускає (*emitting*), пари ключових значень залежно від ситуації.

псевдокод для трансформації кількості слів:

```
Map (String docid, String text):  
  foreach word w in text:  
    Emit (w, 1);
```



Функція *reduce* здійснює перегляд набору значень v , пов'язаних з конкретним ключем k , агрегуючи й обробляючи їх належним чином. Псевдокод цієї функції для кількості слів:

```
Reduce (String term, Iterator < Int > values):  
  int sum = 0;  
  foreach v in values:  
    sum + = v;
```

Ефективність програм **MapReduce** залежить від багатьох речей, проте одне з найважливіших завдань полягає в тому, щоб звести до мінімуму кількість викидів. Створення лічильника для кожного слова запускає повідомлення по різних машинах, і це спілкування та пов'язані з ним записи у пакет виявляються дорогими. Що більше матеріалів трансформується (*mapped*), то більше має бути редуковано (*reduced*).

Ідеальним варіантом вбачається об'єднання підрахунків з окремих вхідних потоків і подальше випромінювання лише суми для кожного окремого слова у файлі. Це можна зробити, додаючи додаткову логіку / структуру даних функцію *map*. Альтернативний підхід — запускати мініредуктори у пам'яті після фази трансформації, але до міжпроцесорного зв'язку — як оптимізація і для зниження мережного трафіку. Зауважимо, що оптимізація для обчислень у пам'яті є однією з основних переваг продуктивності у Spark порівняно з Hadoop для програмування у стилі MapReduce.

На рис. 1.11 показано потік завдань і MapReduce для підрахунку слів із використанням трьох трансформаторів (*mapper*) і двох редукторів (*reducer*). Об'єднання було здійснено локально, тому лічильники кожного слова, використаного більше від одного разу у вхідному файлі (тут *doc i be*), були зведені в таблицю до передачі редукторам.

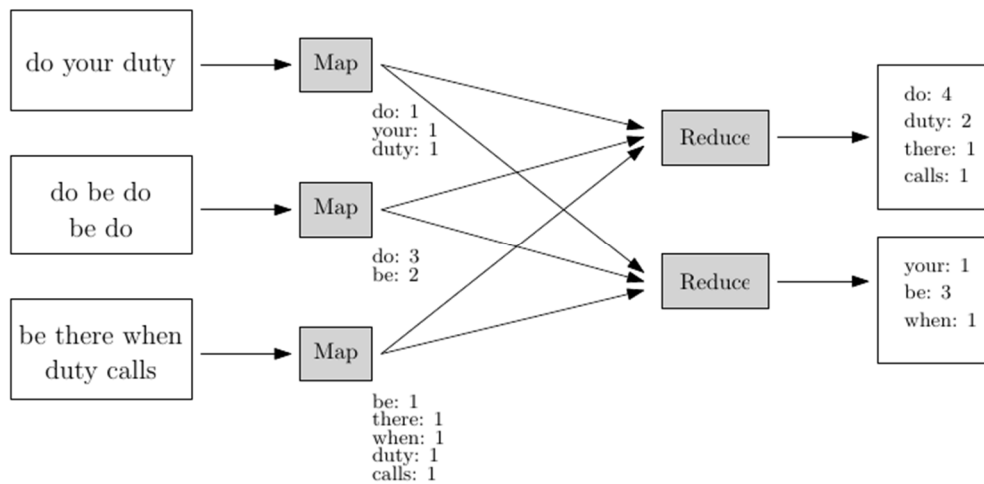


Рис. 1.11. Підрахунок кількості слів у дії (перед трансформацією було здійснено об'єднання підрахунків для зменшення розміру файлів трансформації)

Запитання для перевірки знань



1. Що таке великі дані? Чому це важливо? Звідки беруться великі дані?
2. Яким, на вашу думку, буде майбутнє великих даних? Чи втраять вони свою популярність?
3. Що таке аналітика великих даних? Чим вона відрізняється від звичайної аналітики?
4. Які критичні фактори успіху для аналізу великих даних?
5. Про які основні проблеми слід пам'ятати, розглядаючи впровадження аналітики великих даних?
6. Які основні властивості великих даних?
7. Назвіть поняття технології MapReduce та її складові.

1.4. Штучний інтелект: концепції, драйвери, основні технології та бізнес-додатки



Штучний інтелект має кілька визначень, проте багато експертів погоджуються, що ШІ пов'язаний з двома основними ідеями: 1) вивчення процесів мислення людини, щоб зрозуміти, що таке інтелект; і 2) представлення та дублювання цих процесів мислення у машинах, наприклад, комп'ютерах, роботах. Іншими словами, очікується, що машини будуть мати людські процеси мислення.



Одним із широко розрекламованих трактувань штучного інтелекту, за словником Webster, є «здатність машини імітувати інтелектуальну поведінку людини». Теоретичні основи ШІ базуються на логіці, яка використовується в кількох інноваціях у сфері інформатики. Звідси ШІ вважається підгалуззю інформатики. Добре відомим раннім застосуванням штучного інтелекту була шахова програма Deep Blue, розміщена на суперкомп'ютері IBM, якій програв чемпіон світу, гросмейстер Гаррі Каспаров. Штучний інтелект — це загальний термін для багатьох методів, які мають схожі можливості і характеристики.

Штучний інтелект зараз швидко поширюється завдяки його можливостям. Щоб зрозуміти, що таке штучний інтелект, потрібно спочатку визначити поняття інтелекту.



Інтелект можна вважати загальним терміном і зазвичай його вимірюють за допомогою тестів IQ. Однак деякі дослідники стверджують, що існує кілька типів інтелекту. Наприклад, доктор Говард Гарднер з Гарвардського університету запропонував таку класифікацію типів інтелекту:

- мовно-словесний;
- логічний;
- просторовий;
- тіло / рух;
- музичний;
- міжособистісний;
- внутрішньоособистісний;
- натуралістичний.

Інтелект складається з міркувань, навчання, логіки, здатності вирішувати складні питання, сприйняття і мовних здібностей. Сфери, для яких штучний інтелект корисний:

1. Боротьба з кіберзлочинністю (наприклад, ШІ може визначити майбутні кібератаки та запропонувати рішення).

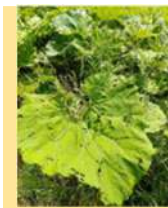
2. Рішення щодо електронної комерції. Маркетингові інструменти штучного інтелекту можуть зробити результати тестування набагато швидшими й адаптувати макет і функції відповіді вебсайту до користувачів. Машини також дають рекомендації, і маркетологи можуть прийняти або відхилити їх.
3. Високочастотна біржова торгівля. Цей процес можна виконувати або повністю автономно, або у деяких випадках за допомогою людського контролю та калібрування.

Отже, щоб зрозуміти, що таке *штучний інтелект*, корисно спочатку вивчити ті здібності, які вважаються ознаками людського інтелекту:

- навчання або розуміння на досвіді;
- вивчення неоднозначних, неповних чи навіть суперечливих повідомлень та інформації;
- швидке й успішне реагування на нову ситуацію (тобто використання найбільш правильних відповідей);
- раціональне розуміння і висновки, вирішення проблем та ефективне керування поведінкою;
- застосування знань для маніпулювання середовищем і ситуаціями;
- розпізнавання та оцінка відносної важливості різних елементів у ситуації.

Штучний інтелект намагається надати певні можливості, проте в цілому досі не здатний зрівнятися з людським розумом. Відсоток невдач ініціатив ШІ також високий. Основною причиною вважається те, що деякі технології штучного інтелекту потребують великої кількості даних, іноді надвеликих. Наприклад, багато мільйонів елементів даних щодня передаються в Alexa, щоб збільшити її знання. Без постійного потоку інформації не було з чого вчитися в ШІ. Питання полягає в тому, чи можна поєднати ШІ та аналітику, інші інтелектуальні системи у такий спосіб, щоб досягти синергії для кращих результатів.

1.4.1. Драйвери розвитку штучного інтелекту



Зростає тенденція робити комп'ютери ще розумнішими. Концепція Web 3.0 передбачає створення комп'ютеризованих систем, які демонструють значно більший інтелект, ніж Web 2.0. Кілька додатків уже базуються на методах ШІ. Сфера машинного перекладу з різних мов допомагає людям співпрацювати, а також купувати онлайн-продукти, які рекламуються мовами, якими не володіють.

Основні елементи штучного інтелекту. Ландшафт штучного інтелекту величезний і містить сотні або тисячі компонентів. Етапи розвитку штучного інтелекту показано на рис. 1.12.

Використання штучного інтелекту було зумовлено такими драйверами:

- інтерес людей до розумних машин і штучного мозку;
- низька вартість додатків штучного інтелекту порівняно з високою вартістю ручної праці (виконання тієї самої роботи);

- бажання великих технологічних компаній набути конкурентну перевагу і частку ринку на ринку штучного інтелекту та їх готовність інвестувати мільярди доларів у штучний інтелект;
- тиск на керівництво з метою підвищення продуктивності та швидкості;
- наявність якісних даних, які сприяють розвитку штучного інтелекту;
- збільшення функціональних можливостей і зниження загальної вартості комп'ютерів;
- розвиток нових технологій, зокрема хмарних обчислень.



Рис. 1.12. Етапи розвитку штучного інтелекту

Типові переваги штучного інтелекту у різних сферах

1. Міжнародна асоціація Swaps and Derivatives (ISDA) використовує штучний інтелект, щоб усунути виснажливу діяльність у контрактних процедурах. Наприклад, за допомогою оптичного розпізнавання символів (OCR), інтегрованого з штучним інтелектом, ISDA оцифровує контракти, а потім визначає, витягує та архівує контракти.
2. Штучний інтелект починає революціонізувати процес бізнес-наймання персоналу, проводячи більш ефективний і справедливий відбір кандидатів, покращуючи відповідність кандидатів робочим місцям і допомагаючи захистити майбутні кадри для організацій.
3. Штучний інтелект змінює визначення менеджменту. За даними Kolbjørnsrud et al. (2016), наступні п'ять практик є результатом використання ШІ: він може виконувати рутинні адміністративні завдання; менеджери можуть зосередитися на частині роботи, пов'язаній із судженнями; до інтелектуальних машин ставляться як до колег (тобто менеджери довіряють порадам, створеним штучним інтелектом); менеджери зосереджуються на творчих здібностях, які можуть підтримуватися машинами ШІ; менеджери розвивають соціальні навички, необхідні для кращої співпраці, лідерства та навчання.
4. Компанія Accenture Inc. розробила рішення на основі штучного інтелекту з використанням обробки природної мови NLP і розпізнавання зображень, щоб допомогти сліпим людям в Індії покращити спосіб сприйняття навколишнього світу. Це дає їм змогу мати якісніше життя, а ті, хто працює, можуть робити це краще, швидше, виконуючи складнішу роботу.

5. Компанія Ford Motor Credit використовує машинне навчання для виявлення позичальників, яких не помітили, крім того, щоб допомогти андерайтерам краще розуміти позичальників. Програма сприяє підвищенню продуктивності як андерайтерів, так і заявників, яких не помічають. Нарешті, система прогнозує кредитоспроможність потенційних позичальників, у такий спосіб мінімізуючи втрати для Ford.
6. Аластер Коул використовує дані, зібрані з кількох джерел разом із IBM Watson, щоб передбачити, чого клієнти очікують від компанії. Згенеровані дані використовують для підтримки більш ефективних бізнес-рішень.
7. Компанії будують бізнес навколо штучного інтелекту. Є багато прикладів стартапів або існуючих компаній, які намагаються створити новий бізнес.

Основні технології штучного інтелекту та їх похідні щодо бізнесу подано на рис. 1.13.

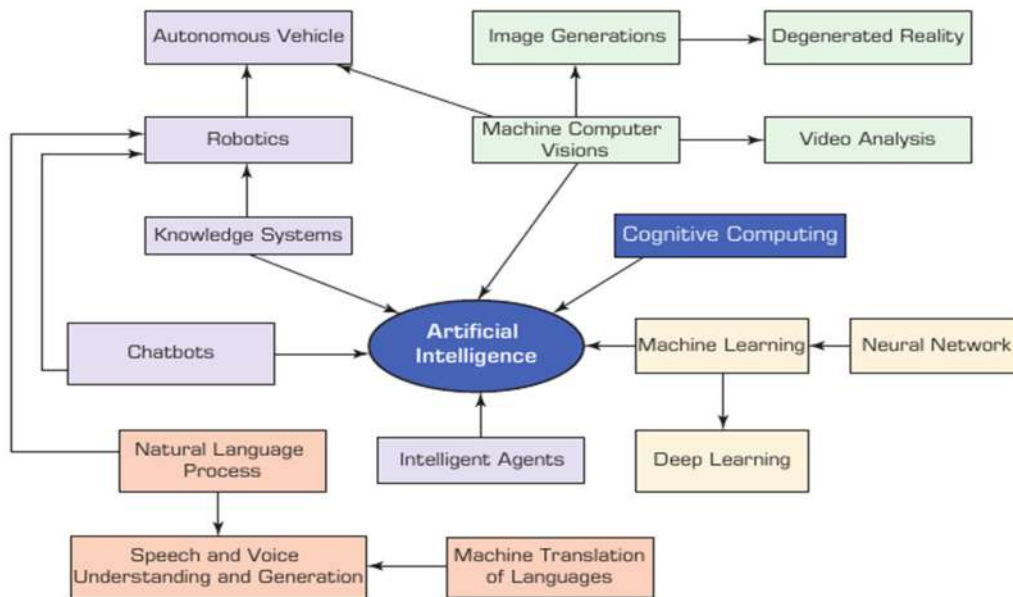


Рис. 1.13. Основні технології штучного інтелекту

1.4.2. Типи штучного інтелекту

Генеративний штучний інтелект — це тип штучного інтелекту, який здатний генерувати зображення, тексти та інші типи контенту у відповідь на спеціальні запити користувача — **prompts***.

Популярність генеративного AI:

- у листопаді 2022 р. був випущений ChatGPT від OpenAI — чат-бот на базі GPT-3 мовної моделі;

- у листопаді 2022 р. була випущена четверта версія моделі Midjourney для генерації зображень;
- листопад став стартом для популярності генеративного AI, оскільки контент, згенерований цими моделями, був дуже якісний.

Мовне моделювання — це завдання присвоєння ймовірності послідовності слів у тексті певною мовою. Ґрунтуючись на статистичному аналізі існуючих текстових послідовностей, прості мовні моделі можуть дивитися на слово та передбачати наступне слово (або слова), яке, швидше за все, йтиме за ним. Щоб створити мовну модель, яка успішно передбачає послідовність слів, ви повинні навчити її на великих наборах даних.

До GPT-3 не існувало загальної мовної моделі, яка могла б добре працювати з масивом завдань NLP. Мовні моделі були розроблені для виконання одного завдання NLP, такого як генерація тексту, резюмування або класифікація.

GPT-3 є генеративною моделлю, оскільки генерує текст. Генеративне моделювання — розділ статистичного моделювання. Це метод математичного наближення світу. Ми оточені неймовірною кількістю легкодоступної інформації — як у фізичному, так і в цифровому світах. Складна частина полягає в розробці інтелектуальних моделей і алгоритмів, які можуть аналізувати та розуміти цю скарбницю даних. Генеративні моделі є одним із найбільш перспективних підходів до досягнення цієї мети.

Щоб навчити модель, ви повинні підготувати та попередньо обробити набори даних, прикладів, що допомагає моделі навчитися виконувати задане завдання. Зазвичай набір даних — це велика кількість даних у певному домені: наприклад, мільйони зображень автомобілів, щоб навчити моделі, що таке автомобіль. Набори даних також можуть мати форму речень або зразків звуку. Після того, як ви показали моделі багато прикладів, ви повинні навчити її створювати схожі дані.

Попередньо підготовлені моделі. У книзі «Outliers» Малкольм Ґладуелл припускає, що для того, щоб стати експертом, достатньо практикувати будь-які навички протягом 10 тис. годин. Ці експертні знання відображаються у зв'язках, які ваш людський мозок розвиває між своїми нейронами. Модель AI робить щось подібне. Щоб створити ефективну модель, її потрібно навчити за допомогою певного набору змінних, які називаються *параметрами*.

Процес визначення ідеальних параметрів для вашої моделі називається навчанням. Модель асимілює значення параметрів через послідовні ітерації навчання. Модель глибокого навчання займає багато часу, щоб знайти ці ідеальні параметри. Навчання — тривалий процес, який залежно від завдання може розтягнутися від кількох годин до кількох місяців і потребує величезної обчислювальної потужності. Повторне використання частини цього довгого процесу навчання для інших завдань стане в нагоді. І тут на допомогу приходять попередньо навчені моделі.

Така модель, яка відповідає теорії «10 тисяч годин Ґладуелла», є першою навичкою, яку ви розвиваєте, щоб швидше здобути іншу. Наприклад, оволодіння ремеслом розв'язування математичних задач може дозволити вам більшою

мірою набути навичок розв'язування інженерних задач. Попередньо навчена модель вивчається для узагальненого завдання і може бути налаштована для різних завдань. Замість того щоб створювати абсолютно нову модель для вирішення проблеми, можна використовувати попередньо навчену модель для виконання більш загальної проблеми. Попередньо навчена модель може бути налаштована відповідно до ваших конкретних потреб, забезпечуючи додаткове навчання завдяки спеціальному набору даних. Цей підхід швидший та ефективніший та дозволяє підвищити продуктивність порівняно зі створенням моделі з нуля.

У машинному навчанні модель тренується на наборі даних. Розмір і тип вибірки даних змінюються залежно від завдання, як вказано у табл. 1.2.

Таблиця 1.2

Назва набору даних	Зміст
Common Crawl	Петабайти даних, включаючи необроблені дані вебсторінок, мета- і текстові дані, зібрані протягом восьми років вебсканування. Дослідники OpenAI використовують підібрану відфільтровану версію цього набору даних
WebText2	Розширена версія набору даних WebText, створеного шляхом копіювання вебсторінок особливо високої якості. Щоб перевірити якість, автори збрали всі вихідні посилання з Reddit, які отримали щонайменше три карми (індикатор того, чи вважали посилання іншим користувачам цікавим, навчальним або просто смішним). WebText містить 40 гігабайт тексту з цих 45 млн посилань і понад 8 млн документів
Books1 і Books2	Містять тексти десятків тисяч книг на різні теми
Wikipedia	Колекція, що включає всі англomовні статті з краудсорсингової онлайн-енциклопедії Wikipedia

- **Моделі трансформерів.** Нейронні мережі лежать в основі глибокого навчання, їхня назва та структура навіяні людським мозком. Вони складаються з мережі або ланцюга нейронів, які працюють разом. Досягнення нейронних мереж можуть підвищити ефективність моделей штучного інтелекту для вирішення різних завдань, що спонукає вчених ШІ постійно розробляти нові архітектури для цих мереж. Одним із таких досягнень є *трансформатор*, модель машинного навчання, яка обробляє послідовність тексту одразу, а не одне слово за раз, і має потужну здатність розуміти зв'язок між цими словами. Цей винахід кардинально вплинув на сферу обробки природної мови.

Основою таких трансформаторних моделей є архітектура послідовності. Моделі послідовності (Seq2Seq) корисні для перетворення послідовності елементів, наприклад, слів у реченні, в іншу послідовність, наприклад, речення іншою мовою. Це особливо ефективно в завданнях перекладу, коли послідовність слів однією мовою перекладається на послідовність слів іншою мовою. **Google Translate** почав використовувати подібну модель на основі Seq2Seq ще у 2016 р. (рис. 1.14).

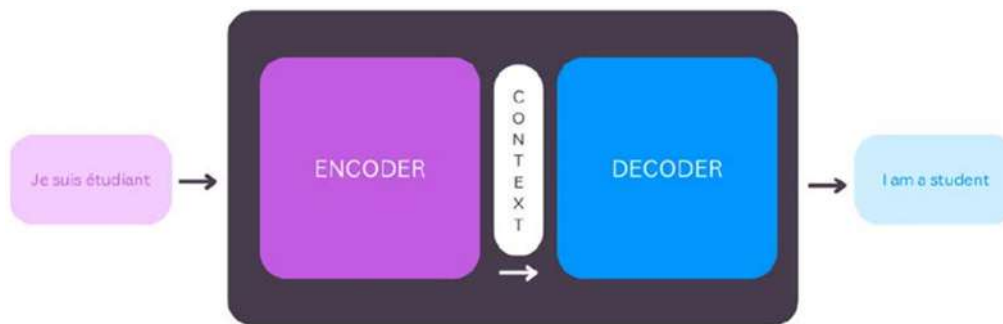


Рис. 1.14. Seq-to-Seq Model Neural Machine Translation

Моделі **Seq2Seq** складаються з двох компонентів: енкодера та декодера. Encoder можна розглядати як перекладача, який розмовляє французькою як своєю першою мовою та корейською — як другою мовою. Декодер — це перекладач, який говорить англійською як своєю першою мовою та корейською — як другою мовою. Щоб перекласти французьку на англійську, кодувальник перетворює французьке речення на корейське (також відоме як контекст) і передає його декодеру. Оскільки декодер розуміє корейську, то може перекладати речення з корейської на англійську.

Для ефективного використання моделей-трансформерів вкрай важливо зрозуміти *концепцію уваги*. Механізми уваги імітують те, як людський мозок зосереджується на певних частинах вхідної послідовності, використовуючи ймовірності, щоб визначити, які частини послідовності є найбільш релевантними на кожному кроці.

Модель трансформатора має два типи уваги: самоувага (зв'язок слів у реченні) і увага кодера-декодера (зв'язок між словами вихідного речення та словами цільового речення). Механізм уваги допомагає трансформатору відфільтрувати шум і зосередитися на тому, що актуально: з'єднання двох слів у семантично одне до одного, які не містять жодних явних маркерів, що вказують одне на одне. Отже, така модель виграє від більшої архітектури та більшої кількості даних. Навчання на великих наборах даних і точне налаштування для конкретних завдань покращують результати. Трансформери ліпше розуміють контекст слів у реченні за будь-який інший вид нейронної мережі. GPT — це лише декодерна частина трансформатора. Тепер ви знаєте, що означає GPT.

• **Персональний штучний інтелект** — це системи штучного інтелекту, які створені для надання, допомоги та взаємодії між окремими особами на персоналізованому рівні з урахуванням їхніх унікальних потреб та уподобань.



Функції персонального штучного інтелекту:

- генерація різних планів та ідей (подорожей, свят, днів народження, весілля);
- швидке пояснення складних тем (поясни мені);
- написання чи відладка програмного коду (Copilot);



- виявлення помилок у тексті, корекція (Grammarly);
 - симуляція бесіди з вигаданим персонажем (оціни моє есе як учитель);
 - генерація маркетингових текстів, листів, привітань, рекомендаційних листів;
- генерація зображень для блогів, презентацій.



Життєвий цикл машинного навчання, схему якого наведено на рис. 1.15, — це періодичний процес, що починається з бізнес-завдання, а його останнім етапом є моніторинг та оптимізація. Якщо у бізнес-вимоги внесено зміни, доведеться знову виконувати всі етапи життєвого циклу ML.



Рис. 1.15. Життєвий цикл проекту з використання штучного інтелекту та машинного навчання

1.4.3. Модельна проблема

Модельні проблеми побудови запитів — prompts.



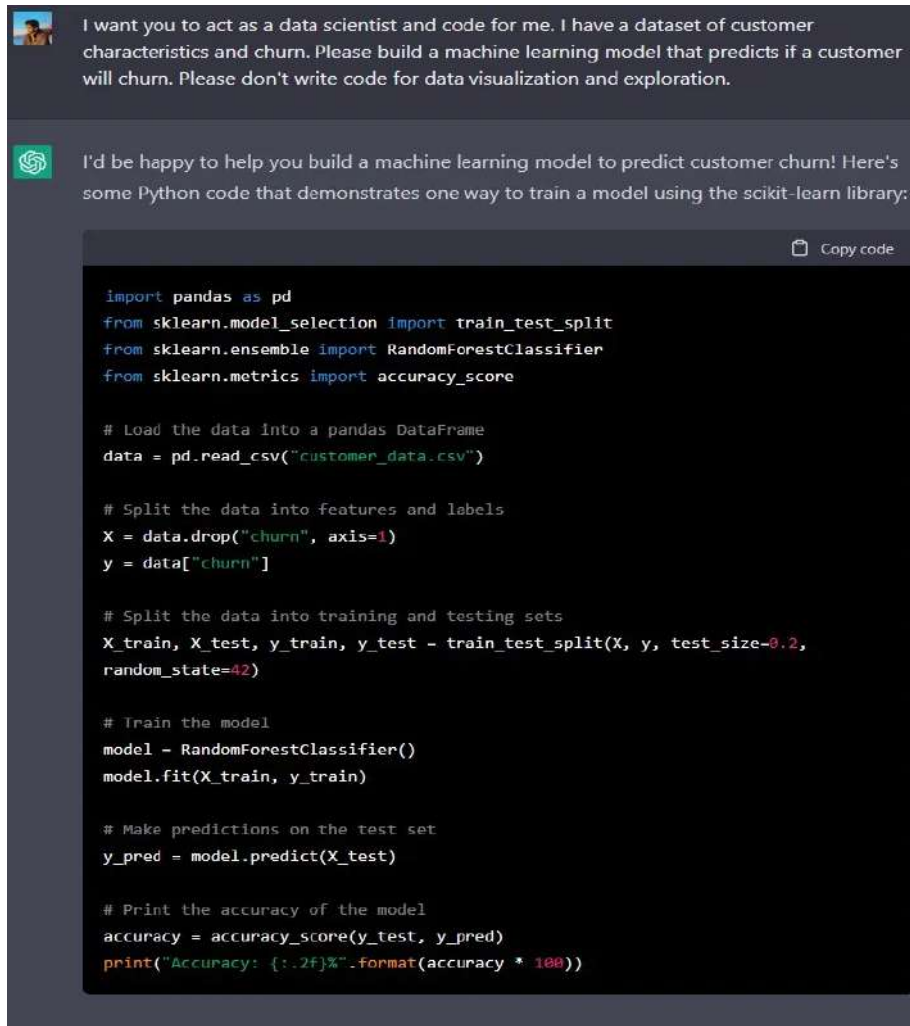
Prompts може містити різні елементи:

- Інструкція — це конкретне завдання або вказівка, яку має виконувати модель штучного інтелекту.
 - Контекст — надає додаткову інформацію або контекст, щоб допомогти моделі генерувати кращі відповіді.
 - Вхідні дані — це стосується питання або вхідних даних, на які ми хочемо, щоб модель дала відповідь.
 - Індикатор виводу — вказує на потрібний тип або формат виводу.
- Не всі чотири елементи потрібні для підказки, а формат залежить від конкретного завдання, яке виконується.

Щоб розробити ефективні запити для простих завдань, використовуйте інструктивні команди, такі як «Написати», «Класифікувати», «Підсумувати», «Перекласти», «Упорядкувати» тощо. Експериментування є ключовим у пошуку найкращого підходу. Деякі рекомендації включають розміщення вказівок на початку підказки та використання чіткого роздільника, наприклад «###», щоб відрізнити вказівки від контексту.

1. Модель класифікації поїздів

Запит. Я хочу, щоб ви діяли як науковець із даних і написали код. У мене є набір даних [describe dataset]. Будь ласка, створіть модель машинного навчання, яка передбачає [target variable], як подано на рис. 1.16.



The screenshot shows a chat window with a user's request and a response from a data scientist. The user asks for a machine learning model to predict customer churn. The response includes a Python script using pandas, sklearn, and RandomForestClassifier to load data, split it into training and testing sets, train the model, and print the accuracy.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load the data into a pandas DataFrame
data = pd.read_csv("customer_data.csv")

# Split the data into features and labels
X = data.drop("churn", axis=1)
y = data["churn"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=42)

# Train the model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

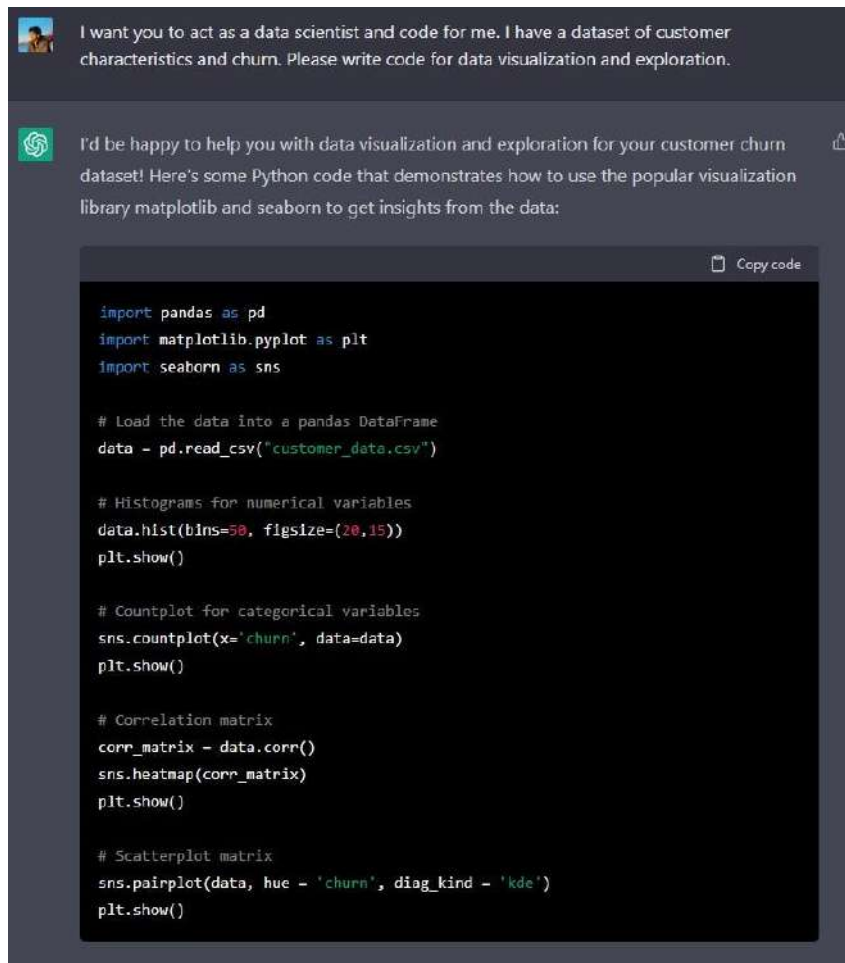
# Print the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy * 100))
```

Рис. 1.16. Код програми

Цей код передбачає, що ваші дані містяться у файлі CSV «customer_data.csv», а останній стовпець — це мітка «churn». Код використовує класифікатор випадкового лісу для навчання моделі, але ви також можете експериментувати з іншими алгоритмами. Код також використовує оцінку точності як метрику для оцінки моделі, але для цього можна застосовувати інші показники оцінки.

2. Дослідіть дані

Запит. Я хочу, щоб ви діяли як аналітик і написали код. У мене є набір даних [описати набір даних]. Будь ласка, напишіть код для візуалізації та дослідження даних, як для візуалізації вказано на рис. 1.17.



The screenshot shows a chat window with a user message and a GPT response. The user message is: "I want you to act as a data scientist and code for me. I have a dataset of customer characteristics and churn. Please write code for data visualization and exploration." The GPT response is: "I'd be happy to help you with data visualization and exploration for your customer churn dataset! Here's some Python code that demonstrates how to use the popular visualization library matplotlib and seaborn to get insights from the data:". Below the response is a code block with the following Python code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data into a pandas DataFrame
data = pd.read_csv("customer_data.csv")

# Histograms for numerical variables
data.hist(bins=50, figsize=(20,15))
plt.show()

# Countplot for categorical variables
sns.countplot(x='churn', data=data)
plt.show()

# Correlation matrix
corr_matrix = data.corr()
sns.heatmap(corr_matrix)
plt.show()

# Scatterplot matrix
sns.pairplot(data, hue = 'churn', diag_kind = 'kde')
plt.show()
```

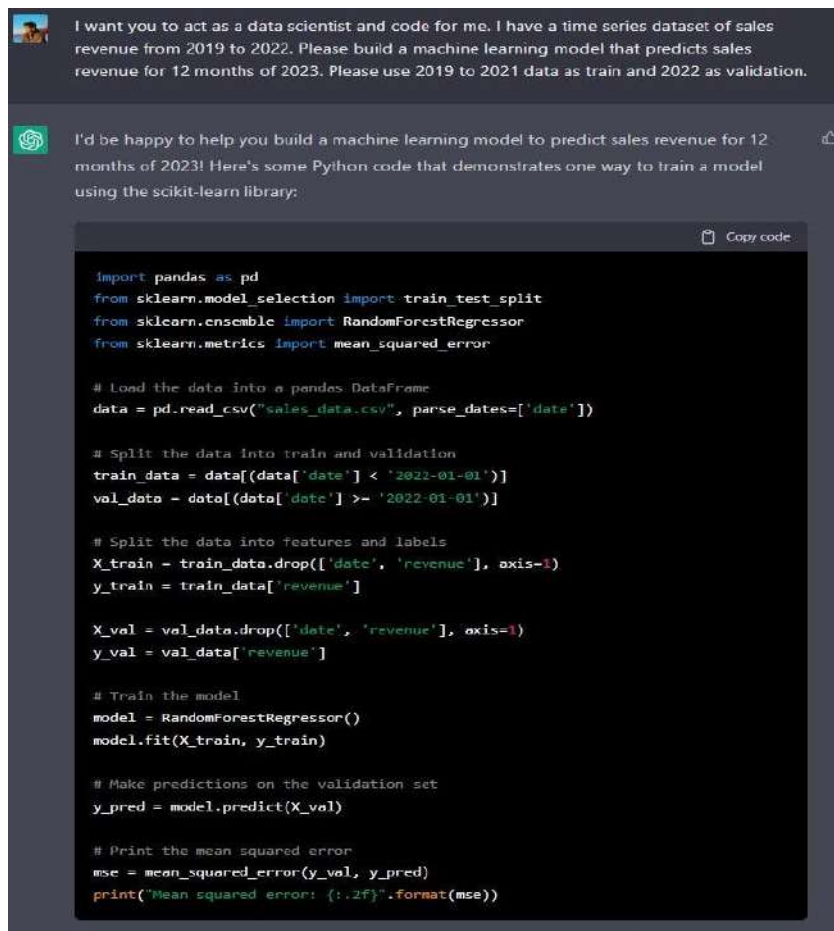
Рис. 1.17. Код програми

Цей код завантажує дані клієнта в **Pandas DataFrame**, а потім створює різні типи візуалізацій, щоб отримати інформацію з даних.

- Гістограми для всіх числових змінних, щоб перевірити розподіл кожної змінної.
- Графічний графік для змінної «відтік», щоб побачити розподіл клієнтів, які відмовилися, і тих, хто ні.
- Кореляційна матриця, щоб побачити, які функції сильно корелюють одна з одною.
- Матриця діаграми розсіювання для візуалізації зв'язку між функціями та цільовою змінною.

3. Часовий ряд поїздів

Запит. Я хочу, щоб ви діяли як науковець із даних і написали код. У мене є набір даних часових рядів [описати набір даних]. Створіть модель машинного навчання, яка передбачає [цільову змінну]. Будь ласка, використовуйте [часовий діапазон] як потяг і [часовий діапазон] як підтвердження, що для візуалізації подано на рис. 1.18.



The screenshot shows a chat window with a user message and a response from a data scientist. The user message asks for a machine learning model to predict sales revenue for 12 months of 2023, using 2019-2021 data for training and 2022 for validation. The response includes a Python code snippet using scikit-learn and pandas to load data, split it into training and validation sets, train a RandomForestRegressor, and print the mean squared error.

```

I want you to act as a data scientist and code for me. I have a time series dataset of sales revenue from 2019 to 2022. Please build a machine learning model that predicts sales revenue for 12 months of 2023. Please use 2019 to 2021 data as train and 2022 as validation.

I'd be happy to help you build a machine learning model to predict sales revenue for 12 months of 2023! Here's some Python code that demonstrates one way to train a model using the scikit-learn library:

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Load the data into a pandas DataFrame
data = pd.read_csv("sales_data.csv", parse_dates=['date'])

# Split the data into train and validation
train_data = data[(data['date'] < '2022-01-01')]
val_data = data[(data['date'] >= '2022-01-01')]

# Split the data into features and labels
X_train = train_data.drop(['date', 'revenue'], axis=1)
y_train = train_data['revenue']

X_val = val_data.drop(['date', 'revenue'], axis=1)
y_val = val_data['revenue']

# Train the model
model = RandomForestRegressor()
model.fit(X_train, y_train)

# Make predictions on the validation set
y_pred = model.predict(X_val)

# Print the mean squared error
mse = mean_squared_error(y_val, y_pred)
print("Mean squared error: {:.2f}".format(mse))

```

Рис. 1.18. Код програми

Цей код передбачає, що ваші дані містяться у файлі CSV під назвою “sales_data.csv” зі стовпцями дати та доходу. Код застосовує «Random Forest Regressor» для навчання моделі. Код також використовує середню квадратичну помилку як метрику для оцінки моделі, але ви можете брати для цього й інші показники оцінки. Код припускає, що у вас є стовпець дати у наборі даних, і він розділив набір даних на дві частини: дані поїзда (2019–2021) і дані перевірки (2022).

Запитання для перевірки знань



1. Які тренди сприяють розвитку штучного інтелекту?
2. Визначте та обґрунтуйте підтримку, яку штучний інтелект може надати тим, хто приймає рішення.
3. Проаналізуйте переваги автоматичного прийняття рішень.
4. Чому штучний інтелект вважається «найважливішою технологією, коли-небудь створеною людьми»?
5. Чому вартість робочої сили зростає, а вартість штучного інтелекту падає?
6. Якщо колись штучний мозок міститиме стільки ж нейронів, скільки мозок людини, чи буде він таким же розумним, як мозок людини?
7. Проаналізуйте, чому програми штучного інтелекту популярні та мають багато застосувань.
8. Що таке мовні моделі?
9. Назвіть основні правила побудови prompt-запиту.

1.5. Машинне навчання



Багато років тому людство під час промислової революції почало автоматизувати фізичні дії за допомогою машин, що викликало величезний бум продуктивності і сприяло появі другої хвилі автоматизації — цифровій революції, де замість фізичних дій застосовуються розумові дії. Якщо є калькулятор і ми знаємо, як виконувати ділення, можна запрограмувати операцію ділення і змусити калькулятор виконувати завдання, тобто автоматизувати операцію, проте все одно це ми придумали, що хочемо зробити і як це зробити, а потім реалізували на машині. Наступним кроком розвитку буде з'ясування проблеми, щоб далі машина сама її вирішила. Для цього й потрібно машинне навчання.



Машинне навчання або ML — це наука про те, щоб змусити комп'ютери діяти, надаючи їм дані та дозволяючи самостійно вивчити кілька шляхів. Тут ми дійсно не програмуємо машину, а передаємо їй багато даних, щоб вона могла вчитися самостійно.

Машинне навчання — це підмножина систем штучного інтелекту, яка дає машинам можливість автоматично навчатися й удосконалюватися на основі досвіду без явного програмування.

Машинне навчання — це набір математичних алгоритмів для пошуку патернів у даних.

Машинне навчання — це певна математична функція, яка з деякою мірою якості описує ваші дані.

Машинне навчання можна застосувати до надзвичайно різноманітних проблем — від забезпечення максимальної продуктивності гоночних автомобілів до виявлення шахрайства. Наприклад, в одній великій страховій компанії високий рівень автомобільних аварій за полісами нових клієнтів свідчив про те, що претензії подаються за існуючі збитки. Модель машинного навчання змогла зіставити різноманітні показники, щоб встановити групи нових полісів із рівнем нещасних випадків у шість разів вищим за середній. Це групування покладено в основу нової стратегії ціноутворення, яка підвищила прибутковість майже на 10%. В іншому прикладі велика платіжна система застосувала методи машинного навчання на своїх великих даних про транзакції, щоб ідентифікувати рахунки, залучені до відмивання грошей.

1.5.1. Класифікація машинного навчання



Можна сказати, що ML — це галузь штучного інтелекту, яка допомагає робити прогнози і приймати рішення за допомогою процесів самонавчання. Це набір методів, які допомагають ідентифікувати та зрозуміти закономірності в даних і приймати розумні рішення та прогнози. Велика кількість даних, отриманих через ці численні



джерела, вимагає методів і стратегій для належної обробки. Алгоритми ML використовують шаблони в даних для розробки прогнозованих моделей і використання їх для прогнозування. Для ефективної обробки великих даних машинне навчання є відповідним варіантом.

Загальна класифікація машинного навчання поділяється на дві категорії:

- **Контрольоване навчання:** вони використовуються, коли результати класифікуються. Під час навчання під наглядом користувачі знають про вхідні та вихідні дані. Контрольоване навчання використовує навчальні набори даних, щоб навчатися з них, і створює моделі, використовуючи їх. Потім ці моделі застосовують у прогнозах і для прийняття рішень. Математичні методи Баєса (наївний метод) і дерева рішень слугують прикладами навчання під наглядом. Сферою їх застосування є класифікація зображень, прогнозування ринку тощо.
- **Неконтрольоване навчання:** у цьому випадку користувачі знають лише про вхідні дані, але не мають інформації про вихідні. Неконтрольовані алгоритми використовуються для створення функції пошуку прихованих або немаркованих шаблонів даних. Їх основне завдання полягає в створенні міток класів.

Існує багато алгоритмів машинного навчання, які можна використовувати для розробки режимів прогнозування. Залежно від категорії типу алгоритми машинного навчання можна розділити на такі типи:

- класифікація;
- регресія;
- кластеризація;
- зменшення розмірності;
- глибинне навчання.

Отже, можемо виділити три різні типи машинного навчання, які подано на рис. 1.19:

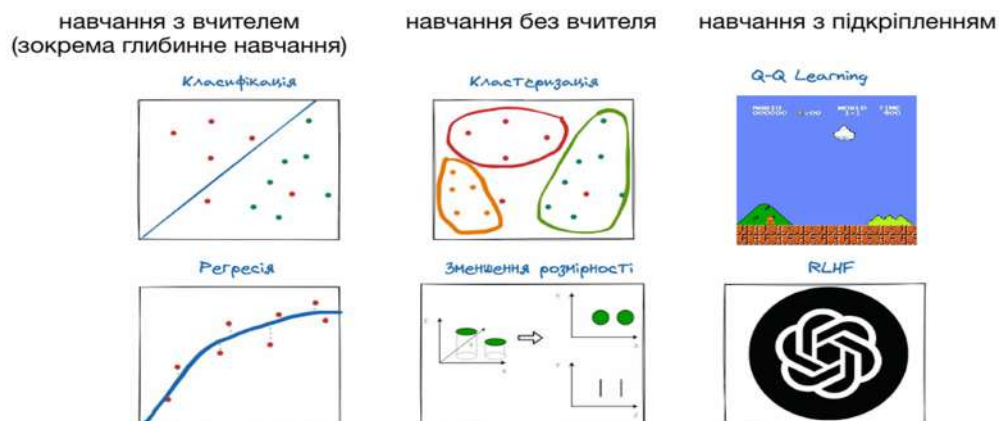


Рис. 1.19. Типи машинного навчання

- 1) навчання з учителем (контрольоване навчання);
- 2) навчання без учителя;
- 3) навчання з підкріпленням.

Існує термін «глибинне навчання», який належить до вирішення деяких завдань машинного навчання із застосуванням специфічної методики. Основна ідея полягає в тому, що ми будуємо наш алгоритм машинного навчання із послідовності цифрових шарів. Якщо вкласти їх вертикально один на одний, можна сказати, що отримаємо глибину, або алгоритм стане глибоким.



Навчання з учителем, або контрольоване навчання. Цей тип машинного навчання означає спрямувати певну діяльність і переконатися, що вона виконується правильно. Це один із найбільш популярних методів машинного навчання, переважно на класифікаційних методах і регресійних моделях.

Коли вибірки даних надходять із попередньо визначеними маркуваннями, ми говоримо про навчання з учителем. Учителем є наші маркування. Прогноз порівнюється з реальним маркуванням. Якщо передбачене маркування узгоджується з тим, якого ми хочемо, то нічого не робимо. В іншому разі ми говоримо, що алгоритм модифікувався.

Існує *два основні типи навчання з учителем* — класифікація та регресія. Класифікація належить до спостереження за цим набором категорій, щоб відшукати одну, яка найкраще описує специфічний вхід. Регресія має справу з безліччю вимірів і передбачення якоїсь іншої величини (часто тієї, яка надходить наступною, але може також і передувати початку множини або бути десь у середині його).

Основні математичні методи, які використовуються у навчанні з учителем:

1. Класифікація використовується для передбачення типу об'єкта в межах обмеженого набору параметрів, а вихідна змінна є категоричною. Одними з важливих моделей для класифікації є метод опорних векторів (SVM), дерева рішень, логістична регресія, наївний метод Байєса тощо.
2. Регресія є базовою технікою, яка набуває безперервні значення для виконання прогнозів. Деякі з основних методів регресійної моделі — це лінійна регресія, регресія Лассо (Least Absolute Shrinkage and Selection Operator), регресія хребта, регресія опорних векторів SVM і регресія дерева рішень.



Навчання без учителя. Цей тип навчання діє без будь-якої вказівки чи контролю. У цьому методі моделі в основному надається набір даних, який не позначений і не класифікований, тому машина не знає, як має виглядати результат. Наприклад, якщо ми вводимо зображення фрукта, ми не збираємося позначати його як «яблуко». Натомість ми лише даємо дані, а машина навчатиметься сама. Це лиш означає, що ми не контролюємо процес навчання за допомогою маркувань. Система має самостійно прове-

сти обчислення без нашої допомоги. Навчання без учителя часто використовують у вирішенні питань, які називаються кластеризація, придушення шумів і зниження розмірності.

1.5.2. Кластерний аналіз



Кластеризація — це модель, яка використовується для групування подібних об'єктів. Це належить до неконтрольованого навчання та ідентифікує об'єкт автоматично без необхідності втручання користувача. Деякими прикладами кластеризації є статистичний аналіз, аналіз соціальних мереж, аналіз ринку тощо. Основні методи кластеризації — це методи k-середніх, k-середніх ++, k-medoids, кластеризація на основі щільності, ієрархічна кластеризація і та ін.

Зменшення розмірності — це техніка, за допомогою якої дані з простору високої розмірності переносяться до простору низької розмірності. Ця методика передбачає зниження кількості функцій для покращення процесу прогнозування. Основні типи моделей зменшення розмірності — PCA (метод головних компонент, статистика), TSNE (Т-розподілене вкладення стохастичної близькості), SVD (сингулярний розклад матриці) та ін.



Кластеризація. Припустимо, що під час закладення фундаменту будинку робітники з'ясовують, що ґрунт заповнений стародавніми глиняними горщиками і вазами. Викликають археолога, який встановлює, що було знайдено перемішану колекцію стародавніх посудин, з багатьох різних місць і різних історичних періодів. Археолог не може встановити маркування і символіку, і тому не впевнений у походженні кожної посудини. На якихось вазах маркування виглядає як варіації на ту саму тему, тоді як на інших — це зовсім різні символи.

Щоб вирішити цю проблему, знімають копії маркувань і намагаються розсортувати їх за групами. Для цього вирішено використовувати алгоритм машинного навчання, щоб автоматично раціональним способом згрупувати маркування. На рис. 1.20 показано скопійовані маркування та групи, які могли бути знайдені алгоритмом. Це називається проблемою кластеризацією і застосовується алгоритм кластеризації.

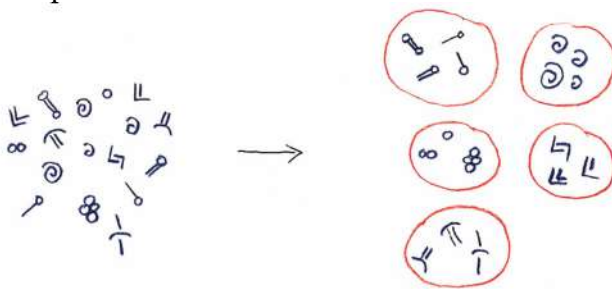


Рис. 1.20. Застосування алгоритму кластеризації маркування глиняних посудин. Копія маркування посудин і маркування, згруповані у кластери



Придушення шуму. Іноді вибірки перекручені шумом. Якщо вибірка є аудіозаписом когось, хто говорить по своєму телефону, шумом може бути фоновий гамір вулиці, що робить нерозбірливими його слова. Якщо вибірка є електронним сигналом, скажімо, телевізійною програмою, — шум може наводитися статичною перешкодою, що руйнує звук, або точками, що миготять у телевізійному зображенні.

У синтетичному, або створюваному комп'ютером, зображенні можна заощадити час, видаляючи деякі пікселі, тому вони стають лише чорними точками. Для комп'ютера це своєрідний шум, хоча насправді це, імовірно, зниклі дані, ніж спотворені дані. На рис. 1.21 показано приклад зашумленого зображення та наскільки алгоритм, що пригнічує шум, може його очистити.



Рис. 1.21. Пригнічення, або зниження, рівня шуму зменшує ефекти, пов'язані з випадковим спотворенням вибірки, і може дозволити заповнити зниклі пікселі у деяких місцях

Шляхом вивчення статистики вибірок алгоритм оцінює, яка частина кожної вибірки є шумом і видаляє її, залишаючи нам очищені дані, що полегшує навчання та інтерпретацію.



Зниження розмірності. Іноді наші вибірки мають більше ознак, ніж потрібно. Наприклад, збираються вибірки погоди в місті в розпал літа. Щодня записують швидкість вітру, його напрямок і наявність дощу. Для цієї пори року та місцезнаходження ознака наявності чи відсутності дощу буде у кожній вибірці. Якщо ж застосувати ці вибірки в системі машинного навчання, комп'ютеру необхідно буде обробляти та інтерпретувати цю непотрібну, постійну частину інформації у кожній вибірці.

У кращому разі ми уповільнимо аналіз; у гіршому — це може вплинути на точність системи, тому що комп'ютер витратить деякі кінцеві тимчасові ресурси та ресурси пам'яті, намагаючись видобути інформацію від цих ознак, що не змінюються. Використання навчання без учителя для пошуку способів зменшення в описі даних кількості величин (розмірності) називається зниженням розмірності. Це означає зменшення кількості ознак (зване розмірністю) у кожній вибірці даних.



Навчання з підкріпленням (Reinforcement Learning / RL) — це тип машинного навчання, де агент вчиться діяти у середовищі, виконуючи дії та спостерігаючи за результатами.

Це один із типів машинного навчання, який набирає популярності за останні кілька років. Використовується в різних додатках, таких як самокеровані автомобілі, методи обробки зображень, у яких ми виявляємо об'єкти за допомогою навчання з підкріпленням.

Припустімо, неодружений студент-старшокурсник несподівано був змушений залишитися із трирічною дочкою свого друга у квартирі, яку вони знімали. Він гадки не мав, що дівчинка любить їсти. Першого вечора, який вони проводили разом, він хлопець приготував макарони із вершковим маслом. Дівчинці сподобалось! Але така їжа щодня протягом тижня набридла обом. Тому студент до макаронів додав твердий сир, і дитині це також було до вподоби. Коли докурила і така страва з сиром, хлопець одного разу додав до макарон соус. Втім дівча це їсти відмовилася. Тоді після макарон студент запропонував смажену картоплю зі сметаною, і ця страва сподобалася дитині. Тепер новоз'явлений кухар спробував розробити меню зі страв, що смакували дівчинці. Єдиним зворотним зв'язком для складання переліку страв стала інформація про те, чи їсть дівчинка цю їжу, чи відмовляється. Цей метод навчання називається навчання із підкріпленням.

Змінімо цю історію, використавши більш наукову термінологію та назвавши студента-кухаря агентом, а дитину — довкіллям. Агент — це те, що приймає рішення та здійснює дії (кухар), а оточення — це все інше у навколишньому середовищі (дитина). Навколишнє середовище налагоджує зворотний зв'язок чи дає винагороду після кожної дії, повідомляє, наскільки хорошими ці дії виявилися. Поряд із оцінкою дій зворотний зв'язок повідомляє агенту про зміни середовища, і якщо це сталося, то як тепер виглядає довкілля (рис. 1.22). Цінність такої абстракції полягає в тому, що ми можемо вигадати багато варіантів комбінацій агента і середовища.



Рис. 1.22. Процес дій і винагороди у навчанні з підкріпленням

Навчання із підкріпленням відрізняється від навчання з учителем, тому що дані не мають ознак. Загальний план навчання від помилок такий самий, втім механізм інший. Відмінність полягає у тому, що система навчання з учителем видає результат (зазвичай категорію чи передбачену величину), і потім ми порів-

нюємо його з правильним результатом, який знаємо. У процесі навчання із підкріпленням правильний результат відсутній. Дані не мають маркування. Існує лише зворотний зв'язок, який свідчить про те, як добре ми діємо. Цим зв'язком може бути лише один біт інформації, що повідомляє про те, що наші дії «хороші» або «погані».

Отже, RL — це тип машинного навчання, де машини вчаться на основі оцінювання, тоді як інші алгоритми навчання (навчання під керівництвом) навчаються за інструкціями. Якщо винагорода позитивна, це підсилює таку конкретну дію, а якщо винагорода негативна, це заважає агенту виконати ту саму дію знову. Цей процес триває протягом багатьох ітерацій, поки агент не навчиться взаємодіяти з середовищем у належний спосіб.

Як правило, навчання з підкріпленням складається з двох основних компонентів, які включають:

- агента;
- навколишнє середовище.

Тут агент визначається як репрезентація для алгоритму Reinforcement Learning, а середовище — параметром, на який діє агент.

Процес підкріплення навчання починається, коли середовище надсилає стан агенту, а потім агент виконує деякі дії на основі спостережень (рис. 1.22).

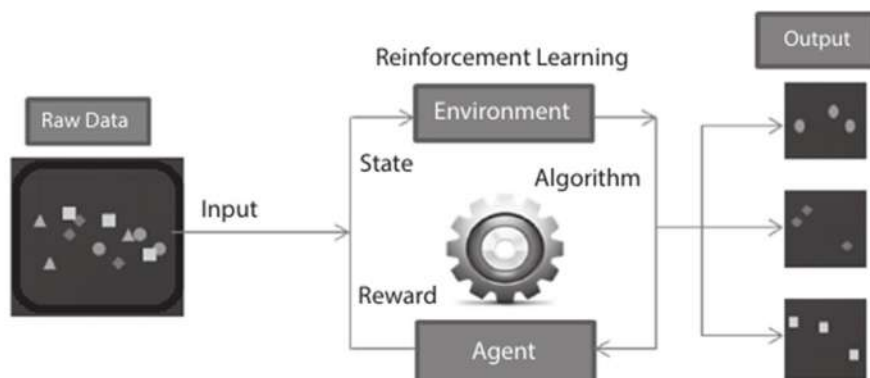


Рис. 1.23. Процес навчання з підкріпленням

За змістом навчання з підкріпленням поділяють на дві категорії:

1. *Позитивний RL* можна визначити лише тоді, коли подія відбувається внаслідок певної поведінки, яка може збільшити сили і частоти поведінки. Цей тип RL може допомогти максимізувати продуктивність і вплинути на збереження змін протягом більш тривалих періодів часу. Занадто багато позитивних RL може призвести до надлишку станів, що може знизити результати шляхом постійного підвищення позитивності винагород.
2. *Негативний RL*. На відміну від цього, негативне підкріплення також робить поведінку більш перспективною. Негативний RL використовується для дотримання мінімального стандарту продуктивності, а не для досягнення максимальної продуктивності моделі. Цей тип RL може гаранту-

вати, що модель утримується від несприятливих дій, проте він не може категорично змусити модель досліджувати очікувані дії. Цей негативний RL може призвести до збільшення поведінки та забезпечує зухвалість до мінімального стандарту продуктивності.

Середовище ж надішле агенту подальший стан і відповідну винагороду. Агент оновить свої знання винагородою, яку повертає середовище, і використовує це для оцінки своїх попередніх дій. Цей цикл продовжується, поки середовище не надішле термінальний стан, який означає, що агент виконав усі завдання та на решті отримав винагороду.

У навчанні з підкріпленням використовуються різні математичні методи та алгоритми, які включають:

- марковські процеси прийняття рішень (математична система для моделювання ухвалення рішень у ситуаціях, в яких наслідки є частково випадковими, а частково контрольованими ухвалювачем рішення);
- Q-Learning (алгоритм безмодельного навчання з підкріпленням);
- проблема «багаторукогого бандита» (задача розподілу обмеженої множини ресурсів між конкуруючими альтернативами у такий спосіб, щоб максимізувати очікуваний виграш);
- вибірка Томпсона (вибір дії, який максимізує очікувану нагороду порівняно з довільно вибраним переконанням);
- SARSA (алгоритм пошуку стратегії марковського процесу вирішування).

Значний прогрес досліджень в галузі навчання з підкріпленням RL підтримав кілька нових комбінованих шляхів, включаючи появу моделей глибокого навчання. Навчання з підкріпленням RL достатньо актуальне з широким спектром застосувань в усіх сферах, здебільшого у фінансах, робототехніці, охороні здоров'я та промисловості. Дослідження навчання з підкріпленням RL стали рушійною силою до базових принципів AI штучного інтелекту.

1.5.2. Моделі глибокого навчання



Вираз «глибоке навчання» стосується вирішення деяких завдань машинного навчання із застосуванням специфічної методики. Основна ідея полягає в тому, що алгоритм машинного навчання вибудовується з послідовності цифрових шарів. Якщо укласти їх вертикально один на один, можна сказати, що буде отримано глибину, або алгоритм стане глибоким.

Існує багато варіантів побудови глибоких мереж і безліч різних типів шарів та варіантів вибору їх для побудови мереж. Простий приклад показано на рис. 1.23. У цьому прикладі входом є 4 числа, а на виході формуються 2 числа.

Мережа має два шари, кожен з яких показаний як ящик із жовтим відтінком. У середині кожного шару ми маємо невеликий елемент обчислення, який має

назву *штучний нейрон*, або *персептрон*. Базовою операцією штучного нейрона є отримання пакета чисел, об'єднання їх особливим чином та отримання на виході нового числа, яке потім передаватиметься наступному шару.

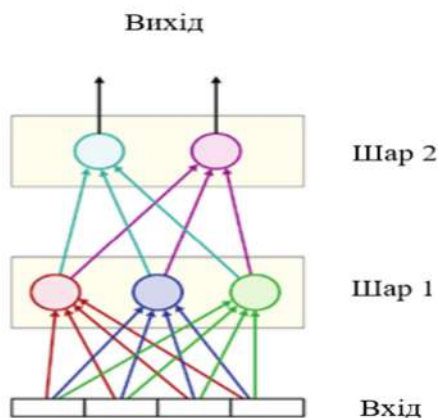


Рис. 1.23. Глибока мережа із двома шарами (у кожному шарі розташовуються штучні нейрони; виходи кожного шару з'єднані із входами наступного шару)

Привабливість архітектури мереж глибокого навчання полягає в тому, що є можливість створити повну мережу, скориставшись звичайною бібліотекою глибокого навчання, яка перебере на себе всю роботу з конструювання мережі. Бібліотека сформує всі внутрішні та зовнішні деталі. Завдяки такій структурі і простоті конструювання багато розробників порівнюють глибоке навчання з будівництвом чогось із блоків.

Штучні нейрони можуть представляти математичну логіку, а математична логіка є основою машинних обчислень, і це означає, що нейрони можуть стати математиками.

Робота одиночного *персептрона* — це мімікрія функціональної поведінки нейрона. *Персептрон* має деяке число входів, кожен є одним числом з плаваючою комою. Кожен вхід множиться на число, яке називається вагою. Результати цих множень підсумовуються, далі результат порівнюється із величиною порога. Якщо результат підсумовування більше від 0, то на виході маємо +1, інакше вихід дорівнює -1 (у деяких версіях замість +1 і -1 виходом є 0 та 1). Хоча персептрон виступає дуже спрощеною версією реального нейрона, він виявився чудовим будівельним блоком для систем машинного навчання.

1.5.3. Модельна проблема



У дитинстві ми не могли одразу відрізнити такі фрукти, як яблука, вишні, апельсини. Це тому, що недостатньо спостерігали за ними з огляду на те, як вони виглядають, але в міру дорослішання ми дізналися, як виглядають яблука, вишні та апельсини, тобто ми дізналися колір, форму та розмір фрукта та інші його параметри. Так само, коли справа доходить до машини, якщо ми вводимо зображення яблука чи будь-якого фрукта, спочатку вона не

зможе їх відрізнити. Це тому, що не має достатньо даних про них, але якщо ми продовжуємо задавати машині кілька зображень про фрукти (вишні, яблука та апельсини), вона навчиться розрізняти їх. Подібно до того, як люди навчаються, спостерігаючи та збираючи дані, машини також навчаються, коли ми надаємо багато інформації, тобто коли вводимо багато даних у машину, вона навчиться розрізняти їх. Машина в основному навчатиметься на даних і намагатиметься розрізнити різні правила за допомогою різних алгоритмів машинного навчання.

Термін «машинне навчання» охоплює велике коло технологій, які переслідують одну мету: витягти інформацію з даних, що має сенс. Розглянемо деякі приклади додатків, які використовують машинне навчання для отримання значущої інформації з даних. Наприклад, поштовому відділенню потрібно щодня сортувати велику кількість листів та посилок із написаними від руки поштовими індексами. Комп'ютер навчили зчитувати ці індекси та автоматично надсилати пошту на адресу, як це показано на рис. 1.24.

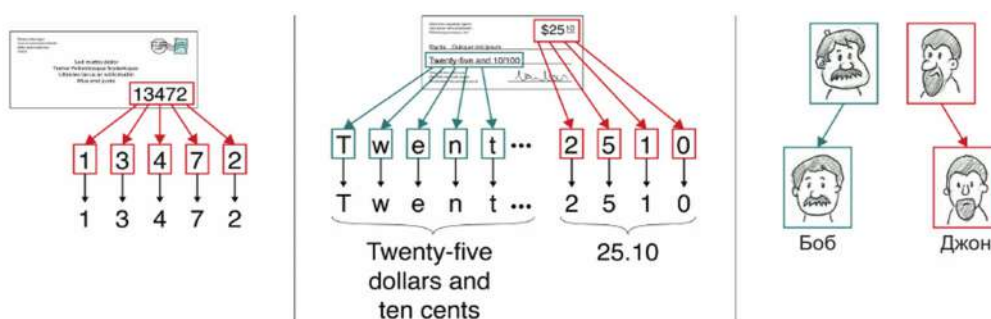


Рис. 1.24. Вилучення смислової інформації з багатьох даних:
 зліва: отримання поштового індексу з конверта;
 у середині: зчитування чисел і літер з чека;
 праворуч: розпізнавання обличь по фотографії

Банки обробляють величезну кількість рукописних чеків. Надійна перевірка вимагає, щоб числова сума, записана вручну у відповідне поле (наприклад, «25.10»), відповідала виписаній сумі в текстовому рядку, наприклад, «двадцять п'ять доларів і десять центів» («Twenty-five dollars and ten cents»). Комп'ютер може прочитати обидва числа і слова та підтвердити відповідність, як це показано на рис. 1.24. Сайти соціальних мереж хочуть ідентифікувати людей за фотографіями учасників мережі. Це означає не лише виявлення осіб на цій фотографії, а й визначення того, де розташовані ці особи, а потім зіставлення кожної особи з особами, що раніше спостерігалися. Це стає ще важчим при розумінні того, що фактично кожна фотографія людини унікальна: освітлення, кут зору, вираз обличчя, одяг та багато інших характеристик відрізнятиметься від будь-якого попереднього знімка.

Запитання для самоперевірки



1. Що таке машинне навчання?
2. Що таке навчання з вчителем?
3. Що таке навчання без вчителя?
4. Що таке навчання з підкріпленням?
5. Назвіть основні математичні методи, які використовуються в навчанні з вчителем?
6. Що таке кластеризація?
7. Що таке глибоке навчання і на якому принципі воно базується?

1.6. Інтернет речей



Вбудовування комп'ютерів та інших пристроїв, які можна вмикати та вимикати, в активні елементи в будь-якому місці та під'єднання всіх пристроїв до Інтернету (або один до одного) забезпечує широке спілкування і співпрацю між користувачами та елементами. Підключаючи багато пристроїв, які можуть спілкуватися один з одним, можна створювати програми з новими функціональними можливостями, підвищувати продуктивність існуючих.



Інтернет речей (IoT) — це термін, що розвивається, і має кілька визначень. Узагальнено, IoT стосується комп'ютеризованої мережі, яка з'єднує багато об'єктів (людей, тварин, пристроїв, датчиків, будівель, предметів), кожен із вбудованим мікропроцесором. Об'єкти підключені, переважно бездротовим способом, до Інтернету, утворюють IoT. Інтернет речей може обмінюватися даними і підтримувати зв'язок між об'єктами та їхнім середовищем.



Кевін Ештон — британський інженер, який зробив значний внесок в стандартизацію IoT, автор терміна «Інтернет речей» (1999 р.), дає таке визначення: «Інтернет речей означає датчики, підключені до Інтернету та ведуть себе подібно до Інтернету, створюючи відкриті спеціальні з'єднання, обмінюючись даними вільно та дозволяючи непередбачувані додатки, щоб комп'ютери могли розуміти навколишній світ і ставати нервовою системою людства».

Технічно можна визначити, що IoT — це мережа підключених комп'ютерних пристроїв, що містить різні типи об'єктів (наприклад, цифрові машини). Кожен об'єкт у мережі має унікальний ідентифікатор UID і здатний автоматично збирати та передавати інформацію по мережі. Зібрані дані не мають жодної цінності, доки їх не проаналізують. Важливо, що IoT дозволяє людям і речам взаємодіяти та спілкуватися в будь-який час, у будь-якому місці, щодо будь-якої бізнес-теми чи послуги.



За П. Міллером, IoT — це підключена мережа, в якій:

- можна з'єднати велику кількість об'єктів (речей);
- кожна річ має унікальне визначення (IP-адресу);
- кожна річ має можливість автоматично отримувати, надсилати та зберігати дані;
- кожна річ доставляється переважно через бездротовий Інтернет;
- кожна річ побудована на зв'язку між машинами (M2M).

1.6.1. Структури IoT

1. Екосистема IoT. Коли мільярди речей підключені до Інтернету з усіма допоміжними службами та під'єднаною IT-інфраструктурою, ми бачимо гігантський комплекс, який можна розглядати як величезну екосистему. Екосистема Інтернету речей відноситься до всіх компонентів, які дозволяють користувачам створювати програми IoT. Ці компоненти включають шлюзи, аналітику, алгоритми штучного інтелекту, сервери, сховище даних, безпеку та пристрої підключення.

Іншими словами, є мережа датчиків, що збирає інформацію, яка передається в центральне місце для обробки та аналізу даних. Звідси додатки IoT — підмножин екосистеми IoT.

2. Структура систем IoT. Речі в IoT стосуються різноманітних об'єктів і пристроїв, починаючи від автомобілів і побутової техніки до медичних пристроїв, комп'ютерів, фітнес-трекерів, апаратного забезпечення, програмного забезпечення, даних, датчиків і багато іншого. З'єднувати речі та дозволяти їм спілкуватися є необхідністю програми IoT. Проте для більш складних застосувань потрібні додаткові компоненти: система управління і бізнес-модель. IoT дозволяє речам бездротово з'єднуватися через мережу, наприклад, система контролю за температурою у кімнаті.

Приклад, не пов'язаний з Інтернетом, — світлофор на перехресті доріг, де датчики камери розпізнають автомобілі, що рухаються з різних боків, а система керування регулює час перемикання світла відповідно до запрограмованих правил.

3. Технологічна інфраструктура IoT. Технологію IoT можна розділити, як показано на рис. 1.25, на чотири основні блоки.

1. *Апаратне забезпечення:* включає фізичні пристрої, датчики та виконавчі механізми, в яких дані виробляються та записуються. Пристрої — це обладнання, яке потрібно контролювати або відстежувати. Сенсорні пристрої IoT можуть містити процесор або будь-який обчислювальний пристрій, який аналізує вхідні дані.

2. *Підключення:* має бути базова станція або концентратор, який збирає дані з об'єктів, завантажених датчиками, і надсилає ці дані в «хмару» для аналізу. Пристрої підключаються до мережі для зв'язку з іншими мережами чи програмами, які можуть бути підключені безпосередньо до Інтернету. Шлюз дозволяє пристроям, які безпосередньо не підключені до Інтернету, отримати доступ до хмарної платформи.

3. *Сервер програмного забезпечення:* на цьому рівні здійснюється керування зібраними даними. Програмне забезпечення керує підключеними мережами та пристроями та забезпечує інтеграцію даних. Це цілком може бути у «хмарі».

4. *Програми:* у цій частині IoT дані перетворюються на значущу інформацію. Багато програм можуть працювати на смартфонах, планшетах і ПК і робити щось корисне з даними. Інші програми можуть працювати на сервері та надавати результати чи сповіщення через інформаційні панелі або повідомлення для зацікавлених сторін.

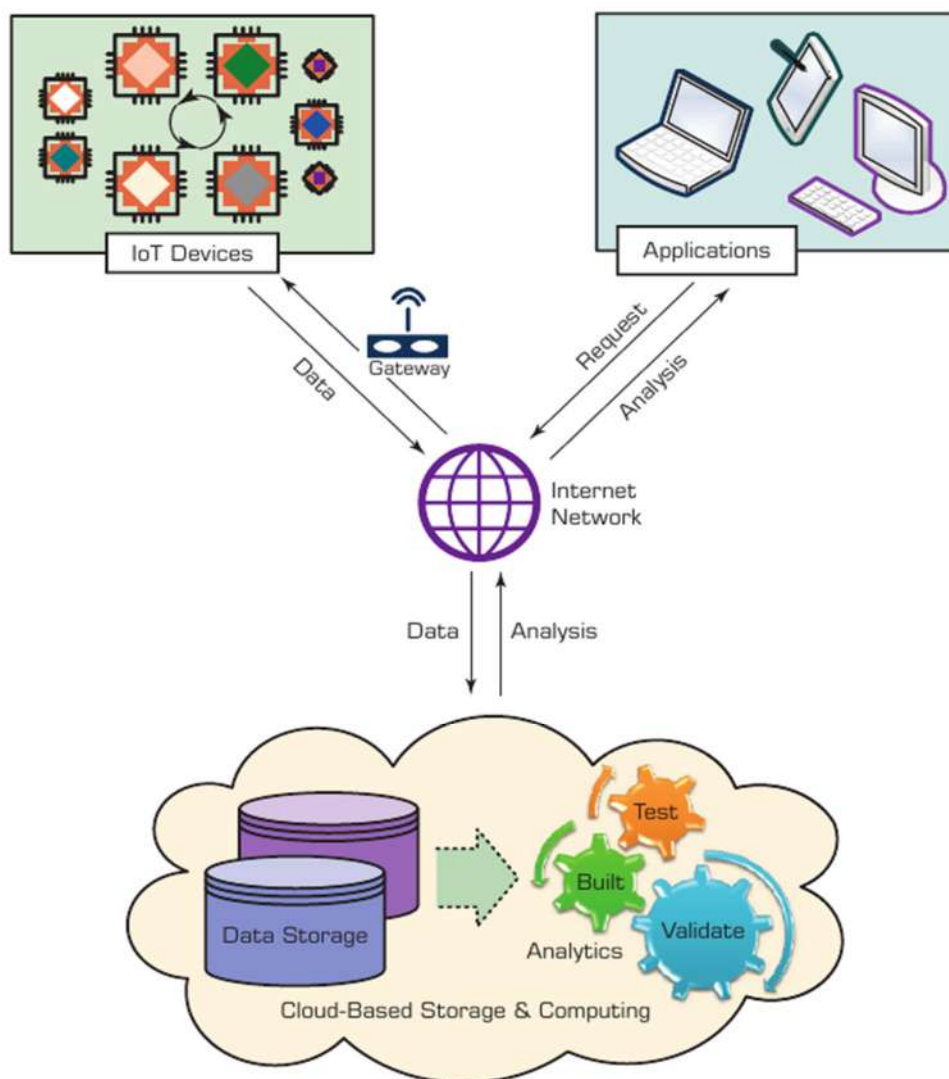


Рис. 1.25. Будівельні блоки IoT

1.6.2. Платформи та основні переваги IoT



Оскільки IoT досі розвивається, багато технологічних платформ, орієнтованих на домени і програми, також розвиваються. Саме тому багато основних постачальників платформ IoT є тими самими компаніями, які надають послуги аналітики та зберігання даних для інших доменів-додатків. Це Amazon AWS IoT, Microsoft Azure IoT Suite, Predix IoT Platform від General Electric (GE) і IBM Watson IoT platform. Уніфікована архітектура даних Teradata застосовувалася і багатьма клієнтами у домені IoT.

Основні переваги IoT:

- Зменшує витрати за рахунок автоматизації процесів.
- Підвищує продуктивність працівників.
- Створює нові джерела доходу.
- Оптимізує використання активів
- Покращує стійкість.
- Може передбачати наші потреби (прогнози).
- Дозволяє аналізувати широке середовище (датчики збирають дані).
- Сприяє розумнішим рішенням/покупкам.
- Забезпечує підвищену точність прогнозів.
- Швидко визначає проблеми (навіть до їх виникнення).
- Забезпечує миттєве формування та розповсюдження інформації.
- Пропонує швидке та недороге відстеження діяльності.
- Робить бізнес-процеси більш ефективними.
- Забезпечує зв'язок між споживачами та фінансовими установами.
- Сприяє стратегії зростання.
- Фундаментально покращує використання аналітики.
- Дозволяє краще приймати рішення на основі інформації в реальному часі.
- Прискорює вирішення проблеми та відновлення несправності.
- Підтримує інтеграцію засобів.
- Надає кращі знання про клієнтів для персоналізованих послуг і маркетингу.

Основні драйвери IoT:

Кількість «речей» — від 20 до 50 мільярдів — може бути підключена до Інтернету до 2025 р.

- Підключені автономні речі / системи (наприклад, роботи, автомобілі) створюють нові програми IoT.
- Широкосмуговий доступ до Інтернету стає все більш доступним і з часом зростає.
- Вартість пристроїв і датчиків постійно знижується.
- Зменшується вартість підключення пристроїв.
- Додаткові пристрої створюються (через інновації) і легко підключаються між собою.
- У пристрої вбудовано більше датчиків.
- Розповсюдження смартфонів стрімко зростає.
- Доступність пристроїв зростає.
- Швидкість переміщення даних зростає до 60 ТГц.
- Розробляються протоколи для IoT (наприклад, WiGig).
- Очікування клієнтів зростають; інноваційне обслуговування клієнтів стає необхідністю.
- Доступність інструментів і платформ IoT зростає.
- Доступність потужних систем аналітики, які використовуються з IoT, зростає.



IoT — це не додаток. Це інфраструктура, платформа або структура, яка використовується для підтримки програм. Нижче наведено комплексний процес для програм IoT. У багатьох випадках IoT слідує лише частинам цього процесу.

Інтернет-екосистема включає велику кількість речей. Датчики та інші пристрої збирають інформацію з екосистеми. Зібрану інформацію можна відображати, зберігати й обробляти аналітично (наприклад, за допомогою методів аналізу даних). Цей аналіз перетворює інформацію на знання та/або інтелект. Експертні системи або програми машинне навчання можуть допомогти перетворити знання на підтримку прийняття рішень створених людьми та/або машинами, про що свідчать покращені дії та результати як на рис. 1.26.



Згенеровані рішення можуть допомогти у створенні інноваційних додатків, нових бізнес-моделей і вдосконалення бізнес-процесів. Це веде до дій, які можуть вплинути на початковий сценарій чи інші речі. IoT може генерувати величезну кількість даних (Big Data), які потрібно аналізувати за допомогою різних методів бізнес-аналітики, включаючи глибоке навчання чи передові методи штучного інтелекту.

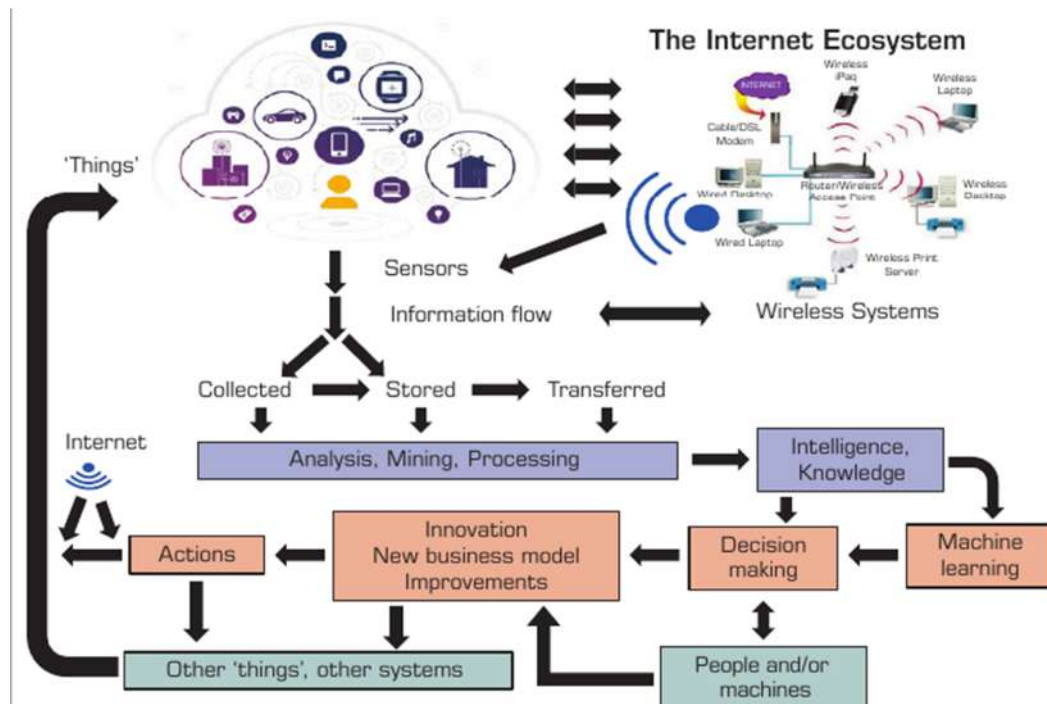


Рис. 1.26. Процес роботи IoT

1.6.3. Датчики та мітки



Датчик — це електронний пристрій, який автоматично збирає інформацію про події чи зміни в навколишньому середовищі. Багато програм IoT містять датчики. Зібрані дані надсилаються на інші електронні пристрої для обробки. Існує кілька типів датчиків і кілька методів збору інформації. Датчики часто генерують сигнали, які перетворюються на зрозумілі для людини дисплеї. На додаток до їхнього використання в IoT датчики являють собою важливі компоненти у робототехніці та автономних транспортних засобах.



Датчики радіочастотної ідентифікації (RFID, Radio-Frequency Identification). Радіочастотна ідентифікація (RFID) — це частина ширшої екосистеми технологій збору даних. RFID — це загальна технологія, яка стосується використання радіочастотних хвиль для ідентифікації об'єктів. За змістом RFID слугує одним із прикладів сімейства технологій автоматичної ідентифікації, які також включають усюдисущі штрих-коди і магнітні смужки. З середини 1970-х років роздрібна торгівля (серед багатьох інших сфер) використовує штрих-коди як основну форму автоматичної ідентифікації. RFID можуть зберігати набагато більший обсяг даних, ніж штрих-коди; крім того, до них можна отримати доступ з більшої відстані без проводів.



У найпростішій формі система RFID складається з мітки (прикріпленої до продукту, який потрібно ідентифікувати), запитувача (тобто зчитувача RFID), однієї або кількох антен, прикріплених до зчитувача, і комп'ютерної програми (для керування зчитувачем і збиранням даних).

Мітки бувають двох таких типів.

Пасивні мітки отримують енергію від електромагнітного поля, створеного запитувачем (наприклад, зчитувачем), і розсіюють інформацію лише тоді, коли її запитують. Пасивна мітка залишається під напругою лише тоді, коли розташовується у магнітному полі запитувача.

Активні мітки мають батарею для живлення. Оскільки активні мітки мають власне джерело живлення, їм не потрібен зчитувач для живлення; замість цього вони можуть ініціювати процес передачі даних самостійно. Порівняно з пасивними тегами, активні теги мають довший діапазон зчитування, кращу точність, складніше перезаписуване зберігання інформації та багатші можливості обробки. З іншого боку, їхні батареї ведуть до того, що активні мітки мають обмежений термін служби, більші за розміром за пасивні мітки і коштують дорожче.

Зараз більшість додатків у сфері роздрібно́ї торгівлі розроблено та працює з пасивними мітками.

Найпоширенішим представленням даних для технології RFID є електронний код продукту (EPC), який багатьма представниками галузі розглядається як наступне покоління універсального коду продукту (UPC), що найчастіше представлено саме штрих-кодом. Як і UPC, EPC складається з серії чисел, які ідентифікують типи продукції та виробників у всьому ланцюгу постачання. EPC також містить додатковий набір цифр для унікальної ідентифікації товарів.

Існує кілька типів розумних датчиків з різними рівнями можливостей у разі інтеграції в IoT.

1.6.4. Розумні (smart) датчики та IoT



Розумний (smart) датчик — це датчик, який сприймає навколишнє середовище й обробляє зібрані дані за допомогою вбудованих обчислювальних можливостей (наприклад, мікрообробки). Обробка попередньо запрограмована. Результати передаються.

Залежно від якості внутрішнього обчислення розумні датчики можуть бути більш автоматизованими і точними, ніж інші датчики, та можуть фільтрувати небажані шуми і компенсувати помилки перед надсиланням даних. Розумні датчики є ключовими та невід’ємними елементами IoT. Вони можуть містити спеціальні компоненти, такі як підсилювачі, аналогові фільтри та перетворювачі, для підтримки IoT.

Відомі приклади використання додатків IoT:

- *Hilton*. Гості можуть зареєструватися безпосередньо в своїх номерах за допомогою смартфонів (не потрібно проходити реєстрацію у холі, ключі не використовуються). Інші готельні мережі наслідують цей приклад.
- *Ford*. Користувачі можуть підключатися до програм за допомогою голосу. Розробляється автоматична оплата за бензин і попереднє замовлення на поїв у Starbucks безпосередньо з автомобілів Ford.
- *Tesla*. Програмне забезпечення Tesla автономно планує спеціального співробітника, щоб забрати автомобіль і відвезти його на завод Tesla, коли авто потребує ремонту чи запланованого обслуговування. Вантажівки Tesla, керовані Інтернетом речей, одного разу стануть безпілотними.
- *Apple*. Apple дає змогу користувачам iPhone, годинників Apple і домашніх комплектів спростити покупки за допомогою Apple Pay.
- *Starbucks Clover Net in the Cloud*. Ця система зв’язує кавоварів із уподобаннями клієнтів, відстежує продуктивність співробітників, моделі споживання, а також покращує рецепти тощо.

IoT впливає на аналітику даних чотирма способами:

1. *Підривний збір даних.* Інтернет речей збирає більше даних про клієнтів із більшої кількості джерел, ніж інші технології. Це включає дані з пристроїв, розумних будинків і всього, що роблять споживачі. Крім того, IoT надає дані про зміни в уподобаннях і поведінці споживачів.

2. *Персоналізація у реальному часі.* Наприклад, Інтернет речей може надавати точнішу інформацію про рішення конкретних клієнтів щодо купівлі. IoT може визначити очікування клієнтів і спрямувати клієнтів до певних брендів.

3. *Екологічна атрибуція.* IoT може контролювати середовище щодо доставки реклами для певних місць, клієнтів, методів і кампаній. IoT може полегшити дослідження бізнес-середовища; таких факторів, як конкуренція, ціноутворення, погодні умови та нові урядові постанови.

4. *Повний шлях спілкування.* Ініціативи IoT розширюють і збагачують цифровий канал спілкування між клієнтами та постачальниками, особливо тими, хто використовує бездротове цифрове спілкування. IoT складає уявлення про шляхи купівлі споживачів.



Концепція розумного дому (smart home) була в центрі уваги протягом кількох років, ще до того, як концепція IoT вийшла на передову сцену. Розумний дім — це дім з автоматизованими компонентами, які взаємопов'язані (часто бездротовим способом), такими як побутова техніка, засоби безпеки, освітлення та розваги, і централізовано керуються та можуть спілкуватися один з одним.

Розумні будинки покликані забезпечити своїм мешканцям комфорт, безпеку, низькі витрати на енергію та зручність, спілкуючись через смартфон або Інтернет. Контроль може відбуватися в режимі реального часу або з будь-якими бажаними інтервалами. Більшість існуючих будинків ще не розумні, але їх можна легко та недорого оснастити для принаймні часткової розумності. Кілька протоколів дозволяють з'єднання, зокрема, **XIO, UPB, Z-Wave та EnOcean**. Ці продукти пропонують масштабованість, тому з часом до розумного дому можна буде підключити більше пристроїв.

Нижче наведено *типові компоненти розумних будинків*:

- *Освітлення.* Користувачі можуть керувати домашнім освітленням, де б вони не були.
- *Телевізор.* Це найпопулярніший компонент.
- *Енергетичний менеджмент.* Домашні системи опалення та охолодження можна повністю автоматизувати та керувати ними за допомогою інтелектуального термостата;
- *Контроль води.* WaterCop (watercop.com) — це система, яка зменшує шкоду від води, відстежуючи витoki води за допомогою датчика. Система надсилає сигнал на клапан, який змушує його закритися.

- *Розумна колонка та чат-боти.* Найпопулярнішими є Echo і Alexa, а також Google Assistant.
- *Домашні розваги.* Аудіо- та відеообладнання можна запрограмувати реагувати на пристрій дистанційного керування. Усі пристрої домашньої автоматизації працюють з одного віддаленого місця та однієї кнопки.
- *Будильник.* Це сповіщає дітей повернутися спати або прокинутися.
- *Пилосос.* Прикладами є iRobot Roomba та LG Roboking vacuum.
- *Камера.* Це дозволяє мешканцям у будь-який час з будь-якого місця бачити, що відбувається в їхніх будинках.
- *Холодильник.* Прикладом цього є Instaview від LG, який працює на основі Alexa.
- *Кібербезпека і безпека будинку.* Такі системи можна запрограмувати на сповіщення власників про події, пов'язані з безпекою на їхній власності. Як зазначалося, певну безпеку можуть підтримувати камери для дистанційного перегляду власності в режимі реального часу.



Розумний прилад (Smart Appliance) містить функції, за допомогою яких можна дистанційно керувати роботою пристрою на основі вподобань користувача. Розумний пристрій може використовувати домашню мережу або Інтернет для зв'язку з іншими пристроями в розумному домі.



Ідея **розумних міст (Smart cities)** виникла приблизно в 2007 р., коли IBM запустила проєкт Smart Planet, а Cisco розпочала програму Smart Cities and Communities. Ідея полягає в тому, що в розумних містах цифрові технології (здебільшого мобільні) сприяють покращенню громадських послуг для громадян, кращому використанню ресурсів і меншому негативному впливу на навколишнє середовище.

Основна мета розумних міст — автоматизувати якомога більше громадських послуг, таких як транспорт, комунальні послуги, соціальні послуги, безпека, медичне обслуговування, освіта та економіка. Отже, у загальному проєкті розумного міста можна знайти кілька підпроєктів, деякі з яких є незалежними від головного проєкту. Транспорт є основною сферою, у якій аналітика та ШІ можуть зробити міста розумнішими. Інші сфери включають економічний розвиток, боротьбу зі злочинністю та охорону здоров'я.

Інші приклади компонентів розумного міста можна знайти в розумному університеті, розумних медичних центрах, розумній електромережі, а також в аеропортах, заводах, портах, спортивних аренах і розумних заводах. Кожен із цих компонентів можна розглядати як незалежний проєкт IoT та/або як частину загального проєкту розумного міста.

1.6.5. Модельні проблеми



Проблема 1. Аналітична модель SAS для розумних міст. Обсяг даних, зібраних мережами IoT у містах, може бути величезним. Дані збираються з багатьох датчиків, комп'ютерних файлів, людей, баз даних тощо. Щоб зрозуміти ці дані, необхідно використовувати аналітику, включаючи алгоритми штучного інтелекту. SAS використовує процес із семи кроків, який розділений на три основні фази: розуміти, розуміти та діяти. Нижче наведено їх визначення:

Сенс. За допомогою датчиків відчуйте все, що має значення. SAS аналізує зібрану інформації. Дані проходять через інтелектуальні фільтри для очищення, щоб лише релевантні дані переходили на наступний етап. IoT збирає та передає дані з датчиків.

Розуміти сигнали в даних. Використовуючи алгоритми інтелектуального аналізу даних, уся відповідна екосистема аналізується на предмет розпізнавання образів. Процес може бути складним, оскільки дані, зібрані датчиками IoT, поєднуються з даними з інших джерел.

Дія. Рішення можна приймати швидко, оскільки всі необхідні дані є на місці. Інструменти управління рішеннями SAS можуть підтримувати цей процес. Рішення варіюються від сповіщень до автоматизованих дій.



Розумні (цифрові) заводи (smart factory). Роботи виготовляють тисячі продуктів від автомобілів до мобільних телефонів. У центрах розподілу **Amazon** можна знайти десятки тисяч роботів. Тому не дивно, що заводи стають розумнішими завдяки технологіям штучного інтелекту та додаткам Інтернету речей. Як такі вони можуть розглядатися як компонент розумних міст і можуть бути взаємопов'язані з іншими компонентами, такими як чисте повітря та транспорт.

Основна ідея полягає в тому, що на фабриці, оснащеній технологіями штучного інтелекту, буде розташований командний центр. Штучний інтелект у поєднанні з датчиками IoT та потоком інформації забезпечить оптимальну організацію та послідовність бізнес-процесів. Увесь виробничий ланцюг, від постачальників сировини, логістики та виробництва до продажів, буде підключено до систем IoT для планування, координації та контролю. Планування базуватиметься на аналітичних прогнозах попиту. Виробничі процеси будуть максимально автоматизовані та керовані бездротовим способом. Логістика буде надаватися на вимогу швидко, а контроль якості буде автоматизований. IoT у поєднанні з датчиками буде використовуватися як для прогнозованого, так і для профілактичного обслуговування. Деякі з цих елементів існують на передових фабриках, і в майбутньому більше фабрик стануть розумнішими.



Проблема 2. Щороку аеропортом користуються понад 20 млн пасажирів, і їх кількість щороку збільшується більш ніж на 10 %. Очевидно, кількість рейсів велика і також підвищується щороку. Зростання відбувається, забруднення повітря також. Аеропорт має серйозну прихильність до захисту навколишнього середовища, тому керівництво шукало рішення для екологічного контролю. Мета полягала в тому, щоб зробити аеропорт нейтралізованим вуглецем. Велика кількість літаків у повітрі та на землі і той факт, що літаки часто рухаються, вимагають передових технологій для вирішення проблеми.

Рішення. Розумним способом боротьби з проблемами було використання Інтернету речей, технології, яка в поєднанні з датчиками на основі штучного інтелекту забезпечує моніторинг навколишнього середовища, аналіз і звітність, які забезпечують базову інформацію для прийняття рішень щодо мінімізації забруднення повітря. Дві компанії об'єднали свій досвід для цього проєкту: ЕХМ з Греції, яка спеціалізується на аналітиці прогнозування Інтернету речей та інноваційних рішеннях для Інтернету речей, і Libelium зі Сполучених Штатів, яка спеціалізується на датчиках, пов'язаних зі штучним інтелектом, зокрема датчиками для навколишнього середовища.

Мета проєкту полягала в тому, щоб належно контролювати якість повітря всередині і за межами аеропорту та визначати у режимі реального часу місцезнаходження літака на землі та вживати коригувальних дій, коли це потрібно. Спеціальний моніторинг та аналіз якості повітря «Аеропорт» тепер має мережу моніторингу якості повітря. Рішення включає сенсорну платформу Libelium, підключену економічно ефективним способом. Різні датчики вимірюють температуру, вологість, атмосферний тиск, рівень озону та тверді частинки. Покази датчиків передаються до IoT для звітування та аналізу. Датчики були вдосконалені за допомогою функцій ШІ, і тому їхня точність зросла. Крім того, контролюється безпека та енергоспоживання.

Розташування літаків в аеропорту. Для визначення точного місця розташування літаків під час зльоту і посадки у проєкті використовуються механізми акустичного вимірювання. Це досягається за допомогою датчиків шуму, розміщених у різних місцях. Датчики в режимі реального часу вимірюють рівень шуму, який оцінюється аналітикою. Загалом система забезпечує неінвазивне рішення IoT. Розміщення датчиків було складним через міркування безпеки та регулювання. Тому підсистема звукового моніторингу мала бути самокерованою (автономною) з сонячними панелями і батареями, що забезпечують електрику. Крім того, система використовує подвійну систему бездротового зв'язку (відому як GPPS).

Зібрані дані про шум співвідносяться з типами літаків і польотів на базі IoT. Уся інформація аналізується екологічним відділом аеропорту та використовується у прийнятті рішень про покращення контролю забруднення.

Технологічна підтримка. Рішення поєднує в собі систему IoT з аналітикою, візуалізацією та звітністю на основі ШІ та виконується в хмарі. Крім того, система має вбудовані датчики та комунікаційну інфраструктуру. Бездротові датчики малої потужності контролюють споживання води та газу в приміщенні, а також якість повітря на парковках. Продукти постачальників, такі як **Microsoft Azure** і **IBM Bluemix**, підтримують проєкт і забезпечують необхідну гнучкість.



Проблема 3. Для управління та координації такого складного бізнесу зі свого корпоративного офісу компанії потрібна була надійна система зв'язку, а також ефективні аналітичні можливості та мережа обслуговування клієнтів. Наприклад, критично важливою є наявність запасних частин. Обладнання клієнта не працює, поки не буде замінено зламану деталь. Конкурентний тиск дуже сильний, особливо в сільськогосподарському секторі, де погодні умови, сезонність робіт можуть ускладнити збирання врожаю. Належний моніторинг і керування обладнанням є важливим фактором конкуренції. Прогнозувати несправності дуже бажано. Швидке підключення до клієнтів і обладнання, яке вони купують у CNH, має важливе значення, а також ефективний моніторинг і збір даних. І CNH, і його клієнти повинні приймати безперервні рішення, для яких важливий потік інформації та зв'язок у реальному часі. CNH Industrial — це міжнародний виробник транспортних засобів для сільського господарства, будівництва та комерційних ринків. Компанія виробляє та обслуговує понад 300 типів транспортних засобів в 190 країнах, де працює понад 65 тис. співробітників. Бізнес компанії постійно розвивається, працюючи в дуже конкурентному середовищі.

Рішення. Компанія CNH створила систему на основі Інтернету речей із внутрішньою структурованою трансформацією, щоб вирішити проблеми та змінити підключені промислові транспортні засоби. Початкове впровадження пройшло в аграрному секторі. IoT дозволяє людям і речам бути взаємопов'язаними у будь-який час і у будь-якому місці. Вбудовані датчики, які збирають і обмінюються даними, складають основну частину об'єктів і IoT, використовуючи концепцію «усюдисущі обчислення».

Аналітики прогнозують, що до 2025 р. майже 50 млрд пристроїв (об'єктів) будуть підключені до Інтернету, створюючи магістраль додатків IoT. Проблеми та можливості цієї революційної технології (наприклад, для скорочення витрат, нових бізнес-моделей, покращення якості) обговорюються в інтерв'ю з П. Уцшнайдером, віцепрезидентом з управління продуктами для **Java** в **Oracle** (2014).

Запитання для самоперевірки знань



1. Що таке IoT?
2. Перелічіть основні характеристики IoT.
3. Чому IoT важливий?
4. Перелічіть деякі зміни, внесені IoT.
5. Що таке екосистема IoT?
6. Які основні сфери використання IoT?
7. Порівняйте IoT зі звичайним Інтернетом.
8. Визначте, що таке датчик IoT.
9. Опишіть роль датчиків в IoT.
10. Що таке RFID? Що таке датчик RFID?

ЗАВДАННЯ ДЛЯ МОЗКОВОГО ШТУРМУ

1. Знайдіть інформацію, що таке **Apple Home Pod**. Як він взаємодіє з пристроями розумного будинку?
2. Alexa тепер підключена до розумних домашніх пристроїв, таких як термостати та мікрохвильові печі. Знайдіть приклади інших пристроїв, підключених до **Alexa**, і напишіть звіт.
3. Обговоріть мету розумних міст для обговорення відновлення економіки України в після воєнний період.
4. Перейдіть на theinternetofthings.eu та знайдіть інформацію про **Council IoT**. Розкажіть про його призначення.
5. Перейдіть на сторінку <https://www.ptc.com/en/resource-center> або інші джерела та виберіть три приклади впровадження IoT. Напишіть короткий зміст кожного.

LAB 1.

ПОПЕРЕДНЯ ОБРОБКА ДАНИХ ЗА ДОПОМОГОЮ БІБЛІОТЕКИ PANDAS

Завдання. Необхідно перевірити та очистити дані з файлу Balloon Race Data Breaches — LATEST — breaches.csv за допомогою бібліотеки Pandas

Дані за адресою:

<https://www.kaggle.com/datasets/joebeachcapital/worlds-biggest-data-breaches-and-hacks>

Видаліть рядки, які містять помилки в даних, привести стовпці з числами до числового формату, визначити та видалити дублікати рядків, перевірити категорії, зберегти очищені дані до нового файлу CSV.

Збір даних. Набір даних містить інформацію про найбільші у світі витоки та хакерські зломи даних, в ньому вибрані події обсягом 30 000 записів. Стовпці містять назву організації, втрачені записи, рік, дату, докладну історію, сектор, метод злому і конфіденційність даних.

Формат даних: набір даних доступний у структурованому форматі, CSV.

Опис набору даних по стовпцях:

```
Data columns (total 16 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   organisation                           418 non-null    object
1   alternative name                       144 non-null    object
2   records lost                           418 non-null    object
3   year                                   418 non-null    object
4   date                                   417 non-null    object
5   story                                  411 non-null    object
6   sector                                 418 non-null    object
7   method                                 418 non-null    object
8   interesting story                       85 non-null     object
9   data sensitivity                       417 non-null    object
10  displayed records                       75 non-null     object
11  Unnamed: 11                             0 non-null      float64
12  source name                             417 non-null    object
13  1st source link                         416 non-null    object
14  2nd source link                         32 non-null     object
15  ID                                       417 non-null    float64
dtypes: float64(2), object(14)
memory usage: 52.4+ KB
```

Використати бібліотеку Pandas, та напишіть скрипт, для виконання таких операцій:

- Завантажте набір даних в DataFrame.
- Відсутні дані: перевірте, чи є в наборі відсутні дані.
- Категорії: якщо є категорії даних, визначте, які вони є.
- Опишіть історію (звіт), яку ви можете розповісти, використовуючи ці дані.

Запишіть коротке відео (3–4 хв.), що демонструє вашу роботу, остаточний результат, та скрипт на Python, і надішліть викладачу як звіт про виконання лабораторної роботи.

ПІДСУМКИ



- Розумні системи можуть впливати на організації різними способами — і як автономні системи, і як інтегровані між собою чи з іншими комп'ютерними інформаційними системами.
- Вплив смарт-аналітики на людей є різним — позитивним, нейтральним або негативним.
- Зростання розумних систем викличе серйозні зміни у структурі галузей економіки та майбутньому сфери зайнятості.
- Розумні технології можуть впливати на бізнес-процеси, організаційну структуру та методи управління.
- Розумні системи можуть сприяти реструктуризації багатьох робочих місць, створити безліч нових робочих місць, які потребуватимуть спеціальної підготовки.
- Використання смарт-аналітики та штучного інтелекту може привести до скорочення робочого тижня та необхідності компенсації тим людям, які втратять робочі місця.

ЧОТИРИ ЕРИ АНАЛІТИКИ



Аналітика 1.0. Почалася в 1950-х роках, коли окремі компанії збирали власні офлайн-дані та вручну аналізували їх, щоб шукати шляхи покращення своєї діяльності. Замість того щоб шукати корисну інформацію, люди витратили свій час на обробку технології та управління даними. Робота була повільною, значною мірою вручну та базувалася лише на обмежених джерелах даних.

Аналітика 2.0. Початок 2000-х став епохою перших ініціатив щодо BigData, коли комп'ютери та підключення до Інтернету почали широко поширюватися серед компаній.

Аналітика 3.0. Наприкінці 2000-х і на початку 2010-х років кількість можливих джерел даних значно розширилася завдяки зростанню кількості підключених пристроїв, широко розповсюдженим ініціативам зі збору даних і все більш інтелектуальним аналітичним технологіям.

Аналітика 4.0. Дозволила організаціям масово застосовувати прогнозовану аналітику та розширене моделювання рішень. Тепер вони можуть отримувати дані з сотень джерел у надскладні аналітичні програми та розгортати значною мірою автоматизовані інструменти прийняття рішень за допомогою хмарних і смарт-технологій.

РОЗДІЛ 2. СМАРТ-АНАЛІТИКА

2.1. Візуалізація даних



Інформативна візуалізація (яка також називається побудовою графіків) — одне з найважливіших завдань аналізу даних, може бути частиною процесу дослідження, наприклад, застосовувати для виявлення викидів, визначення необхідних перетворень даних або пошуку ідей для побудови моделей. Дані мають цінність, лише якщо відомо, як їх візуально відобразити та надати контекст, бо слід з'ясувати історію, яку розповідають дані. Діаграми, карти та інфографіка допомагають аудиторії зрозуміти складні числа, виявити закономірності, встановити тенденції та здобути інформацію.



Важливо дотримуватися найкращих практик у створенні діаграм. Візуалізація має містити цифри, які належно складаються, і мають відповідний масштаб. Уважне вивчення даних і постановка цілей допоможе вибрати тип візуалізації, який найкраще відповідає завданню.

Візуалізація даних — це метод, який дозволяє фахівцям з аналізу даних перетворювати сирі дані на діаграми і графіки, які несуть цінну інформацію. Діаграми зменшують складність даних і роблять більш зрозумілими для будь-якого користувача.

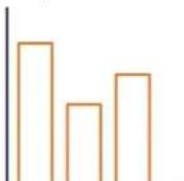
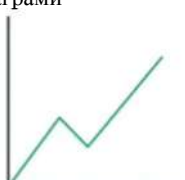
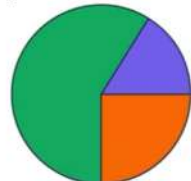
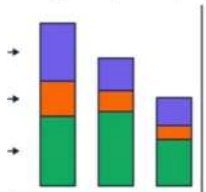
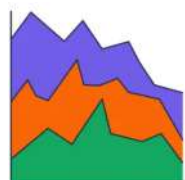
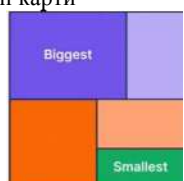
Загалом виділяють **чотири основні типи діаграм**:

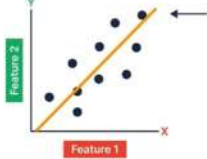
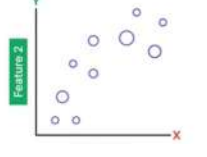
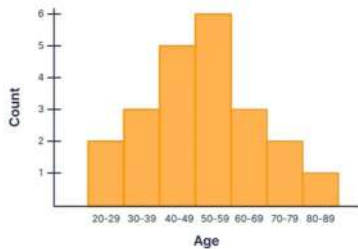
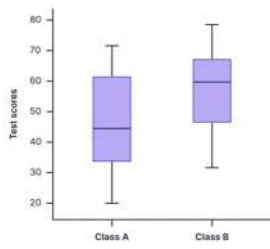
- порівняння (Comparison);
- композиція (Composition);
- зв'язки (Relationship);
- розподіл (Distribution).

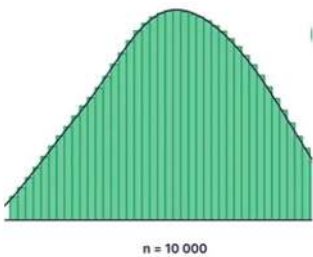
Опис видів діаграм і сфер їх застосування наведено у табл. 2.1.

Таблиця 2.1

ВИБІР ПРАВИЛЬНОЇ ВІЗУАЛІЗАЦІЇ ДЛЯ СВОЇХ ДАНИХ

Типи візуалізації	Вид діаграм	Призначення
Порівняльні візуалізації — дозволяють вивчити відмінності між різними сутностями	Стовпчасті діаграми 	- порівняння значень для різних категорій - відстеження змін у часі - коли зміни значні
	Лінійні діаграми 	- відстеження змін у часі - коли зміни невеликі - порівняти зміни з часом для кількох груп
Композиційні візуалізації — дають зрозуміти, як окремі компоненти поєднуються, щоб утворити цілу систему або набір даних.	Кругові діаграми 	Кругова діаграма підходить для відображення основної композиції, але вона обмежена кількістю частин, які ви можете відобразити. Не слід вибирати кругову діаграму, якщо вона буде поділена на більше ніж п'ять частин
	Стовпчасті діаграми з накопиченням 	Використовуються, коли корисно порівняти загальні показники по категоріях, а також показати, що кожна підкатегорія є частиною цілого
	Діаграми з накопиченням 	Є розширенням лінійної діаграми, може робити те саме, але одночасно показуючи склад кожної категорії в цілому
	Деревовидні карти 	Якщо ціле складається з п'яти чи більше частин, є хорошим вибором

Типи візуалізації	Вид діаграм	Призначення
Візуалізація зв'язків — розкриває зв'язки між змінними.	Діаграми розсіювання 	З'ясувати зв'язок між двома числовими змінними. Якщо існує лінійна залежність, до графіка можна додати лінію, яка найкраще її відображує.
	Бульбашкові діаграми 	З'ясувати зв'язок між трьома числовими ознаками. Бульбашкова діаграма будується так само, як діаграма розсіювання для 2 функцій, тоді як третя функція визначатиме розмір кожної нанесеної точки, створюючи бульбашки різних розмірів
	Теплові карти 	Теплова карта допомагає визначити, які дві функції пов'язані одна з одною в ефективний спосіб. Теплова карта показує кореляцію між двома функціями в наборі даних
Візуалізація розподілу — відображає частотний розподіл значень у наборі даних. Вони допомагають проаналізувати, як значення групуються, і виявити викиди	Гістограми 	Гістограми групують безперервні (числові) дані у контейнери фіксованої ширини і підраховують кількість спостережень у кожному контейнері. Контейнери відображаються як суміжні прямокутники, нанесені на вісь абсцис, з висотою, яка вказує на частоту. Вони ефективно показують загальну форму та кластери в розподілі набору даних
	Коробчасті діаграми 	Коробкові діаграми використовуються для показу розподілу значень числових даних, особливо коли ви хочете порівняти їх між кількома групами. За своєю суттю коробчаста діаграма відображає аспекти «підсумку з п'яти чисел» — набору значень, які описують центральну тенденцію і поширення набору даних

Типи візуалізації	Вид діаграм	Призначення
	Графіки щільності 	Діаграми щільності використовують оцінку щільності ядра (KDE), щоб згладити біни гістограм у криві, виявляючи більш тонкі закономірності. Вони дуже корисні при роботі з великими розмірами вибірки



Найкраща практика — вивчити власні дані та розглянути свої цілі, перш ніж робити вибір. Запитайте себе, на які запитання ви найбільше хочете дістати відповіді з даних. Потім виберіть тип візуалізації, який відповідає потрібним уявленням. Існує безліч інструментів для візуалізації даних, таких як **Tableau**, **Power BI**, **ChartBlocks** та ін., які є no-code інструментами. Вони дуже потужні, і кожен має свою аудиторію. Однак для роботи з сирими даними, що вимагають обробки, Python підійде найкраще. Для Python функціонує багато бібліотек статичної та динамічної візуалізації, але найпопулярнішою з них залишається **matplotlib** (<http://matplotlib.sourceforge.net>) і надбудовані над нею бібліотеки.

Matplotlib (скорочення від «Бібліотека математичних графіків») — це фундаментальний пакет для побудови графіків (здебільшого двовимірних) поліграфічної якості. Проект був заснований Джоном Хантером у 2002 р. з метою реалізувати на Python інтерфейс, аналогічний MATLAB. Згодом співтовариства matplotlib і IPython спільно працювали над тим, щоб спростити інтерактивну побудову графіків з оболонки IPython (а тепер і Jupyter-блокнотів). Matplotlib підтримує різноманітні графічні інтерфейси користувача у всіх операційних системах, а також вміє експортувати графічні дані у всіх векторних і растрових форматах: PDF, SVG, JPG, PNG, BMP, GIF тощо.

На базі matplotlib було створено багато додаткових бібліотек візуалізації.

Plotly — потужна бібліотека Python (<https://github.com/plotly/plotly.py>), яка дозволяє створювати інтерактивні та візуально привабливі діаграми та графіки для візуалізації даних за допомогою дуже малого коду.

Seaborn — це бібліотека для створення статистичних графіків на Python (<http://seaborn.pydata.org/>), що ґрунтується на matplotlib і тісно взаємодіє зі структурами даних pandas.

Як зазвичай буває у проєктах з відкритим вихідним кодом, засобів створення графіки для Python занадто багато, щоб перерахувати все. Починаючи з 2010 р. зусилля розробників також були зосереджені на створенні інтерактивної графіки для публікації в Інтернеті.

Завдяки таким інструментам, як Altair (<https://altair-viz.github.io/>), Bokeh (<http://bokeh.pydata.org/>) стало можливо описувати на Python динамічну інтерактивну графіку, орієнтовану на відображення в браузері.

Документи на основі даних (D3). D3 — це проєкт Стенфордської групи візуалізації, розроблений Майком Бостоком (<https://d3js.org/>). Бібліотека JavaScript D3 надає нам вебвізуалізацію, яка слугує чудовим способом розгортання інформації та допомагає нам бачити такі речі, як пропорції, зв'язки, кореляції та шаблони, а також відкривати раніше невідомі речі. Оскільки веббраузер надають нам дуже гнучкий та інтерактивний інтерфейс практично на будь-якому пристрої, наприклад, ПК, планшеті і смартфоні, D3 — гарний інструментом для візуалізації на основі даних за допомогою HTML, JavaScript, SVG і CSS.

Taipy — це платформа додатків на Python, проєкт із відкритим вихідним кодом (ліцензія Apache 2.0), який дозволяє створювати бізнес-додатки, такі як інформаційні панелі для відображення даних, конвеєри даних або програми AI (<https://www.taipy.io/>). Taipy складається з двох основних компонентів: Taipy GUI — дозволяє створювати інформаційні панелі та інтерактивні програми (GUI = графічний інтерфейс користувача) для кінцевих користувачів і Taipy Core- «сучасний бек-енд», який дозволяє створювати конвеєр, перетворювати дані та запускати моделі машинного навчання.



Бібліотека matplotlib — це засіб досить низького рівня. Графік збирається з базових компонентів: спосіб відображення даних (тип графіка: лінійний графік, стовпчаста діаграма, коробчаста діаграма, діаграма розсіювання, контурний графік і та ін.), напис пояснення, назва, мітки рисок та інші анотації).

Усі елементи, з яких складається діаграма, наприклад, заголовок, мітки осей, маркери тощо, являють собою екземпляри об'єкта Artist бібліотеки. Кожен із цих екземплярів виконує свою роль в ієрархічній структурі (як показано на рис. 2.1).

Як правило, працюючи на цьому рівні, доведеться часто мати справу з об'єктами ієрархії, такими як «Фігура», «Ахес» та «Axis». Тому важливо повністю зрозуміти, що це за об'єкти та яку роль вони відіграють у графічному представленні. На рис. 2.2 показано ці три основні складові (компонитні), які традиційно використовуються в усіх візуалізаціях, що виконуються на цьому рівні.

- Фігура — це об'єкт із найвищим рівнем в ієрархії, що відповідає всьому графічному представленню і, як правило, може містити багато осей.

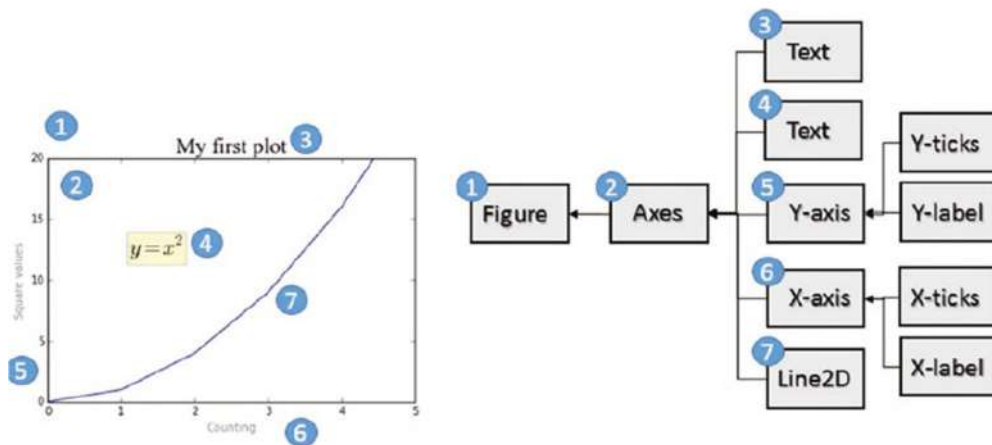


Рис. 2.1. Кожен елемент діаграми відповідає екземпляру об'єкту *Artist*, структурованого за ієрархією

- Axes — це, як правило, те, що ви маєте на увазі під графіком або діаграмою. Кожен об'єкт Axis належить лише одній фігурі та характеризується двома вісями Artist (трьома у тривимірному випадку). Інші об'єкти, як-от заголовок, мітка x і y , належать цьому композитному виконавцю.
- Об'єкти осей враховують числові значення, які мають бути представлені на осях, визначають межі та керують галочками (позначками на осях) і мітками позначок (текстом позначки, представленим на кожній галочці).

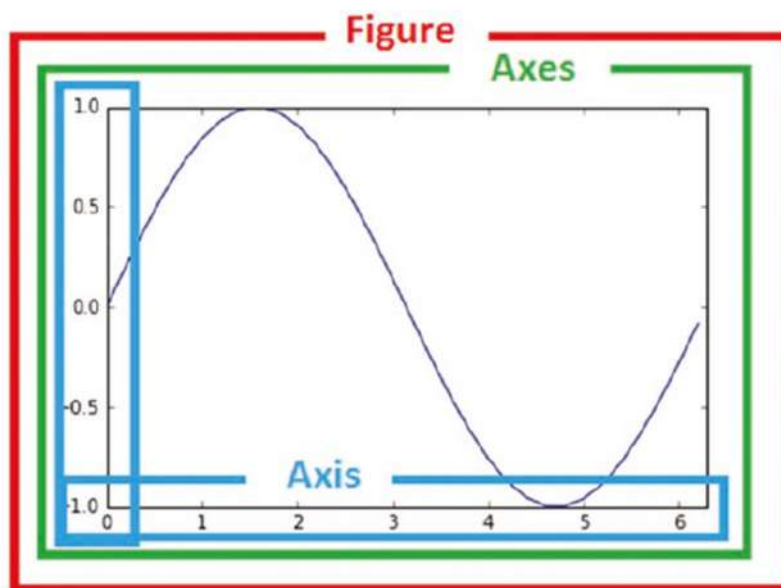


Рис. 2.2. Три основні об'єкти *Artist* в ієрархії

Положення галочки регулюється об'єктом під назвою `Locator`, тоді як мітка галочки форматування регулюється об'єктом під назвою `Formatter`.

2.1.1. Основи побудови

Перш за все нам потрібно дістати доступ до функцій побудови, встановивши бібліотеку: `import matplotlib.pyplot`.



Створення простих лінійних графіків. Як випливає з назви, точки даних з'єднані прямими лініями, які корисні для відображення даних, які безперервно змінюються в діапазоні, що полегшує визначення закономірностей і тенденцій. Використовуйте `plt.plot(x, y)` для простого лінійного графіка та `plt.show()` для показу сюжету.

Код:

```
import matplotlib.pyplot as plt
# Давайте візьмемо дані, що представляють місячний трафік веб-сайту (у тисячах)
months = [ 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun' ]
traffic = [ 150, 200, 180, 220, 250, 210 ]
# Щоб створити лінійний графік
plt.plot(months, traffic)
plt.show()
```

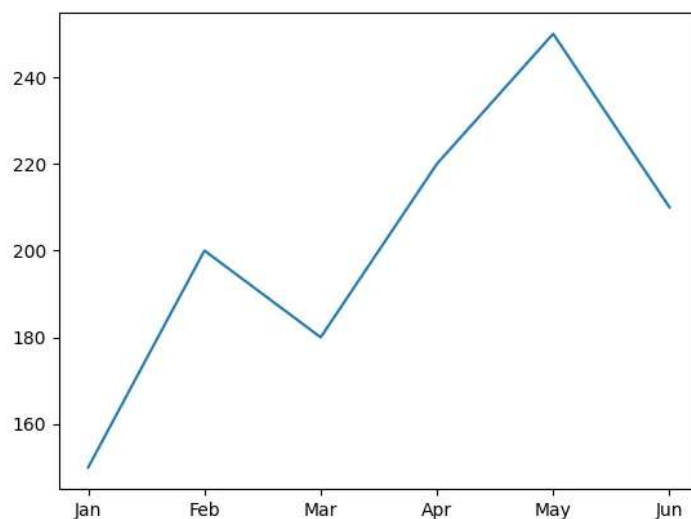


Рис. 2.3. Лінійний графік

Але, як ми бачимо, він не має жодних міток чи назв як таких.

Параметри налаштування міток, кольорів і стилів. Ми можемо передавати інформацію з великою ясністю, налаштовуючи сюжети. Matplotlib пропонує численні параметри налаштування, дозволяючи вам керувати кольором, стилем ліній, маркерами тощо.

Додайте мітки і заголовок: використовуйте `plt.xlabel()`, `plt.ylabel()` і `plt.title()`.

Код:

```
import matplotlib.pyplot as plt
# Зразок даних, що відображає місячний трафік веб-сайту (у тисячах)
months = [ 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun' ]
traffic = [ 150, 200, 180, 220, 250, 210 ]

# Створити лінійний графік
plt.plot(months, traffic)

# Додати мітки та назву
plt.xlabel( 'Місяць' )
plt.ylabel( 'Щомісячний трафік (в тисячах)' )
plt.title( 'Щомісячний трафік веб-сайту' )

plt.show()
```

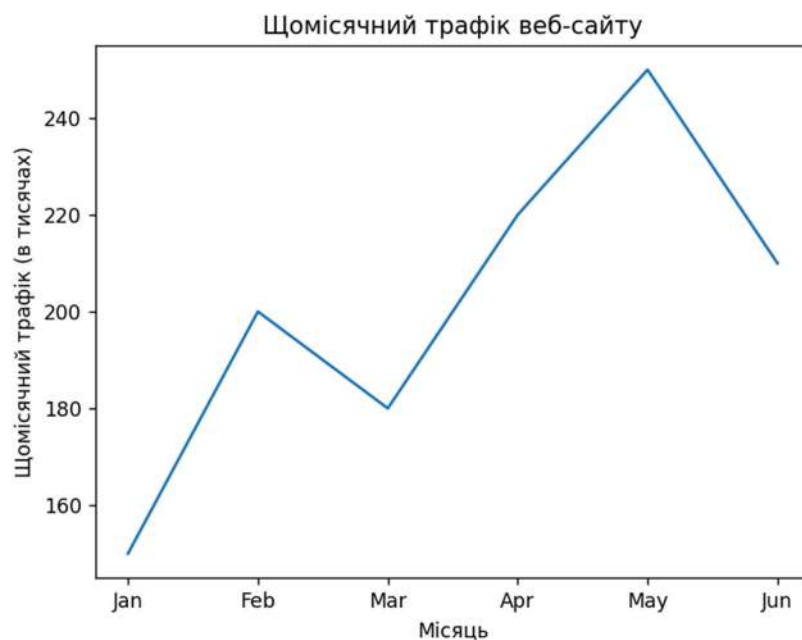


Рис. 2.4. Проста лінійна схема з мітками та заголовком

Примітка. Пам'ятайте, що `plt.show()` завжди має бути в кінці налаштувань сюжету. Якщо ви введете команди міток після `plt.show()` вони не відобразяться!

Зміна кольору і стилю лінії. Використовуйте параметри: `marker`, щоб позначити точки; `linestyle`, щоб змінити стиль лінії та додати сітку до графіків за допомогою `plt.grid (True)`.

Код:

```
# Створіть лінійну діаграму з нестандартним виглядом
plt.plot(months, traffic, marker= 'o', linestyle= '--', color=
'g' )

# Додайте мітки та назву
plt.xlabel( 'Місяць' )
plt.ylabel( 'Щомісячний трафік (у тисячах)' )
plt.title( 'Місячний трафік веб-сайту' )

# Додати сітку до графіка
plt.grid( True )
# Відобразити графік
plt.show()
```

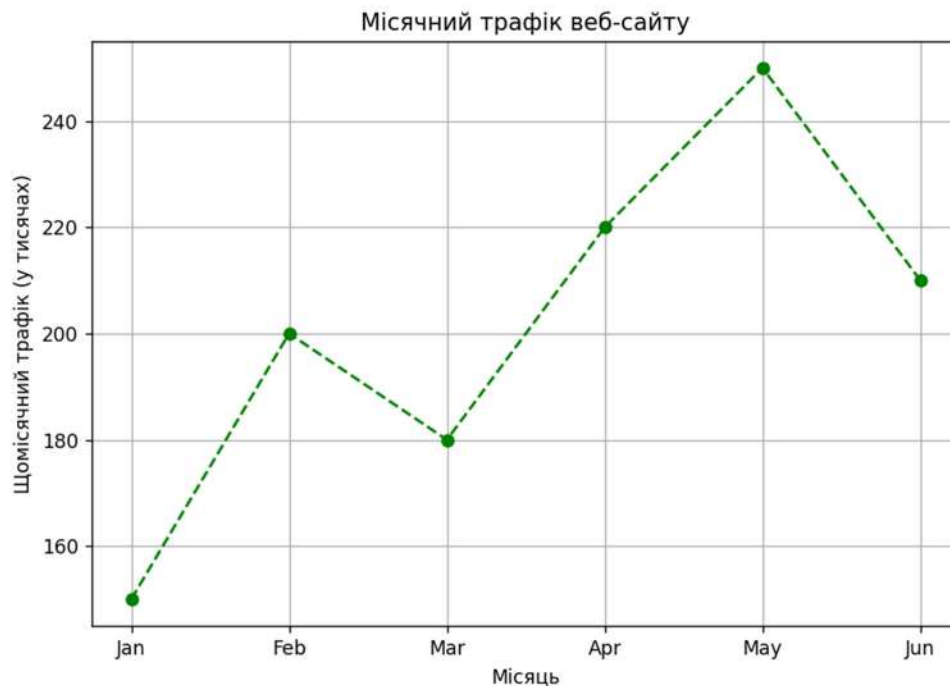


Рис. 2.5. *Переоформлений лінійний графік*

Зміна стилю графіка. У Matplotlib доступні різні стилі, щоб перевірити доступні стилі, скористайтеся командою `plt.style.available`. Використовуйте `plt.style.use('desired_style')` для перемінювання стилю усього графіка. Щоб використовувати графік у стилі коміксів, ви можете використовувати `plt.xkcd()`, це надасть такого вигляду, як показано нижче (рис. 2.6).

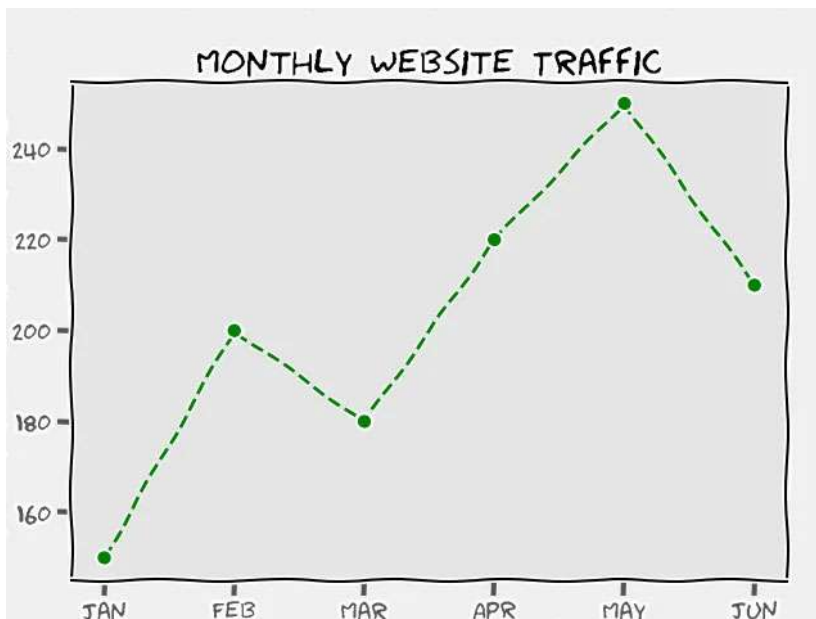


Рис. 2.6. Зміна стилю лінійного графіка

Це перетворить стилі малюнку для всього блокнота. Якщо ви бажаєте змінити стиль за замовчуванням, скористайтеся `plt.style.use('default')`.

Примітка. Обов'язково використовуйте ці команди стилю перед `plt.show()`

Щоб змінити розмір фігури. Для візуалізації часто доводиться коригувати розмір графіка, і для цього потрібно використовувати `plt.figure(figsize=(x_size, y_size))`.

Використання мітки. У випадку побудови кількох ліній на одному графіку, ви можете зробити це, використавши команду `plot` двічі для потрібних змінних. Але проблема полягає в тому, щоб правильно їх розрізнити, для цього у нас є параметр під назвою `label`, разом з яким вам також потрібно використовувати `plt.legend()`.

Код:

```
# Зразок даних для місячного доходу від двох продуктів (у тисячах
# доларів)
months = [ 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun' ]
product_a_revenue = [ 45, 55, 60, 70, 80, 75 ]
product_b_revenue = [ 35, 40, 50, 55, 70, 68 ]

# Створіть лінійний графік для продукту A із синьою лінією та
# круглими маркерами
plt.plot(months, product_a_revenue, marker= 'o', linestyle= '-',
         color= 'blue', label= 'Product A' )
```

```
# Створення лінійного графіка для продукту В з червоною пунктирною лінією та квадратними маркерами
plt.plot(months, product_b_revenue, marker= 's', linestyle= '--', color= 'red', label= 'Product B' )

# Додати мітки та назву
plt.xlabel('Month')
plt.ylabel('Monthly Revenue (in $1000)')
plt.title('Monthly Revenue Comparison')

# Показати легенду, щоб відрізнити продукт А від продукту В
plt.legend()

# Показати графік
plt.show()
```

Результат можна побачити на рис. 2.7.

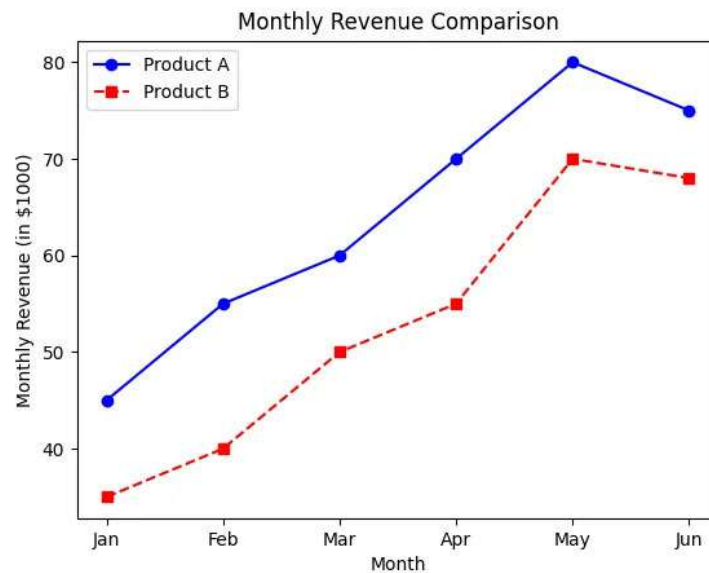


Рис. 2.7. Багатолінійний графік із відповідними мітками для кожної ділянки

Matplotlib дозволяє зберегти графік як файл зображення:

1. У Jupyter Notebook: під час роботи в Jupyter Notebook, якщо ви бажаєте зберегти графік як файл зображення, потрібно використовувати команду `plt.savefig('path/to/directory/plot_name.png')`. Ви можете вказати повний шлях до файлу, а також потрібне ім'я та формат файлу (наприклад, .jpg, .png, .pdf).

2. У Google Colab: під час роботи в Google Colab, якщо ви бажаєте зберегти графік як файл зображення, вам потрібно спочатку підключити диск і використати команду `plt.savefig()`.

Код:

```
from google.colab import drive

# Змонтувати Google Drive
drive.mount( '/content/drive' )

# Зберегти графік як файл зображення в Colab
# Можете змінити розташування на бажану папку.
plt.savefig( '/content/drive/My Drive/' + 'my_plot.png' )
```

2.1.2. Типи графіків

Ми побачили створення основного лінійного графіка, але Matplotlib пропонує набагато більше видів графіків, таких як стовпчасті діаграми, гістограми, точкові діаграми, кругові діаграми, коробчастий графік (діаграма в формі коробки та вусів), теплові карти, відображення зображень і та ін. Розглянемо, в яких випадках їх використовувати.

Стовпчаста діаграма. Гістограми представляють категоричні дані за допомогою прямокутних стовпчиків, де довжина чи висота кожного стовпчика представляє значення метрики. Використовується команда `plt.bar(x,y)` — для створення вертикальних і `plt.barh(x,y)` — горизонтальних гістограм.

Приклади використання

1. Порівняння показників продажів різних продуктів.
2. Відображення розподілу населення по країнах.

Код:

```
# категорій витрат
categories = ['Housing', 'Transportation', 'Food',
              'Entertainment', 'Utilities']

# Щомісячні витрати для Аліси, Боба та Керол
alice_expenses = [1200, 300, 400, 200, 150]
bob_expenses = [1100, 320, 380, 180, 140]
carol_expenses = [1300, 280, 420, 220, 160]

# Створити масив для позицій осі
x = np.arange(len(categories))

# Ширина смуги, нам це потрібно, оскільки ми маємо намір #побудувати кілька смуг.
bar_width = 0.2

# Створіть стовпчики для витрат Аліси, відніміть ширину #стовпця з масиву x,
```

```

# таким чином, щоб він був розміщений ліворуч.
plt.bar(x - bar_width, alice_expenses, width=bar_width,
label='Alice', color='skyblue')

# Створення стовпчиків для витрат Боба
plt.bar(x, bob_expenses, width=bar_width, label='Bob',
color='lightcoral')

# Створіть стовпчики для витрат Керол, додайте ширину #стовпчика
до масиву x,
# щоб він розташовувався праворуч.plt.bar(x + bar_width,
carol_expenses, width=bar_width, label='Carol', color='lightgreen')

# Додати мітки, назву та легенду
plt.xlabel('Expense Categories')
plt.ylabel('Monthly Expenses (USD)')
plt.title('Monthly Expenses Comparison')

# Для відображення назв категорій у положеннях осі X.
plt.xticks(x, categories)
plt.legend()

# Показати графік
plt.show()

```

Результат подано на рис. 2.8.

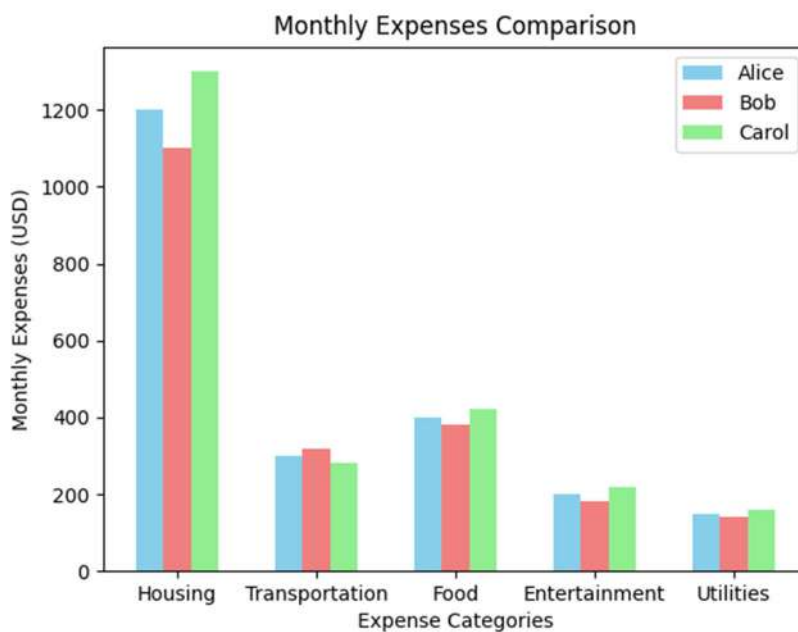


Рис. 2.8. *Стовпчаста діаграма*

Щоб отримати ці стовпчики, для першого стовпчика ми відняли мітки x із шириною стовпчика, а для останнього стовпчика ми додали мітку з шириною стовпчика. Параметр ширини встановлюємо рівним ширині смуги для всіх.

Гістограми. Гістограми використовуються для візуалізації розподілу безперервних або числових даних і допомагають ідентифікувати закономірності в даних. На графіку гістограми дані згруповані в «bins», а висота кожного стовпчика відповідає частоті або кількості точок даних у цьому бункері. Він бере нижню та верхню межі заданих даних і розділяє їх на кількість наданих бункерів.

Можна використовувати команду `plt.hist(x)` для створення гістограм. На відміну від стовпчастої діаграми, тут не потрібен `y`, оскільки представляє лише частоту одного безперервного даних. За замовчуванням бункерів є 10, їх можна змінювати. Ви також можете замінити діапазон бункерів бажаним діапазоном бункерів. Ви також можете додати `edgecolorfor bars`.

Разом із графіком гістограми на тому самому графіку, якщо треба додати будь-яку лінію, скажімо, середнє чи медіану, це можна зробити, обчислюючи значення і переходячи до команди `plt.axvline(calculated_mean, label=desired_label)`

Це можна використовувати з будь-яким іншим сюжетом.

2.1.3. Приклади використання

1. Аналіз вікового розподілу населення.
2. Перевірка розподілу екзаменаційних балів у аудиторії.

Код:

```
# Зразок результатів іспиту
exam_scores = [68, 72, 75, 80, 82, 84, 86, 90, 92, 95, 98, 100]

# Користувачський діапазон бінів
bin_ranges = [60, 70, 80, 90, 100]

# Створіть гістограму зі спеціальними діапазонами бінів, призначивши її бінам.
plt.hist(exam_scores, bins=bin_ranges, color='lightblue',
         edgecolor='black', alpha=0.7)

# Додати мітки та назву
plt.xlabel('Exam Scores')
plt.ylabel('Frequency')
plt.title('Exam Scores Histogram with Custom Bins')

# Обчислити та додати середню лінію
median_score = np.median(exam_scores)
plt.axvline(median_score, color='red', linestyle='dashed',
            linewidth=2, label=f'Median Score: {median_score}')
```

```
# Додати легенду
plt.legend()

# Відобразити графік
plt.show()
```

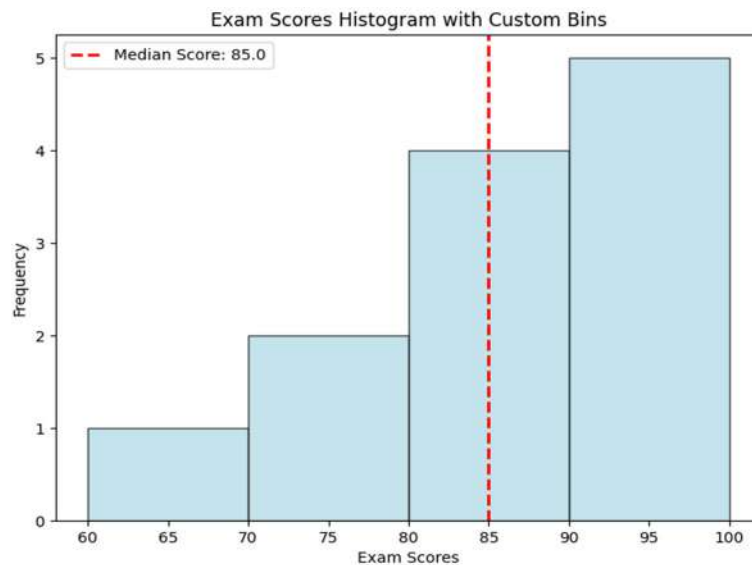


Рис. 2.9. Гістограма з власним діапазоном бінів і доданою середньою лінією

Діаграма розсіювання. Точкові діаграми відображають окремі точки даних як точки на двовимірній площині. І вони використовуються для дослідження зв'язків або кореляції між двома числовими змінними. У цьому випадку кожна вісь представляє одну змінну, а крапки представляють точки даних.

Ви можете використовувати `plt.scatter(x, y)` для створення діаграм розсіювання. Для зміни розміру точок використовуйте параметр `s`, `c` — для кольору, а `marker` — для зміни маркера замість точки. А `alpha` параметр контролює інтенсивність кольору. Для розміру можливо надіслати інший список розмірів для кожної точки.

Наприклад:

1. Дослідження зв'язку між навчальними годинами та результатами іспитів.
2. Аналіз кореляції між температурою і продажем морозива.

Код:

```
# Візьмемо приклад даних для магазинів
# Розмір магазину
stores = ['Store A', 'Store B', 'Store C', 'Store D', 'Store E']
customers = [120, 90, 150, 80, 200]
```

```

revenue = [20000, 18000, 25000, 17000, 30000]
store_size = [10, 5, 15, 8, 20]

# Тут ми масштабуємо розміри магазинів для розмірів точок у
діаграмі розсіювання
point_sizes = [size * 100 for size in store_size]

# Щоб створити діаграму розсіювання з різними розмірами точок,
# Тут альфа контролює інтенсивність кольору.
plt.scatter(customers, revenue, s=point_sizes, c='skyblue',
alpha=0.7, edgecolors='b')

# Додати мітки, назву та легенду
plt.xlabel('Number of Customers')
plt.ylabel('Revenue (USD)')
plt.title('Relationship between Customers, Revenue, and Store Size')

# Для відображення графіка
plt.show()

```

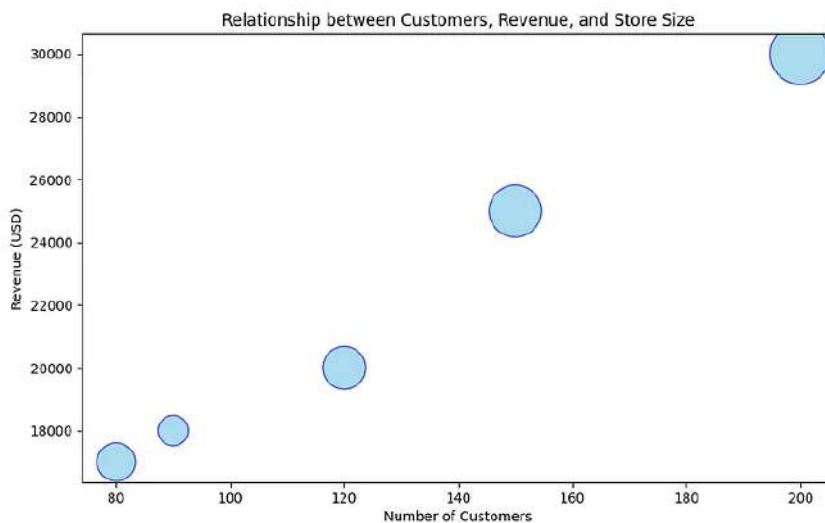


Рис. 2.10. Діаграма розсіювання з розмірами магазинів, масштабованими як розміри точок

Кругові діаграми представляють частини цілого як шматочки круглого сектора. Вони підходять для відображення складу однієї категоріальної змінної у відсотках. Але це не виглядатиме добре, якщо є більше шести категорій, оскільки вони стають незграбними, у таких випадках горизонтальна панель може бути кращою.

Скористайтеся командою, якщо є потрібний список кольорів, `plt.pie(x, labels=your_category_names, colors=desired_colors_list)`, можете надати його, а також змінити колір країв секторної діаграми за допомогою параметра `wedgeprops={'edgecolor': your_color}`.

Можемо виділити окремі сегменти за допомогою `explode` параметра, передавши кортеж, де кожен елемент є величиною, на яку він має розгорнутися. Параметр `autopct` дає змогу вибрати, скільки значень після коми має відображатися на графіку.

Варіанти використання:

1. Відображення розподілу бюджету за категоріями витрат.
2. Відображення частки ринку різних брендів смартфонів.

Наприклад: розширення окремого сегмента для кращого оповідання під час презентацій.

Код:

```
# Категорії продуктів
categories = ['Electronics', 'Clothing', 'Home Decor', 'Books',
'Toys']

# Дані про продажі для кожної категорії
sales = [3500, 2800, 2000, 1500, 1200]

# Розділіть певний сегмент (наприклад, 'Одяг'),
# друге значення (0,1) – це величина, на яку сегмент 'Одяг'.
explode = (0, 0.1, 0, 0, 0)

# Створення кругової діаграми з рознесенням і тінню
plt.pie(sales, labels=categories, explode=explode, shadow=True,
autopct='%1.1f%%')
plt.title('Sales by Product Category')

# Показати діаграму
plt.show()
```

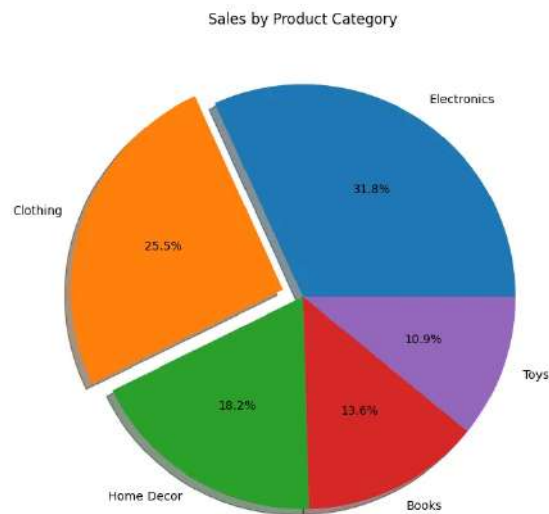


Рис. 2.11. Кругова діаграма, що виділяє сегмент одягу

Теплова карта — це візуалізація кореляційної матриці, яка дасть нам уявлення про те, як кожна змінна корелюється з іншою змінною.

`plt.imshow()` — це функція Matplotlib, яка використовується для відображення даних 2D зображень, візуалізації 2D масивів або показу зображень у різних форматах.

Використання `imshow` для теплової карти: створимо теплову карту, щоб візуалізувати кореляційну матрицю, і використаємо кольорову карту, щоб візуально показати цей зв'язок. Передаємо кореляційну матрицю в `imshow`, щоб візуалізувати теплову карту.

Код:

```
# Створить зразкову кореляційну матрицю
correlation_matrix = np.array([[1.0, 0.8, 0.3, -0.2],
                               [0.8, 1.0, 0.5, 0.1],
                               [0.3, 0.5, 1.0, -0.4],
                               [-0.2, 0.1, -0.4, 1.0]])

# Створення теплової карти для кореляційної матриці
plt.imshow(correlation_matrix, cmap='coolwarm', vmin=-1,
           vmax=1, aspect='auto', origin='upper')

# Додайте кольорову панель за допомогою таких команд
cbar = plt.colorbar()
cbar.set_label('Correlation', rotation=270, labelpad=20)

# Додайте мітки та назву
plt.title('Correlation Matrix Heatmap')
plt.xticks(range(len(correlation_matrix)), ['Var1', 'Var2', 'Var3',
                                             'Var4'])
plt.yticks(range(len(correlation_matrix)), ['Var1', 'Var2', 'Var3',
                                             'Var4'])

plt.show()
```

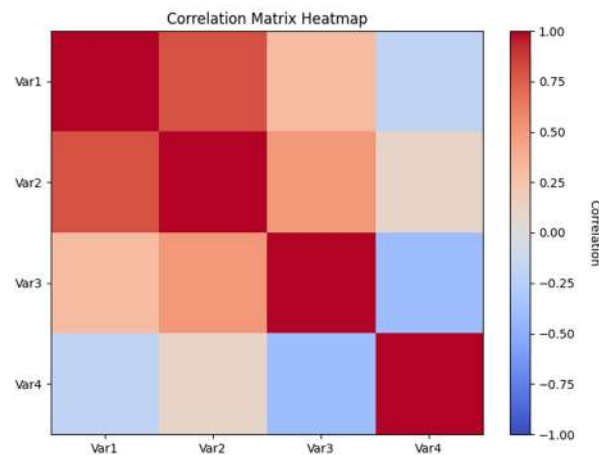


Рис. 2.12. Створення теплової карти за допомогою методу matplotlib imshow

Відображення зображень за допомогою `imshow:imread` — цей метод часто використовується для читання та завантаження зображень у різних форматах, включаючи JPEG, PNG, BMP, TIFF тощо, які потім можна відобразити за допомогою `imshow`. Якщо у вас уже є дані зображення в масиві, їх можна відобразити безпосередньо за допомогою `imshow`.

Код:

```
import matplotlib.image as mpimg

# Завантажити та відобразити зображення
img = mpimg.imread('sample_image.jpg')
# Відобразити зображення
plt.imshow(img)
# Вимкнути мітки осі та галочки
plt.axis('off')
# Додати заголовок
plt.title('Sample Image')
# Показати графік
plt.show()
```

Запитання для самоперевірки



1. Що таке візуалізація даних?
2. Опишіть чотири основні типи діаграм.
3. Опишіть основні інструменти візуалізації даних.
4. Застосування бібліотеки *Matplotlib*?
5. Як використовується об'єкт *Artist* бібліотеки *Matplotlib*, та які його складові.
6. Що таке теплова карта, як використовується та які має особливості?

2.2. Статистичні методи



Статистичний аналіз зосереджується на зборі, організації, аналізі, інтерпретації та представленні даних. Застосовуючи статистичні дані до наукової, промислової чи соціальної проблеми, прийнято починати зі статистичної сукупності або статистичної моделі, яку необхідно дослідити.

Статистика в основному поділяється на два типи:

- описова статистика (Descriptive statistics);
- статистичне висновування (Inferential statistics).

Статистичні дані можна розділити на дві категорії: кількісні та якісні. Кількісні дані — це числові дані, які можна виміряти, тоді як якісні дані — це нечислові дані, які неможливо виміряти чисельно. Розуміння типу даних, що аналізуються, має вирішальне значення для визначення найкращих статистичних методів аналізу.

Кількісні дані можна розділити на два типи: дискретні та безперервні. Безперервні дані можуть набувати будь-які значення в межах певного діапазону, тоді як дискретні дані можуть приймати лічильну кількість значень, наприклад, кількість учнів у класі.

З іншого боку, номінальні та порядкові дані є підкатегоріями якісних даних. Дані, які використовуються для іменування або позначення змінних, наприклад виду фруктів у кошику, називаються номінальними даними. Порядкові дані, з іншого боку, можна ранжирувати або впорядковувати та включати такі речі, як рівень задоволеності респондентів.

2.2.1. Описова статистика

У цій формі статистики дані узагальнюються з використанням наданих спостережень. Підсумок — це представлення вибірки сукупності з використанням таких показників, як середнє або стандартне відхилення.



За допомогою таблиць, графіків і підсумкових статистичних даних описова статистика є засобом упорядкування, зображення та опису колекції даних. Наприклад: кількість людей, які користуються Інтернетом або телебаченням у місті.

Описову статистику також поділяють на чотири різні категорії:

- міра частоти;
- міра дисперсії;
- міра центральної тенденції;
- міра положення.

Вимірювання частоти показує, як часто трапляється така частина даних. Міра дисперсії включає діапазон, дисперсію та стандартне відхилення і показує, як дані

поширюються. Міри центральної тенденції — це статистичні заходи, які визначають центральне або типові значення набору даних. Ці показники вказують, де розташовано центр розподілу даних. Основними тенденціями є середнє значення, медіана та мода. Процентиль і кватиль рангів описуються за допомогою міри позиції.

2.2.2. Статистичне висновування (statistical inference)

За допомогою цього типу статистики інтерпретується описова статистика. Іншими словами, після того, як дані були зібрані, досліджені та узагальнені, ми застосовуємо цей тип статистики, щоб пояснити важливість даних. Або, інакше кажучи, він використовується для отримання висновків із даних, які піддаються випадковим помилкам, таким як помилки спостереження, дисперсія вибірки тощо.

За допомогою статистичного висновування ми можемо вводити дані, зібрані з вибірки, щоб екстраполювати висновки про сукупності. Це дозволяє нам робити узагальнення, які виходять за межі наявних фактів або даних.



Розподіл ймовірностей. Ймовірність різних подій можна описати за допомогою розподілу ймовірностей. Найбільш типовий розподіл дзвоноподібний і має нормальний розподіл. Ймовірність успіху або невдачі в послідовності незалежних випробувань моделюється біноміальним розподілом. Коли розмір вибірки невеликий, використовується t -розподіл. Для перевірки гіпотез використовується розподіл χ^2 -квадрат.

Ось кілька прикладів того, як вони використовуються в реальному світі:

- Вимірювання зросту та ваги, показники IQ і результати тестів часто мають нормальний розподіл.
- Підкидання монети, кидання кубика або проведення опитування з запитаннями «так/ні» — усе це приклади біноміального розподілу.
- Якщо розмір вибірки невеликий і дисперсія сукупності невідома, у медичних дослідженнях використовується t -розподіл.
- Розподіл χ^2 -квадрат: використовується для визначення відповідності спостережених даних теоретичній моделі, наприклад визначення справедливості монети.



Перевірка гіпотези — це статистичний метод, який використовується для визначення правильності чи хибності гіпотези сукупності на основі вибірки даних. Багато статистичних бібліотек, включаючи SciPy і StatsModels, надають модулі Python для перевірки гіпотез.

Існує кілька ключових компонентів перевірки гіпотези.

- **Нульова та альтернативна гіпотези.** Нульова гіпотеза стверджує, що немає суттєвої різниці між вибіркою та параметром сукупності, тоді як альтернативна гіпотеза стверджує, що суттєва різниця існує.

- **Значення Р.** Значення Р представляє ймовірність побачити статистику вибірки такою ж екстремальною, як і розрахована, припускаючи, що нульова гіпотеза вірна.
- **Помилки типу I і типу II.** Помилка типу I виникає, коли нульову гіпотезу відхилено, навіть якщо вона істинна, тоді як помилка типу II виникає, коли нульову гіпотезу не відхилено, навіть якщо вона є хибною.
- **Одновибірковий t -тест:** одновибірковий t -тест використовується, щоб побачити, чи суттєво відрізняється вибіркове середнє від відомого середнього сукупності.
- **Двовибірковий t -тест.** Щоб визначити, чи існує значна різниця між середніми значеннями двох незалежних вибірок, використовується двовибірковий t -тест.
- **Тест χ^2 -квадрат:** Тест χ^2 -квадрат використовується, щоб побачити, чи існує значущий зв'язок між двома категоріальними змінними.

У реальному житті фармацевтична компанія хоче перевірити ефективність нового препарату від артеріального тиску. Нульова гіпотеза стверджує, що ліки не впливають на артеріальний тиск, тоді як альтернативна гіпотеза стверджує, що ліки мають ефект. Щоб перевірити цю гіпотезу, фармацевтична компанія може запустити t -тест з одним зразком на групі людей з високим кров'яним тиском, яким давали ліки. Це робиться для того, щоб порівняти середній артеріальний тиск цієї вибірки із середнім артеріальним тиском населення.

Крім того, фармацевтична компанія може використовувати двовибірковий t -тест для порівняння середнього артеріального тиску тих, хто отримував ліки, з тими, хто отримував плацебо. Це дозволить компанії визначити, чи є статистично значуща різниця в середньому артеріальному тиску між двома групами.

Критерій χ^2 -квадрат використовується, щоб визначити, чи існує статистично значущий зв'язок між двома категоріальними змінними. Виробник автомобіля, наприклад, може використовувати тест χ^2 -квадрат, щоб побачити, чи існує значний зв'язок між кольором автомобіля та типом трансмісії.



Довірчі інтервали допомагають оцінити діапазон значень, у який, ймовірно, потрапить справжній параметр сукупності. Рівень довіри вказує на ймовірність того, що справжній параметр сукупності потрапляє в розрахований інтервал. Найбільш часто використовуваний рівень гарантії становить 95%. **Похибка** — це величина невизначеності, яку можна очікувати при оцінці справжнього параметра сукупності.

Вибірковий розподіл статистики використовується для обчислення довірчих інтервалів. Довірчий інтервал для середнього, наприклад, розраховується за допомогою вибіркового середнього, стандартного відхилення, розміру вибірки та рівня довіри. Ось приклад обчислення довірчого інтервалу для середнього за допомогою Python:

```

Код:
import numpy as np
from scipy.stats import norm

# Дані
data = [10.2, 9.5, 8.7, 9.9, 11.1, 8.5, 10.3, 9.8]

# Середнє значення вибірки та стандартне відхилення
sample_mean = np.mean(data)
sample_std = np.std(data, ddof=1)

# Рівень довіри
conf_level = 0.95

# Ступені свободи
df = len(data) - 1

# Критичне значення
crit_val = norm.ppf((1 + conf_level) / 2)

# Похибка
moe = crit_val * sample_std / np.sqrt(len(data))

# Довірчий інтервал
conf_int = (sample_mean - moe, sample_mean + moe)

print("Confidence interval: ", conf_int)

```

Інтерпретація. Ми можемо сказати з 95% упевненістю, що справжнє середнє значення населення перебуває у діапазоні від 8,54 до 10,76.



Похибка. Рівень достовірності та розмір вибірки впливають на похибку. Більший розмір вибірки призводить до більш широкого інтервалу, тоді як вищий рівень достовірності веде до більш вузького інтервалу. Збільшуючи обсяг вибірки або зменшуючи рівень достовірності, можна знизити межу похибки.

Припустімо, що опитування проводиться для оцінки середньої кількості годин, які студенти витрачають на домашнє завдання на тиждень. Вибірку зі 100 студентів роблять у випадковий спосіб, середнє значення вибірки становить 5,2 год зі стандартним відхиленням 1,8 год. Ми шукаємо 95% довірчий інтервал навколо справжнього середнього значення сукупності.

Рішення. Використовуючи формулу для довірчого інтервалу для середнього значення, ми отримуємо:

$$moe = z \cdot (\sigma / \sqrt{n})$$

де z — критичне значення для 95% рівня довіри;

σ — стандартне відхилення генеральної сукупності (невідомо);
 n — розмір вибірки.

Ми використовуємо стандартне відхилення вибірки як оцінку, оскільки стандартне відхилення сукупності невідоме. Критичне значення 95% рівня довіри та 99 ступенів свободи — 1,984. У результаті похибка становить:

$$\text{moe} = 1.984 * (1.8 / \sqrt{100}) = 0.35712$$

Інтерпретація. Ми можемо сказати з 95% упевненістю, що справжня середня кількість населення перебуває у діапазоні від 4,84 до 5,56 год на тиждень.

2.2.3. Регресійний аналіз



Регресійний аналіз — це статистичний метод для аналізу зв'язку між залежною змінною та однією чи кількома незалежними змінними.

Існує три типи регресійного аналізу.

1. **Проста лінійна регресія.** Використовується для встановлення зв'язку між залежною змінною та окремою незалежною змінною. Лінійне рівняння визначає залежність, причому нахил і перетин лінії представляють вплив незалежної змінної на залежну змінну. Наприклад, агент з нерухомості хоче визначити співвідношення між розміром будинку (у квадратних метрах) і його ціною продажу. Агент може використовувати просту лінійну регресію, щоб передбачити ціну продажу будинку на основі його розміру.

Код:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Створення набору даних
X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]).reshape(-1, 1)
y = np.array([2, 4, 5, 7, 8, 10, 11, 13, 14, 16])

# Створення моделі лінійної регресії
model = LinearRegression().fit(X, y)

# Отримати нахил і відрізок лінії
slope = model.coef_
intercept = model.intercept_

# Побудувати точки даних і лінію регресії
plt.scatter(X, y)
plt.plot(X, slope*X + intercept, color='red')
plt.show()
```

2. **Множинна лінійна регресія.** Використовується, коли дві або більше незалежні змінні впливають на залежну змінну. Лінійне рівняння визначає залежність, причому коефіцієнти незалежних змінних представляють вплив кожної змінної на залежну змінну. Наприклад, виробник автомобілів хоче спрогнозувати паливну ефективність своїх транспортних засобів на основі різних незалежних змінних, таких як розмір двигуна, потужність і вага.
3. **Логістична регресія.** Це статистичний метод для дослідження зв'язку між двійковою залежною змінною (результатом) та однією або декількома незалежними змінними (предикторами). Мета логістичної регресії полягає в тому, щоб знайти найкращу модель для опису зв'язку між залежною змінною та незалежними змінними, а потім використовувати цю модель для прогнозування змінної результату. Наприклад, банк хоче передбачити, чи клієнт не виплатить кредит, виходячи з його кредитного рейтингу, доходу та інших факторів. Використовуючи логістичну регресію, банк може оцінити ймовірність дефолту клієнта та вжити відповідних заходів для мінімізації їх ризику.

Код:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression

# Завантаження даних
data = pd.read_csv('data.csv')

# Розділіть дані на навчальні та тестові набори
train = data[:800]
test = data[800:]

# Визначити незалежні змінні
X_train = train[['age', 'gender', 'income']]
X_test = test[['age', 'gender', 'income']]

# Визначити залежну змінну
y_train = train['buy_product']
y_test = test['buy_product']

# Підібрати модель логістичної регресії
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

# Передбачити результати тестових даних
y_pred = logreg.predict(X_test)

# Оцініть продуктивність моделі
from sklearn.metrics import accuracy_score
accuracy = accuracy
```


Статистичне висновування має широкий спектр застосувань у сценаріях реального світу:

- медичні випробування: статистичні дані можуть допомогти визначити ефективність нового препарату порівняно з плацебо;
- опитування: може допомогти передбачити результати виборів на основі даних опитувань;
- контроль якості: може допомогти визначити, чи виробничий процес виробляє продукти в певному діапазоні допусків.
- фінанси: може допомогти оцінити очікуваний прибуток і ризик, пов'язаний з різними інвестиціями.

2.2.4. Статистичні тести

Статистичні тести використовуються під час перевірки гіпотез і можуть бути використані для визначення того, чи мають прогностична змінна та змінна результату статистично значущий зв'язок. Для цього необхідно оцінити різницю між двома або більше групами. Параметричні тести використовуються, якщо дані розподілені нормально. Нульова гіпотеза — це твердження, що між двома популяціями немає різниці, тоді як альтернативна гіпотеза — це твердження, що різниця існує.

Статистичні тести поділяються на дві основні категорії:

- параметричний;
- непараметричний.

Припущення, які слід зробити або перевірити:

1. **Незалежність спостережень.** Окремі спостереження (кожен запис змінної) не пов'язані одне з одним (наприклад, повторення того самого тесту на одному пацієнті призводить до незалежних вимірювань, тобто повторних вимірювань).
2. **Нормальність даних.** Дані мають нормальний розподіл. Це припущення потрібне лише для кількісних даних.
3. **Однорідність дисперсії.** Кожна порівнювана група має подібну дисперсію (тобто розподіл, або розкид, балів навколо середнього). Потужність тесту для виявлення відмінностей буде зменшена, якщо одна група має значно більші варіації від інших.

Крім того, ви можете застосувати тести, які враховують те, що ваші дані не відповідають передумові незалежності спостережень (тобто тести повторних вимірювань).

Якщо ваші дані натомість не задовольняють припущення про нормальність або однорідність дисперсії, ви можете виконати непараметричний статистичний тест, який дозволить вам робити порівняння без цих двох припущень.

Існують наступні типи тестів (рис. 2.2.1):

- Z-тест;
- T-тест;
- тест ANOVA;
- тест χ^2 -квадрат;
- кореляційний тест.



Слід мати чітке розуміння концепції перевірки гіпотез у статистиці, тому що робити статистичні висновки є важливою справою, недостатньо розв'язувати задачі за допомогою математики та коду Python.

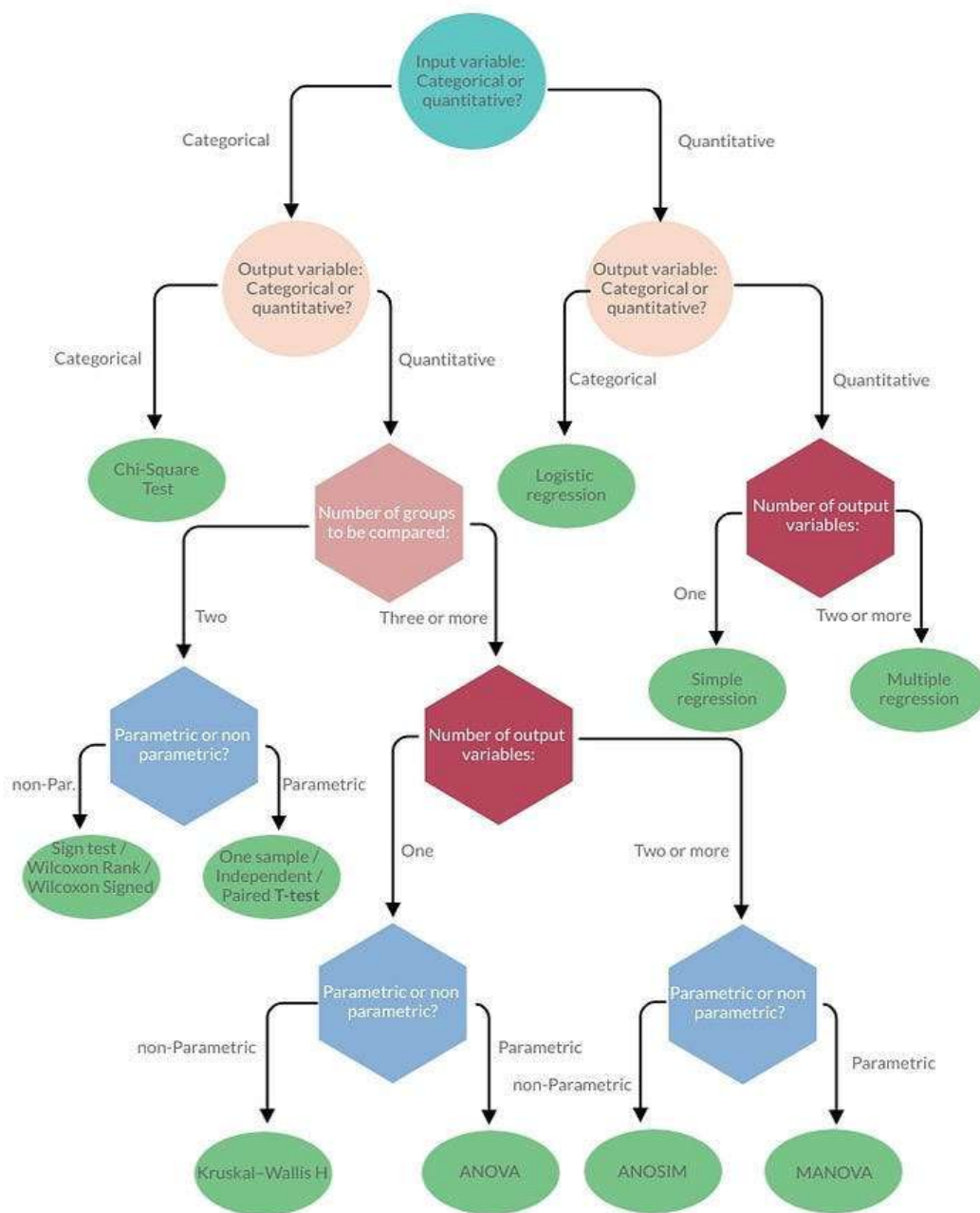


Рис. 2.2.1. Алгоритм вибору статистичного тесту



Z-тест — це статистичний тест, який використовується для порівняння середнього значення вибірки із середнім значенням генеральної сукупності, якщо відомо стандартне відхилення генеральної сукупності. Простіше кажучи, це статистичний тест, який визначає, чи відрізняються два середні сукупності, коли дисперсії відомі та розмір вибірки великий.

Даний тест застосовується при перевірці гіпотез, у яких z -статистика відповідає нормальному розподілу. Z -статистика або Z -оцінка — це число, що представляє результат Z -тесту.

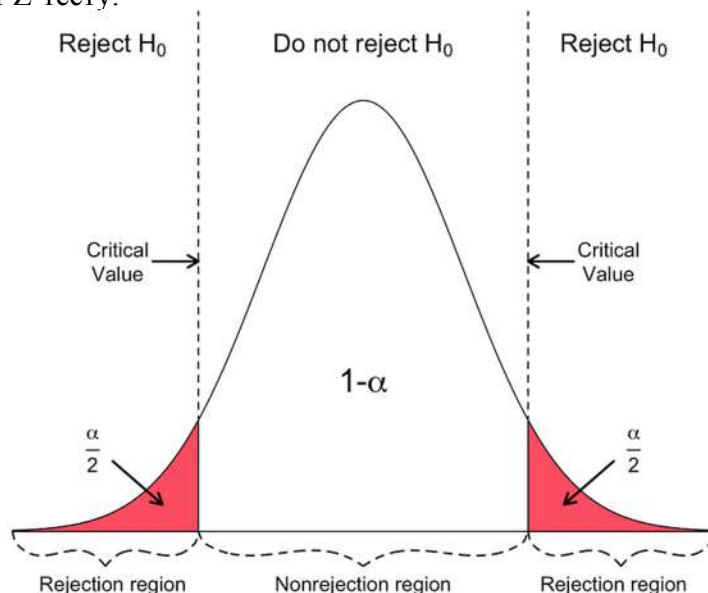


Рис. 2.2.2. Візуалізація Z -тесту

Щоб зрозуміти Z -тест, розглянемо реальний приклад. Скажімо, взуттєва фірма стверджує, що типовий термін служби її взуття становить 500 днів. Ми зібрали зразки із 50 пар взуття, щоб перевірити це твердження, і виявили, що середня тривалість життя виробу становить 490 днів зі стандартним відхиленням у 25 днів. Тепер ми можемо запустити Z -тест, щоб побачити, чи зразок підтверджує твердження компанії чи ні.

Коли використовувати Z -тест:

- Розмір вибірки має бути більше від 30, інакше треба використовувати t -тест.
- Вибірки слід відбирати випадковим чином із популяції.
- Необхідно знати стандартне відхилення генеральної сукупності.
- Вибірки, відібрані з сукупності, повинні бути незалежними одна від одної.
- Дані мають бути нормально розподілені, однак для великого розміру вибірки передбачається, що вони мають нормальний розподіл.

Алгоритм виконання Z-тесту:

- Спочатку визначте нульову та альтернативну гіпотези.
- Визначте рівень значущості (α).
- Знайдіть критичне значення z у z -критерії за допомогою.
- Обчисліть статистику Z-тесту.

Формула Z-тесту:

$$Z = \frac{x - \mu}{\sigma}$$

де Z = Z-тест; x — значення, яке оцінюється; μ — середнє значення; σ — стандартне відхилення.

Якщо p -значення *нижче* за порогове значення (що означає менше), то ви можете зробити висновок про статистично значущий зв'язок між вхідними та цільовими змінними.

Задача. Уявімо, що IQ у певній популяції людей нормально розподілений із середнім значенням $\mu = 100$ і стандартним відхиленням $\sigma = 15$.

Дослідник хоче знати, чи впливає новий препарат на рівень IQ (Так чи Ні [категорично]), тому він набирає 20 пацієнтів, щоб випробувати його, і записує їхній рівень IQ.

Код для виконання Z -тесту з одним зразком на Python, щоб визначити, чи викликає новий препарат значну різницю в рівнях IQ:

```
from statsmodels.stats.weightstats import ztest as ztest
#H0 : Новий препарат не впливає на IQ
#H1 : Новий препарат впливає на IQ

#введіть рівні IQ для 20 пацієнтів
data = [88, 92, 94, 94, 96, 97, 97, 97, 99, 99,
        105, 109, 109, 109, 110, 112, 112, 113, 114, 115]

#виконати один зразок z-test
ztest(data, value=100)
```

Статистика тесту для z -критерію однієї вибірки становить 1,5976, а відповідне значення p — 0,1101.

Оскільки це значення p не менше, ніж 0,05, ми не маємо достатніх доказів, щоб відхилити нульову гіпотезу. Іншими словами, новий препарат істотно не впливає на рівень IQ.



T-тест — це перевірка статистичної гіпотези, яка використовується для визначення суттєвої різниці між середніми значеннями двох груп. Це тип параметричного тесту, який припускає, що дані розподілені нормально і що дисперсії двох груп рівні.

Існує два основних типи *t*-тестів:

- Т-критерій для однієї вибірки (використовується для порівняння середнього значення однієї групи з відомим значенням або гіпотетичним значенням);
- Т-критерій для двох вибірок (використовується для порівняння середніх двох незалежних груп).

Т-критерій працює шляхом обчислення *t*-статистики, яка вимірює різницю між середніми значеннями двох груп відносно варіації в кожній групі. Потім *t*-статистика порівнюється з критичним значенням *t*-розподілу на основі ступенів свободи та бажаного рівня значущості (зазвичай 0,05). Якщо розрахована *t*-статистика більша за критичне значення, нульова гіпотеза (про те, що середні значення рівні) відхиляється на користь альтернативної гіпотези (що середні значення різні).

Формула для *t*-статистики:

$$t = (x_1 - x_2) / \left(s * \sqrt{\frac{2}{n}} \right)$$

де:

- x_1 і x_2 є вибірковими середніми для двох груп
- s – об'єднане стандартне відхилення двох груп
- n – розмір вибірки кожної групи/

Код Python для проведення двовибіркового *t*-тесту:

```
from scipy.stats import ttest_ind

group1 = [1, 2, 3, 4, 5]
group2 = [6, 7, 8, 9, 10]

t_stat, p_value = ttest_ind(group1, group2)

print("t-statistic:", t_stat)
print("p-value:", p_value)
```

Парний *t*-тест використовується для порівняння відмінностей між парою залежних змінних для того самого суб'єкта. Наприклад, у нас є сорт рослин А, і ми хочемо порівняти врожайність А до та після внесення певного добрива.

Парний *t*-тест — це *t*-тест для однієї вибірки на відмінності між двома залежними змінними.

Гіпотези:

- Нульова гіпотеза — немає різниці між двома залежними змінними (різниця = 0).
- Альтернативна гіпотеза — існує різниця між двома залежними змінними (двостороння або двостороння).
- Альтернативна гіпотеза — різниця між двома змінними відповіді, більша або менша за нуль (одностороння або одностороння).

Припущення:

- Відмінності між двома залежними змінними відповідають приблизно нормальному розподілу (тест Шапіро — Вілкса).
- Незалежна змінна повинна мати пару залежних змінних.
- Відмінності між двома залежними змінними не повинні мати викидів.
- Спостереження відбираються незалежно один від одного.

Бібліотека `bioinfokit` надає різноманітні прості у використанні функціональні можливості для аналізу, візуалізації та інтерпретації біологічних даних, отриманих у результаті експериментів

Код:

```
# install bioinfokit as !pip install bioinfokit
from bioinfokit.analys import get_data, stat
# завантажте набір даних як фрейм даних pandas
# набір даних не повинен містити пропущених (NaN) значень. Якщо так,
то буде пропущено
df = get_data('t_pair').data
df.head(2)

res = stat()
res.ttest(df=df, res=['AF', 'BF'], test_type=3)
print(res.summary)
```

Р-значення, отримане за t-критерієм, є значущим ($p < 0,05$), і отже, ми робимо висновок, що врожайність сорту рослини А значно збільшилася у ході застосування добрив.



Тест χ^2 -квадрат — використовується для визначення наявності суттєвого зв'язку між двома категоріальними змінними. Він вимірює, наскільки дані спостереження відрізняються від очікуваних даних за конкретною гіпотезою.

Тест обчислює статистику χ^2 -квадрат, яка порівнюється з критичним значенням із розподілу χ^2 -квадрат, щоб визначити, чи результати є статистично значущими. Формула для статистики χ^2 -квадрат така:

$$\chi^2 = \sum \frac{(\text{спостережувана частота} - \text{очікувана частота})^2}{\text{очікувана частота}}$$

У Python `scipy.stats` модуль надає функції для виконання тесту χ^2 -квадрат.

```

Код:
import numpy as np
from scipy.stats import chi2_contingency

# Створити таблицю непередбачуваності
observed = np.array([[50, 30], [20, 40]])

# Виконати тест хі-квадрат
chi2_stat, p_val, dof, expected = chi2_contingency(observed)

print("Chi-squared statistic:", chi2_stat)
print("P-value:", p_val)
print("Degrees of freedom:", dof)
print("Expected frequencies:\n", expected)

```



Тест ANOVA (Analysis of Variance). Дисперсійний аналіз використовується, щоб визначити, чи існує значна різниця між середніми значеннями трьох або більше груп. Він розподіляє загальну дисперсію даних на різні джерела та обчислює F -статистику, яка порівнюється з критичним значенням F -розподілу, щоб визначити, чи результати є статистично значущими.

Формула для розрахунку:

$$F = \frac{(\text{дисперсія між групами}(k - 1))}{(\text{дисперсія всередині групи} / (n - k))}$$

де k — кількість груп; n — загальний розмір вибірки.

У Python statsmodels модуль надає функції для виконання ANOVA.

```

Код:
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols

# Завантажити дані
data = pd.read_csv( "data.csv" )

# Виконати ANOVA
model = ols('score ~ group', data=data).fit()
anova = sm.stats.anova_lm(model, typ=2)

print (anova)

```


Z-критерій використовується для оцінки припущень щодо середніх популяцій, тоді як Т-тест використовується для визначення того, чи існує значна різниця між двома групами.

Запитання для самоперевірки



1. *Поняття статистичної гіпотези.*
2. *Що таке похибка?*
3. *Що таке довірчий інтервал?*
4. *Статистичні тести: z — тест та t — тест, їх відмінності*
5. *Поняття регресії, її типи*
6. *Що таке тест ANOVA?*

2.3. Аналіз часових рядів



Ситуація 1. Ви відповідаєте за центр доставки піци та хочете знати, чи ваші продажі йдуть за певною схемою, оскільки ви відчуваєте, що щосуботи ввечері кількість ваших замовлень зростає...

Ситуація 2. Ваша компанія продає продукт, і ви відповідаєте за передбачення чи прогнозування поставок, необхідних для цього продукту в певний момент у майбутньому...

Ситуація 3. Ви відстежуєте центр обробки даних і хочете виявити будь-яку аномалію, як-от ненормальне використання ЦП, яка може призвести до простою ваших серверів. Ви стежите за кривою використання ЦП і хочете знати, коли виникає аномалія...

У кожній із цих ситуацій ви маєте справу з часовими рядами.

2.3.1. Визначення часового ряду



Визначення часового ряду — це упорядкована послідовність значень змінної, отримана в послідовний момент часу. Це послідовність даних, упорядкована (або проіндексована) за часом. Він дискретний, а інтервал між кожною точкою постійний.

Приклад: температура, яка реєструється кожні дві години протягом дня, створить часовий ряд (рис. 2.3.1).



Рис. 2.3.1. Часовий ряд температури

Залежно від частоти спостережень часовий ряд зазвичай може бути погодинним, щоденним, тижневим, місячним, кварталним і річним. Іноді також можуть бути секундні та хвилинні часові ряди, наприклад, кількість кліків і відвідувань користувачів сайту щохвилини.

Навіщо взагалі аналізувати часовий ряд? Тому що це є підготовчий етап перед розробкою прогнозу.

Крім того, прогнозування часових рядів має величезне комерційне значення, оскільки такі важливі для бізнесу речі, як попит і продажі, кількість відвідувачів веб-сайту, ціна акцій тощо, по суті, є даними часових рядів.

Що є можливим робити з даними часових рядів:

- отримати розуміння керівних сил та структури, які створили спостережувані дані;
- моделювати дані спостережень і робити прогнози.

Деякі застосування аналізу часових рядів включають:

- прогнозування продажів;
- прогнози врожайності;
- прогнози навантаження.

2.3.2. Складові часових рядів

Часовий ряд можна розкласти на кілька компонентів, і розуміння цих компонентів має вирішальне значення для ефективного прогнозування: рівень — середнє значення за певний період часу; тренд — довгостроковий рух або напрямок даних, він може бути вгору, вниз або залишатися відносно постійним; тенденції можуть бути лінійними та нелінійними; сезонність — повторювані моделі або цикли протягом фіксованого інтервалу часу. Сезонні ефекти часто зумовлені зовнішніми факторами, такими як погода чи свята. Сезонність може бути щоденною, тижневою, місячною або річною.

Також треба згадати шум (залишки) — випадкові коливання або нерівності в даних, які не можна пояснити тенденцією чи сезонністю. Представлено різницю між спостережуваними та прогнозованими значеннями.

Складова тенденції (тренду) — це довгострокова модель часового ряду. Може бути висхідний тренд (upward), спадний тренд (downward) і стаціонарний / горизонтальний тренд (stationary) (рис. 2.3.2).

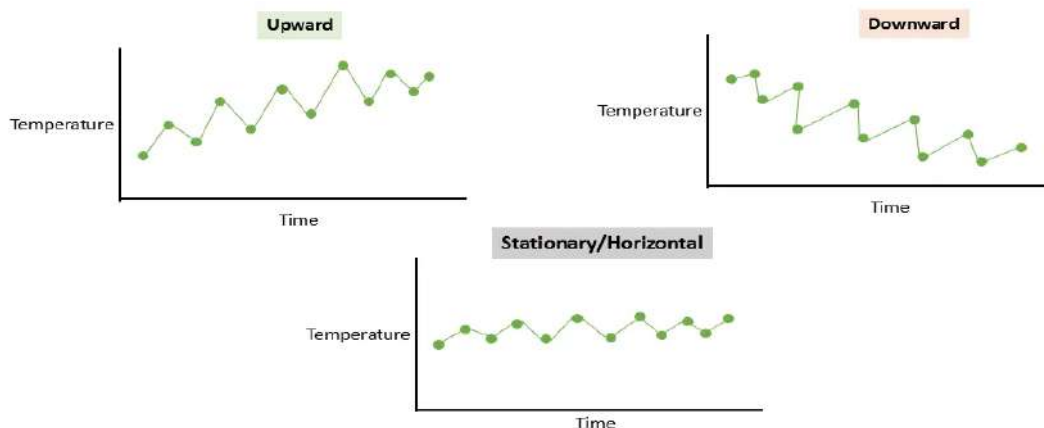


Рис. 2.3.2. Тренди

Циклічна складова. Рух вгору і вниз навколо лінії тренду (рис. 2.3.3).

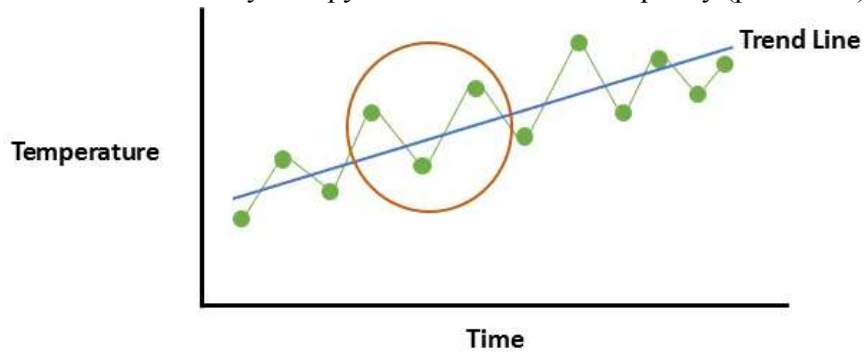


Рис. 2.3.3. Циклічна складова

Сезонна складова. Регулярні коливання навколо лінії тренду протягом фіксованого інтервалу. Це може відбуватися протягом одного сезону або місяців кожного року (див.рис.2.3.4).

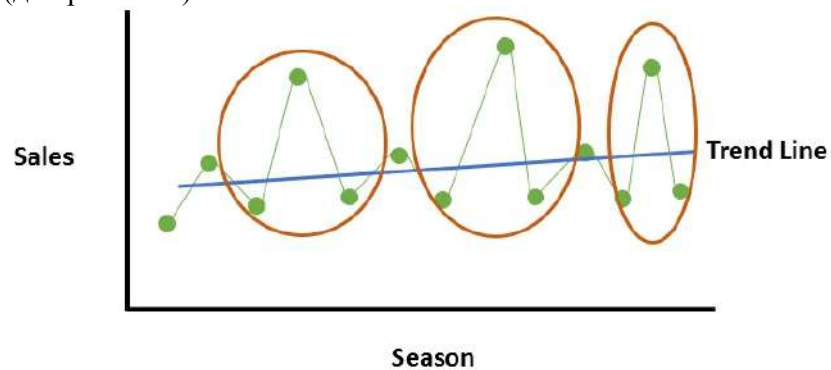


Рис. 2.3.4. Сезонна складова

Нерегулярна складова. Непередбачуваний компонент. Наприклад, це може бути велике коливання (рис. 2.3.5).

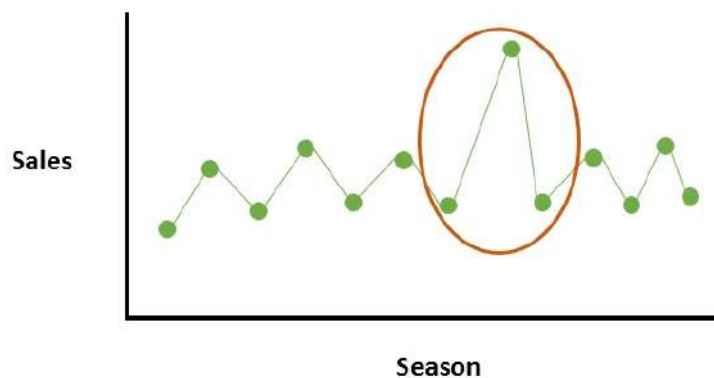


Рис. 2.3.5. Нерегулярна складова



Стаціонарні та нестаціонарні часові ряди. Стаціонарний часовий ряд — це ряд, властивості якого не залежать від часу, в який цей ряд спостерігається. Часові ряди з трендами або сезонністю не є стаціонарними, оскільки тренд і сезонність впливають на значення часових рядів у різний час.

Часто стаціонарний часовий ряд не має передбачуваної моделі в довгостроковій перспективі. В аналізі часових рядів стаціонарність належить до статистичних властивостей часового ряду, які залишаються постійними з часом. Існує два основні типи стаціонарності: адитивна і мультиплікативна.

- *Адитивна стаціонарність.* Кажуть, що часовий ряд демонструє адитивну стаціонарність, якщо середнє значення та дисперсія залишаються незмінними протягом часу.
- *Мультиплікативна стаціонарність.* Кажуть, що часовий ряд демонструє мультиплікативну стаціонарність, якщо середнє значення та дисперсія пропорційні та залишаються незмінними протягом часу.
- *Сезонна стаціонарність.* Сезонна стаціонарність належить до часового ряду, який демонструє повторювану модель або сезонність протягом фіксованих інтервалів часу.
- *Стаціонарність тенденції.* Стаціонарність тенденції стосується часового ряду, який зберігає послідовну тенденцію з часом.

Перетворення, що використовуються для стаціонаризації даних.

Усунення трендів: видаляємо основну тенденцію в ряді. Це можна зробити кількома способами залежно від характеру даних:

1. Індексовані дані: дані, виміряні у валютах, пов'язані з індексом цін або пов'язані з інфляцією. Отже, поділ ряду на цей індекс (тобто дефляція) поелементно є рішенням для зменшення тренду даних.
2. Неіндексовані дані: чи необхідно оцінити, чи тенденція є постійною, лінійною чи експоненціальною. Перші два випадки прості, для останнього необхідно оцінити темп зростання (інфляція чи дефляція) і застосувати той самий метод, що і для індексованих даних.

Розбіжність: сезонні або циклічні моделі можна видалити шляхом віднімання періодичних значень. Якщо дані сезонні за 12 місяців, віднімання ряду з різницею ряду з 12 лагами дасть більш плоский ряд.

Логування: у випадку, коли складена ставка в тренді не зумовлена індексом цін (тобто ряд не вимірюється у валюті), журналювання може допомогти лінеаризувати ряд із експоненціальним трендом (нагадаємо, що $\log(\exp(x)) = x$). На відміну від дефляції, це не усуває кінцевої тенденції.

На практиці те, чи розглядати часовий ряд як додатковий або мультиплікативний, а також врахування сезонності та тренду залежить від характеристик конкретних даних і цілей аналізу.

2.3.3. Методи аналізу часових рядів

Розглянемо базові три методи, які використовуються в аналізі часових рядів.

Моделі ковзного середнього. Модель ковзного середнього використовує регресійну модель, що складається з минулих помилок прогнозу для майбутніх прогнозів.

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \dots + \theta_q \varepsilon_{t-q}$$

де ε — білий шум; а від ϕ_1 до ϕ_q — параметри.

Моделі авторегресії. Використовується лінійна комбінація минулих значень змінної, щоб робити прогнози:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \varepsilon_t$$

де ε — білий шум; а від ϕ_1 до ϕ_q — параметри.

Моделі машинного навчання. Використовуються регресійні моделі машинного навчання, такі як лінійна регресія, Light GBM або регресор нейронної мережі, щоб робити прогнози на основі даних часових рядів.



Дані для часового ряду зазвичай зберігаються у файлах csv або інших форматах електронних таблиць і містять два стовпці: дата та вимірне значення (метрика).

Щоб прочитати набір даних часових рядів (файл csv про продажі ліків в Австралії) треба перетворити його в фрейм даних pandas. Додавання аргументу призведе до аналізу стовпця дати як поля дати (рис. 2.3.5).

`read_csv(parse_dates=['date']).`

Код:

```
from dateutil.parser import parse
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
plt.rcParams.update({'figure.figsize': (10, 7), 'figure.dpi': 120})
# Перетворення в фрейм даних
df =
pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse_dates=['date'])
df.head()
```

	date	value
0	1991-07-01	3.526591
1	1991-08-01	3.180891
2	1991-09-01	3.252221
3	1991-10-01	3.611003
4	1991-11-01	3.565869

Рис. 2.3.5. Набір даних часового ряду

Крім того, є можливим імпортувати його як серію pandas із датою в якості індексу. Для цього просто потрібно вказати `index_col` аргумент у `pd.read_csv()`

Код:

```
ser =
pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse_dates=['date'], index_col='date')
ser.head()
```

Панельні дані також є набором даних на основі часу. Різниця полягає в тому, що, окрім часових рядів, вони також містять одну або кілька пов'язаних змінних, які вимірюються за ті самі періоди часу. Як правило, стовпці в панелі даних містять пояснювальні змінні, які можуть бути корисними для прогнозування Y , за умови, що ці стовпці будуть доступні в майбутньому періоді прогнозування.

Приклад панелі даних:

```
# адреса набору даних: https://github.com/rouseguay
df =
pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/MarketArrivals.csv')
df = df.loc[df.market=='MUMBAI', :]
df.head()
```

	market	month	year	quantity	priceMin	priceMax	priceMod	state	city	date
6654	MUMBAI	January	2004	267100	719	971	849	MS	MUMBAI	January-2004
6655	MUMBAI	January	2005	275845	261	513	387	MS	MUMBAI	January-2005
6656	MUMBAI	January	2006	228000	315	488	402	MS	MUMBAI	January-2006
6657	MUMBAI	January	2007	205200	866	1136	997	MS	MUMBAI	January-2007
6658	MUMBAI	January	2008	267550	348	550	448	MS	MUMBAI	January-2008

Рис. 2.3.6. Панель даних

Візуалізація часового ряду. Використаємо бібліотеку matplotlib для візуалізації ряду:

```
# адреса даних часового ряду: fpp package in R.
import matplotlib.pyplot as plt
df =
pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse_dates=['date'], index_col='date')

# Побудувати графік
def plot_df(df, x, y, title="", xlabel='Date', ylabel='Value', dpi=100):
    plt.figure(figsize=(16,5), dpi=dpi)
    plt.plot(x, y, color='tab:red')
    plt.gca().set(title=title, xlabel=xlabel, ylabel=ylabel)
    plt.show()

plot_df(df, x=df.index, y=df.value, title='Monthly anti-diabetic
drug sales in Australia from 1992 to 2008.')
```

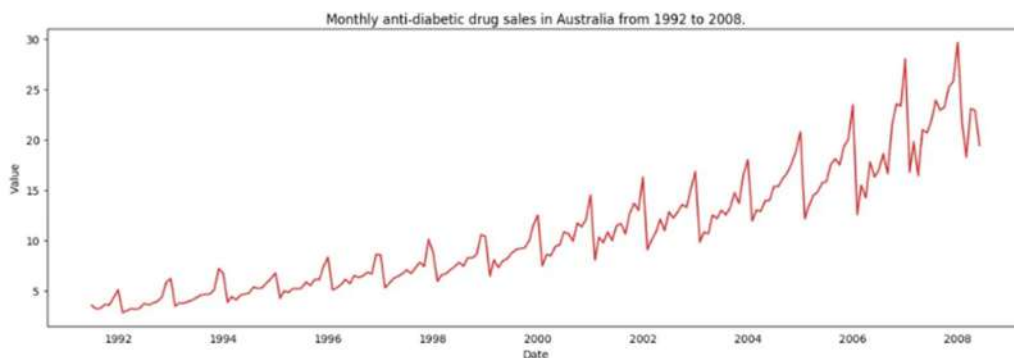


Рис. 2.3.7. Візуалізація часового ряду

Якщо всі значення додатні, є можливість показати це по обидві сторони осі Y, щоб підкреслити зростання (рис. 2.3.8).

```
# імпорт даних
df = pd.read_csv('datasets/AirPassengers.csv', parse_dates=['date'])
x = df['date'].values
y1 = df['value'].values

# Графік
fig, ax = plt.subplots(1, 1, figsize=(16,5), dpi= 120)
plt.fill_between(x, y1=y1, y2=-y1, alpha=0.5, linewidth=2,
color='seagreen')
plt.ylim(-800, 800)
plt.title('Air Passengers (Two Side View)', fontsize=16)
plt.hlines(y=0, xmin=np.min(df.date), xmax=np.max(df.date), linewidth=.5)
plt.show()
```

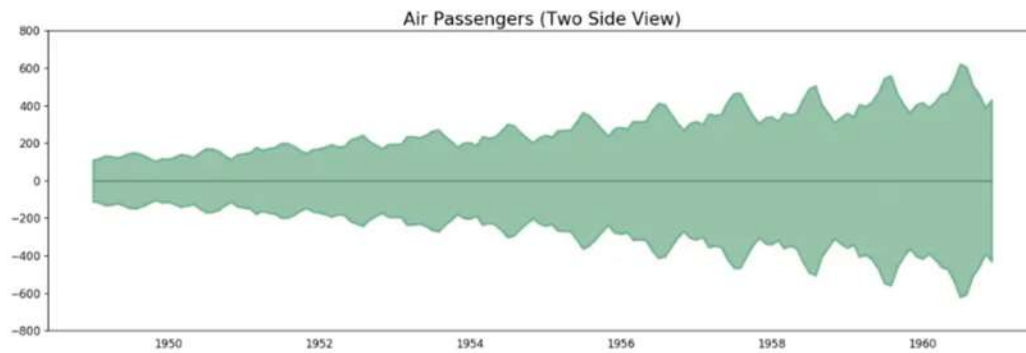


Рис. 2.3.8. Візуалізація часового ряду — дві бічні серії

Якщо це місячний часовий ряд і щороку він повторюється за певною схемою, є можливим нанести кожен рік як окремий рядок на одному графіку. Це дає змогу порівняти річні зразки.

```
# Імпорт даних
df =
pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse_dates=['date'], index_col='date')
df.reset_index(inplace=True)

# Виділення даних
df['year'] = [d.year for d in df.date]
df['month'] = [d.strftime('%b') for d in df.date]
years = df['year'].unique()
# визначення кольорів
np.random.seed(100)
mycolors = np.random.choice(list(mpl.colors.XKCD_COLORS.keys()),
len(years), replace=False)
# Побудова графіків
plt.figure(figsize=(16,12), dpi= 80)
for i, y in enumerate(years):
    if i > 0:
        plt.plot('month', 'value', data=df.loc[df.year==y, :],
color=mycolors[i], label=y)
        plt.text(df.loc[df.year==y, :].shape[0]-.9,
df.loc[df.year==y, 'value'][-1:].values[0], y, fontsize=12,
color=mycolors[i])
# Написи
plt.gca().set(xlim=(-0.3, 11), ylim=(2, 30), ylabel='$Drug Sales$',
xlabel='$Month$')
plt.yticks(fontsize=12, alpha=.7)
plt.title("Seasonal Plot of Drug Sales Time Series", fontsize=20)
plt.show()
```

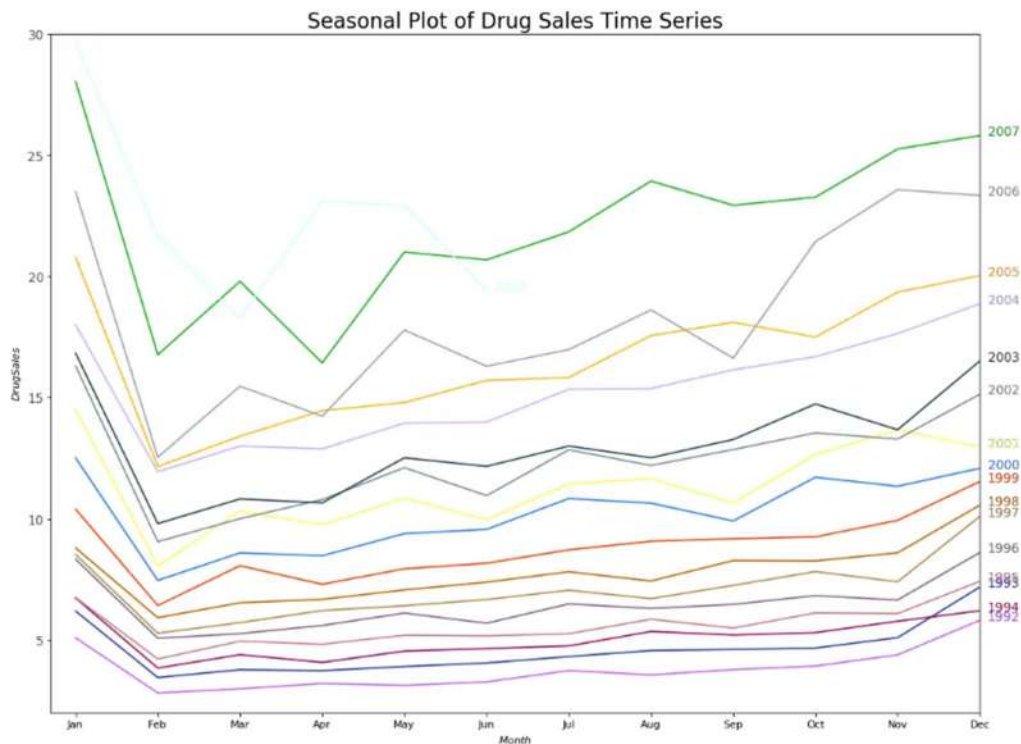


Рис. 2.3.9. Візуалізація часового ряду — сезонність

Що лютого продажі ліків різко падають, у березні знову зростають, у квітні знову падають і так далі. Зрозуміло, що картина повторюється протягом певного року, кожного року.

Будь-який часовий ряд можна розділити на такі компоненти:

Базовий рівень + Тренд + Сезонність + Помилка

Тенденція спостерігається, коли в часовому ряді спостерігається збільшення або зменшення нахилу. У той час як сезонність спостерігається, коли існує чітка повторювана картина між регулярними інтервалами через сезонні фактори. Це може бути через місяць року, день місяця, дні тижня чи навіть час доби.

Однак необов'язково, щоб усі часові ряди мали тенденцію та/або сезонність. Часовий ряд може не мати чіткої тенденції, але мати сезонність. Може бути і протилежне.

Отже, часовий ряд можна уявити як комбінацію тенденції, сезонності та умов помилки.

Іншим аспектом, який слід враховувати, є циклічна поведінка. Це трапляється, коли модель зростання та падіння в серії не відбувається у фіксованих календарних інтервалах. Варто бути обережним, щоб не плутати циклічний ефект із сезонним.

Якщо шаблони не мають фіксованих календарних частот, то вони циклічні. Оскільки, на відміну від сезонності, циклічні ефекти зазвичай залежать від бізнесу та інших соціально-економічних факторів.

Залежно від характеру тенденції та сезонності часовий ряд можна моделювати як адитивний, або мультиплікативний, де кожне спостереження в ряді можна виразити як суму, або добуток, компонентів:

Адитивний часовий ряд:

значення = базовий рівень + тренд + сезонність + помилка

Мультиплікативний часовий ряд:

значення = базовий рівень $\times$ тенденція $\times$ сезонність $\times$ помилка

Досягнення стаціонарності часто є метою аналізу часових рядів, оскільки багато методів прогнозування припускають стаціонарний часовий ряд. Якщо часовий ряд не є стаціонарним, для стабілізації середнього значення та дисперсії можна застосувати такі методи, як розрізнення або перетворення (наприклад, логарифмічне перетворення).

Найбільш поширеним і зручним методом стаціонаризації часового ряду є розрізнення ряду принаймні один раз, поки не стане приблизно стаціонарним.

Якщо Y_t це значення в момент часу t , то перша різниця $Y = Y_t - Y_{t-1}$. Простіше кажучи, розрізнення ряду — це не що інше, як віднімання наступного значення від поточного значення.

Якщо перша різниця не робить ряд стаціонарним, можете скористатися другою різницею. І так далі.

Наприклад, розглянемо такі ряди: [1, 5, 2, 12, 20].

Перша різниця дає: [5 - 1, 2 - 5, 12 - 2, 20 - 12] = [4, -3, 10, 8].

Друга різниця дає: [-3 - 4, -10 - 3, 8 - 10] = [-7, -13, -2].

Стаціонарність ряду можна визначити, розділивши ряд на 2 або більше суміжні частини, і обчислюючи підсумкову статистику, як-от середнє, дисперсія та автокореляція. Якщо статистика зовсім інша, то серія навряд чи буде стабільною.

Навіщо робити нестаціонарний ряд стаціонарним перед прогнозуванням?
Прогнозувати стаціонарні ряди відносно легко, і прогнози більш надійні. Важливою причиною є те, що авторегресійні моделі прогнозування є, по суті, моделями лінійної регресії, які використовують лаг(и) самого ряду як предиктори.

Зараз прогнозування часових рядів можна умовно розділити на два типи.

Якщо ви використовуєте лише попередні значення часового ряду для прогнозування його майбутніх значень, це називається однофакторним прогнозуванням часового ряду.

Якщо для прогнозування використовуються предиктори, відмінні від рядів (так звані екзогенні змінні), це називається багатоваріантним прогнозуванням часових рядів.

Відомо, що лінійна регресія працює найкраще, якщо предиктори (змінні X) не корельовані одна з одною. Отже, стаціонаризація ряду вирішує цю проблему, оскільки усуває будь-яку постійну автокореляцію, тим самим роблячи предиктори (лаги ряду) у моделях прогнозування майже незалежними.

Запитання для самоперевірки



1. Що таке часового ряду?
2. В чому є основні задачі аналізу часового ряду?
3. Які є основні компоненти часового ряду?
4. Що таке стаціонарні та нестаціонарні часові ряди?
5. Які є методи аналізу часових рядів?

2.4. Аналіз соціальних мереж (SNA)



Мережі — це загальний, але потужний спосіб подання та вивчення простих та складних взаємодій. Мережі допомагають нам розуміти закономірності з'єднань та взаємин, які формують наше життя.

За своєю суттю **мережа** — це найпростіший опис безлічі взаємопов'язаних сутностей, які ми називаємо вузлами, та їх з'єднань, які ми називаємо зв'язками.

Мережі використовуються вивчення найрізноманітніших систем. Вузли можуть представляти всі види сутностей: людей, міста, комп'ютери, вебсайти, концепції, клітини, гени, види тварин тощо. Зв'язки являють собою взаємовідносини або взаємодії між цими сутностями: дружні зв'язки між людьми, рейси між аеропортами, пакети, якими комп'ютери обмінюються в інтернеті, зв'язки між сторінками Всесвітньої павутини, синапси між нейронами і та ін.

2.4.1. Визначення соціальних мереж

Соціальна мережа — це група людей, пов'язаних будь-яким типом взаємовідносини. Дружба, співпраця, романтика чи просте знайомство — це приклади соціальних взаємин, які поєднують пари людей. Коли ми говоримо про соціальну мережу, то зазвичай думаємо про певний тип взаємовідносини. Людина представляється вузлом у соціальній мережі, а взаємини подаються зв'язком між двома людьми. Отже, мережа є уявленням взаємовідносини. Це дозволяє нам говорити про взаємини, описувати їх та аналізувати на рівні, що виходить за рамки пари людей.

Існує багато різних типів соціальних мереж, і їх важливо вивчати. Медичні працівники аналізують мережі відносин, щоб знайти способи боротьби з поширенням захворювань, що передаються контактним шляхом. Економісти вивчають мережі пошуку роботи для вирішення проблеми нерівності та сегрегації на ринках праці. А вчені перевіряють мережі співавторства у наукових публікаціях, щоб виявляти впливових мислителів та ідеї.



Аналіз соціальних мереж (SNA) — це метод дослідження, який вивчає соціальні структури за допомогою теорій мереж і графів. Витоки SNA можна простежити на початку XX ст., коли соціологи почали вивчати відносини між окремими людьми та групами. Проте лише в 1960-х роках SNA набула популярності як окремий метод дослідження.

Одним із піонерів аналізу соціальних мереж був Дж. Л. Морено, психіатр і соціолог, який розробив концепцію соціограми на початку XX ст. Морено використовував соціограму як інструмент для візуального представлення стосунків між членами групи чи спільноти.

Соціограма — це графічне зображення соціальної мережі, де вузли представляють окремих осіб або групи, а посилання — відносини між ними. Соціограму

можна використовувати для визначення ключових осіб або підгруп у мережі, а також для аналізу потоку інформації та впливу в мережі.



Платформи соціальних мереж, такі як Facebook, Twitter і LinkedIn, стали багатим джерелом даних для аналізу соціальних мереж. Ці платформи дозволяють користувачам створювати та підтримувати соціальні зв'язки з іншими користувачами, що робить можливим картографування та аналіз соціальних мереж.

У детективному кіно пробкова дошка, що містить відповідні докази, з'єднані червоними нитками, часто зображується як метод для слідчих візуалізувати та відстежувати зв'язки між особами та подіями в розслідуванні. Червоний рядок використовується для фізичного з'єднання різних вузлів (часто представлених фотографіями чи нотатками) на дошці чи стіні, показуючи зв'язки між ними. У галузі теорії мереж цей метод аналізу зв'язків між людьми, організаціями чи документами відомий як аналіз зв'язків, і він використовується вже понад століття в правоохоронних органах, при виявленні шахрайства, в соціологічних дослідженнях.

Завдяки аналізу даних соціальних медіа дослідники та маркетологи можуть отримати уявлення про структуру та динаміку соціальних мереж, зокрема, про те, хто є ключовими гравцями, як інформація протікає через мережу та як спільноти формуються та розвиваються з часом.

Аналіз соціальних мереж має багато застосувань у цифровому маркетингу. Наприклад, його можна використовувати для маркетингу впливу, управління спільнотою та контент-маркетингу. Аналіз соціальних мереж може допомогти визначити впливових користувачів і лідерів громадської думки в мережі, найактивніших членів спільноти, а також теми та вміст, які найбільше резонують із цільовою аудиторією.



Аналіз мережі — це метод пошуку закономірностей у структурі мережі. Він спирається на теорію графів. Мережа складається з вузлів, які представляють окремих осіб, людей або речі, а ребра представляють зв'язок або відносини між вузлами. Аналіз соціальних мереж (SNA) цікавлять відносини між соціальними суб'єктами, такими як окремі особи чи організації.

Вузли та ребра мають різне значення в різних мережах. Наприклад, для банку вузли представляють банківські рахунки, тоді як ребра представляють транзакції між цими рахунками. Для додатків соціальних медіа, таких як Facebook і Twitter, вузли представляють окремих людей, тоді як краї вказують на наявність зв'язку між цими людьми.

Вузли та ребра разом створюють граф, а ребра можуть мати напрямку. Якщо спосіб зв'язку між вузлами незначний, то ребра не мають напрямку і граф стає неорієнтованим. У прикладі програми соціальних медіа, якщо межа існує між двома вузлами (або, іншими словами, двома людьми), це означає, що вони обидва

є друзями один одного. Тому цей зв'язок є ненаправленим. У прикладі з банком відносини є спрямованими, оскільки транзакція відбувається з одного рахунку на інший. Отже, ми проєктуємо нашу мережу з орієнтованими графами.



Граф складається з вершин (також званих вузлами), які з'єднані ребрами (або зв'язками). Є три різні співвідношення ребер:

- двійковий і значний (вага);
- симетричний і асиметричний (спрямованість) — розрізняють неорієнтовані графи, де ребра з'єднують дві вершини симетрично, і орієнтовані графи, де ребра з'єднують дві вершини асиметрично;
- одномодовий і багаторежимний.



Підграфи — підмножина вузлів у мережі разом із усіма ребрами, що з'єднують ці вузли. Компоненти — підграф мережі, від'єднаний один від одного.

Розмір мережі визначається як кількість вузлів або ребер. *Щільність мережі* обчислюється як загальна кількість існуючих ребер, поділена на кількість усіх можливих ребер. Набуває значення від 0 до 1. *Ступінь вузла* представляє кількість ребер, пов'язаних із цим вузлом. Якщо ребра мають напрямок у мережі, то кожен вузол має значення *ступеня введення та виходу* (in-degree, out-degree). In-degree представляє вхідні ребра до вузла, тоді як out-degree представляє вихідні ребра з вузла. Розподіл ймовірностей цих ступенів усіх вузлів створює розподіл ступенів, і соціальні мережі можна охарактеризувати за допомогою цієї статистики.

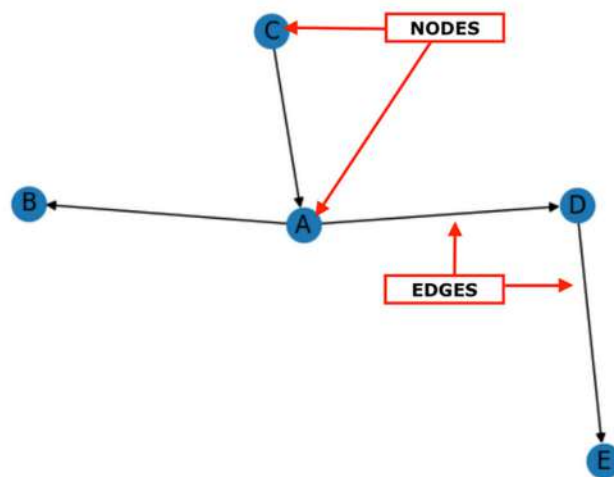


Рис. 2.4.1. Приклад графа, де позначені вузли та ребра

2.4.2. Основні поняття соціальних мереж

Розглянемо основні з цих термінів:

- *Актор*. Соціальні мережі представляють відносини між соціальними суб'єктами. Кожен соціальний суб'єкт називається актором. Кожна сутність може бути як одного типу, так і різних типів.
- *Відносини*: зв'язки — це зв'язки між акторами;
- *Діада, триада та підгрупа*. пара існуючих або можливих двох акторів називається діадами, тоді як підмножина з трьох існуючих акторів називається триадами. Узагальнюючи цю логіку, підмножина будь-якої підмножини акторів називається підгрупою.
- *Група* — це сукупність усіх акторів.
- *Відношення* — це сукупність зв'язків у конкретній групі.

Метрики підключення у соціальних мережах. Кореляції спостерігаються між вузлами в різних моделях. Наступні метрики допомагають розшифрувати значення зв'язків між вузлами:

- *Транзитивність* (Transitivity) демонструє кількість відповідного відношення для двох вузлів, що з'єднуються ребром. У соціальних мережах, коли ми бачимо акторів як людей, це вказує на ймовірність того, що друзі двох людей стануть друзями.
- *Взаємність* (Reciprocity) показує схильність актора, з яким актор пов'язаний, створити зв'язок із самим собою (іншими словами, відповідати один одному).
- *Асортативність* (Assortativity) вказує на схильність актора зв'язуватися з акторами, подібними до себе за ступенем.
- Гомофілія показує тенденцію до утворення зв'язків між подібними акторами.



Міри центральності. Для використання інформації вузлів (акторів у соціальних мережах) у мережі використовуються різноманітні заходи центральності. Потрібно відповісти на запитання, які вузли важливіші, а іноді і який з них найважливіший.

Є різні міри центральності:

- *ступінь центральності* вузла означає кількість ребер, з'єднаних із цим вузлом;
- *центральність втрати* — це міра того, наскільки вузол розташований близько до інших вузлів у наборі вузлів;
- *центральність між ними* визначається шляхом вимірювання частки шляхів, які з'єднують усі пари вузлів і включають вузол, який цікавить аналітика;
- *центральність власного вектору* визначає центральність вузла, дивлячись не тільки на кількість ребер, пов'язаних з цим вузлом, а й також на його якість.

Мережі мають різні властивості, і вони можуть давати уявлення не лише про два вузли, а й про взаємозв'язки в межах однієї невеликої частини мережі або про всю мережу.

Дослідження в мережах можна розділити на три різні рівні абстракції:

1. Аналіз на рівні елементів досліджує важливість окремих вузлів і ребер.
2. Аналіз на рівні групи визначає згуртованість і щільність груп у мережі.
3. Аналіз мережевого рівня цікавить топологічні властивості мережі.

Одним з аспектів, який робить SNA таким потужним інструментом, є можливість візуалізувати ці зв'язки на графіку, використовуючи вузли для представлення окремих осіб і ребра для представлення зв'язків між ними. Візуалізація індивідів і стосунків дозволяє легше відчувати динаміку соціального впливу, формування соціальних груп і потік інформації між групами та індивідами.

Інструменти аналізу соціальних мереж:

- **Gephi** — це безкоштовне програмне забезпечення з відкритим вихідним кодом, яке використовується для мережевого аналізу та візуалізації, написане за допомогою Java та Netbeans. Додаток надає інструменти, за допомогою яких можна аналізувати мережі, отримувати статистичну інформацію, завантажувати дані вручну, а також завантажувати дані через API.
- **NetworkX** — це бібліотека Python для створення, керування та вивчення складних мереж. Може працювати з мережами з мільйонами вузлів і ребер, а також надає функції для створення випадкових мереж, обчислення мережових показників і візуалізації мережових структур; також має широкий спектр алгоритмів для виявлення спільнот, прогнозування посилянь і візуалізації мереж.

Щоб продемонструвати, давайте створимо просту мережу на Python за допомогою NetworkX, щоб представити симетричний зв'язок між старостою та іншими студентами групи.

```
import networkx as nx
import matplotlib.pyplot as plt
# створити новий граф
G_symmetric = nx.Graph()

# додати ребра та вузли
G_symmetric.add_edge('Глеб Сахань', 'Боярчук Віталій')
G_symmetric.add_edge('Глеб Сахань', 'Кононенко Влад')
G_symmetric.add_edge('Глеб Сахань', 'Єфімова Анастасія')

# намалювати графік
fig = plt.figure(figsize=(12, 8))
nx.draw_networkx(G_symmetric, font_size=10)
plt.show();
```

Цей код виведе мережевий графік, який виглядає так (рис. 2.4.2).

Опис соціальних мереж з центром. Одним із критеріїв, який використовується для опису окремих осіб і стосунків у мережі, є центральність.

Центральність — це міра важливості або впливу вузлів у мережі. Дві міри центральності включають ступінь центральності та центральність між вузлами.

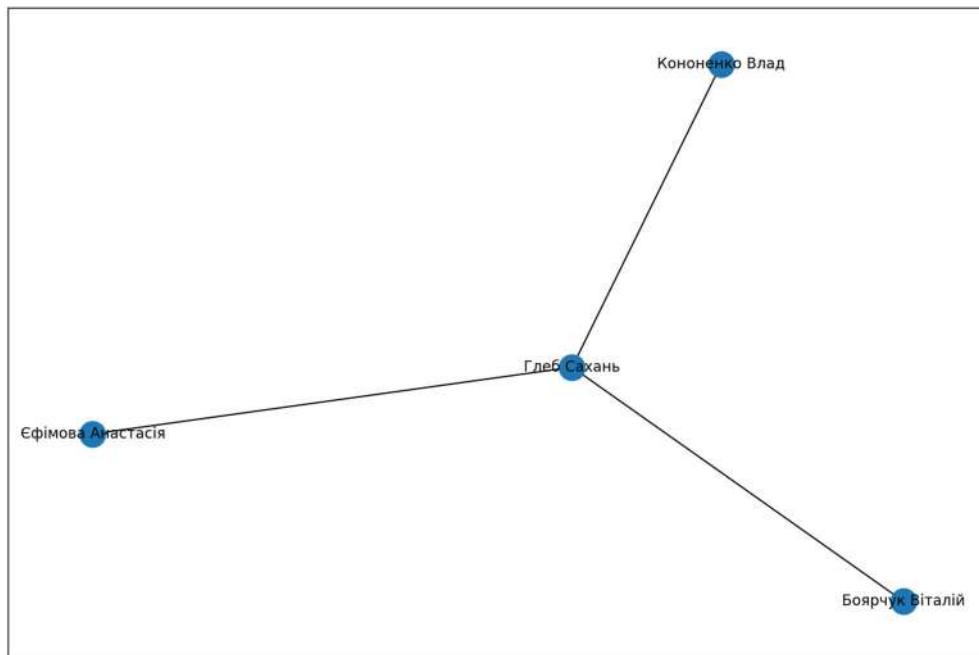


Рис. 2.4.2. Графічне відображення мережі



Ступінь центральності — це міра важливості вузла в мережі на основі кількості з'єднань, які він має. Вузли з високим ступенем центральності вважаються більш впливовими або центральними в мережі.

Ступінь центральності можна розрахувати шляхом ділення кількості з'єднань, які має вузол, на максимально можливу кількість з'єднань у мережі. Ступінь центральності 1 для вузла в мережі означає, що вузол безпосередньо підключений до всіх інших вузлів у мережі.

Центральність між вузлами вимірює, наскільки важливий кожен вузол у мережі на основі того, як часто він з'являється на найкоротшому шляху між парами інших вузлів. Центральність дозволяє нам оцінити, які вузли в мережі вважаються більш впливовими, оскільки вони відіграють вирішальну роль у з'єднанні різних частин мережі.

Центральність між ними можна розрахувати шляхом підрахунку кількості найкоротших шляхів, які проходять через вузол, і ділення їх на загальну кількість найкоротших шляхів у мережі.

Розрахунок центральності за допомогою NetworkX. Використаємо графічний об'єкт, створений за допомогою NetworkX, для обчислення ступеня центральності та центральності між вузлами в мережі:

```

import networkx as nx
import pandas as pd
import matplotlib.pyplot as plt

# створити новий граф
G_symmetric = nx.Graph()

# додати ребра та вузли
G_symmetric.add_edge('Глеб Сахань', 'Боярчук Віталій')
G_symmetric.add_edge('Глеб Сахань', 'Кононенко Влад')
G_symmetric.add_edge('Глеб Сахань', 'Єфімова Анастасія')

# обчислити центральність для nx
degree centrality = nx.degree_centrality(G_symmetric)
betweenness centrality = nx.betweenness_centrality(G_symmetric)

# визначте мітки вузлів
nodes = ["Глеб Сахань", "Боярчук Віталій", "Кононенко Влад", "Єфімова Анастасія"]
# отримати ступінь центральності для кожного вузла
dc_calculations = []
bc_calculations = []

for node in nodes:
    dc_calculations += [degree_centrality[node]]
    bc_calculations += [betweenness_centrality[node]]

# створити df для зберігання значень
centrality_data = pd.DataFrame()

# заповнити df вузлами та значеннями їх центральності
centrality_data["Вузли"] = nodes
centrality_data["Ступінь центральності"] = dc_calculations
centrality_data["Між-Центральність"] = bc_calculations
print (centrality_data)

```

У наведеному коді демонструється, як використовувати графічний об'єкт, створений за допомогою NetworkX, для обчислення ступеня центральності та центральності між вузлами в мережі.

У цьому коді згенеровано графічний об'єкт, який представляє симетричний зв'язок між nx Глебом Саханем і студентами групи ІА-301. Потім метод `degree_centrality` застосовується для створення словника, де кожен вузол і його ступінь центральності зберігаються як пари ключ-значення. Ми повторюємо цю процедуру з `betweenness_centrality`, щоб створити словник, який включає кожен вузол nx і розрахунок центральності між ними. Результат наведено на рис. 2.4.3.

	Вузли	Ступінь центральності	Між-Центральність
0	Глеб Сахань	1.000000	1.0
1	Боярчук Віталій	0.333333	0.0
2	Кононенко Влад	0.333333	0.0
3	Ефімова Анастасія	0.333333	0.0

Рис. 2.4.3. Словник групи ІА-301

Інтерпретація результатів. Глеб Сахань має ступінь центральності 1, що означає, що він має максимальну кількість зв'язків з іншими студентами групи. Це узгоджується з даними і тим, як було побудовано мережу, щоб продемонструвати симетричні стосунки між Глебом Саханем і кожним із його колег по групі. Оскільки Глеб Сахань з'явився в кожному кортежі, він матиме посилання на будь-який інший вузол.

З другого боку, інші однокласники з'являються лише один раз. Щоб відновити увагу, центральність ступеня обчислюється так:

$$\text{degree}(\text{node}) / (\text{total nodes} - 1)$$

Оскільки ступінь входить до кількості ребер, з'єднаних з вузлом, кожен напарник, який з'єднаний з Глебом Саханем ребром, додасть ще один ступінь.

Якщо $\text{degree}(\text{node})=3$ і $(\text{total nodes}-1)=(4-1)=3$ тоді $\text{degree centrality}=3/3=1$.

Ми можемо застосувати розрахунок подібним чином до кожного учасника групи. Оскільки кожен з них має один ступінь, то $1/3 = 0.333$.

Інтерпретація міжцентральності. Центральність для вузла обчислюється як сума частки всіх найкоротших шляхів між парами інших вузлів у мережі, які проходять через цей вузол, поділена на загальну кількість найкоротших шляхів між цими парами вузлів.

У даному прикладі це не має особливого сенсу, оскільки у нас невелика мережа, і ми лише візуалізували стосунки старости та трьох студентів, а не стосунки з іншими студентами в групі та один з одним (або будь-якими іншими групами).

Але як дійти до центральної ролі Глеба Саханя в цьому сценарії? Центральність 1 для вузла означає, що вузол лежить на всіх найкоротших шляхах між парами інших вузлів у мережі. Іншими словами, він служить критичним мостом, або з'єднувачем, між різними частинами мережі.

Коли вузол має центральність між вузлами, рівну 1, це означає високий рівень впливу та контролю над потоком інформації або ресурсів у мережі. Будь-яка інформація або взаємодія, що проходить через мережу, ймовірно, пройде через цей вузол. І тому його можна розглядати як ключове вузьке місце або посередника у комунікації мережі.

Вузли з високою центральністю між ними часто вважаються необхідними з точки зору динаміки мережі, оскільки вони можуть контролювати розповсюдження інформації, впливати на процеси прийняття рішень або діяти як важливі з'єднувачі для ефективного зв'язку в мережі.

Важливо відзначити, що інтерпретація центральності між собою також повинна враховувати контекст і специфічні характеристики мережі, що аналізується. Значимість оцінки центральності між проміжками 1 може змінюватися залежно від розміру, структури та призначення мережі.

2.4.3. Модельна задача. Мережевий аналіз «Гри престолів».



«Гра престолів» — це надзвичайно популярний телесеріал від НВО, заснований на циклі книг «Пісня льоду й полум'я» Джорджа Р. Р. Мартіна. Набір даних містить серію книг «Пісня льоду та полум'я» з першої по п'яту. Ми також використовуватимемо цей набір даних для аналізу спільної мережі персонажів у книгах «Гри престолів».

Перш за все, просто завантажимо дані першої книги «Гри престолів», щоб перевірити назви стовпців.

Код:

```
import pandas as pd
book1 = pd.read_csv('datasets/book1.csv')
book1.head()
```

1	Source	Target	Type	weight	book
2	Addam-Marbrand	Jaime-Lannister	Undirected	3	1
3	Addam-Marbrand	Tywin-Lannister	Undirected	6	1
4	Aegon-I-Targaryen	Daenerys-Targaryen	Undirected	5	1
5	Aegon-I-Targaryen	Eddard-Stark	Undirected	4	1
6	Aemon-Targaryen-(Maester-Aemon)	Alliser-Thorne	Undirected	4	1
7	Aemon-Targaryen-(Maester-Aemon)	Bowen-Marsh	Undirected	4	1
8	Aemon-Targaryen-(Maester-Aemon)	Chett	Undirected	9	1
9	Aemon-Targaryen-(Maester-Aemon)	Clydas	Undirected	5	1
10	Aemon-Targaryen-(Maester-Aemon)	Jeor-Mormont	Undirected	13	1
11	Aemon-Targaryen-(Maester-Aemon)	Jon-Snow	Undirected	34	1
12	Aemon-Targaryen-(Maester-Aemon)	Samwell-Tarly	Undirected	5	1
13	Aerys-II-Targaryen	Brandon-Stark	Undirected	4	1
14	Aerys-II-Targaryen	Eddard-Stark	Undirected	10	1
15	Aerys-II-Targaryen	Gerold-Hightower	Undirected	3	1

Рис. 2.4.4 Структура DataFrame book1

Отриманий DataFrame **book1** має 5 стовпців: **Source, Target, Type, weight** і **book**. Джерело та ціль — це два вузли, які з'єднані ребром. Атрибут ваги кожного ребра говорить про кількість взаємодій, які персонажі мали протягом книги.

Якщо вже є дані для створення мережі, використаємо бібліотеку Networkx для мережевого аналізу та створимо графічний об'єкт для першої книги.

Код:

```
import networkx as nx
G_book1 = nx.Graph()
```

Наразі об'єкт графіка G_book1 порожній. Таким чином, потрібно заповнити його ребрами book1 та завантажити решту книг.

Код:

```
# Створення списку мереж для всіх книг
books = [G_book1]
book_fnames = ['datasets/book2.csv', 'datasets/book3.csv',
'datasets/book4.csv', 'datasets/book5.csv']
for book_fname in book_fnames:
    book = pd.read_csv(book_fname)
    G_book = nx.Graph()
    for _, edge in book.iterrows():
        G_book.add_edge(edge['Source'], edge['Target'],
weight=edge['weight'])
    books.append(G_book)
```

Ступінь центральності, щоб знайти найважливішого персонажа. По-перше, використаємо ступінь центральності, щоб виміряти важливість вузла в мережі, подивившись на кількість сусідів і кількість вузлів, до яких він підключений.

Код:

```
# Розрахунок центрального ступеня книги 1 і книги 5
deg_cen_book1 = nx.degree_centrality(books[0])
deg_cen_book5 = nx.degree_centrality(books[4])
# Сортуння словників за ступенем центральності та збереження перших 10
sorted_deg_cen_book1 = sorted(deg_cen_book1.items(), key=lambda x:
x[1], reverse=True)[0:10]
('Eddard-Stark', 0.3548387096774194)
sorted_deg_cen_book5 = sorted(deg_cen_book5.items(), key=lambda x:
x[1], reverse=True)[0:10]
('Jon-Snow', 0.1962025316455696)
```

Згідно зі ступенем центральності розуміємо, що важливість персонажів змінюється з часом, тому треба перевірити еволюцію ступеня центральності кількох персонажів.

Код:

```
%matplotlib inline
evol = [nx.degree_centrality(book) for book in books]
degree_evol_df = pd.DataFrame.from_records(evol)
degree_evol_df[['Eddard-Stark', 'Tyrion-Lannister', 'Jon-Snow']].plot()
```

Побудувавши лінійну діаграму (рис. 2.4.5), бачимо, що важливість Еддарда Старка зменшується в міру розвитку серії книг. З Джоном Сноу спостерігається падіння в четвертій книзі, але раптове зростання в п'ятій.

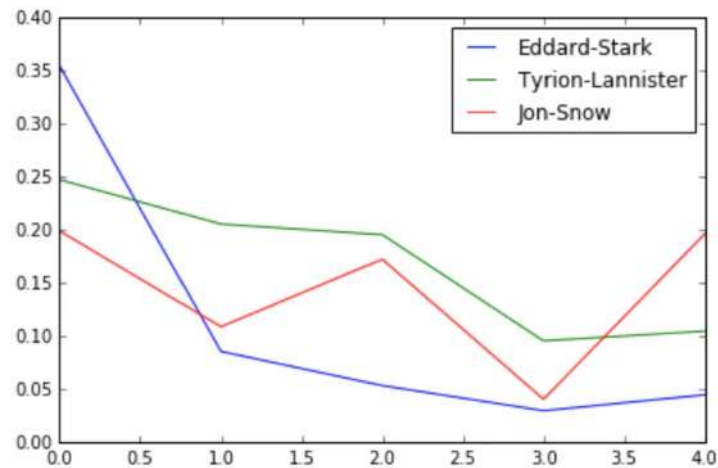


Рис. 2.4.5. Лінійна діаграма персонажів

Знайти найважливішого персонажа. Далі впроваджуємо міжцентральність, щоб знайти важливих персонажів у списку персонажів «Гри престолів» і візуалізуємо еволюцію центральності цієї мережі протягом п'яти книг.

Код:

```
evol = [nx.betweenness_centrality(book, weight='weight') for book in books]
betweenness_evol_df = pd.DataFrame.from_records(evol).fillna(0)
# Пошук 4 топ персонажів у кожній книзі
set_of_char = set()
for i in range(5):
    set_of_char |=
    set(list(betweenness_evol_df.T[i].sort_values(ascending=False)[0:4].
    index))
list_of_char = list(set_of_char)
betweenness_evol_df[list_of_char].plot(figsize=(13,7))
```

Бачимо на рис. 2.4.6 особливе зростання важливості Станіса Баратеона в міру розвитку серії книг. У п'ятій книзі він значно важливіший за інших персонажів мережі.

PageRank — щоб знайти найважливішого персонажа. Тепер спробуємо PageRank, щоб відшукати важливих персонажів у циклі «Гри престолів» і побудувати еволюцію PageRank цієї мережі за п'ятьма книгами.

PageRank (PR) — це алгоритм, який використовується пошуком Google для ранжування вебсторінок у результатах, отриманих пошуковою системою, та спосіб вимірювання важливості сторінок вебсайту. PageRank оцінює вхідні та вихідні посилання вебсторінок Інтернету (спрямована мережа).

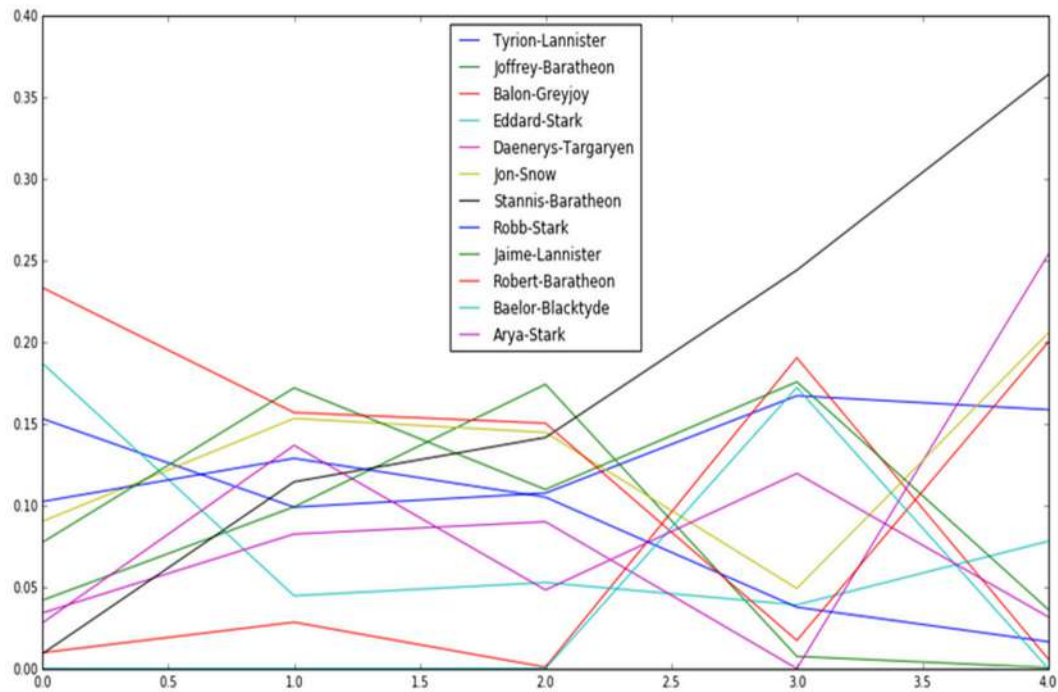


Рис. 2.4.6. Еволюція центральності персонажів

Код:

```
evol = [nx.pagerank(book) for book in books]
pagerank_evol_df = pd.DataFrame.from_records(evol)
set_of_char = set()
for i in range(5):
    set_of_char |=
set(list(betweenness_evol_df.T[i].sort_values(ascending=False)[0:4].
index))
list_of_char = list(set_of_char)
pagerank_evol_df[list_of_char].plot(figsize=(13,7))
```

Отже, за рейтингом PageRank, Станніс Баратеон і Джон Сноу — найважливіші персонажі п'ятої книги (рис. 2.4.7).

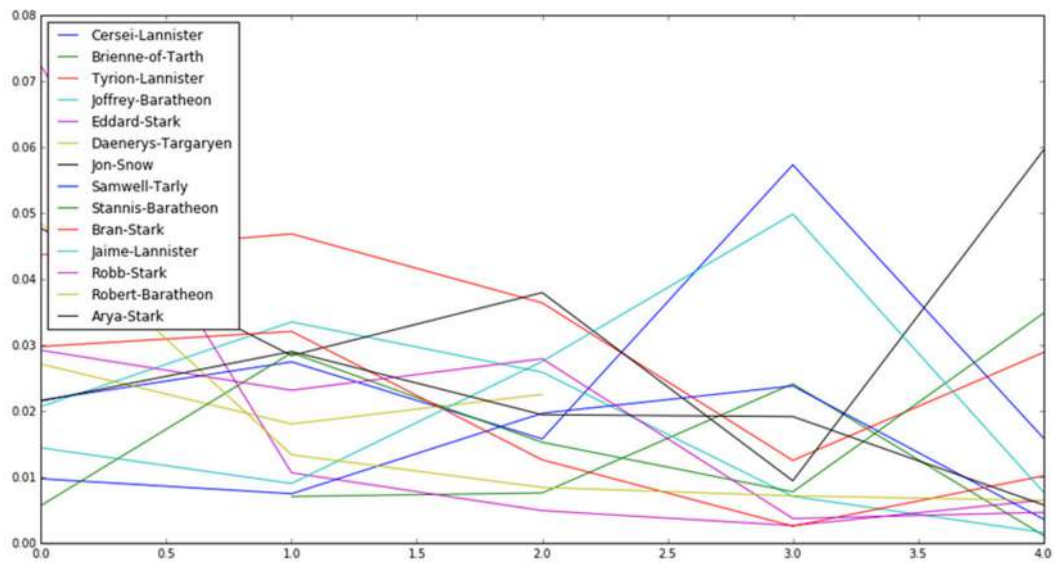


Рис. 2.4.7. Рейтинг PageRank

Кореляція між різними показниками. Ми бачили три різні показники для обчислення важливості вузла в мережі, але хочемо побачити кореляцію між рейтингом PageRank, між центральністю та ступенем центральності за допомогою кореляції Пірсона (рис.2.4.8).

Код:

```
measures = [nx.pagerank(books[4]),
            nx.betweenness_centrality(books[4], weight='weight'),
            nx.degree_centrality(books[4])]
cor = pd.DataFrame.from_records(measures)
cor.T.corr()
```

	0	1	2
0	1.000000	0.793372	0.971493
1	0.793372	1.000000	0.833816
2	0.971493	0.833816	1.000000

Рис. 2.4.8. Кореляція Пірсона

Можемо зробити висновок, що існує висока кореляція між цими трьома показниками для мережі персонажів.

```
p_rank, b_cent, d_cent = cor.idxmax(axis=1)
print(p_rank)      Jon-Snow
print(b_cent)      Stannis-Baratheon
print(d_cent)      Jon-Snow
```

Рис. 2.4.9. Підсумки аналізу



За ступенем центральності і PageRank, Джон Сноу є найважливішим персонажем у п'ятій книзі, а з огляду на центральність, у п'ятій книзі Станіс Баратеон значно важливіший за інших персонажів у мережі книг.

Висновки



Кілька базових визначень та термінів, які дозволяють описувати мережу:

- Мережа складається з двох множин елементів: вузлів та зв'язків, що з'єднують пари вузлів.
- Підмережа — це підмножина мережі, що включає кілька її вузлів та всі зв'язки між ними.
- У спрямованих мережах зв'язку мають напрямок. Може існувати зв'язок від вузла 1 до вузла 2 і не обов'язково з вузла 2 до вузла 1.
- У ненаправлених мережах зв'язку є взаємними.
- У зважених мережах зв'язку мають асоційовані ваги, які представляють атрибути зв'язку, такі як важливість, подібність, відстань, трафік і та ін.
- Багатошарові мережі мають різні типи вузлів та зв'язків, поділених на взаємопов'язані шари. Якщо у кожному шарі вузли однакові, то така багатошарова мережа називається мультиплексною.
- Щільність мережі — частка пар з'єднаних вузлів. Мережа є повною, якщо всі пари вузлів з'єднані, внаслідок чого щільність дорівнює одиниці. Більшість реально існуючих мереж є розрідженими, а отже, вони мають дуже малу щільність.
- Ступінь вузла — це кількість сусідів. У спрямованих мережах вузли мають «ступінь-на-вході» (in-degree) і «ступінь-на-виході» (out-degree), що вимірюють відповідно кількість вхідних та вихідних зв'язків. Якщо мережа є зваженою, то сила вузла

дорівнює сумі ваги її зв'язків. Вузли зважених спрямованих мереж мають «силу-на-вхід» (in-strength) і «силу-на-вихід» (out-strength).

- Списки суміжності та списки ребер є ефективними уявленнями, що служать для зберігання розріджених мереж.
- NetworkX — це популярна та зручна програмна бібліотека для програмування мереж на Python.

У великих мережах потрібно використовувати статистичні інструменти аналізу глобальних ознак мережі. Гістограма забезпечує наочну ілюстрацію розподілу значень даного атрибуту вузлів чи зв'язків (наприклад, ступеня). Нормалізована гістограма являє собою оцінку ймовірнісного розподілу значень метрики, що нас цікавить.

Запитання для самоконтролю



1. Що таке мережа?
2. Які бувають типи мереж?
3. Що таке щільність мережі?
4. Назвіть метрики соціальних мереж
5. З чого складається аналіз соціальних мереж (SNA)?
6. Які засоби використовуються для аналізу соціальних мереж?

2.5. Аналітика природної мови



Аналітика тексту. Останніми роками, з появою великих даних і величезної кількості текстових даних, що надходять з Інтернету, було розроблено багато методів аналізу тексту. Насправді таку форму даних дуже важко аналізувати, але в той же час вона є джерелом великої кількості корисної інформації, також враховуючи величезну доступність даних.



Згадайте лише всю створену літературу — наприклад, численні публікації, опубліковані в Інтернеті. Коментарі в соціальних мережах і чатах також можуть бути чудовим джерелом даних, особливо для розуміння ступеня схвалення чи несхвалення певної теми. Тому аналіз текстів став дуже актуальним, і для нього було розроблено багато методів, фактично створивши нову дисципліну.

2.5.1. Методи аналітики тексту

Для попередньої обробки тексту:

- перетворення регістру (верхній, нижній);
- токенизація слів і речень;
- зняття розділових знаків;
- видалення стоп-слів;
- стемінг (процес скорочення слова до основи шляхом відкидання допоміжних частин, таких як закінчення чи суфікс);
- лематизація (розбиття слова на його кореневе значення для виявлення подібності).

Для аналізу тексту:

- аналіз частотного розподілу слів;
- розпізнавання образів та об'єктів (вилучення тексту);
- тегування;
- аналіз зв'язків і асоціацій;
- аналіз настроїв.

Одним із найпростіших і основних прикладів для аналізу тексту є обчислення частоти слів, що містяться в ньому.

Попередня обробка тексту є одним із найважливіших і фундаментальних етапів аналізу тексту. Зібравши текст для аналізу з різних доступних джерел, ви незабаром зрозумієте, що для того, щоб використовувати текст для аналізу, його необхідно очистити, трансформувати, а потім підготувати спеціально для використання.

Перетворення регістру є, мабуть, найпоширенішою операцією не лише на етапі попередньої обробки, але й на наступних етапах аналізу. Насправді текст

містить багато слів, у яких деякі символи написані з великої літери. Більшість методів синтаксичного аналізу вимагають, щоб усі слова були написані в нижньому регістрі, щоб «слово» і «слово» вважалися одним словом.

Стоп-слова — це слова, які не мають великого значення в аналізі та повинні бути відфільтровані. Немає загального правила визначення того, чи є слово стоп-словом (підлягає видаленню) чи ні.

Біграми та колокації. Іншим основним елементом аналізу тексту є розгляд пар слів (біграм) замість окремих слів. Слова «є» і «жовтий» є, наприклад, біграмою, оскільки їх поєднання можливе і має значення. Тож «жовтий» можна знайти в текстових даних. Ми всі знаємо, що деякі з таких біграм настільки поширені в нашій літературі, що майже завжди використовуються разом. Приклади: «швидке харчування», «зверніть увагу», «доброго ранку» тощо. Такі біграми називаються колокаціями.

Токенізація слів є ще однією дуже поширеною операцією в обробці текстів. Текст складається зі слів із пробілами між ними. Тому операція перетворення тексту в список слів є фундаментальною для того, щоб мати можливість обчислювати текст.

Іншою поширеною операцією попередньої обробки є видалення знаків пунктуації. Часто доводиться надсилати на аналіз коментарі, взяті з соціальних мереж або огляди продуктів в Інтернеті. Багато з цих текстів багаті на знаки пунктуації, які компрометують правильну токенизацію слів, що містяться в них.

Інший тип операцій попередньої обробки включає лінгвістику. Такі операції, як стемінг та лематизація, діють на окремих словах, оцінюючи їх лінгвістичний корінь (у першому випадку) та лему (у другому випадку). Потім визначення кореня групує всі слова, що мають один і той самий корінь, вважаючи їх одним словом. Лематизація натомість шукає всі відмінювані форми слова чи дієслова та групує їх під тією самою лемою, вважаючи їх усіх одним словом.



Обробка природної мови (NLP) — це галузь штучного інтелекту, яка забезпечує основу для комп'ютерів для розуміння та інтерпретації людської мови.

Обробка природної мови використовується для полегшення аналізу неструктурованих даних, наприклад тексту з електронних листів та інших документів, або голосових даних, зібраних за допомогою таких продуктів, як Amazon Alexa та Google Home.

Величезна кількість зібраних неструктурованих даних форматується як текстові документи, як-от електронні листи, транскрибовані виступи тощо. Обробка природної мови допомагає таким обчислювальним системам, як штучний інтелект, зрозуміти, як ці дані слід аналізувати та класифікувати, спрощуючи кількість людського втручання, необхідного для виділення корисної інформації з даних.

NLP перетворює мову в числові представлення, які використовуються алгоритмами машинного навчання. Наприклад, машинне навчання виконує математичні операції з числовими даними, щоб знаходити зв'язки та створювати прогнози.

Оскільки текст не є числовим, математичні операції, необхідні для машинного навчання, неможливо виконати, доки ми не перетворимо текст. Тоді завданням для NLP стає пошук способу зберегти значення мови, одночасно представляючи його в числовому форматі. Дослідники виявили, що перетворення слів у вектори чисел добре працює для кодування значення в мові, створюючи найсучасніші результати в завданнях моделювання мови.

Іншою формою обробки та аналізу даних є процес відбору слів, що містяться в тексті, на основі конкретних характеристик. Наприклад, вас може зацікавити виділення слів на основі їх довжини.



Аналіз настроїв — це нова галузь дослідження, яка розвинулась зовсім недавно для оцінки думок людей щодо певної теми. Ця дисципліна базується на різних техніках, які використовують аналіз тексту та його сферу роботи у світі соціальних медіа та форумів (видобуток думок).

Завдяки коментарям і відгукам користувачів алгоритми аналізу настроїв можуть оцінювати ступінь вдячності або оцінки на основі певних ключових слів. Цей рівень оцінки називається думкою і має три можливі значення: позитивне, нейтральне або негативне. У такий спосіб оцінка цієї думки стає формою класифікації.

Аналіз настроїв використовує NLP, щоб визначити, чи мова відображає позитивні чи негативні емоції. Наприклад, біржовий трейдер може використовувати аналіз настроїв, щоб проаналізувати те, що люди говорять і як вони ставляться до компанії в Інтернеті. Якщо загальні настрої хороші, трейдер може купити акції компанії. Деякі з найбільших інвестиційних компаній у світі відстежують настрої в соціальних мережах, щоб зрозуміти, як трейдери можуть діяти на ринку.

Моделювання тем використовує NLP для аналізу текстового корпусу та узагальнення його, розбиваючи його на відповідні теми. Це звичайне завдання, яке виконується під час видобутку великих масивів тексту. Моделювання тем може зменшити обсяги тексту до списку тем, виявляючи семантичні структури, які людям важко виявити.

Вилучення тексту — це широко використовуваний метод обробки природної мови, який автоматично виявляє певну інформацію в тексті, також відомий як розпізнавання іменованих об'єктів. Розпізнавання іменованих об'єктів можна використовувати для отримання ключових слів, імен, місць, компаній і конкретних фраз із великих пакетів даних, щоб визначити тенденції та знайти корисну інформацію. Його також можна використовувати для оптимізації операцій, налашту-

вавши автоматичні тригери, які вводять отримані дані в бази даних або отримують дані про певних клієнтів у службі підтримки клієнтів.

Чат-боти і віртуальні помічники використовують NLP, щоб автоматично розуміти та надавати належні відповіді на запити користувачів за допомогою генерації природної мови, яка відповідає заздалегідь визначеним правилам. Сьогодні чат-боти використовуються, щоб відповісти в середньому на 80 відсотків рутинних запитань клієнтів.

Якщо програмуєте на Python і хочете аналізувати дані в текстовій формі, одним із найбільш часто використовуваних інструментів на даний момент є Python Natural Language Toolkit (NLTK). NLTK — це не що інше, як бібліотека Python (www.nltk.org), в якій є багато інструментів, що спеціалізуються на обробці та аналізі текстових даних. NLTK був створений у 2001 році з освітньою метою, потім з часом розвинувся настільки, що став справжнім інструментом аналізу.

Розробимо проект текстової аналітики з використанням NLP і класифікації.

Імпорт основних бібліотек. Python має великий арсенал засобів для аналітики текстів. По-перше, давайте імпортуємо бібліотеки `pandas`, `numpy`, `matplotlib`, `seaborn`, `regex` і `random`, які досить широко використовуються.

Код:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random
import regex as re
# нижче код призначений для того, щоб не відображати жодних попереджень, які можуть з'явитися для поточних удосконалень у функціях
import warnings
warnings.filterwarnings('ignore')
```

Для прикладу використаємо набір даних про огляди фільмів із Kaggle. Набір даних фільмів — це набір даних, який найчастіше використовують для вивчення класифікації настроїв.

Код:

```
# Зчитування файлу csv у фрейм даних
df = pd.read_csv('movie.csv')

# Перші десять записів
df.head(10)
# якщо параметр не надано, функція head() покаже 5 записів
```

Результат ми бачимо на рис. 2.5.1.

Ми бачимо в наборі даних, що настрої, позначені 1, є позитивними, а ті, що мають 0, — негативними.

	text	label
0	I grew up (b. 1965) watching and loving the Th...	0
1	When I put this movie in my DVD player, and sa...	0
2	Why do people who do not know what a particula...	0
3	Even though I have great interest in Biblical ...	0
4	Im a die hard Dads Army fan and nothing will e...	1
5	A terrible movie as everyone has said. What ma...	0
6	Finally watched this shocking movie last night...	1
7	I caught this film on AZN on cable. It sounded...	0
8	It may be the remake of 1987 Autumn's Tale aft...	1
9	My Super Ex Girlfriend turned out to be a plea...	1

Рис. 2.5.1. Набір даних огляду фільмів

Проаналізуємо кількість позитивних і негативних відгуків.

Код:
`df['label'].value_counts()`

Є 4318 негативних коментарів і 4170 позитивних коментарів. Оскільки присутність обох типів зворотного зв'язку майже однакова, ми не будемо виконувати вибірку в наборі даних. Однак ми можемо отримати більшу точність, якщо виконаємо вибірку. Давайте побудуємо візуалізацію настроїв (рис. 2.5.2.).

Код:
`sns.countplot(x = 'label', data = df)`
`plt.xlabel('Sentiments')`
`plt.show()`

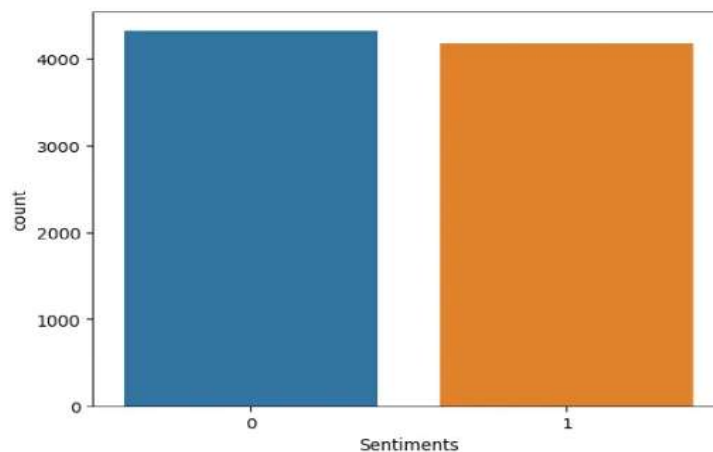


Рис. 2.5.2. Візуалізація настроїв, набір даних огляду фільмів

2.5.2. Обробка тексту

Коли ми використовуємо текстові дані, ми не маємо функцій, які ми бачили б у структурованих табличних даних. Отже, нам потрібно застосування деяких заходів, щоб отримати функції з текстових даних. Можна взяти кожне слово з речення та отримати якийсь вимір, за допомогою якого зможемо дізнатися, чи існує слово в іншому реченні чи ні, а також його важливість? Це можливо завдяки процесу, який називається моделлю «торби зі словами» (Bag Of Word Model, BoW). Іншими словами, кожне речення в наборі даних перегляду фільмів розглядається як пакет слів, тому кожне речення називається документом. Усі документи разом складають корпус.

Спочатку створюємо словник усіх унікальних слів, які використовуються в корпусі (що означає всі документи чи відгуки, присутні в наборі даних). Не беремо до уваги граматику, а також слова «як-от», «це», «є» тощо, коли враховуємо слова, оскільки це не має істотного значення для розуміння контексту тексту. Потім конвертуємо всі документи (окремі відгуки) у вектори, які відображатимуть наявність слів із нашого словника в конкретних документах. Існує три способи визначення важливості слів у моделі BoW:

- кількісна векторна модель (Count Vector Model);
- векторна модель частоти термінів (Term Frequency Vector Model) — tf;
- частотно-інверсна модель частоти термінів документа (Term Frequency-Inverse Document Frequency Model) — tf-idf.

Кількісна векторна модель підраховує кількість появ слів у повному реченні. Це краще зрозуміти візуально, тож припустімо, що ми маємо наведені нижче твердження:
review1 = «Фільм був дуже-дуже хорошим»;
review2 = «Фільм розчарував».

У векторній моделі підрахунку відгуки відображатимуться як такі (рис. 2.4.3).

	Documents	Movie	was	very	good	disappointing
1	Movie was very very good	1	1	2	1	0
2	Movie was disappointing	1	1	0	0	1

Рис. 2.4.3. Кількісна векторна модель

Векторна модель частоти термінів — у цьому випадку частота кожного слова обчислюється в кожному документі (або реченні) відносно загальної кількості слів, які спостерігаються в повному документі. Це визначається як:

$$TF = \frac{\text{кількість входжень слова в } i \text{ — му документі}}{\text{загальна кількість слів в } i \text{ — му документі}}$$

Частотно-інверсна модель частоти термінів документа (tf-idf) — вимірює важливість слова в конкретному реченні. Важливість слова в реченні прямо пропо-

рційна його кількості в документі та обернено пропорційна частоті того самого слова в повному корпусі, і визначається як:

$$TF - IDF = TF \ln \left(\frac{1+N}{N_i} \right),$$

де N — загальна кількість документів в наборі, а N_i — документ, який містить слово « i ».

Підрахуємо, скільки разів кожне слово з'являється в кожному документі за допомогою функції `CountVectorizer`.

Код:

```
from sklearn.feature_extraction.text import CountVectorizer

#ініціалізація CountVectorizer
count_vector = CountVectorizer()

# створення словника слів із корпусу
features = count_vector.fit(df['text'])

# перегляд назв функцій, витягнуті CountVectorizer
feature_names = features.get_feature_names_out()
feature_names
print('Total Number of features extracted are - ',len(feature_names))
```

Результат виконання коду наведено на рис. 2.5.4.

```
array(['00', '000', '001', ..., 'über', 'ünfaithful', 'zimjewski'],
      dtype=object)

Total Number of features extracted are - 48618
```

Рис. 2.5.4. Кількість слів у кожному документі

Це означає наявність 48 618 унікальних об'єктів у нашому наборі даних. Це тому, що не було очищено тексти від непотрібних слів. Дану операцію необхідно зробити.

Код:

```
# Випадковим чином виберемо з нього 10 назв об'єктів
random.sample(set(feature_names), 10)

['discirnable',
 'slough',
 'gonzales',
 'hermandad',
 'lilia',
 'cartilage',
 'aau',
 'faboulous',
 'tings',
 'sept']
```

Рис. 2.5.5. Назви об'єктів

Як бачимо, це назви об'єктів, а не їх векторна форма. Ми повинні перетворити об'єкти у векторну форму.

Код:

```
feature_vector.getnnz()
1158500
# To get the non-zero value density in the document
feature_vector.getnnz()/(feature_vector.shape[0]*feature_vector.shape[1])
0.0028073307190965642
```

Так менше ненульових значень у векторі ознак, більшість з них нульові. Також є можливим візуалізувати розріджену матрицю, як показано нижче (рис. 2.5.6):

`feature_vector.todense()`

```
matrix([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

Рис. 2.5.6. Матриця

Очевидно, що потрібно виправити набір даних, перш ніж продовжити правильну класифікацію.

Видалення стоп-слів. По-перше, видалимо стоп-слова з речень, оскільки вони не мають істотного значення. Стоп-слова — це такі слова, як the, an, was, is тощо. Це зменшить кількість функцій. Ми імпортуємо стоп-слова з бібліотеки nltk.

Код:

```
from nltk.corpus import stopwords

#оскільки текст англійською мовою, стоп-слова будуть англійською мовою, яку нам потрібно встановити, як показано нижче-
all_stopwords = set(stopwords.words('english'))

#так стоп-слова виглядають -
list(all_stopwords)[:10]

["doesn't",
 "weren't",
 'each',
 "she's",
 'himself',
```



```
'did',
'about',
'through',
'the',
'should']
```

Тепер знову почнемо з початку, викличемо векторизатор лічильника, але цього разу з додатковим параметром `stop_words`, який зупинить появу стоп-слів у векторі лічильника.

Код:

```
count_vector2 = CountVectorizer(stop_words=list(all_stopwords))
feature_names2 = count_vector2.fit(df['text'])

feature_vector2 = count_vector2.transform(df['text'])
feature_vector2.shape
(8488, 48473)
```

Раніше кількість функцій становила 48 618, а тепер 48 473. Це невелике зменшення. Потрібно ще більше зменшити. Подивимося на назви деяких функцій (рис. 2.5.7).

Код:

```
feature_names = feature_names2.get_feature_names_out()
feature_counts = np.sum(feature_vector2.toarray(), axis = 0)

pd.DataFrame(dict(Features = feature_names, Count = feature_counts))
```

	Features	Count
0	00	41
1	000	115
2	001	1
3	007	7
4	0080	1
...	...	...
48468	onsjón	1
48469	über	1
48470	über	6
48471	ünfaithful	1
48472	zmijewski	1

48473 rows x 2 columns

Рис. 2.5.7. Назви функцій

Якщо розглянути функції, то побачимо, що є багато неанглійських слів і чисел, які забруднюють набір даних. Приберемо їх:

```
for word in df.text[:][:10]:
    review = re.sub('[^a-zA-Z]', ' ', word)
```

Тепер перетворити всі слова в реченні для перегляду в нижній регістр, а потім видалити стоп-слова, оскільки векторизатор має тенденцію створювати два вектори для сказати «hello» і «HELLO» через різницю в регістрі.

```
count_vector3 = CountVectorizer(stop_words=list(all_stopwords))
feature_names3 = count_vector3.fit(sentences)
feature_vector3 = count_vector3.transform(sentences)
feature_vector3.shape
(8488, 47672)
```

Функції зменшено з 48473 до 47672. Але ще є такі відгуки, які не мають такого значення.

2.5.3. Стимулювання та лематизація

Створення кореня — це процес скорочення слів до кореневої форми. Створення кореня обрізає кінцеву частину слова та повертає його до кореневої форми, оскільки векторизатор розглядає слова зі схожим значенням, але написані по-різному, як дві різні ознаки. Наприклад, любов, люблячий мають однакові значення в різних формах. Стимулювання відрубає кожне слово до його кореневої форми. Це призведе до створення однієї ознаки замість 3. Одна проблема з корінням полягає в тому, що слова, створені після кореня, не є частиною словника, а коріння також не може перетворити слова з урахуванням їх морфологічного значення. Наприклад, і жінки, і жінка стосуються одного і того ж, але стимулювання цього не може зрозуміти. Проте лематизація бере до уваги морфологічний аналіз слів. Вона використовує словник для перетворення слів у їхній корінь. Наприклад, лематизація може зрозуміти, що жінка та жінки належать до однієї сутності, і зведе їх до жінки. Для цього існує два алгоритми — «PorterStemmer» і «LancasterStemmer»

```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()

sentences_stemmed = []
for texts in sentences:
    reviews = [ps.stem(word) for word in texts.split() if not word in all_stopwords]
    sentences_stemmed.append(' '.join(reviews))

#Let's call the Countvectorizer process now
count_vector4 = CountVectorizer()
feature_names4 = count_vector4.fit(sentences_stemmed)
```

```
feature_vector4 = count_vector4.transform(sentences_stemmed)

feature_vector4.shape
(8488, 32342)
```

Тут ми бачимо, що Features зменшено з 47 672 до 32 342. Тепер використаємо лематизацію, алгоритм «WordNetLemmatizer»:

```
from nltk.stem import WordNetLemmatizer
lemma = WordNetLemmatizer()
sentences_lemma = []
for texts in sentences:
    reviews = [lemma.lemmatize(word) for word in texts.split() if not
word in all_stopwords]
    sentences_lemma.append(' '.join(reviews))

count_vector5 = CountVectorizer()
feature_names5 = count_vector5.fit(sentences_lemma)
feature_vector5 = count_vector5.transform(sentences_lemma)

feature_vector5.shape
(8488, 42521)
```

Обираємо алгоритм «PorterStemmer» для стемінгу. Створимо функцію, яка виконуватиме всі ці завдання, і дасть кінцевий результат речень (рис. 2.5.8):

```
def get_clean_text(df, col):
    sentence = []

    for word in df[col][:]:
        review = re.sub('[^a-zA-Z]', ' ', word)
        review = review.lower()
        review = review.split()
        review = [ps.stem(word) for word in review if not word in
all_stopwords]
        review = ' '.join(review)
        sentence.append(review)

    return sentence

df['clean_text'] = get_clean_text(df, 'text')
df.head(10)
```

Тепер потрібно його векторизувати. Зробимо це так само, як і раніше, тобто за допомогою «countvectorizer»:

```
cv = CountVectorizer()
features = cv.fit_transform(df[ 'clean_text' ])
```

	text	label	clean_text
0	I grew up (b. 1965) watching and loving the Th...	0	grew b watch love thunderbird mate school watc...
1	When I put this movie in my DVD player, and sa...	0	put movi dvd player sat coke chip expect hope ...
2	Why do people who do not know what a particula...	0	peopl know particular time past like feel need...
3	Even though I have great interest in Biblical ...	0	even though great interest biblic movi bore de...
4	Im a die hard Dads Army fan and nothing will e...	1	im die hard dad armi fan noth ever chang got t...
5	A terrible movie as everyone has said. What ma...	0	terribl movi everyon said made laugh cameo app...
6	Finally watched this shocking movie last night...	1	fnal watch shock movi last night disturb mind...
7	I caught this film on AZN on cable. It sounded...	0	caught film azn cabl sound like would good fil...
8	It may be the remake of 1937 Autumn's Tale aft...	1	may remak autumn tale eleven year director mab...
9	My Super Ex Girlfriend turned out to be a plea...	1	super ex girlfriend turn pleasant surpris real...

Рис. 2.5.8. Остаточний результат

Класифікація. Розділимо набір даних і використаємо класифікацію для аналізу настроїв. Використовуємо наївну класифікацію Байєса:

```
#Розділимо набір даних на train та test
x = features.toarray()
y = df['label']
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =
0.10, random_state = 42)
```

2.5.4. Модельне навчання

Побудуємо два типи наївного класифікатора Байєса та порівняємо точність.



Наївний баєсів класифікатор — ймовірнісний класифікатор, що використовує теорему Баєса для визначення ймовірності приналежності спостереження (елемента вибірки) до одного з класів при припущенні (наївному) незалежності змінних. Тобто, якщо на основі значень змінних можна однозначно визначити, до якого класу належить спостереження, баєсів класифікатор повідомить ймовірність приналежності до цього класу.

У проміжних же випадках, коли спостереження може з різною ймовірністю належати до різних класів, результатом роботи класифікатора буде вектор, компоненти якого є ймовірностями приналежності до того чи іншого класу.



Можна бачити, що ідеальний баєсів класифікатор в якомусь сенсі є оптимальним. Його результат не може бути поліпшений, тому що в усіх випадках, коли можлива однозначна відповідь, він її дасть — а в тих випадках, коли відповідь неоднозначна, результат кількісно характеризує міру цієї неоднозначності.

По-перше, це модель Гауса «GaussianNB»:

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train, y_train)
```

Зробимо прогноз:

```
y_pred = classifier.predict(x_test)
```

Перевіримо зараз на точність:

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[305 124]
 [194 226]]
```

```
round(accuracy_score(y_test, y_pred), 3)
```

0.625

Маємо 194 хибно-негативні результати/ Точність моделі досить низька. Це результат наявності дивних коментарів. Перевіримо точність навчального набору, щоб перевірити на переобладнання:

```
y_pred_train = classifier.predict(x_train)
round(accuracy_score(y_train, y_pred_train), 3)
```

0.902

Модель занадто переобладнана. Тепер давайте застосуємо іншу модель класифікації NB, перш ніж обробити дані:

```
from sklearn.naive_bayes import BernoulliNB
```

```
classifier2 = BernoulliNB()
classifier2.fit(x_train, y_train)
```

```
y_pred2 = classifier2.predict(x_test)
cm = confusion_matrix(y_test, y_pred2)
sns.heatmap(cm, annot = True, fmt='.2f')
```

На рис. 2.5.9. рядки показують фактичні мітки з даних, а стовпці показують передбачені мітки. З цього отримуємо, що існує 111 позитивних тверджень, які помилково класифікуються як негативні, які є хибно-негативними. Існує 50 негативних тверджень, які класифікуються як позитивні, вказуючи на помилкові позитивні результати.

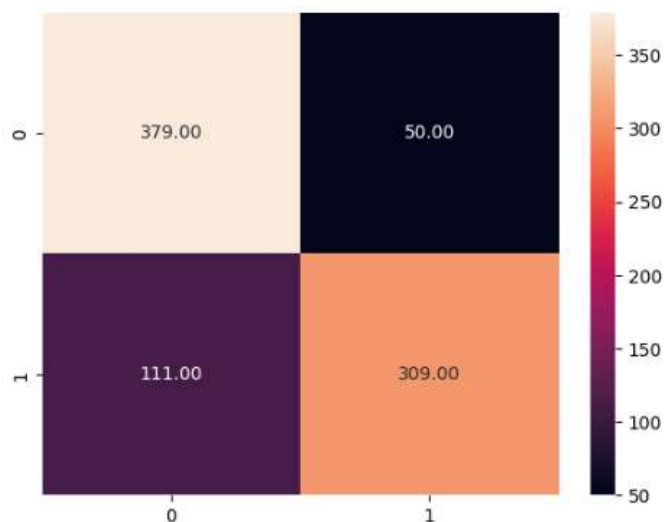


Рис.2.5.9. Модель Бернуллі

Це показує, що модель Бернуллі «BernoulliNB» може бути кращою моделлю для цього набору даних.

```
round(accuracy_score(y_test, y_pred2), 3)
```

0.81

Точність тесту також значно зросла завдяки використанню моделі! Перевіримо перенавчання моделі.

```
y_pred_train2 = classifier2.predict(x_train)
round(accuracy_score(y_train, y_pred_train2), 3)
```

0.916

Значно менше, ніж раніше. Давайте розглянемо повний класифікаційний звіт (рис. 2.5.10):

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred2))
```

	precision	recall	f1-score	support
0	0.77	0.88	0.82	429
1	0.86	0.74	0.79	420
accuracy			0.81	849
macro avg	0.82	0.81	0.81	849
weighted avg	0.82	0.81	0.81	849

Рис. 2.5.10. Повний звіт

Отже, ми бачимо, що для цього набору даних найкраще використовувати модель «BernoulliNB», а не модель Гауса.

Давайте протестуємо модель на повних невидимих даних:

```
#Це було написано для тесту
reviews = [
    "I didn't liked the movie. It was so boring.",
    'I am not happy that the movie ended so badly',
    "The movie is terrific. It's a must watch for every one."
]

dataframe = pd.DataFrame({'Text':reviews})
dataframe
```

	Text
0	I didn't liked the movie. It was so boring.
1	I am not happy that the movie ended so badly
2	The movie is terrific. It's a must watch for e...

Рис. 2.5.11. Вигляд набору даних

```
# очищення даних і застосування стемінгу
test_sent = get_clean_text(dataframe, 'Text')

# перетворення у вектори за допомогою останньої навченої моделі n-грам
x1 = tfidf.transform(test_sent).toarray()

# прогнозування невидимих даних за допомогою останнього навченого класифікатора
y_pred_res = classifier.predict(x1)

dataframe['predictions'] = y_pred_res.tolist()
dataframe
```

	Text	predictions
0	I didn't liked the movie. It was so boring.	0
1	I am not happy that the movie ended so badly	0
2	The movie is terrific. It's a must watch for e...	1

Рис. 2.5.12. Результат

Прогнози негативних і позитивних відгуків є правильними відповідно до вхідних даних, які не видно для нашої моделі.

Тепер ви можете створити власну модель текстової аналітики за допомогою описаної вище процедури.

Запитання для самоперевірки



1. Назвіть найбільш важливі методи аналізу текстів?
2. Що таке обробка природної мови (NLP)?
3. Дайте визначення аналізу настроїв та його основним методам.
4. Які бібліотеки Python використовуються для аналізу тексту?
5. Що таке найвщий Баєсів класифікатор?
6. Які принципи лежать в основі процесу обробки текстів?

2.6. Смарт-аналітика в проблемах міжнародної безпеки



Завдяки розвитку інформаційних технологій і активній цифровізації суспільства початок XXI ст. нерідко називають ерою інновацій, в основі якої — системи та технології штучного інтелекту. По суті, штучний інтелект — це головна складова третьої комп'ютерної ери, заснованої на інноваціях і передових технологіях. Укладачі британського словника англійської мови *Collins Dictionary* навіть назвали штучний інтелект (ШІ) «головним (найпопулярнішим) словом 2023 року». Зазначається, що у 2023 р. використання терміну зросло в чотири рази.

Нові технології ШІ виникли на новій хвилі технологічного розвитку. При цьому технології штучного інтелекту сприяють трансформації економіки, ринку праці, державних інституцій і суспільства загалом.

2.6.1. Основні сфери застосування технологій штучного інтелекту



Позитивні моменти в активному розвитку ШІ вбачаються у звільненні людей від важких, небезпечних і некваліфікованих видів робіт; збільшенні продуктивності в усіх галузях економіки; запобіганні хворобам, подовженні людського життя; убезпеченні людства від стихійних лих і катастроф; запобіганні злочинності тощо. Штучному інтелекту навіть можливо довірити правоохоронні функції, розроблення і вдосконалення законодавства, розгляд справ у суді. Активні розробки у сфері створення систем штучного інтелекту призвели до того, що зі сфери цивільного життя ці технології поступово перейшли до військової сфери. За таких умов суспільство активно прямує до цифрового суперінтелекту, який значно перевершує інтелект людини. Саме тому керівник *Google* С. Пічаї зауважив, що досягнення у сфері штучного інтелекту зроблять для людства більше, аніж відкриття вогню чи електрики.

Отже, великі дані, потужність їх опрацювання, а також застосовувані алгоритми ШІ є ключовими чинниками науково-технологічного лідерства держав. ШІ на основі опрацювання великих наборів даних дає змогу оптимізувати процеси та підвищити якість цифрових продуктів та послуг. Як наслідок, завдяки ШІ підвищується конкурентоспроможність бізнесу через оптимізацію бізнес-процесів, а також якість державних послуг, зменшується державний апарат завдяки автоматизації рутинних повторюваних процесів. Слід додати, що штучний інтелект може бути новітньою технологією, яка здатна знизити інфляцію.

Використання штучного інтелекту розповсюджується на все більше сфер і галузей економіки. До галузей, у яких прогнозується максимальний ефект від інвестицій у технології ШІ, належать: ІТ-технології, телекомунікації, комерційні та

професійні послуги, обслуговування споживачів і сектор фінансових послуг. Зокрема, найбільше від розвитку ШІ з точки зору збільшення доходів виграють галузі високих технологій, роздрібно́ї торгівлі, банківської справи, туризму й логістики, передових виробництв і охорони здоров'я. Вже зараз такі компанії, як *Amazon*, *Apple* і *Google* сформували цифрові платформи з використанням штучного інтелекту, а соціальна мережа *Facebook* запустила технологію *DeepTex*, яка надає можливість за повідомленнями розпізнавати тенденції поведінки користувачів.



На сьогодні основними сферами застосування технологій штучного інтелекту є: державне управління, місцеве самоврядування, національна і громадська безпека (інформаційна та кібербезпека), СМАРТ-інфраструктура, житлово-комунальне господарство, бізнес-процеси й системи, промислове виробництво, електроенергетика, ринок товарів і послуг (алгоритми, орієнтовані на клієнта), торгівля, трансфертне ціноутворення, банківська справа (управління ризиками; оцінювання, прогнозування й аналітика; чат-боти у мобільних банківських додатках), транспорт (оптимізація управління автомобільним транспортом, розширення засобів круїз-контролю, автопілот) і логістика (покращення продуктивності, зниження простоїв), телекомунікації, медицина (ведення документації, діагностика), освіта, наука, культура та спорт. Наприклад, чат-боти активно використовуються в різних сферах публічного управління та бізнесу.

Вже запроваджуються світові практики застосування технологій ШІ у сфері реалізації онлайн-торгів, навігаційних систем і діяльності поліції у розкритті злочинів на основі аналізу поведінки зловмисників. Втім, досі ключовим питанням при застосуванні практик впровадження технологій ШІ у різних країнах залишається легальність систем ШІ: наявність в законодавстві правової бази для їх функціонування, умови й необхідність втручання державних органів у конкретних обставинах, передбачуваність втручання для пересічних громадян, а також належні запобіжники від зловживань.



Вкрай важливою сферою, де знайдуть у майбутньому своє застосування технології штучного інтелекту, є дистанційне навчання, у тому числі із застосуванням технологій віртуальної реальності. Відповідно дедалі більше уваги науковці приділяють особливостям адаптації технологій доповненої реальності у навчальних дисциплінах для студентів різних навчальних закладів. При цьому, на думку дослідників, застосування технологій доповненої реальності підвищує мотивацію до навчання майбутніх фахівців, а також рівень засвоєння інформації завдяки різноманітності та інтерактивності її візуального подання.

2.6.2. Цілі і завдання ШІ в міжнародній безпеці

Фундаментально перед теорією штучного інтелекту постає низка цілей і завдань:

- стратегічною метою розвитку штучного інтелекту є наукове пояснення розумового процесу, оцінювання можливості передавання розумових функцій технічним системам;
- теоретична наукова мета ШІ — це пізнання механізму виконання різних функцій мозку і переробки інформації та створення моделей цих функцій;
- практична, технічна мета ШІ — це вирішення нагальних невідкладних завдань високого ступеня складності, з якими людський інтелект не може впоратися без допомоги технічних засобів.



Головними особливостями штучного інтелекту є:

- ШІ вивчає методи розв'язання задач, які потребують людського розуміння;
- ШІ вивчає методи розв'язання задач, для яких не існує способів розв'язання або вони не коректні;
- ШІ займається моделюванням людської вищої нервової діяльності;
- ШІ — це системи, які можуть оперувати знаннями, а найголовніше — навчатися.

Важливою особливістю ШІ є те, що він може значно швидше й ефективніше за людину вирішувати проблеми, з якими стикнулося людство на сучасному етапі розвитку — переходу від інформаційного до цифрового суспільства. Насамперед можливості ШІ реалізуються завдяки опрацюванню великих баз даних, розподіленим реєстрам, швидкісним комунікаціям у мережі, інтернету речей. Основу технологій ШІ складають можливості цифрової техніки проводити обчислення у багато разів швидше й обробляти значно більшу кількість даних, ніж людина.

Водночас штучний інтелект — це технічна система, що має певні ознаки інтелекту, тобто здатна: розпізнавати і розуміти; знаходити спосіб досягнення результату та приймати рішення; вчитися. Характерними рисами штучного інтелекту є накопичення досвіду (знань, навичок), адаптація до зміни умов, узагальнення, розуміння природної мови, здатність до навчання, а також здатність мислити і, що важливо, діяти.

Системи на основі штучного інтелекту можуть бути заснованими винятково на програмному забезпеченні та працювати у віртуальному світі (наприклад, голосові помічники, програмне забезпечення для аналізу зображень, пошукові системи, системи розпізнавання мовлення та обличчя) або можуть бути інтегрованими (вбудованими) в апаратне забезпечення (наприклад, робототехніка, безпілотні транспортні засоби, дрони чи об'єкти мережі «Інтернет речей» (англ. Internet of Things, IoT)).



Сьогодні ШІ об'єднує багато технологій, які працюють з базами даних і здатні до самонавчання, основними з яких вважаються:

- *машинне навчання* (Machine Learning, ML) — вивчає методи побудови алгоритмів;
- *комп'ютерний зір* (Computer Vision, CV) — спеціалізується на технологіях аналізу, класифікації та розпізнавання зображень і відео;
- *штучні нейронні мережі* (Artificial Neural Networks, ANN) — відтворює роботу людського мозку;
- *глибоке навчання* (Deep Learning, DL) — фокусується на класифікації даних;
- *опрацювання природної мови* (Natural Language Processing, NLP) — спеціалізується на технологіях розпізнавання мови.

Прикладом машинного навчання є віртуальні особисті помічники, які допомагають створювати особистий розклад, планувати зустрічі, будувати маршрути, надавати рекомендації під час прийняття рішень; відеоспостереження, яке відстежує незвичну поведінку, тощо. Іншими словами, штучний інтелект включає широкий спектр можливостей, таких як голос, розпізнавання зображень, машинне навчання та семантичний пошук¹.

Сучасні технології штучного інтелекту реалізуються за наступними напрямками: розпізнавання і синтез мови; інтелектуальні системи підтримки прийняття рішень; перспективні методи штучного інтелекту.

2.6.3. «Штучний інтелект: не упустити вигоду»



До переваг застосування ШІ належать: точність і швидкість обробки даних; аналіз значної кількості інформації у стислі строки; зменшення помилок і збільшення шансів досягнення результатів більш високої точності; зниження собівартості та підвищення продуктивності праці; невтомність / відсутність потреби у сні або перерві; здатність працювати у потенційно небезпечному для людини середовищі.

Згідно з прогнозом звіту «Штучний інтелект: не упустити вигоду» міжнародної консалтингової компанії *PricewaterhouseCoopers* (PwC), до 2030 р. завдяки прискореному розвитку СШІ глобальний ВВП може зрости на 14% порівняно з поточними показниками (це орієнтовно 15,7 трлн дол. США). Основним чинником зростання є автоматизація виробничих процесів, яка до 2030 р. може збільшитися на 11%, або майже 9 трлн дол. США. Зі свого боку, дослідження консал-

¹ *Семантичний пошук* — це вдосконалення пошуку з використанням інформації про структурні елементи того інформаційного елементу, який потрібно знайти, про його властивості та відношення з іншими інформаційними ресурсами. Під час семантичного пошуку система сама збирає і видає користувачеві потрібну йому інформацію, а не направляє його до безлічі знайдених ресурсів.

тингової компанії *McKinsey* вказує, що впровадження штучного інтелекту на рівні програмного забезпечення може щорічно приносити світові економічну вигоду від 2,6 до 4,4 трлн дол. США на рік, що становить 4,4% усієї світової економіки. Для порівняння: за весь 2022 рік світова економіка [зросла](#) на 3,6 трлн дол. США.



Експерти зазначають, що обороти інвестицій у технології штучного інтелекту щорічно зростають і до 2024 р. можуть досягти майже 140 млрд дол. США. Загалом, за даними дослідницької компанії IDC, у 2022 р. світові витрати на штучний інтелект досягли 79,2 млрд дол. США. У 2021 р. загальні інвестиції в штучний інтелект склали 77,5 млрд дол. США, що значно перевищує показник 2020 р. в 36 млрд дол. США. А у 2018 р. обсяг інвестицій у галузь штучного інтелекту складав усього 27 млрд дол. США. У 2022 р. розмір глобального ринку штучного інтелекту оцінювався у 119,78 млрд дол. США, очікується, що він досягне 1591,03 млрд дол. США до 2030 р.

З початку 2023 р. середня ціна акцій компаній із списку S&P 500 з найбільшим впровадженням технологій ШІ у свою діяльність зросла на 11%. Капіталізація компаній, які повільно рухаються в цьому напрямі, майже не змінилася. Однак, за даними *The Economist*, майже 70 компаній із списку S&P 500 досі не зацікавлені в штучному інтелекті.

Технологія генеративного штучного інтелекту, яка вже вміє створювати тексти й почасти навіть музику, захопила світ після випуску американською компанією *OpenAI* всесвітньовідомого чат-боту *ChatGPT*. Цей інструмент став світовим рекордсменом за швидкістю розвитку.

Так, **генеративний ШІ найбільш ефективний у чотирьох сферах**: обслуговування клієнтів, маркетинг і продажі, розробка програмного забезпечення, дослідження і розробка продуктів. Штучний інтелект уже зараз здатний підтримувати взаємодію з клієнтами, генерувати контент для маркетингу і писати код.

Згідно з прогнозами американської консалтингової компанії *Gartner* до 2025 р. генеративний штучний інтелект генеруватиме 10% усіх вироблених даних. Для порівняння, сьогодні ця цифра складає менше 1%.

Аналітики консалтингової компанії *McKinsey* прогнозують, що приблизно 70% компаній запровадять принаймні один тип технології ШІ до 2030 р.



У На світовому рівні потенційний вплив від впровадження технологій генеративного ШІ щорічно може мати прояв у збільшенні прибутку в розмірі 400–660 млрд дол. США.

Український бізнес теж відкритий до новітніх технологій ШІ. Фахівці освітньої лабораторії *Projector AI Lab* опитали 150 українських компаній, з них 51% компаній час від часу використовують ці технології, 20% — рідко користуються ШІ або взагалі не використовують, 15% компаній самі розробили інструменти ШІ, 24% — роблять це або планують цим зайнятися.

Національна академія наук України навіть створила Інститут проблем штучного інтелекту та розробила [Стратегію розвитку](#) штучного інтелекту в Україні. Відповідно до положень цієї стратегії штучний інтелект повинен допомагати розвивати й охоплювати усі сфери життєдіяльності держави. Оборона, економіка, освіта, промисловість, медицина, телекомунікації, аграрний сектор — залучення ІІ до цих галузей і пошуку рішень у разі пришвидшить і покращить процес їх зростання. Отже, активне запровадження штучного інтелекту в усі сфери життєдіяльності суспільства сприятиме зростанню екосистеми та безпечному функціонуванню її суб'єктів.

Слід наголосити, що ще у 2017 р. Організація економічного співробітництва та розвитку (ОЕСР) виокремила чотири ключові драйвери цифрової трансформації: штучний інтелект, інтернет речей, технологія аналізу великих даних і технологія блокчейн.



Однак, незважаючи на тривалу історію розвитку штучного інтелекту, досі немає єдиного розуміння його сутності та чіткого визначення терміну «штучний інтелект». Крім того, **застосування цієї технології досі не стандартизовано**. Більшість наукових досліджень присвячена філософському аналізу феномену ІІ. Тому вчені надають різні визначення штучного інтелекту.

У *Державному стандарті України ДСТУ 2938-94* («Системи оброблення інформації») штучний інтелект визначається як «здатність систем оброблення даних виконувати функції, що асоціюються з інтелектом людини, — логічне мислення, навчання та самовдосконалення».

Як зазначено у *«Білій книзі» Європейської Комісії*, штучний інтелект є сукупністю технологій, які об'єднують дані, алгоритми й обчислювальні потужності. Штучний інтелект дає змогу автоматизувати ті процеси, які раніше виконувала людина, використовуючи алгоритми, які значною мірою копіюють мислення самої людини.



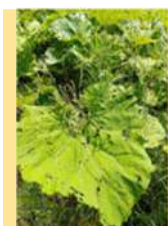
Експерти НАТО у своїй діяльності оперують спорідненими тлумаченнями штучного інтелекту: «спроможність, що надається алгоритмами оптимального або неоптимального вибору з широкого простору можливостей для досягнення цілей через застосування стратегій, які можуть спиратися на навчання або адаптацію до навколишнього середовища»; «системи, які створені людиною і діють у фізичному або цифровому світі, враховують складну мету й обирають найкращі дії (відповідно до заздалегідь визначених параметрів), які необхідно виконати для досягнення поставленої мети на основі сприйняття свого середовища, інтерпретації зібраних структурованих або неструктурованих даних і обґрунтування знань, отриманих із цих даних».

Згідно з резолюцією Європарламенту (European Parliament resolution of 16 February 2017 with recommendations to the Commission on Civil Law Rules on Robotics, 2015/2103(INL)), штучний інтелект — це розумний робот, який має такі особливості: набуває автономності за допомогою сенсорних датчиків та/або обміну даними; самостійно вивчає досвід чи взаємодію; має слабку фізичну підтримку; адаптує модель поведінки до середовища; є неживим у біологічному сенсі. У повідомленні Єврокомісії штучний інтелект трактовано як системи, що демонструють розумову діяльність у формі аналізу навколишнього середовища та вживання заходів (із певним ступенем автономії) для досягнення конкретної мети.

Водночас відсутність нормативно-правового акту у формі закону, який би регулював застосування штучного інтелекту в життєдіяльності суспільства в Україні, є однією із прогалин вітчизняного законодавства. В Україні на законодавчому рівні не визначено поняття, види (форми) штучного інтелекту, принципи, межі, умови й порядок його застосування тощо: ці та інші проблеми донині розглядаються лише у теоретичній площині. З іншого боку, міжнародне законодавство та іноземні доктрини також не дають чіткої відповіді на окремі питання у сфері застосування штучного інтелекту. Отже, у реаліях сьогодення важливими є підготовка та прийняття в Україні нормативно-правового акту у формі закону, який би регулював застосування штучного інтелекту в різних галузях держави. У формуванні законодавчого визначення поняття «штучний інтелект», необхідно зважати на напрацювання зарубіжних науковців, позитивний досвід країн ЄС, США, Японії, Китаю з цих питань.

Зокрема, актуальними постають питання щодо визначення у національному законодавстві штучного інтелекту як об'єкту цивільних правовідносин; питання реалізації цивільних прав і обов'язків у сфері застосування ШІ; гарантії захисту відповідних прав і юридичної відповідальності. Аналіз наукових досліджень різних авторів з цих питань дозволив дійти таких висновків: штучний інтелект слід розглядати як об'єкт цивільних правовідносин; штучний інтелект є об'єктом цивільних прав або ж об'єктом права інтелектуальної власності; з метою реалізації юридичної відповідальності за заподіяну шкоду штучним інтелектом мають застосовуватися норми законодавства про діяльність, пов'язану з джерелом підвищеної небезпеки, а вона повинна покладатися на володільця штучного інтелекту.

При цьому у світі системи на основі штучного інтелекту переважно розуміються як програмне забезпечення, яке використовують у віртуальному світі для аналізу даних, зображень, як-от системи розпізнавання мовлення й обличчя, пошукові системи тощо. Штучний інтелект може бути вбудовано в апаратні пристрої (наприклад, автономні машини, дрони або об'єкти інтернету речей).



Водночас аналіз сучасних світових концепцій штучного інтелекту свідчить, що інтелект людини і штучний інтелект — це взаємопов'язані явища, оскільки функціонування штучного інтелекту (основними характеристиками якого є здатність мислити, сприймати, розв'язувати проблеми, навчатися тощо) базується на принципах функціонування людського; єдина відмінність

Нині застосування штучного інтелекту в основному проходить у формах систем, заснованих на знаннях (*knowledgebased systems*), у яких для розв'язання проблем використовуються людські знання. Найпопулярнішим типом таких систем є **експертні системи**. Галузь штучного інтелекту займається створенням систем (механізмів, коли йдеться про роботів, або комп'ютерних програм), які можуть вирішувати складні задачі без допомоги людини.



Водночас слід розмежовувати поняття «робот», «комп'ютерні технології та програми», «інформаційні системи передачі інформації», «штучний інтелект» тощо. Так, *поняття «робот» і «штучний інтелект» співвідносяться між собою як форма та зміст*. Штучний інтелект вченими розглядається як моделювання здатності до абстрактної, творчої думки, насамперед здатності до навчання, за допомогою цифрової комп'ютерної логіки. Натомість робот є пристроєм, машиною, основною здатністю якої є автоматизоване виконання одного чи кількох завдань за зразком дій людини, якій властиві ознаки рухливості, сенситивності, аналітичності тощо. Вченими доведено, що саме штучний інтелект дає роботу ті властивості, якими прийнято його характеризувати, а саме: розумність, здатність до аналізу та оброблення інформації, а також до виконання завдань, на які він запрограмований.

Штучний інтелект зазвичай поділяють на такі різновиди:

1) **«слабкий штучний інтелект»** (англ. Weak Artificial Intelligence, WAI), або «вузький штучний інтелект» / «обмежений штучний інтелект» (англ. Artificial Narrow Intelligence, NAI) — це штучний інтелект, який орієнтовано на вирішення одного чи декількох завдань, які виконує або може виконувати людина;

2) **«сильний штучний інтелект»** (англ. Strong artificial intelligence, ASI, термін запропонував американський філософ Джон Серль (John Searle)), або «загальний штучний інтелект» (англ. Artificial General Intelligence, AGI) — це штучний інтелект, який орієнтовано на вирішення всіх завдань, які виконує або може виконувати людина;

3) **«штучний суперінтелект»** (англ. Artificial Superintelligence, ASI, термін запропонував шведський філософ Нік Бострем (Nick Bostrom)) — це інтелект, який набагато розумніший, ніж найкращий людський інтелект, практично в кожній галузі, включаючи наукову творчість, загальну мудрість і соціальні навички.

2.6.4. Міжнародні рейтинги з розвитку технологій ШІ

Міжнародні рейтинги з розвитку технологій ШІ очолюють США і Китай (табл. 2.1). США залишаються світовим лідером у розвитку ШІ, зокрема з комерційних інвестицій у цю галузь. За рівнем розвитку технологій штучного інтелекту Китай сьогодні упевнено посідає друге місце у світі після США, а за десять років має намір стати безумовним світовим лідером у цій галузі. Технології ШІ в

найближче десятиліття будуть головним драйвером розвитку економіки Китаю, які є основою для створення унікальних технологічних переваг цієї країни під час підкорення нових ринків у глобальному поділі праці. У листопаді 2023 р. гендиректор *Alphabet*, яка є материнською компанією *Google*, Сундар Пічаї теж наголосив, що Китай може стати світовим лідером у галузі ІІІ. Наприклад, найближчими роками КНР планує розпочати виробництво людиноподібних роботів і стати світовим лідером у робототехніці.

Таблиця 2.1

Країни, у яких найкраще розвинена галузь ІІІ

Країна	Місце в рейтингу									
	Загальний рейтинг	А	Б	В	Г	Ґ	Д	Е	Є	Ж
США	1	1	1	28	1	1	8	1	1	5
Китай	2	20	2	3	2	2	3	2	2	21
Сінгапур	3	4	3	22	3	5	16	4	10	1
Велика Британія	4	5	24	30	5	8	10	5	4	10
Канада	5	6	23	8	7	11	5	7	7	7
Південна Корея	6	12	7	11	12	3	6	18	8	6
Ізраїль	7	7	28	23	11	7	47	3	17	2
Німеччина	8	3	12	13	8	9	2	11	3	15
Швейцарія	9	9	13	30	4	4	56	9	16	3
Фінляндія	10	13	8	4	9	14	16	12	13	4
Нідерланди	11	8	16	15	10	13	28	20	11	8
Японія	12	11	5	10	20	6	18	23	6	25
Франція	13	10	11	25	15	18	13	10	9	20
Індія	14	2	59	12	30	21	38	13	5	51
Австралія	15	14	44	62	6	16	14	22	15	14

Умовні позначення:

А — кількість фахівців; Б — інфраструктура; В — закони та громадська думка; Г — дослідження; Ґ — розробка; Д — стратегія уряду; Е — інвестиції та стартапи; Є — масштаб галузі ІІІ; Ж — масштаб галузі ІІІ відносно економіки чи населення.

Джерело: The Global AI Index.

Згідно зі звітом Всесвітньої організації інтелектуальної власності (ВОІВ, WIPO) «Про стан трендів у галузі штучного інтелекту» Китай значно випереджає інші країни за окремими показниками розвитку штучного інтелекту. Зокрема, за фінансовими показниками загальний обсяг китайської індустрії технологій ІІІ до 2020 р. перевищив 22 млрд дол. США, до 2025 р. — має вийти на позначку 60,3 млрд дол. США, а до 2030 р. — досягти 1 трлн юанів (приблизно 150,8 млрд дол. США).

Крім того, Китай досяг значних успіхів у галузі ІІІ за кількістю міжнародних опублікованих науково-технічних робіт і кількістю запатентованих винаходів.

Також Китай є лідером у розвитку технологій розпізнавання голосу, візуального розпізнавання обличчя і біометричної ідентифікації. Водночас Китай займає перше місце у світі в питанні доступу до баз великих даних.



Також у рейтингу розвитку інфраструктури для ШІ у першій десятці держав визначені країни Південно-Східної Азії, а саме: Південна Корея, Японія, Сінгапур. Канада отримала 100% у крос-національному дослідженні готовності урядів до ШІ (за параметром урядової підтримки впровадження ШІ у країні). Разом із Бразилією, Індією, Сінгапуром і Південною Африкою, Канада є країною з найвищим зростанням найму в галузі, пов'язані з ШІ, у період з 2016 до 2020 рр. На високих позиціях рейтингу країн з відповідальним використанням технологій ШІ разом з Канадою також знаходяться Естонія, Норвегія, Фінляндія, Швеція.

В Україні поки що спостерігається низький рівень впровадження і реалізації суб'єктами господарювання інноваційних проєктів з використанням технологій штучного інтелекту порівняно із провідними країнами світу, що призводить до зниження продуктивності праці та появи великого відсотка робочих місць, які необхідно автоматизувати, а також низький рівень інвестицій у розроблення технологій штучного інтелекту. Крім того, як зазначено в Концепції розвитку штучного інтелекту в Україні, схваленій Розпорядженням Кабінету Міністрів України від 2 грудня 2020 р. №1556-р, в Україні — недостатній рівень інформаційної безпеки й захисту даних в інформаційно-телекомунікаційних системах державних органів внаслідок застарілості автоматичних систем виявлення та оцінки інформаційних загроз і недостатнього розвитку з цією метою новітніх технологій ШІ, невикористання потенціалу прогнозування та передбачення загроз на основі ШІ з метою своєчасної підготовки системи до можливої кібератаки. Слабкий розвиток в Україні галузі ШІ також призводить до недосконалих механізмів прийняття управлінських рішень у публічній сфері.



Країни з високими показниками ШІ розташовані у Західній Європі, Східній Азії та Північній Америці. У країнах Європи та США штучний інтелект активно використовується в оборонній сфері та сфері забезпечення публічної безпеки і громадського порядку. Сінгапур, який за останні два роки піднявся з десятої на третю сходинку, має високі результати щодо ШІ, Велика Британія має перевагу в дослідженнях і комерційних інвестиціях у ШІ, а Німеччина отримала високі оцінки у впровадженні ШІ.

Якщо співвіднести масштаб розвитку ШІ в країні з її населенням, то лідерами є Сінгапур, Ізраїль і Швейцарія: вони можуть масово впроваджувати ШІ. Це важливо, оскільки кількість інновацій не так сильно впливає на розвиток країни, як впровадження і поширення інновацій серед населення. Отже, Китай і США, які є лідерами за інвестиціями й інноваціями у сфері ШІ, можуть не отримати достат-

ньої вигоди від ШІ для власних економік через низький рівень поширення технології серед громадян. Не менш важливим чинником у галузі ШІ є позитивне ставлення суспільства до розвитку штучного інтелекту.



Зі свого боку, Велика Британія хоче закріпити себе як арбітра між Сполученими Штатами, Китаєм і Європейським Союзом у технологічному секторі штучного інтелекту. Велика Британія має у планах стати центром безпеки штучного інтелекту. З цією метою 1–2 листопада 2023 р. у Лондоні Велика Британія провела перший у світі глобальний саміт з безпеки штучного інтелекту.

Мета саміту — розпочати міжнародний діалог щодо регулювання штучного інтелекту. Порядок денний заходу було присвячено обговоренню непередбачуваних досягнень технології та можливості втрати людьми контролю над ШІ. На саміті розглядалися питання, як скоординовані міжнародні дії можуть подолати ризики, викликані поширенням у світі штучного інтелекту.

Зокрема, у перший день саміту двадцять вісім країн, серед них Британія, США, ЄС і Китай, підписали *декларацію Блетчлі* (Bletchley declaration) про «катастрофічну» небезпеку штучного інтелекту для людства. Країни домовилися працювати разом над дослідженнями безпеки ШІ. З цього приводу Прем'єр-міністр Великої Британії Ріші Сунак наголосив, що «ми зобов'язані гарантувати безпечний і відповідальний розвиток штучного інтелекту, враховуючи ризики, які він створює на досить ранніх стадіях процесу».

Учасники саміту, зокрема [технологічний американський мільярдер Ілон Маск](#), віцепрезидентка США Камала Гаріс і високопосадовець уряду Китаю, погодилися, що штучний інтелект становить серйозну небезпеку для людства. Так, у кулуарах саміту Ілон Маск попередив: «Вперше ми зіткнулися з ситуацією, коли відбувається щось, що буде набагато розумнішим за найрозумнішу людину... мені не зрозуміло, чи можемо ми насправді це контролювати».

Після саміту з безпеки штучного інтелекту в Лондоні [Прем'єр-міністр Великої Британії Ріші Сунак](#) також заявив, що найпередовіші технологічні компанії, які впроваджують технології ШІ, зокрема *Meta*, *Google DeepMind* та *OpenAI*, погодилися дозволити державним регуляторам тестувати (перевіряти) свої новітні продукти штучного інтелекту перед оприлюдненням. Це, на думку експертів, сповільнить перегони з розробки ШІ, які можуть конкурувати з людьми. Прем'єр-міністр Великої Британії підкреслив, що угоди, досягнуті з кількома країнами й компаніями, які займаються штучним інтелектом, значно знизили загрозу, що походить від цієї технології, однак він змушений захищати добровільний характер угоди про тестування технологій ШІ, оскільки уряд Великої Британії відмовився ухвалити закон, який обмежує розвиток ШІ.

Загалом сьогодні розвинуті країни світу, які активно впроваджують технології ШІ у різних галузях, впевнені у необхідності впровадження і подальшого дотримання світових стандартів штучного інтелекту з метою забезпечення його розвитку.

2.6.5. Аналітика міжнародних торговельних мереж



Модельна задача. Міжнародні торговельні мережі відіграють ключову роль у світовій економіці, і розуміння їх взаємопов'язаного характеру має вирішальне значення для компаній, політиків і дослідників. Дані служать важливим ресурсом для розуміння глобальних економічних тенденцій, розкриття ринкових можливостей і інформування про політичні рішення. Аналіз і візуалізація подібних мереж надають потужні інструменти для виявлення прихованих закономірностей і отримання глибшого розуміння міжнародної політики.

Аналіз мереж дозволяє вивчати міжнародні торговельні мережі як взаємопов'язані системи, забезпечуючи цілісне уявлення про їх структуру, характеристики та динаміку. Представляючи країни або організації як вузли, а торгові відносини як межі, мережевий аналіз виявляє такі важливі властивості, як центральність, кластеризація та структури спільноти. Цей підхід дає змогу зрозуміти потік товарів, послуг та інформації через кордони та визначити ключових гравців і впливових організацій у мережі.

NetworkX — це потужна бібліотека **Python** для створення, керування та аналізу складних мереж, а **Plotly** — універсальна бібліотека для створення інтерактивних та візуально привабливих візуалізацій. Завдяки інтеграції NetworkX і Plotly є можливим поєднувати мережевий аналіз і візуалізацію, щоб отримати всебічне розуміння міжнародних мереж і ефективно аналізувати їх.

Налаштування середовища. Щоб розпочати аналіз торгових мереж за допомогою NetworkX і Plotly, потрібно налаштувати середовище Python із необхідними бібліотеками та залежностями, код наведено далі:

```
import networkx as nx
import pandas as pd
import plotly.graph_objects as go
import plotly.express as px

from plotly.subplots import make_subplots
```

Збір і підготовка даних. Для аналізу міжнародної торговельної мережі потрібні надійні та вичерпні торгові дані. Для задачі отримаємо та підготуємо дані про торгівлю з платформи Світового інтегрованого торговельного рішення (WB-WITS) Світового банку, яка надає велику кількість даних про міжнародну торгівлю.

Для демонстрації буде використовуватися дані експортної торгівлі (INDICATOR= XPRT-TRD-VL), попередньо зібрані з WB-WITS SDMX API для мінеральних продуктів. Потім фільтруємо дані лише для торгівлі за 2018 рік.

```
df = pd.read_csv('XPRT-TRD-VL_25-26_Minerals.csv')
df = df[(df['TIME_PERIOD'] == 2018) &
        (~ (df['REPORTER'] == 'EUN') & ~ (df['PARTNER'] ==
        'EUN'))] # EUN = European Union
]
```

FREQ	REPORTER	PARTNER	PRODUCTCODE	INDICATOR	TIME_PERIOD	DATASOURCE	OBS_VALUE
A	AGO	AZE	25-26_Minerals	XPRT-TRD-VL	2018	WITS-CMT	37.952553
A	AGO	BEL	25-26_Minerals	XPRT-TRD-VL	2018	WITS-CMT	39.118574
A	ALB	ARE	25-26_Minerals	XPRT-TRD-VL	2018	WITS-CMT	3558.955112
A	ALB	ARM	25-26_Minerals	XPRT-TRD-VL	2018	WITS-CMT	11.626449
A	ALB	AUT	25-26_Minerals	XPRT-TRD-VL	2018	WITS-CMT	502.512541

Рис. 2.6.1. *Набір даних*

Інформація про дані (рис. 2.6.1):

- **FREQ** — частота даних, де A = рік.
- **REPORTER** — це код країни експорту. Причина, чому він використовує цей термін, полягає в тому, що дані про торгівлю надходять зі звітів країн, надісланих до Світового банку, тому вони називаються **REPORTER**.
- **PARTNER** — код країни торгового контрагента.
- **PRODUCTCODE** — це код товару, що продається.
- **INDICATOR** є показником значення, де **XPRT-TRD-VL** означає Export Trade Value (US\$ Thousand)
- **TIME_PERIOD** — це період часу значення.
- **DATASOURCE** — це додаткова інформація про те, звідки це значення.
- **OBS_VALUE** — це значення даних.

Побудова міжнародної мережі. Після підготовки набору даних, наступним кроком є створення представлення міжнародної торгової мережі та її властивостей (наприклад, центральності) за допомогою бібліотеки NetworkX. Розглянемо це покроково:

```
1. Ініціалізація (спрямованого) графа: G = nx.DiGraph()
2. Додавання вузлів до графа:
country_set = set(df['REPORTER'].values.tolist() +
df['PARTNER'].values.tolist()) # об'єднати та прибрати дублікати
G.add_nodes_from(country_set)
3. Додавання ребер до графа:
for _, row in df.iterrows():
    exporter = row['REPORTER']
    importer = row['PARTNER']
    export_value = row['OBS_VALUE']
    G.add_edge(exporter, importer, weight=export_value)
```

Коли міжнародна торгова мережа побудована за допомогою NetworkX, застосовуються різні методи аналізу мережі, щоб отримати цінну інформацію про структуру та динаміку мережі.

Міри центральності допомагають визначити найважливіші вузли в міжнародній торговій мережі. Існує багато методів обчислення центральності вузла. На даний момент застосуємо лише загальні вимірювання центральності, які підраховують кількість з'єднань, які має вузол. Розраховуючи показники центральності, є можливим визначити впливові країни чи організації, які відіграють ключову роль у торгових мережах.

```
# Отримайте центральність кожного вузла
degree_centralities = nx.degree_centrality(G)
betweenness_centralities = nx.betweenness_centrality(G)
pagerank_scores = nx.pagerank(G)
eigenvector_centrality = nx.eigenvector_centrality(G)

centralities = degree_centralities # центральність |
betweenness_centralities | pagerank_scores | eigenvector_centrality
print(centralities)
# {'AGO': 0.0008879530226908054, 'AZE':
0.00017629039646938503, 'BEL': 0.017597369348793378,..., 'TKL': 0.0}

# Отримайте кількість з'єднаних ребер
adjacencies = {k: len(v) for k, v in dict(G.adjacency()).items()}
print(adjacencies)
# {'AGO': 33, 'AZE': 25, 'BEL': 152, 'ALB': 40, 'ARE': 120,...,
'TUV': 0, 'WLF': 0, 'TKL': 0}

nx.set_node_attributes(G, centralities, 'centrality')
nx.set_node_attributes(G, adjacencies, 'adjacency')

# Перевірте атрибути вузла
print(G.nodes['IDN']) # {'centrality': 0.6467889908256881,
'adjacency': 75}

df_node_attributes = pd.DataFrame(['country': node,
**G.nodes[node]] for node in G.nodes())
df_node_attributes = df_node_attributes \
    .sort_values(by=['centrality'], ascending=True)
```

Отже, маємо вузли з атрибутами, і є можливість візуалізувати це.

1. Візуалізація центрів торгівельної мережі. Визначити координату кожного вузла на основі оцінки центральності. Перш ніж створювати візуалізацію вузлів і вершин, потрібно визначити розташування кожного вузла (x, y), щоб нанести його на діаграму `networkx.spring_layout()`. Для цього використовується такий метод:

```
# Отримайте x, y кожного вузла за допомогою spring_layout
layout = nx.spring_layout(G
    , weight='centrality')
```


2. Створити ребра. Додайте ребра як роз'єднані лінії в одній трасі. Код:

```
edge_x = []
edge_y = []

for reporter, partner in G.edges():
    x0, y0 = layout[reporter]
    x1, y1 = layout[partner]

    edge_x.append(x0)
    edge_x.append(x1)
    edge_x.append(None)
    edge_y.append(y0)
    edge_y.append(y1)
    edge_y.append(None)

edge_trace = go.Scatter(
    x=edge_x, y=edge_y,
    line=dict(width=0.1, color='#888'),
    hoverinfo='none',
    mode='lines')
```

3. Створення вузлів. Код:

```
node_x = []
node_y = []
node_text = []

for node in df_node_attributes['country']:
    x, y = layout[node]
    text = node
    node_x.append(x)
    node_y.append(y)
    node_text.append(text)

node_trace = go.Scatter(
    x=node_x,
    y=node_y,
    text=node_text,
    mode='markers+text',
    hoverinfo='text',
    textfont=dict(
        size=10 # Set the text size
    ),
    marker=dict(
        showscale=True,
        # colorscale options
        #'Greys' | 'YlGnBu' | 'Greens' | 'YlOrRd' | 'Bluered' | 'RdBu' |
```

```

# 'Reds' | 'Blues' | 'Picnic' | 'Rainbow' | 'Portland' | 'Jet' |
# 'Hot' | 'Blackbody' | 'Earth' | 'Electric' | 'Viridis' |
colorscale='Reds',
reversescale=False,
color=[],
size=20,
colorbar=dict(
    thickness=10,
    title='Node Connections',
    xanchor='left',
    titleside='right'
),
line_width=2
))

```

4. Точки кольорових вузлів. Встановіть колір вузлів за кількістю з'єднаних країв, а також вкажіть, який текст буде показано під час наведення курсора. Код:

```

node_text = []
node_adjacencies = []
node_hovertext = []

for country in df_node_attributes['country']:
    node_text.append(country)

    connections = df_node_attributes[df_node_attributes['country']
    == country]['adjacency'].values[0]
    node_centrality =
df_node_attributes[df_node_attributes['country'] ==
country]['centrality'].values[0]

    hovertext = f'[{country}]<br># of connections:
{connections}<br>Centrality Score: {node_centrality:.2f}'
    node_adjacencies.append(connections)
    node_hovertext.append(hovertext)

node_trace.text = node_text
node_trace.marker.color = node_adjacencies
node_trace.hovertext = node_hovertext

```

5. Візуалізація мережі. Код:

```

fig = go.Figure(layout=go.Layout(
    title='<br>Trade Network Minerals, 2018',
    titlefont_size=16,
    showlegend=False,
    hovermode='closest',
    margin=dict(b=20,l=5,r=5,t=40),

```

```

        xaxis=dict(showgrid=False, zeroline=False,
showticklabels=False),
        yaxis=dict(showgrid=False, zeroline=False,
showticklabels=False))
    )

fig.add_trace(edge_trace)
fig.add_trace(node_trace)

fig.show()

```

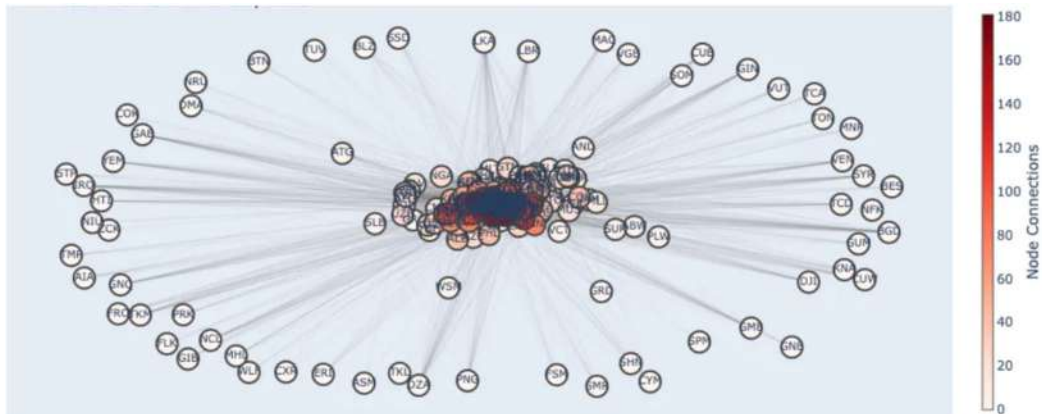


Рис. 2.6.2. Міжнародна торговельна мережа

На мережевому графіку на рис. 2.6.2 визначено, наскільки країна займає центральне місце з точки зору експорту корисних копалин у 2018 р. Колір вузла показує, який торговельний контрагент має країна.

Однією з потужних особливостей Plotly є те, що бібліотека дозволяє інтерактивно збільшувати візуалізацію. Нижче наведено збільшену версію графіка (рис. 2.6.3)

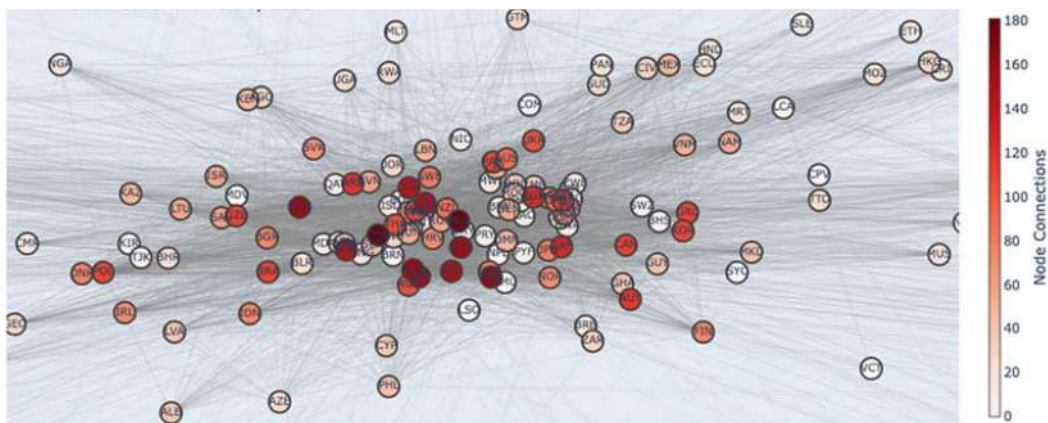


Рис. 2.6.3. Міжнародна торгівельна мережа (збільшена візуалізація)

Код:

```
n = 30
sort_by_cols = [ 'adjacency' ]
top_n = df_node_attributes.nlargest(n, sort_by_cols)

fig = make_subplots(specs=[[{ "secondary_y" : True }]])

fig.add_trace(go.Bar(name= "adjacencies ) ",
                    x=top_n[ 'country' ],
                    y=top_n[ 'adjacency' ],
                    text=top_n[ 'adjacency' ],
                    textposition= 'outside' )
                )

fig.add_trace(go.Scatter(name= "centrality_score",
                        x=top_n[ 'country' ],
                        y=top_n[ 'centrality' ]),
              secondary_y= True )

# Додати заголовок фігури
fig.update_layout(
    title_text= f"Оцінки суміжності та центральності (ступінь
центральності), 2018, Top {n} ",
    yaxis= dict (title= 'Суміжності' ),
    yaxis2= dict (title= 'Оцінка центральності',
                  накладення= 'y',
                  side= 'right' ),
    legend= dict (orientation= 'h', xanchor= ' center',
x= 0,5, y= -0,2)
)

fig.update_traces(hovertemplate= None )
fig.update_layout(hovermode= "x unified" )

fig.show()
```

В обох візуалізаціях на рис. 2.6.4 і 2.6.5 центральний вплив Китаю в мережі торгівлі корисними копалинами 2018 року достатньо очевидний через його високий ступінь центральності. Вплив Китаю додатково підкреслюється простим спостереженням — його кількістю торговельних партнерів порівняно з іншими країнами.

Показники центральності, яких є багато, часто виявляються корисними в частинних випадках.

Adjacencies and Centrality Scores (Degree Centrality), 2018

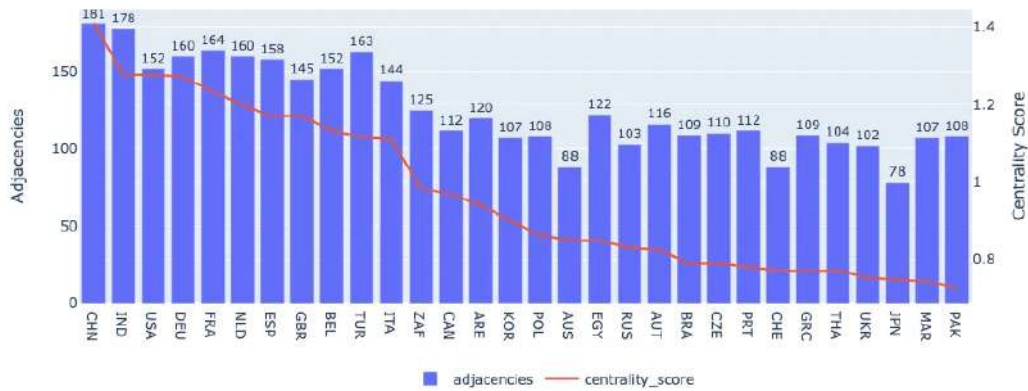


Рис. 2.6.4. Вплив у мережі

```

Код:
fig = px.scatter(df_node_attributes,
                x='centrality',
                y='adjacency',
                title='Adjacencies and Centrality Scores (Degree
Centrality), 2018',
                text='country'
                )

fig.update_layout(yaxis=dict(title='Adjacencies'),
                  legend=dict(orientation='h', xanchor='center',
x=0.5, y=-0.2)
                  )
    
```

Adjacencies and Centrality Scores (Degree Centrality), 2018

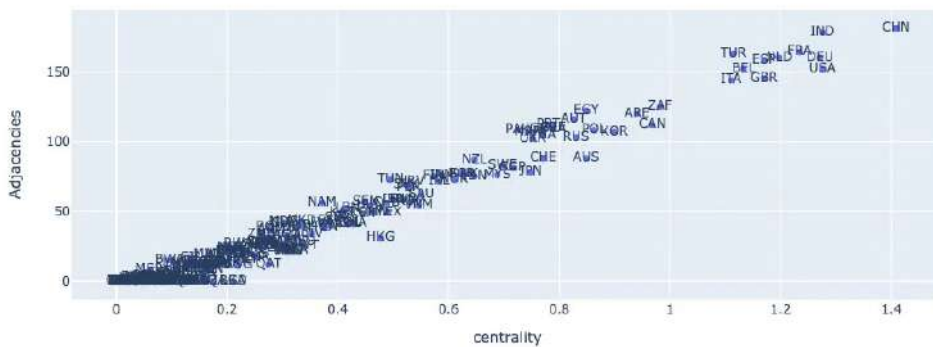


Рис. 2.6.5. Вплив у мережі (інший тип діаграми)

ЗАВДАННЯ ДЛЯ МОЗКОВОГО ШТУРМУ



1. Смарт-аналітика охоплює різноманітні методи та процеси, спрямовані на визначення сенсу великих даних для дослідницьких і бізнес-цілей. Спираючись на літературні джерела, доступні як в академічних журналах, так і в Інтернеті, обговоріть відмінності між звітністю, аналітикою даних і онлайн-аналітичними процесами, а також спосіб, у який вони всі сприяють наданню розвідувальної інформації про предмет дослідження.

2. Завантажте інструмент візуалізації інформації, наприклад «Tableau» або «Microsoft Power BI». Для цієї вправи буде достатньо пробної версії. Використовуйте власні дані (якщо такі є) або використовуйте один із наборів даних, що постачається разом із інструментом (такі інструменти зазвичай мають один або кілька наборів даних для демонстраційних цілей). Вивчіть дані, придумайте кілька бізнес-проблем і використайте візуалізацію даних для аналізу та потенційного вирішення цих проблем.

3. Визначіть, для створення кожного з наведених нижче додатків, що було б краще використовувати: статистичний аналіз чи машинне навчання? Поясніть свої відповіді, включаючи можливі винятки чи особливі умови.

- a) діагностика точно встановленого, але складного захворювання;
- b) підсистема пошуку цін для продавця великого обсягу товарів;
- c) автоматизована система обробки голосових запитів;
- d) навчання нових співробітників;
- e) розпізнавання рукописного тексту.

4. Аналіз конкурентів є однією зі сфер, де методи смарт-аналізу тексту особливо корисні для бізнесу. Проведіть деякі дослідження в Інтернеті, напишіть звіт, у якому пояснюється, як цього можна досягти за допомогою методології, розглянутої в цьому розділі.

5. Netflix зберігає дані про інтереси клієнтів, використовуючи велику дводольну мережу, що з'єднує користувачів з кінофільмами, які вони переглянули та/або оцінили. Бібліотека фільмів Netflix містить близько 100 000 назв, якщо зараховувати поточкову передачу та відправку DVD поштою. У четвертому кварталі 2013 року Netflix повідомила, що має близько 33 млн. користувачів. Припустимо, що середній ступінь користувача в цій мережі становить 1000. Приблизно скільки зв'язків у цій мережі? Чи вважаєте ви цю мережу розрідженою чи щільною? Поясніть.

LAB 2.

РОЗВІДУВАЛЬНИЙ АНАЛІЗ ЦІН НА AMAZON.COM.

Безпека міжнародних контрактів

Мета: Дослідити набір даних цін на ноутбуки на Amazon (Amazon.com — один із перших інтернет-сервісів, орієнтованих на продаж реальних товарів масового попиту, найбільша у світі за обігом компанія, що продає товари та послуги через Інтернет) за 2023 р.

Формат даних: Набір даних доступний у структурованому форматі amazon_laptop_prices_v01, CSV. Доступний за посиланням: <https://www.kaggle.com/datasets/talhabarkaataahmad/laptop-prices-dataset-october-2023>

Збір даних: Цей набір даних складається з повної колекції останніх доступних ноутбуків, взятих із Amazon.com. Дані містять такі відомості про продукт, як рейтинг, ціна, операційна система, назва, кількість відгуків і розмір дисплея. Цей набір даних містить приблизно 4.5K даних.

Опис набору даних по стовпцях:

- brand: Марка ноутбука
- model: Модель ноутбука
- screen size: Розмір дисплея ноутбука в дюймах.
- color: Колір ноутбука
- hard disk: Жорсткий диск в GBs або TBs
- CPU: Встановлений процесор
- RAM: обсяг RAM
- Operating system (OS): Операційна система ноутбука
- Special Features: Додаткові функції
- Graphics card: Тип графічної карти
- Graphics coprocessor: Графічна карта
- CPU rating: рейтинг процесора
- Rating: Середня оцінка клієнтів з 5 на жовтень 2023 року.
- Price: Ціна товару в доларах США на жовтень 2023 року.



Завдання. Виконайте дослідницький аналіз даних (EDA) з даним набором, та побудуйте відповідну візуалізацію. Дослідницький аналіз даних (EDA) є важливим кроком у будь-якому науковому проєкті даних, оскільки він допомагає отримати уявлення, визначити закономірності та зрозуміти структуру нашого набору даних.

1. *Завантаження даних.* Почніть із завантаження набору даних у Python за допомогою таких бібліотек, як Pandas або NumPy.
2. *Дослідження даних.* Використовуйте Pandas для дослідження набору даних. Перевірте відсутні значення та визначте тип даних кожного стовпця, перевірте всі унікальні категорії даних. Розрахуйте підсумкову статистику.
3. *Обробка відсутніх даних.* Визначте відсутні значення у своєму наборі даних і вирішіть, як з ними впоратися.
4. *Виявлення викидів.* Виявіть та обробіть викиди за допомогою статистичних методів або методів візуалізації.
5. *Кореляційний аналіз.* Дослідите зв'язки між змінними, обчисліть коефіцієнти кореляції та побудуйте теплові карти кореляції. Визначте високо корельовані метрики та визначте їхній вплив на дані.
6. *Візуалізація даних.* Використайте Matplotlib або Seaborn для створення значущих візуалізацій. Оберіть такі методи, як гістограми, діаграми розсіювання, прямокутні діаграми та кореляційні матриці, щоб отримати уявлення про розподіл, зв'язки та викиди у даному наборі даних.
7. *Перевірка гіпотези.* Визначте описові питання щодо даного набору даних. (описові запитання мають на меті отримати розуміння чогось конкретного. Це може включати опис сукупності, зв'язок між різними змінними або різними тенденціями. Ці типи запитань, як правило, найлегше визначити — вони зазвичай стосуються теперішнього стану чи минулого). Зробіть висновки або гіпотези щодо даного набору даних.

ЧОТИРИ ЦІЛІ СМАРТ АНАЛІТИКИ



1. У Державному стандарті України ДСТУ 2938-94 («Системи оброблення інформації») штучний інтелект визначається як «здатність систем оброблення даних виконувати функції, що асоціюються з інтелектом людини, — логічне мислення, навчання та самовдосконалення». У Концепції розвитку штучного інтелекту в Україні, схваленій Розпорядженням Кабінету Міністрів України від 2 грудня 2020 р. №1556-р., надається таке визначення: ШІ — це організована сукупність інформаційних технологій, із застосуванням якої можливо виконувати складні комплексні завдання за допомогою використання системи наукових методів досліджень і алгоритмів обробки інформації, отриманої або самостійно створеної під час роботи, а також створювати та використовувати власні бази знань, моделі прийняття рішень, алгоритми роботи з інформацією та визначати способи досягнення поставлених завдань.
2. У свою чергу, галузь штучного інтелекту — це напрям діяльності у сфері інформаційних технологій, який забезпечує створення, впровадження та використання технологій штучного інтелекту.
3. Аналіз міжнародних торгових мереж за допомогою бібліотек **NetworkX** і **Plotly** забезпечує потужну основу для розуміння складної динаміки глобальних торгових відносин. Завдяки інтеграції цих бібліотек є можливим виявити приховані моделі, визначити ключових гравців і отримати цінну інформацію про торговельні мережі.
4. Хоча вимірювання центральності дає цінну інформацію про торговельні мережі, важливо зазначити, що доступна маса інших показників і алгоритмів аналізу мережі, включаючи коефіцієнти кластеризації, виявлення спільноти та інші корисні метрики.

Розділ 3. СМАРТ-БЕЗПЕКА

3.1. Ідентифікація систем кібербезпеки



Ідентифікація — це процес побудови математичних моделей за експериментальними даними, а також на основі деякої апіорної інформації з точністю, яка задовольняє вибраному критерію ідентифікації. **Ідентифікація систем** — це сукупність методів для побудови математичних моделей динамічної системи за даними спостережень. Ключовим завданням є знайти такі умови для експерименту і способи параметризації проблеми, при яких знайдена модель наблизиться до єдиного точного опису реальної системи.

Початок ідентифікації систем, як предмета побудови математичних моделей на основі спостережень, пов'язують з роботою Карла Фрідріха Гаусса, в якій він використовував розроблений ним метод найменших квадратів для передбачення траєкторії руху планет. Згодом цей метод знайшов застосування в безлічі інших додатків, в тому числі і для побудови математичних моделей керованих об'єктів, які використовуються в автоматизації.

3.1.1. Математичне моделювання



Математична модель у даному контексті означає математичний опис поведінки будь-якої системи, пристрою чи процесу в частотній або часовій сфері, наприклад, фізичних процесів (рух механічної системи під дією сили тяжіння), економічного процесу (реакція біржових котирувань на зовнішні обурення) і та ін.

Математична модель — сукупність математичних об'єктів (рівнянь, систем рівнянь і нерівностей, алгебраїчних виразів тощо), математичних символів, що описують мовою, досліджуваній об'єкт і його відносини із навколишнім світом.

Особливість математичного моделювання полягає в тому, що абстрактним віддзеркаленням існуючого чи створюваного об'єкта є його модель, кількісний аналіз якої дозволяє здобути нові знання про нього. Отже, математичне моделювання — це побудова математичної моделі (або вибір наявної «моделі-заготовки»), її дослідження з метою отримання нової інформації про об'єкт і використання для опису властивостей і прогнозу поведінки об'єкта.

Загальна схема моделювання має такий вигляд:

- проблема;

- постановка завдання;
- опис початкових даних;
- формалізація завдання, вибір тимчасової і просторової шкал моделі;
- вибір методу побудови моделі;
- планування експерименту;
- отримання і обробка експериментальних даних;
- ідентифікація невідомих характеристик моделі;
- вибір методу рішення задачі;
- обчислювальний експеримент;
- аналіз і інтерпретація результатів моделювання;
- ухвалення рішень про використання результатів;
- практичне використання моделі.

Спрощено систему в умовах data-driven підходу до прийняття рішень, можна представити діаграмою на рис. 3.1.

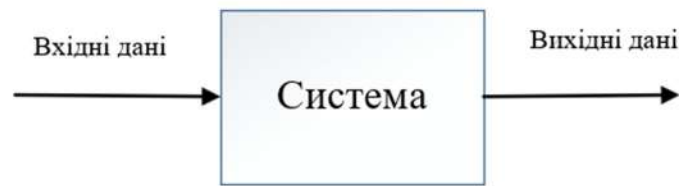


Рис. 3.1. Система

Вхідні дані — це дані, які ми можемо вибрати, наприклад:

- Наскільки відкривати крани для подання гарячої / холодної води?
- Скільки миючого засобу використовувати?
- Скільки їжі з'їсти?

Вихідні дані є наслідками нашого вибору:

- Яка температура води?
- Чи ефективне прибирання?
- Який рівень глюкози в крові?
- Яка вага тіла?

Апріорна інформація існує ще до отримання вимірювальної інформації і характеризує структуру ідентифікованого об'єкта.

Узагальнюючи, можна стверджувати, що люди дуже добре вміють розумно вибирати вхідні дані, щоб досягти бажаного результату, проте людина також:

- схильна до помилок;
- страждає від втоми;
- обмежується відносно простими рішеннями;
- покладається на навчання / досвід;
- вразлива у небезпечних середовищах (спека, холод, хімічні речовини, війни).

Прийняття рішень людиною можна замінити надійною автономною системою, як це показано в табл. 3.1.

Автономні системи

Душ	Душові кабінки можуть мати гарячий і холодний крани (вхід). Ми обираємо, на скільки обертати їх, щоб контролювати відповідні низькі значення і у такий спосіб впливати на вихідну температуру.
Управління двигуном	Тут використовуватиме клапани та інші обмеження для контролю потоку повітря та палива (вхід), що впливає на потужність двигуна і викиди (вихід). На практиці це автоматизовано. Це приклад автономної системи, яка не потребує активного втручання людини.
Національна енергомережа	Оператори мереж вирішують, скільки електроенергії повинен виробляти кожен генератор (вхід), щоб забезпечити відповідність загальної пропозиції (виробництва) попиту. Ми могли б додатково уточнити, що це потрібно робити дуже швидко (за кілька секунд або менше), щоб уникнути перебоїв з електропостачанням та інших небажаних ефектів, коли попит змінюється.
Супутники	Супутникове підключення залежить від позиціонування, тому вони постійно змінюються (вихід) за допомогою змін у ракетах і маховиках (входи). На практиці це автоматизовано. Ми могли б уточнити, що ми також зацікавлені в тому, щоб зміна позиціонування була достатньо швидкою та економічною, оскільки супутники не можна дозаправляти.
Рівень цукру в крові	Підшлункова залоза постачає інсулін (вхід) у кров пропорційно рівням цукру в крові (вихід), щоб підтримувати його на бажаному рівні. Діабетики впорядуються з цим вручну, але «роботизовані / автономні» заміни підшлункової залози зараз випробовуються.
Виробництво мінючих засобів	Виробники підтримують температуру, тиск і концентрацію (вхід) у хімічних процесах, щоб забезпечити високу якість продукту та відповідність специфікаціям (вихід).
Швидкість автомобіля	Водій змінює педаль акселератора (вхід), щоб змінити швидкість (вихід). Ми могли б уточнити, що ми також зацікавлені в тому, щоб прискорення було достатньо швидким.

У наведених вище прикладах залежність виходу від входу багато в чому відрізняється від типу (напруга, рух, тиск тощо) до одиниць і масштабу. Наприклад, зміна педалі акселератора на 1–2 см може сильно вплинути на швидкість автомобіля, тоді як зміна на 1–2 см у внутрішньому клапані може виявитися катастрофічною. Кран змінюється обертанням, у виробника може бути джойстик чи миша, а у велосипедиста — лише груба сила в ногах. Отже, потрібно знайти або визначити відповідну мову, щоб уніфікувати дослідження зазначених проблем вище, а також встановити необхідну точність і продуктивність у систематичний спосіб. Недостатньо лише зробити вихідні значення бажаними в якомусь довільному часовому масштабі, потрібно також керувати динамічними змінами вихідних значень.

Функціонування системи засновано на тому, що сонячні промені розігрівають повітря у сонячній панелі, що являє собою набір прозорих трубок. Це повітря подається в тепловий акумулятор, яким використовується заповнений камінням ящик. Так що накопичувана теплова енергія може братися для опалення будинку.

Інтерес становить питання про те, який вплив чинять на температуру теплового акумулятора зміни сонячного випромінювання і швидкості піддування. Заміри проводилися кожні 10 хв.

Ця система динамічна, тобто така, в якій поточне значення вихідного сигналу залежить не лише від поточних, а й від більш ранніх значень зовнішніх впливів.



Лінійна динамічна система — це динамічна система, еволюція в часі якої описується лінійним диференціальним рівнянням. Хоча загалом динамічні системи не мають розв'язку у замкненій формі, лінійні динамічні системи можна розв'язати достатньо точно, і вони мають багатий набір математичних властивостей.

Наприклад, по процесу споживання електричної енергії комп'ютер є безперервною динамічною системою, а по обробці цифрової інформації — дискретною; резервуар з насосом для закачування рідини — лінійна динамічна система за рівнем рідини в резервуарі, а цей же резервуар з компресором для закачування газу є нелінійною динамічною системою по тиску газу в резервуарі; електропривод компресора, який має зворотний зв'язок по тиску газу на початку трубопроводу і стабілізує цей тиск.

За структурою об'єкти поділяються на динамічні і статичні; стохастичні і детерміновані; нелінійні і лінійні; дискретні і безперервні.

Задача ідентифікації. Подамо модель об'єкта у вигляді такої схеми як на рис. 3.2.

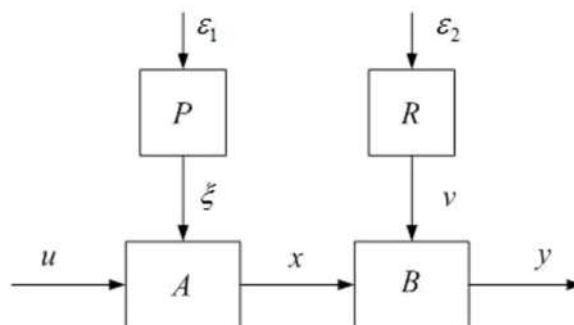


Рис. 3.2. Структурна схема моделі об'єкта

де u і y — вхідні і вихідні сигнали, які можуть бути детермінованими чи випадковими, можуть бути сумішшю (зазвичай адитивної) детермінованою і випадкової складових. Вхідні сигнали можуть спеціально подаватися в систему для ідентифікації (активний експеримент), а можуть існувати в системі як керуючі або впливові (пасивний експеримент);

x — неспостережуваний сигнал, який оцінюється побічно по сигналу y , отриманому в результаті перетворення в об'єкті оператором B ;

ε_1 і ε_2 — неспостережувані перешкоди, які є, як правило, випадковими процесами на зразок білого шуму, в деяких випадках містять детерміновані складові;

ξ і ν — частіше неспостережувані, зазвичай корельовані у часові випадкові сигнали, в деяких випадках містять детерміновані складові;

A, B, P, R — оператори, в деяких випадках їх вид невідомий, в інших — відомий, проте невідомі параметри.

За наведеною структурною схемою моделі об'єкта, основними завданнями ідентифікації можна визначити:

1. *Завдання пошуку характеристик (параметрів) об'єкта.* За відомими спостережуваними змінними u і y потрібно визначити оператори (або параметри операторів) A і B . Часто одночасно із визначенням параметрів A і B потрібно встановити параметри операторів P і R , що перетворюють неспостережувані білі шуми ε_1 і ε_2 у неспостережувані сигнали ξ і ν .
2. *Завдання оцінювання змінних стану.* Стан об'єкта характеризується багатовимірною змінною стану, вектором, однозначно визначаючим усі його характеристики. По відомих операторах сигналів u і y при відомих операторах A, B, P, R з відомими параметрами потрібно оцінити неспостережний випадковий сигнал x . При цьому можливі такі постановки задачі: а) оцінювання x у поточній момент часу — задача фільтрації, або, власне, оцінювання; б) оцінювання x у майбутній момент часу, зрушений на t щодо поточного моменту — завдання прогнозування, або екстраполяції; в) оцінювання x у минулий момент часу — завдання згладжування, або інтерполяції.
3. *Завдання генерації випадкових сигналів із заданими характеристиками або визначення характеристик випадкових сигналів.* За спостережуваними змінними ξ або ν потрібно визначити оператор (або параметри оператора) P (або R).

У деяких випадках виникає задача, при якій одночасно проводиться параметрична ідентифікація A, B, P, R та оцінювання x (одночасна ідентифікація та оцінювання), а також можливий ряд інших постановок задач ідентифікації та оцінювання.

У більшості робіт, присвячених ідентифікації, виділяються такі основні складові, які потрібно виконати на етапі ідентифікації:

1. Сформулювати вимоги до даних спостережень: як виконати збір експериментальних даних, як використовувати ці дані, зібрані в реальних умовах проведення експерименту.
2. Визначити клас об'єктів — сукупність моделей-кандидатів, з яких згодом буде відібрана найкраща модель.
3. Сформулювати функцію втрат або ризику, що характеризує адекватність об'єкта і налаштовуваної моделі, та на її основі сформулювати критерій якості ідентифікації.

4. Вибрати спосіб оцінки рівня відповідності досліджуваної моделі експериментальним даним.
5. Визначити процедуру верифікації моделі: провести перевірку і підтвердження адекватності моделі, тобто з'ясувати, якою мірою модель дійсно пояснює поведінку досліджуваної системи.

3.1.2. Процедура ідентифікації



Процедура ідентифікації має природний логічний порядок: спочатку збираємо дані, потім формуємо безліч моделей і потім вибираємо найкращу модель. Звичайним явищем буває, що перша обрана модель не проходить тест на відповідність експериментальним даним. Тоді слід повернутися назад і вибрати іншу модель або змінити критерії пошуку.

Модель може бути незадовільною через такі випадки:

- Чисельний метод не може знайти модель, відповідну до обраного критерію.
- Неправильно обраний критерій.
- Неправильно сформований безліч моделей, у ньому взагалі може не бути якісної моделі.
- Зібрані дані неінформативні.

При ідентифікації передбачається експериментальне вивчення і зіставлення вхідних та вихідних процесів, і завдання ідентифікації полягає у виборі відповідної математичної моделі. Модель повинна бути такою, що її реакція і реакція об'єкта на один і той же вхідний сигнал повинні бути в даному разі близькими. Результати рішення задачі ідентифікації є вихідними даними для проектування систем управління, оптимізації, аналізу параметрів систем і та ін.

Кожне ідентифікаційне дослідження складається з певних стадій / етапів, хоча різні види і форми ідентифікації мають неоднакову кількість стадій та відмінні назви. Найбільш чітко стадії виражені в експертній формі ідентифікації:

- 1) попереднє дослідження, або огляд об'єктів;
- 2) роздільне дослідження;
- 3) експериментальне дослідження (необов'язкова стадія);
- 4) порівняльне дослідження;
- 5) аналіз результатів дослідження і формулювання висновків.

На етапі попереднього дослідження, або огляду об'єктів, відбувається підготовча робота: встановлюється наявність потрібних матеріалів, придатність об'єктів для дослідження тощо.

Під час роздільного дослідження як такі підлягають вивченню ознаки ототожнюваного і ототожнюваних об'єктів, для чого використовують як власне об'єкти, так і їх відображення або фрагменти, зразки виявлені ознаки окремо фіксують, установлюється специфічна для кожного об'єкта сукупність загальних і окремих ознак.

Експериментальне дослідження спрямовано на одержання в умовах, максимально наближених до тих, за яких сталася подія, порівняльних зразків, що містять достатню сукупність ознак отожднюваного об'єкта.

На стадії порівняльного дослідження відбувається порівняння різними способами (зіставлення, суміщення, накладення) виявлених ідентифікаційних ознак (від групових до окремих), якими наділений кожен із досліджуваних об'єктів; визначається, які ознаки збігаються, а які відрізняються.

Заключною стадією є аналіз результатів порівняльного дослідження і формулювання висновку. Виявлені ідентифікаційні ознаки, що збігаються і відрізняються, підлягають аналізу й оцінюванню з позицій їх закономірностей та значущості, суттєвості відмінностей, наявності необхідної та достатньої сукупності / комплексу загальних / групових, окремих ознак для відповідного висновку.

Існують такі *критерії ідентифікації*:

- Q критерій Кохрена;
- інформаційний критерій Акаїке;
- інформаційний критерій відхилення;
- фокусний інформаційний критерій;
- інформаційний критерій Геннена — Куїнна;
- критерій перехресної перевірки;
- Баєсівський інформаційний критерій;
- критерій Шварца.



Інформаційний критерій Акаїке — це міра відносної якості статистичних моделей для заданого набору даних. Маючи сукупність моделей для цих даних, ІКА оцінює якість кожної з моделей відносно кожної з інших моделей. Отже, ІКА пропонує засоби для обирання моделі. ІКА засновано на теорії інформації: він пропонує відносні оцінки втраченої інформації у разі застосування цієї моделі для представлення процесу, що породжує дані. Роблячи це, ІКА має справу з компромісом між пристосованістю моделі та її складністю.

Узагальнено методи ідентифікації класифікуються за такими ознаками.

1. За способом тестування розрізняють активні і пасивні методи ідентифікації. В активних методах на вхід об'єкта подаються спеціально сформовані — тестові сигнали — детермінованого чи випадкового характеру. Перевагами цього підходу є мінімальні вимоги до апріорних відомостей про об'єкт, цілеспрямований характер ідентифікації і як наслідок — зменшення тимчасових і матеріальних витрат на проведення експерименту.
2. У разі використання пасивних методів об'єкт перебуває в умовах нормального функціонування, і параметри моделі відшукуються за результатами статистичної обробки спостережень. Перевагами цього підходу можна вважати відсутність необхідності проводити спеціальні дослідження об'єкта,

досить лише виміряти спостережувані сигнали в режимі робочого функціонування об'єкта з подальшим розрахунком параметрів моделі. Недоліками такого підходу є значні тимчасові витрати на збір і необхідну статистичну обробку даних і жорсткі вимоги до частотного спектра вхідного впливу — він не повинен бути менше смуги частот динамічної характеристики ідентифікованого об'єкта.

3. За характером використовуваних сигналів розрізняють детерміновані і статистичні методи. Під час проведення активної ідентифікації на основі детермінованих сигналів можливе застосування детермінованих методів ідентифікації. В реальних умовах сигнали завжди схильні до дії перешкод і дуже зашумлені, і тому детерміновані алгоритми слід доповнювати статистичним усередненням (згладжуванням) одержуваних результатів.
4. За ознакою тимчасових витрат методи поділяють на оперативні і ретроспективні. За оперативної ідентифікації забезпечується поточне відстеження мінливих характеристик об'єкта.

Функції передачі TF — це простий спосіб моделювання фізичної динаміки за допомогою наближеної лінійної моделі. Ці функції подано у формі дробу поліномів, які мають чисельник, корені якого називаються нулями, і знаменник, корені яких називаються полюсами. Ці корені характеризують поведінку системи.

Функції передачі легко реалізуються за допомогою **бібліотеки Scipy**. Потрібно лише створити список або масив **numru** з коефіцієнтами полінома чисельника та інший для полінома знаменника. Потім створити об'єкт TF.



Ідентифікація системи першого порядку. Функція передачі моделює відношення між входом і відповідним виходом. Отже, якщо потрібно ідентифікувати якусь систему за допомогою функції передачі, слід проаналізувати її вихідний сигнал як відповідь на ступінчастий вхід.

Систему першого порядку можна налаштувати лише за двома параметрами: K і τ , або посилення і постійна часу. Співвідношення «приріст вихідного значення / приріст вхідного значення» відповідає посиленню системи. Завжди вимірюється сигналами в постійному стані:

$$K = \frac{\Delta y}{\Delta u} = \frac{(y_{ss} - y_0)}{(u_{ss} - u_0)}$$



Якщо ми не маємо доступу до даних введення–виведення, неможливо ідентифікувати систему. Принаймні цим методом.

Постійна часу, τ , регулює швидкість реакції системи. Чим менша константа, тим швидше система. Значення цієї константи відповідає часу, за який вихідний сигнал досягає $\sim 63,2\%$ свого значення в стаціонарному стані з моменту зміни вхідного кроку.

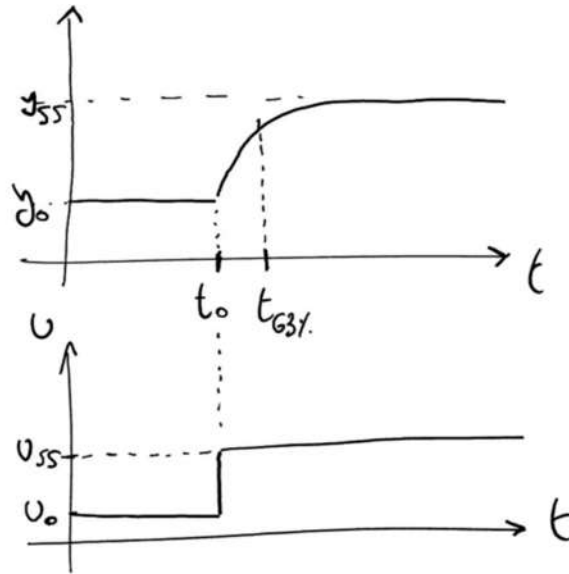


Рис. 3.3. Ідентифікація системи першого порядку

Постійна часу τ регулює швидкість реакції системи. Чим менша константа, тим швидше система. Значення цієї константи відповідає часу, за який вихідний сигнал досягає $\sim 63,2\%$ свого значення в стаціонарному стані з моменту зміни вхідного кроку:

$$\tau = t(y_{63.2}) = t(0.632(y_{ss} - y_0) + y_0)$$

Обчисливши ці параметри, матимемо просту лінійну модель як наближення реальної системи з рівнянням (з формулюванням Лапласа):

$$G(s) = \frac{K}{\tau s + 1}$$

По-перше, запишемо код, щоб отримати коефіцієнт підсилення і значення y , яке *потрібно* знайти, щоб записати час; і по-друге, щоб запустити моделювання за допомогою системи:

```
import numpy as np
from scipy import signal as sig

def param_identification(u0, u_ss, y0, y_ss):
    # Попередні значення мають бути витягнуті з відповіді на крок
```

```

# Зверніть увагу, що y63 має бути знайдено в оригінальному сигналі,
щоб отримати час
gain = (y_ss - y0) / (u_ss - u0)
y63 = 0.632*( y_ss-y0 ) + y0
return [gain, y63]

def run_second_order_mdl(t, u, k, tau):
    self.tf = sig.TransferFunction(k, [tau, 1])
    to, yo, xo = sig.lsim2(self.tf, U=u, T=t)
    return yo

```

3.1.3. Автоматична ідентифікація систем



У випадку якщо є більш складна динаміка, визначити параметри вручну нелегко. Отже, потрібно використовувати чисельний метод для їх наближення.

Існує досить багато методів ідентифікації системи, від метода Монте-Карло до штучних нейронних мереж, але всі вони працюють схожим чином. Ідея полягає в тому, щоб вибрати певні критерії оцінки та налаштувати за допомогою певного алгоритму параметри моделі, щоб відповідати наявним даним.

У багатьох випадках використовується критерій найменших квадратів. Для нашої задачі матимемо два сигнали: вихідний (реакція системи на кроковий вхід) і сигнал апроксимованої системи. Потім для кожного моменту матимемо різницю між вихідним сигналом і наближенням та зводимо його в квадрат. Показником якості апроксимації є сума всіх доданків: чим вона менша, тим краще ідентифікація.

Послідовно параметри змінюються в ітераційному процесі, перевіряючи, які модифікації параметрів покращують використовувані критерії, поки він не досягне певної стабільності протягом ітерацій, і ідентифікація завершується.

Бібліотека Scipy має функцію для оптимізації параметрів запропонованої моделі (користувачем), щоб зменшити квадратичну помилку щодо заданого сигналу даних (користувачем). Це дуже часто використовується в статистиці для перевірки шаблону деяких зразків. Але в цьому випадку будемо використовувати безперервне моделювання лінійної системи, щоб налаштувати модель.

Код:

```

import numpy as np
from scipy import signal as sig
from scipy import optimize as opt
class TF_identificator:
    def __init__(self):
        self.tf = None
        self.inputs = None

```

```

def first_order_md1(self, t, k, pole):
    self.tf = sig.TransferFunction(k, [pole, 1])
    to, yo, xo = sig.lsim2(self.tf, U=self.inputs, T=t)
    return yo

def second_order_md1(self, t, k, wn, delta):
    self.tf = sig.TransferFunction(k*(wn**2), [1, 2*delta*wn, wn**2])
    to, yo, xo = sig.lsim2(self.tf, U=self.inputs, T=t)
    return yo

def identify_first_order(self, t, u, orig_output, method='lm',
p0=[1.0, 1.0]):
    self.inputs = u
    params, params_cov = opt.curve_fit(self.first_order_md1, t,
orig_output,
                                     method=method, maxfev=1000, p0=p0)
    return {'k': params[0], 'tau': params[1]}

def identify_second_order(self, t, u, orig_output, method='lm',
p0=[1.0, 1.0, 0.1]):
    self.inputs = u
    params, params_cov = opt.curve_fit(self.second_order_md1, t,
orig_output,
                                     method=method, maxfev=1000, p0=p0)
    return {'k': params[0], 'wn': params[1], 'zeta': params[2]}

```

Це невеликий клас, створений для цієї мети. Наразі він працює лише з моделями першого та другого порядків, але можна параметризувати його для будь-якої поліноміальної функції передачі. По суті, ви додаєте метод із системою, яка, на вашу думку, відповідатиме краще, і запускаєте його.

Дано часовий масив із відповідним вихідним сигналом; щоб використовувати цей клас із системою першого або другого порядків, далі просто треба створити екземпляр класу TF_identificator і передати йому вхідні, вихідні і часові сигнали. Це означає, що попередньо потрібно експериментувати з досліджуваною системою, щоб зібрати ці сигнали.

Нижче наведено деякі приклади використання класу. Код:

```

tf_ident = TF_identificator()
params = tf_ident.identify_second_order(t_list, u_list, y_list,
p0=[200, 0.05, 0.8])
print("Obtained params:")
print(params)

```

Як бачите, є деякі додаткові параметри, хоча вони не абсолютно необхідні.

- **p0:** Початкові значення для параметрів. Це може бути обов'язковим через низьку здатність методу curve_fit() досліджувати простір рішень.

- **method:** Існує кілька методів оптимізації з різними властивостями. Наприклад, метод "trf" дозволяє застосовувати обмеження для параметрів.
- **maxfev:** Максимальна кількість ітерацій для пошуку параметрів.

Якщо сигнали, які були отримані від реальної системи, мають невеликий шум, інструмент автоматичної ідентифікації системи все одно надасть правильні параметри без додаткових зусиль (рис. 3.3).

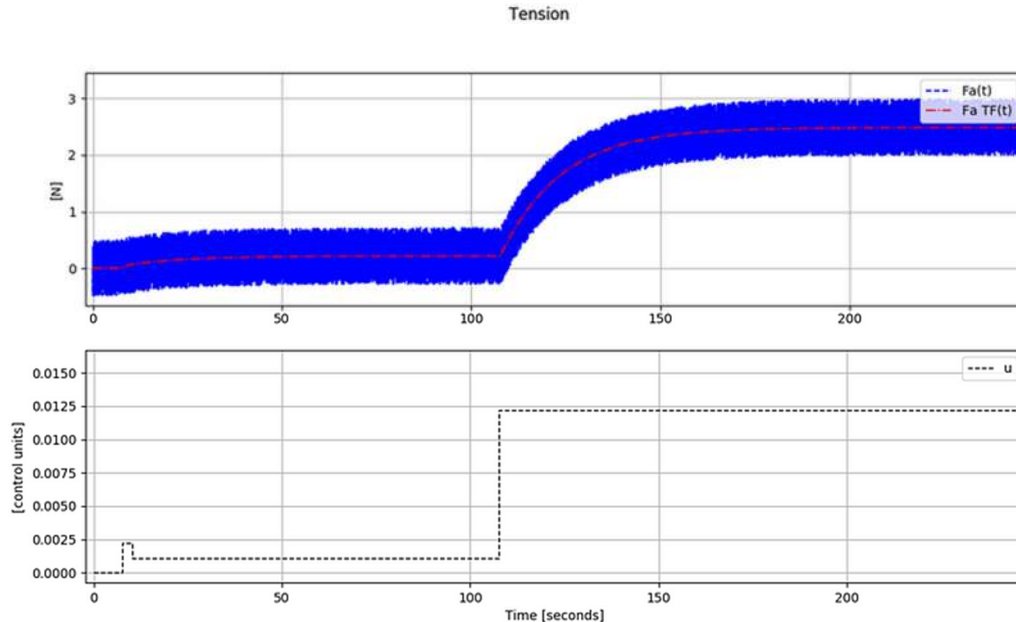


Рис. 3.4. Параметри передавальної функції, отримані з зашумленого сигналу (синій)

Якщо система не має вхідного сигналу, оскільки він залежить лише від часу, ви можете замінити модель (наприклад, метод `second_order mdl`) математичною функцією-кандидатом. **Перегляньте приклад у довіднику Scipy.**

Запитання для самоперевірки



1. Що таке система?
2. Ідентифікація систем, що це?
3. На що за структурою поділяються об'єкти?
4. Опишіть процедуру ідентифікації.
5. Назвіть шляхи вирішення задач ідентифікації?
6. Які бібліотеки мови Python застосовуються для ідентифікації систем?
7. Як проходить ідентифікація систем першого порядку?
8. Як налаштувати автоматичну ідентифікацію систем?

3.2. Аналітика даних щодо кіберзлочинів



Уся інформація в Інтернеті передається *пакетами*, які можна подати у вигляді конверта з даними, що підлягають надсиланню. Як і у випадку зі звичайною поштою, ці пакети надсилаються за вказаною адресою.

Розділ «Від кого» на конверті містить два важливі фрагменти інформації: 1) ім'я відправника; 2) його адресу. Так само у пакетів є MAC-адреса джерела, яка представляє машину, що відправила пакет, і IP-адреса джерела, який вказує, звідки надійшов цей пакет.

3.2.1. Пакети

Інші аналогічні поля, які називаються **заголовками пакета**, визначають пункт його призначення. Для сортування і пересилання пакетів в Інтернеті використовуються пристрої, які називають **маршрутизаторами**. Пакети переміщуються від одного маршрутизатора до іншого, як листи пересилаються між поштовыми відділеннями.

Наприклад, ноутбук оснащений мережевою картою (network interface card, NIC), що дозволяє йому підключатися до маршрутизаторів Wi-Fi. Ця карта має унікальну **MAC-адресу** (media access control, управління доступом до середовища), що дозволяє ідентифікувати ваш комп'ютер у мережі. Перед відправкою інформації на ваш комп'ютер маршрутизатор позначає відповідний пакет MAC-адресою ноутбука, а потім передає його у вигляді радіосигналу.

Всі комп'ютери, підключені до цього маршрутизатора, отримують цей радіосигнал і перевіряють MAC-адресу пакета, щоб з'ясувати, якому з них він призначений. Як правило, MAC-адреси є 48-бітовими значеннями, записаними в шістнадцятковому форматі (наприклад, 08:00:27:3b:8f:ed).

Відомо, що IP-адреси також дозволяють ідентифікувати комп'ютери у мережі. Навіщо ж ще й MAC-адреси? Справа в тім, що мережі складаються з ієрархічних регіонів, як деякі країни діляться на штати, які у свою чергу діляться на міста. IP-адреси враховують структуру, що дозволяє їм визначати місце пристрою у більшій мережі.

Якщо ви перейдете в інше кафе, то вашому ноутбуку буде призначено нову IP-адресу, що відображатиме його нове місце, проте при цьому ваша MAC-адреса залишиться незмінною. IPv4-адреса кодує інформацію про мережну ієрархію у вигляді 32-бітного числа. Зазвичай воно розділяється крапками на чотири розділи (наприклад, 192.168.3.1). Кожен розділ — це восьмибітне двійкове число. Наприклад, 3 в 192.168.3.1 — це восьмибітне двійкове число 00000011.

Як маршрутизатор дізнається MAC-адресу комп'ютера з IP-адресою 128.143.67.11? Для цього використовується протокол ARP. Маршрутизатор

відправляє повідомлення, зване ARP-запитом, усім комп'ютерам у мережі, пропонуючи машині з IP-адресою 128.143.67.11 надіслати ARP-відповідь зі своєю MAC-адресою.

Потім маршрутизатор зберігає результат зіставлення IP-адреси і MAC-адреси в спеціальній ARP-таблиці, що позбавляє його необхідності повторно відправляти ARP-запити в найближчому майбутньому.



MAC-адреса ідентифікує ваш комп'ютер, IP-адреса визначає його розташування, а ARP-таблиці зберігають результати зіставлення цих фрагментів інформації. Під час проведення ARP-спуфінгу ми прагнемо видати себе за когось іншого.

Крім IP-адрес, заголовок пакета містить поля для номерів портів джерела і приймача, які призначаються операційною системою, коли вона дозволяє процесу обмінюватися даними по мережі. Номери портів унікальні. Це означає, що жодні два процеси, запущені на комп'ютері, не можуть використовувати один і той самий номер порту.

3.2.2. Модельні проблеми



Процес — це абстракція, що представляє програму, яка виконується. Наприклад, у разі відкриття браузера операційна система, встановлена на комп'ютері, запускає процес, пов'язаний із цим браузером. Коли процес збирається надсилати й отримувати інформацію по мережі, операційна система призначає йому номер порту, який можна уподібнити до морського порту.

Наприклад, пакети, призначені для вебсерверу, зазвичай надходять на IP-адресу 192.168.1.100 на порт 443. Іншими словами, порти відкривають для мережі доступ до внутрішніх процесів.

Пакет Wireshark — це інструмент, який дозволяє перехоплювати та переглядати пакети, що проходять через мережну карту комп'ютера. За замовчуванням він встановлений у більшості систем Kali Linux. Щоб запустити Wireshark, виберіть пункт меню Applications Sniffing and Spoofing Wireshark (Програми — сніфінг і спуфінг Wireshark) або відкрийте вікно терміналу та виконайте наступну команду:

```
kali@kali:~$ sudo wireshark
```

Якщо інструмент Wireshark не встановлений, то встановіть його, виконавши команду:

```
kali@kali:~$ sudo apt install wireshark
```

Важливо запустити **Wireshark**, володіючи привілеями користувача root, щоб надати інструменту необмежений доступ до інтерфейсів комп'ютера.

Спочатку згенеруємо якийсь вебтрафік, отримавши доступ до вебсервера на машині Metasploitable. Оскільки не було налаштовано DNS-сервер у процесі налаштування, не можна отримати доступ до сервера Metasploitable, ввівши URL-адресу, наприклад, `http://www.evil.corp/`. Натомість доведеться отримати IP-адресу сервера вручну. Авторизуйтесь у системі Metasploitable, використовуючи ім'я користувача `msfadmin` та пароль `msfadmin`. Після входу в систему виконайте наступну команду для отримання IP-адреси сервера:

```
msfadmin@metasploitable:~$ ifconfig eth0
```

```
eth0      Link encap:Ethernet HWaddr 00:17:9A:0A:F6:44
inet addr: 192.168.1.101  Bcast:192.168.1.255 Mask:255.255.255.0
```

Рис. 3.7. Результат виконання команди

Значення після фрагмента `inet addr:` це і є IP-адреса.

Поверніться до **Kali Linux** і запустіть процес перехоплення пакетів, клацнувши на позначку у вигляді акул'ячого плавця () у верхньому лівому куті екрана Wireshark. Тепер вдамо з себе жертву і згенеруємо пакети, відкривши браузер Firefox і ввівши IP-адресу сервера в адресний рядок, наприклад: `http://192.168.1.101/` (у іншого комп'ютера адреса може бути іншою). На рис. 3.5 показано три основні розділи екрана перехоплення Wireshark.

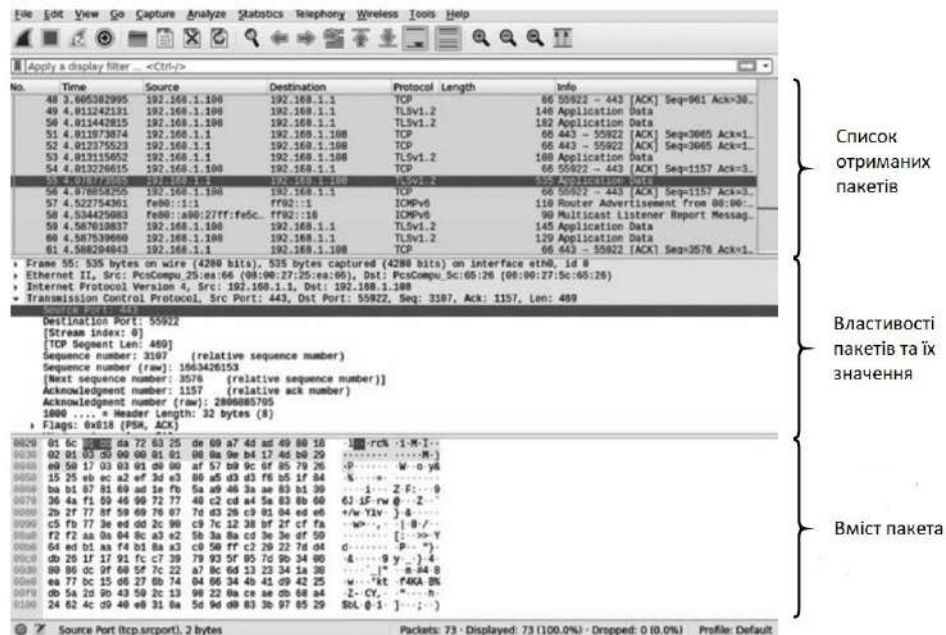
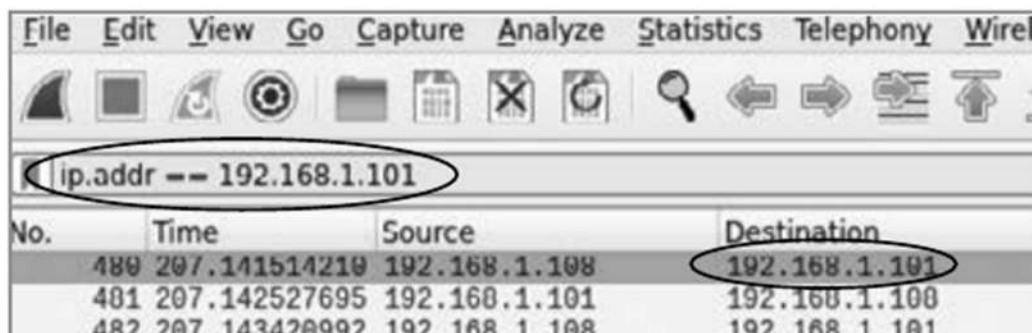


Рис. 3.5. Екран перехоплення Wireshark

Щоб розібратися у всій цій інформації та більше дізнатися про жертву, зловмисник може використовувати фільтрацію, що міститься в Wireshark, що дозволяє знаходити цікаві для нього пакети. Припустимо, що нас цікавлять лише ті пакети, які були відправлені на сервер Metasploitable з IP-адресою 192.168.1.101 (пам'ятайте, що у вашому випадку IP-адреса може бути іншою). Введіть команду, наведену нижче, у полі фільтра Wireshark, щоб відобразити лише ті пакети, якими жертва обмінювалася із сервером Metasploitable.

```
ip.dst == 192.168.1.101
```

За допомогою Wireshark було обрано лише пакети з IP-адресою приймача (ip.dst) 192.168.1.101. Результат наведено на рис. 3.6.



The screenshot shows the Wireshark interface with the filter bar set to 'ip.addr == 192.168.1.101'. Below the filter bar, a table of filtered packets is displayed:

No.	Time	Source	Destination
480	207.141514210	192.168.1.108	192.168.1.101
481	207.142527695	192.168.1.101	192.168.1.108
482	207.143420992	192.168.1.108	192.168.1.101

Рис. 3.6. Результат фільтрації Wireshark

Фільтрування пакетів для відображення лише тих, що були надіслані на сервер, скорочує кількість пакетів, що підлягають аналізу. Як тільки зрозумієте загальний синтаксис фільтрів **Wireshark**, зможете створювати власні. Структура фільтра Wireshark виглядає так:

```
[Protocol].[header/field] [operator: +,==,!=] [value]
```

Спочатку вкажіть протокол ([Protocol]), наприклад, TCP або IP. Потім вкажіть поле пакета, за яким хочете виконати фільтрацію, наприклад IP-адресу джерела (src) або IP-адресу приймача (dst). Нарешті, вкажіть оператора та значення: наприклад, не дорівнює (!=) 192.168.1.10. Використовуючи таку структуру, створюється фільтр, який відображає лише пакети, чия IP-адреса джерела збігається з IP-адресою сервера:

```
ip.src == 192.168.1.101
```

Wireshark також дозволяє фільтрувати пакети на основі їхнього вмісту. Наприклад, є можливість знайти пакети з такими словами, як password, Email або @virginia. Щоб знайти слово login у всіх TCP-пакетах, використовуйте наступний фільтр:

```
tcp contains login
```

Саме у такий спосіб зловмисник може використовувати Wireshark для крадіжки особистих даних із незашифрованих пакетів, перехоплених за допомогою ARP-спуфінгу. І тому важливо забезпечити шифрування вебтрафіку по протоколу HTTPS.

3.2.3. Спосіб виявлення ознак злому мережі



Тепер розглянемо *спосіб виявлення ознак злому мережі* за допомогою цього інструменту. Найпростіший спосіб це зробити — зберегти всі пакети, що проходять через міжмережевий екран, які мають відношення до порту 80. Він майже завжди використовується для обміну даними по протоколу HTTP, тоді як порт 443 зазвичай служить для передачі зашифрованого HTTPS-трафіку.

Запустіть машину Kali Linux і перейдіть на сайт <http://cs.virginia.edu/>. Оскільки увесь трафік нашої мережі проходить через міжмережевий екран pfSense, використовуємо команду tcpdump на pfSense для перехоплення TCP-пакетів з машини Kali Linux. Запустіть pfSense і побачите на екрані таке:

```
Welcome to pfSense                                (amd64) on pfSense

WAN (wan)      -> em0      -> v4/DHCP4: 10.0.1.11/24
LAN (lan)      -> em1      -> v4: 192.168.1.1/24

0) Logout (SSH only)          9) pfTop
1) Assign Interfaces          10) Filter Logs
2) Set interface(s) IP address 11) Restart webConfigurator
3) Reset webConfigurator password 12) PHP shell + pfSense tools
4) Reset to factory defaults    13) Update from console
5) Reboot system              14) Disable Secure Shell (sshd)
6) Halt system                15) Restore recent configuration
7) Ping host                  16) Restart PHP-FPM
8) Shell

Enter an option:
```

Рис. 3.7. Аналіз трафіку

Запустіть оболонку, ввівши значення 8:

Enter an option: 8

[RELEASE][root@pfSense.localdomain]/root:

Потім введіть команду tcpdump в оболонці. Програма, запущена без параметрів, перехоплюватиме усі пакети, що проходять через усі інтерфейси системи, доки не буде завершено її виконання, комбінацією клавіш Ctrl+C. Приклад виведення команди tcpdump виглядає так:

```
[RELEASE][root@pfSense.localdomain]/root: tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on em0, link-type EN10MB (Ethernet), capture size 262144 byte
...
...
15:18:44.372924 IP 192.168.1.100.41193 > z.arin.net.domain
57745% [1au] DS? 41.198.in-addr.arpa (40)
...
```

Звертаємо увагу на те, що трафік організований у вигляді рядків. Проаналізуймо один із них, щоб зрозуміти, які дані виводяться на екран. 15:18:44.372924 — тимчасова мітка, яка вказує на конкретний момент перехоплення трафіку. IP — протокол пакета, а 192.168.1.100.41193 — результат об'єднання IP-адреси і номера порту джерела (номер порту в даному випадку — 41193). z.arin.net.domain. 57745 — IP-адреса і порт приймача. Для покращення читаності tcpdump перетворює цю IP-адресу на відповідне доменне ім'я. Можна вимкнути цю функцію, додавши до команди прапорець -n. Решта — це інформація, що стосується конкретного пакета.

Як і у випадку з Wireshark, є можливим перехоплювати пакети, що передаються за певним протоколом, вказавши його як аргумент tcpdump. Також можна прослуховувати пакети через певний порт, вказуючи його номер.

До того ж замість відображення пакетів у терміналі можна записати їх у файл, отримавши у такий спосіб набір даних, який потім можна буде проаналізувати:

```
tcpdump -i <інтерфейс> -s <кількість пакетів, що підлягають перехопленню> -w <файл.pcap>
```

де параметр -i означає інтерфейс, на якому необхідно перехоплювати пакети. (У попередньому прикладі перехоплювали пакети на інтерфейсі em0.) Для отримання списку всіх інтерфейсів на пристрої виберіть варіант оболонки на стартовому екрані та виконайте команду ifconfig;

прапор -s вказує кількість пакетів, що підлягають перехопленню;

прапор -w — ім'я файлу, в якому зберігатимуться дані.

Щоб не витратити сили на аналіз цих даних, ви можете скористатися такими онлайн-інструментами, як <https://packettotal.com>, які дозволяють виявляти ознаки підозрілої активності у файлах.pcap.



Одними із найбільш корисних журналів в аналізі великомасштабних інцидентів є журнали брандмауера, які надають інформацію про зовнішні та внутрішні мережеві підключення, внутрішній трафік організації та в деяких випадках навіть про доступ користувачів до VPN.

Перший крок — перетворення на формат CSV. Оскільки місія полягає в тому, щоб працювати з журналами Pandas, щоб одержати докази, потрібні журнали у форматі, який може читати Pandas.

Наприклад:

```
def load_log(log_string_list):
    import re
    import pandas as pd
    pattern = re.compile('(\w+)=(?:"([^\"]*)"|(\S*))')
    lines_s = pd.Series(log_string_list)
    data = lines_s.str.extractall(pattern)
    data.index = data.index.get_level_values(0)
    data[1] = data[1].fillna(data[2])
    return data.pivot(index=None, columns=0, values=1)
```

Ця функція дозволяє витягувати заголовки та поля кожного рядка файлу таким чином:

- Видалення груп атрибутів шляхом окремого захоплення ключа даних, розділювача(iv) і даних. Отже, для кожного рядка ми зможемо розділити не дані, а ключі.
- Призначення однакового індексу всім значенням.
- Оскільки є поля, які мають роздільники, генерується три групи. Необхідно згрупувати результати в одному стовпці, виключаючи нульові результати.
- Створення одного рядку із даними, використовуючи стовпець 0 для заголовків і стовпець 1 — для значень.

Через тип форматів журналу дуже часто виявляється, що у файлах є різна кількість стовпців, і це несумісно з бібліотеками аналізу даних. Щоб виправити це, існує бібліотека під назвою `dbtstack`, яка виконує аналіз файлів CSV і повідомляє нам, які стовпці присутні в кожному файлі, а які ні.

Боротьба з потребою у пам'яті. Це дуже поширена проблема в аналізі даних, коли багато даних і недостатньо оперативної пам'яті для їх обробки.

Оскільки наша місія полягає в тому, щоб зберігати стандартизовану інформацію на регулярній основі, зазвичай вибираємо зберігати інформацію в базі даних, таких як Spark або Elastic, але у випадку реагування на інцидент зазвичай доводиться швидко аналізувати тип події. Щоб впоратися з такими ситуаціями, існує Dask.

Це бібліотека аналізу даних дає можливість виконувати запити до файлів журналів у паралельний спосіб, використовуючи ресурси наявного комп'ютера. Іншими словами, вона дозволяє ефективно працювати з великими обсягами даних без необхідності завантажувати їх у пам'ять.

Процес читання файлів швидкий, але займає багато часу, але під час читання файлів за допомогою Dask відповідь миттєва.

```
[1]: import dask.dataframe as dd
import pandas as pd

[2]: %time
df = dd.read_csv(r'C:\logs_vpn_out_fix\*.csv', dtype={'error_num': 'object',
'group': 'object', 'locport': 'object', 'peer_notif': 'object', 'reason': 'object', 'remport': 'object',
'seq': 'object', 'spi': 'object', 'tunnelip': 'object', 'tunneltype': 'object', 'user': 'object',
'desc': 'object', 'espauth': 'object', 'esptransform': 'object', 'in_spi': 'object', 'out_spi': 'object',
'rcvdbyte': 'float', 'stage': 'object', 'tunnelid': 'object', 'Unnamed: 0': 'object', 'assignip': 'object',
'dst_host': 'object', 'duration': 'float', 'eventtime': 'object', 'logid': 'object', 'nextstat': 'object',
'phase2_name': 'object', 'sentbyte': 'float', 'exch': 'object', 'version': 'object'})

Wall time: 64.6 ms
```

Рис. 3.9. Створення Dask DataFrame

Важливо, що на цьому етапі Dask не прочитав файли, він розділив дані за індексами та прочитає їх, коли буде потрібно, саме тому він швидкий на цьому етапі.

Об'єкт `df` не є `Pandas.DataFrame`, це `Dask.DataFrame`, і це треба пам'ятати. Під час створення DataFrame дуже корисно позначити тип даних кожного стовпця, щоб пришвидшити запит і уникнути проблем.

Одним із найпоширеніших випадків використання аналізу журналів є пошук викраденої інформації під час інциденту. Для цього виконаємо запит за допомогою Dask, щоб отримати невелику підмножину даних, з якою можна зручно працювати в пам'яті:

```
df_exf = df[["remip", "tunnelid",
"sentbyte"]].groupby(["remip", "tunnelid"]).max().compute()
```

Виконання цього запиту займе 1 хв. 28 сек., у цьому полягає велика різниця між Pandas і Dask. У Pandas цей запит був би набагато швидшим, якби дані поміщалися в пам'яті. Функція `.compute()` дозволяє створити Pandas DataFrame з результатом Dask.

Важливо знати, як формуються та реєструються такі журнали. Кожного разу, коли користувач здійснює сеанс до іншої IP-адреси через VPN, йому призначається унікальний ідентифікатор під назвою `tunnelid`. У полі `sendbyte` записується кількість інформації, яка залишає організацію під час цього сеансу, але замість того, щоб записувати кількість байтів у кожному з підключень, вони відображаються кумулятивно, так що враховується лише найвище значення з усіх підключень, створених під час сесії.

Щоб полегшити читання інформації в байтах, використаємо такий код:

```
# Конвертація для людини
def sizeof_fmt(num, suffix='B'):
    for unit in ['', 'Ki', 'Mi', 'Gi', 'Ti', 'Pi', 'Ei', 'Zi']:
        if abs(num) < 1024.0:
            return "%3.1f%s%s" % (num, unit, suffix)
        num /= 1024.0
    return "%.1f%s%s" % (num, 'Yi', suffix)
df_exf.reset_index(inplace=True)
```



```
df_exf["h_sentbyte"] = df_exf.sentbyte.apply(sizeof_fmt)
df_ext_10 = df_exf.nlargest(10,"sentbyte").sort_values(
by="sentbyte", ascending=False).reset_index()
```

Перші дві IP-адреси законні і не мають значення для дослідження. Побудуємо графік перших 5 IP-адрес для дослідження (рис. 3.10).

```
import plotly.express as px
import plotly.io as pio
pio.renderers.default = 'iframe'
fig = px.bar(df_ext_10.iloc[2:8], y="remip", x="sentbyte",
title='Destiantion IPs by data sent', orientation='h')
fig.show()
```

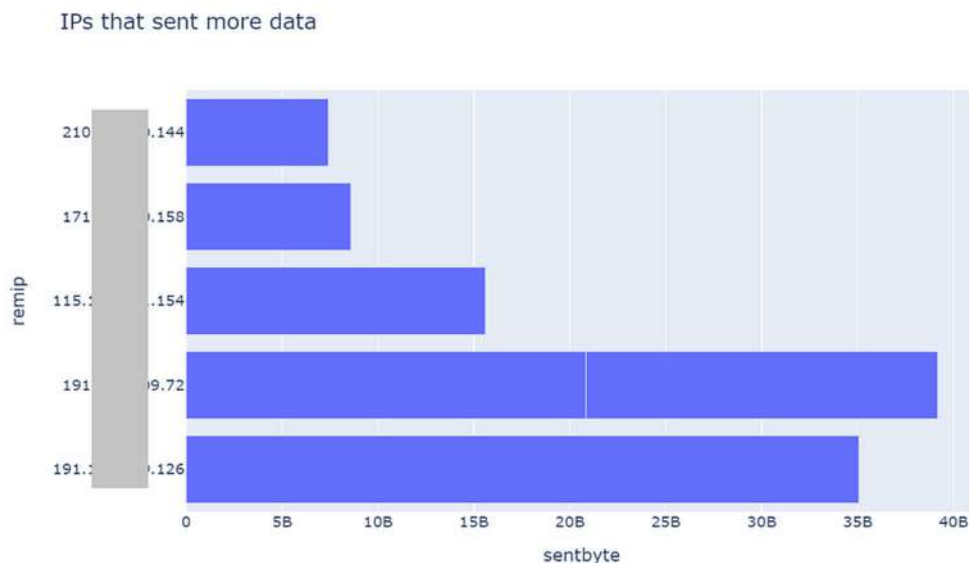


Рис. 3.10. Інформація надсилається на зовнішні IP-адреси

MaxMind ASN (<https://www.maxmind.com>). Ця послуга дозволить отримати AS потрібних нам IP-адрес. Компанія MaxMind пропонує базу даних, яку можна завантажити і переглянути офлайн дуже швидко. Застосуємо таку програму:

```
import geoip2.database
def get_asn(ip):
    try:
        x = reader.asn(ip)
    except:
        return pd.np.nan
    try:
        return x.autonomous_system_organization if x else pd.np.nan
    except KeyError:
```



```

return pd.np.nan
with geoip2.database.Reader(r'C:\maxmind\GeoLite2-ASN.mmdb') as
reader:
    unique_ips = df_exf['remip'].unique()
    unique_ips = pd.Series(unique_ips, index = unique_ips)
    df_exf['ASN'] = df_exf['remip'].map(unique_ips.apply(get_asn))

```

За допомогою цієї функції зможемо отримати назву компанії, якій належать IP-адреси подій, як показано на рис. 3.11.

	index	remip	tunnelid	sentbyte	h_sentbyte	ASN
495	495	103.246	2973221928.0	3.470560e+12	3.2TiB	ons HK Limited
1	1	103.246	3054928288.0	1.646262e+12	1.5TiB	ons HK Limited
164	164	191.108.126	1826793719.0	3.506824e+10	32.7GiB	unications Inc.
214	214	191.108.9.72	1826793700.0	2.084387e+10	19.4GiB	unications Inc.
291	291	191.108.9.72	1826793661.0	1.833587e+10	17.1GiB	unications Inc.
361	361	115.133.154	3068782335.0	1.560168e+10	14.5GiB	ervice Provider
293	293	171.95.158	3069050613.0	8.586856e+09	8.0GiB	ERNET Co.,Ltd.

Рис. 3.11. Інформація про власників IP-адрес

Ця ж компанія пропонує базу даних геолокації IP, яка дозволяє досить надійно знати місцезнаходження IP-адреси.

AbuseIPDB — відома служба репутації доменів і IP-адрес. Через свій API він дозволяє виконувати 1 тис. запитів на день безкоштовно. Зробимо відповідну програму:

```

def abuseipdb_req(ip):
    import json
    try:
        key = <api-key>
        url = 'https://api.abuseipdb.com/api/v2/check'
        querystring = {
            'ipAddress': ip,
            'maxAgeInDays': '90'
        }
        headers = {
            'Accept': 'application/json',
            'Key': key
        }
        response = requests.request(method='GET', url=url,
headers=headers, params=querystring)
        decodedResponse = json.loads(response.text)
    except e:

```

```

    print(e)
    return pd.np.nan
try:
    return decodedResponse["data"]["abuseConfidenceScore"] if
decodedResponse else pd.np.nan
except KeyError:
    return pd.np.nan
unique_ips = df_exf["remip"].unique()
unique_ips = pd.Series(unique_ips, index = top10)
df_exf['remip_abuse'] =
df_exf['remip'].map(unique_ips.apply(abuseipdb_req))
df_exf.nlargest(10, "remip_abuse")

```

У цьому випадку бачимо кілька з'єднань з IP-адресами з дуже поганою репутацією, які потрібно проаналізувати, щоб перевірити їх законність (рис. 3.12).

remip	tunnelid	sentbyte	h_sentbyte	ASN	remip_country	is_tor_node	remip_abuse
162.142.125.39	0.0	NaN	nanYiB	CENSYS-ARIN-01	United States	False	100
192.241.210.30	0.0	NaN	nanYiB	DIGITALOCEAN-ASN	United States	False	100
167.248.133.53	0.0	NaN	nanYiB	CENSYS-ARIN-03	United States	False	100
111.7.96.166	0.0	NaN	nanYiB	Guangdong Mobile Communication Co.Ltd.	China	False	100
184.105.139.69	0.0	NaN	nanYiB	HURRICANE	United States	False	100
65.49.20.69	0.0	NaN	nanYiB	HURRICANE	United States	False	100
74.120.14.40	0.0	NaN	nanYiB	CENSYS-ARIN-02	United States	False	100
74.120.14.39	0.0	NaN	nanYiB	CENSYS-ARIN-02	United States	False	100
162.142.125.128	0.0	NaN	nanYiB	CENSYS-ARIN-01	United States	False	100
184.105.247.195	0.0	NaN	nanYiB	HURRICANE	United States	False	100

Рис. 3.12. Репутація IP-адрес

Створимо новий запит, щоб виконати нові візуалізації даних, щоб спробувати виявити аномалії.

```
df = df_dask[~df_dask["tunnelid"].isnull()][["remip", "duration",
"tunnelid", "sentbyte"]].compute()
```

Цей запит поверне підмножину всіх даних, і через наступний код отримаємо:

```

duration_t = df_work[["remip", "duration",
"tunnelid"]].groupby(["remip", "tunnelid"]).max().dropna().reset_index()
df_final["n_tunnels"] = df_final["remip"].map(duration_t[["remip",
"duration"]].groupby("remip").count()["duration"])
df_final["n_connections"] =
df_final["remip"].map(df_work.value_counts("remip"))
df_final["duration_sum"] = df_final["remip"].map(duration_t[["remip",
"duration"]].groupby("remip").sum()["duration"])
sentby_t = df_work[~df_work["tunnelid"].isnull()][["remip", "sentbyte",
"tunnelid"]].groupby(["remip", "tunnelid"]).max().dropna().reset_index()

```

```
df_final["sentbyte_sum"] = df_final["remip"].map(sentby_t[["remip",
"sentbyte"]].groupby("remip").sum()["sentbyte"])
df_final = df_final[~df_final["n_tunnels"].isnull()]
```

Тепер маємо 4 змінні для малювання графіків, кількість тунелів, створених до певної IP-адреси, загальна кількість з'єднань, здійснених до кожної IP-адреси, сума тривалості з'єднань, здійснених до кожної IP-адреси, і кількість надісланих байтів.

Побудова на їх основі графіків може допомогти візуалізувати як викрадання даних у кількох з'єднаннях, завантаження файлів у «хмару» чи надсилання файлів через SSH, так і викрадання даних у прихованих каналах, таких як DNS або ICMP, оскільки буде видно багато з'єднань і дещо високу кількість надісланої інформації.

Загалом, очевидно, що аналіз даних — це один із ключів до майбутнього практично в кожній галузі, але у випадку кібербезпеки це є і буде основною навичкою для аналітиків.

3.2.4. Аналітика аномалій



У сфері аналітики аномалії представляють дані, які виходять за межі очікуваного діапазону. Аномалія може виникнути випадково, або хтось, можливо, помістив її туди, втім аномалія зазвичай є несподіваною та потенційно небажаною.

Аномалії бувають двох форм:

- *Викиди*: коли дані не збігаються з рештою даних, це викид. Викид може мати різні форми, але визначальною характеристикою є те, що він точно не потрібний, оскільки спотворює будь-який аналіз, який виконується з ним.
- *Новинки*: іноді дані виходять за межі нормального діапазону, але насправді вони збігаються з рештою даних. У цьому разі дані являють собою новий приклад, який слід розглядати як частину будь-якого аналізу. В іншому випадку аналіз не зможе відобразити справжній стан того, що аналіз мав би виявити.

Отже, частина проблеми полягає в тому, що обидва типи аномалій виходять за межі нормального діапазону, але одна — бажана, а інша — ні. У деяких випадках для прийняття правильного рішення може знадобитися перевірка даних людиною для прийняття рішення.

Конкретизація причин і наслідків виявлення аномалії. Перш ніж дізнатися про наявність аномалії, треба її виявити.

Виявлення аномалій — це поєднання таких методів:

- *Розробка*: використання відомих правил і законів, які можна зробити за допомогою методів машинного навчання та штучного інтелекту.
- *Наука*: визначення гіпотези, а потім її доведення. Науці потрібна людина, щоб визначити гіпотезу, проте вона може покладатися на програму, щоб довести її.
- *Мистецтво*: лише отримати відчуття, що щось не так, що, безперечно, є виключно сферою людського інтелекту.

На відміну від багатьох сфер безпеки, не існує жодних методів прогнозування виникнення аномалії, оскільки аномалії за природою абсолютно несподівані та непередбачувані.

Коли аномалії плануються, їх надзвичайно важко виявити, оскільки хакери можуть просто приховати атаку в усіх інших даних.

Частина проблеми з виявленням аномалій полягає в тому, що люди мають схильність до упередженості, що означає схильність не бачити несподіваного, якщо воно не справляє достатнього враження. Як приклад із реального світу, одна людина повністю зосереджена на роботі, а інша — перефарбовує будинок. Перший приходить з роботи і не бачить, що будинок пофарбований, тому що упередженість не дозволяє йому це зробити. Їхня увага зосереджена на роботі; більше нічого не має значення. Те саме відбувається з аномаліями, які впливають на бізнес-середовище, особливо на дані. Можна дивитися на дані, але насправді їх не бачити, тому що в цей час увага зосереджена на чомусь іншому. Саме тут смарт-аналітика може прийти на допомогу, створюючи додаткове враження, що щось не так. Аномалії трапляються в різних ситуаціях, і це не завжди джерело даних, про що багато досліджень забувають згадувати. Новинки дійсно виникають як частина зміни вихідних даних, але викиди не настільки обмежені. Зважаючи на це, у наведеному нижче списку подано деякі джерела, які слід перевірити, для пошуку аномалій у своїх даних.

- *Пошкодження набору даних*. Будь-яке джерело даних може бути пошкоджено. Користувачі можуть ввести неправильні дані, збій мережі може пошкодити дані, хакери можуть змінити дані, природа може змінити дані, або інформація може просто застаріти настільки, що вони більше не будуть корисними. Це джерело аномалій завжди пов'язано з викидами.
- *Зміщення концепції*. Значення даних може змінюватися з часом залежно від фізичних, політичних, економічних, соціальних чи інших зовнішніх сил, що веде до інтерпретації даних. Ви часто розглядаєте таку аномалію як новинку, але вона також може бути винятком.
- *Зміна навколишнього середовища*. Датчики IoT особливо чутливі до цієї проблеми та, ймовірно, створять аномалії у вигляді новинок. Однак така аномалія також може виглядати як викид, коли хакери або інші особи навмисно змінюють середовище, щоб створити непридатні дані.

- *Зміна вихідних даних.* Будь-яке джерело даних, яке використовується, але не під особистим контролем, може бути несподівано змінено. Програма, імовірно, виявить зміни у форматі чи інших фізичних факторах. Однак вона не виявить такі проблеми, як зміни політики організацією, яка створює та керує даними.
- *Людина у середині (Man-in-the-middle).* Джерело даних не змінюється, і код доступу до нього не змінюється, але в якийсь момент між надсиланням інформації з джерела та отриманням даних, дані змінюються, як правило, непомітно. Це просто новий поворот у випробуваному методі, який використовують хакери для створення проблем.
- *Помилка користувача.* Неможливо навчити людей постійно виконувати завдання з повною коректністю.

У більшості випадків викиди, які треба знайти, пов'язані з певною помилкою введення даних. Наприклад, малоімовірно, що хтось у віці 12 років якимось чином матиме ступінь доктора філософії. Однак відхилення також можуть бути незначними. Наприклад, хакер може підвищити ціни на будинки в певному районі на невелику суму, щоб конфедерація з будинками на продаж могла отримати кращу ціну.

Справа в тім, що дані, як правило, мають певні характеристики та потрапляють у певні діапазони, тому фахівець може використовувати інженерний підхід для пошуку потенційних викидів. Потім людина вже повинна вдатися до інтуїції, щоб вирішити, чи помилкові значення виявляються новинками чи фактичними викидами, які ви потім досліджуєте та виправляєте.

3.2.5. Модельна проблема



Однофакторний підхід до визначення викидів був спочатку запропонований Джоном Тьюкі як дослідницький аналіз даних (EDA). Щоб побачити медіану, значення у 25% і 75%, використовується прямокутний графік. Віднявши 25-відсоткове значення від 75-відсоткового значення, щоб отримати IQR, і помноживши на 1,5, визначається очікуваний діапазон значень для певної функції набору даних.

Будь-які значення за межами цього діапазону вважаються викидами. Щоб побачити, як працює цей процес, виконайте наведені нижче дії.

1. Імпортуйте необхідні бібліотеки:

```
from sklearn.datasets import fetch_california_housing
import pandas as pd
%matplotlib inline
```

2. Маніпулюйте набором даних, щоб він відображався в належній формі:

```
california = fetch_california_housing(as_frame = True)
X, y = california.data, california.target
X = pd.DataFrame(X, columns=california.feature_names)
print(X)
```

На рис. 3.13 відображено отриманий набір даних, щоб була можливість побачити, як він виглядає. Зверху показано функції, які використовуються для набору даних:

```
MedInc HouseAge AveRooms AveBedrms Population AveOccup Latitude \
0      8.3252      41.0  6.984127  1.023810      322.0  2.555556      37.88
1      8.3014      21.0  6.238137  0.971880     2401.0  2.109842      37.86
2      7.2574      52.0  8.288136  1.073446      496.0  2.802260      37.85
3      5.6431      52.0  5.817352  1.073059      558.0  2.547945      37.85
4      3.8462      52.0  6.281853  1.081081      565.0  2.181467      37.85
...
20635  1.5603      25.0  5.045455  1.133333      845.0  2.560606      39.48
20636  2.5568      18.0  6.114035  1.315789      356.0  3.122807      39.49
20637  1.7000      17.0  5.205543  1.120092     1007.0  2.325635      39.43
20638  1.8672      18.0  5.329513  1.171920      741.0  2.123209      39.43
20639  2.3086      16.0  5.254717  1.162264     1387.0  2.616981      39.37

Longitude
0      -122.23
1      -122.22
2      -122.24
3      -122.25
4      -122.25
...
20635  -121.09
20636  -121.21
20637  -121.22
20638  -121.32
20639  -121.24

[20640 rows x 8 columns]
```

Рис. 3.13. Маніпуляції з даними створюють табличне подання зі стовпцями, позначеними назвами функцій

3. Створіть графічний малюнок:

```
X.boxplot('MedInc', return_type='axes')
```

Коробчаста діаграма конкретно показує значення функції середнього доходу (MedInc). Вихідні дані, показані на рис. 3.13, вказують на те, що повний діапазон значень коливається приблизно від 9 тис. до 80 тис. дол. США. Середнє значення становить близько 35 тис. дол., з 25-відсотковим значенням 25 тис. дол. і 75-відсотковим значенням 50 тис. дол. США (для IQR 25 тис. дол. США). Однак деякі значення перевищують 80 тис. дол. і, як підозрюється, є викидами.

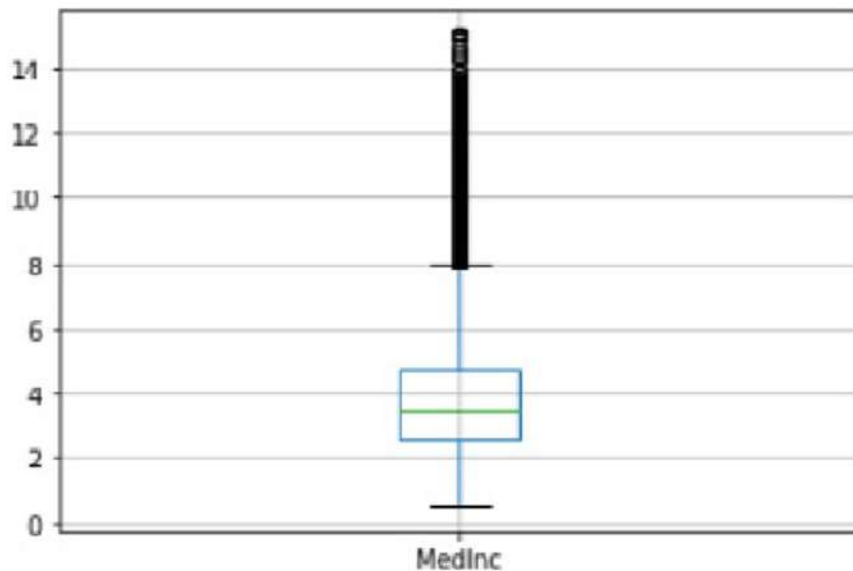


Рис. 3.14. Детальна інформація про діапазон даних, яка вказує на наявність потенційних викидів

Переглядаючи результат, показаний на рис. 3.14, можна побачити, що існують потенційні викиди, проте складно сказати, скільки викидів і як вони пов'язані з іншими даними в наборі даних. Необхідність краще розуміти викиди є однією з причин, чому потрібно застосовувати багатовимірний підхід, такий як PCA (Principal component analysis). Фактична математика PCA може бути складною та вимагає знання статистики, але насправді непотрібно добре її розуміти, щоб успішно використовувати PCA.

Кількість змінних, які фахівець обирає, залежить від сценарію, який він намагається зрозуміти. У цьому випадку приклад шукає кореляцію між середнім доходом і віком будинку. Вирішуючи подібну проблему безпеки, фахівець може шукати кореляцію між невдалими спробами входу в систему та часом доби або викликами програмних API, які не дуже часто використовуються, і показниками навантаження на мережу. Це все про пошук закономірностей у вигляді аномалій, які можна використовувати для виявлення хакерської активності.

Наступний приклад також покладається на діаграму розсіювання, щоб показати результати аналізу. Діаграма розсіювання має тенденцію підкреслювати, де в даних з'являються викиди. Ті самі принципи, які були використані в аналізі середнього доходу, застосовуються до інших джерел даних, наприклад, журналів доступу до API:

1. Імпортуйте необхідні бібліотеки:

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
import matplotlib.pyplot as plt
```

2. Подумайте про необхідність масштабування прикладу, запустивши наступний код:

```
print(X[["MedInc", "HouseAge"]][0:5])
print(scale(X[["MedInc", "HouseAge"]])[0:5])
```

На рис. 3.15, показано, як вихідні дані масштабуються в стовпцях «MedInc» і «HouseAge», щоб мати однаковий відносний розмір. Масштабування є важливою частиною роботи з PCA (Principal component analysis), оскільки воно робить можливим порівняння двох змінних. В іншому разі це буде порівняння яблук і апельсинів, яке ніколи не дасть жодної достовірної інформації:

	MedInc	HouseAge
0	8.3252	41.0
1	8.3014	21.0
2	7.2574	52.0
3	5.6431	52.0
4	3.8462	52.0


```
[[ 2.34476576  0.98214266]
 [ 2.33223796 -0.60701891]
 [ 1.7826994   1.85618152]
 [ 0.93296751  1.85618152]
 [-0.012881   1.85618152]]
```

Рис. 3.15. Масштабування є невід'ємною частиною процесу багатовимірного аналізу

3. Використовуйте PCA, щоб адаптувати модель до масштабованих даних:

```
pca = PCA(n_components=2)
pca.fit(scale(X))
C = pca.transform(scale(X))
print(C)
print("Original Shape: ", X.shape)
print("Transformed Shape: ", C.shape)
```

Метод `pca.transform()` виконав зменшення розмірності. Ви можете побачити результат процесу підгонки та зменшення розмірності на рис. 3.16.

```
[[ 1.88270434 -0.50336186]
 [ 1.37111955 -0.12140565]
 [ 2.08686762 -0.5011357 ]
 ...
 [ 1.40235696 -1.09642559]
 [ 1.5429429  -1.05940835]
 [ 1.40551621 -0.89672727]]
Original Shape: (20640, 8)
Transformed Shape: (20640, 2)
```

Рис. 3.16. Доопрацювання даних і подальше їх перетворення дають дані, які можна використати для моделі

4. Побудуйте діаграму, щоб побачити результат аналізу:

```
plt.title('PCA Outlier Detection')
plt.xlabel('Component 1')
plt.ylabel('Component 2')plt.scatter(C[:,0],C[:,1], s=2**7,
edgecolors='white',
alpha=0.85, cmap='autumn')
plt.grid(which='minor', axis='both')
plt.show()
```

Вихідні дані на рис. 3.17 чітко показують викиди у верхньому правому куті графіка. Дані осей x і y з'являються першими під час виклику `plt.scatter()`. Решта аргументів впливає на представлення: s — для розміру маркера; `edgecolors` — для лінії навколо кожного маркера (щоб їх було легше побачити); `alpha` — для керування прозорістю маркера; і `cmap` — для керування кольорами маркера:

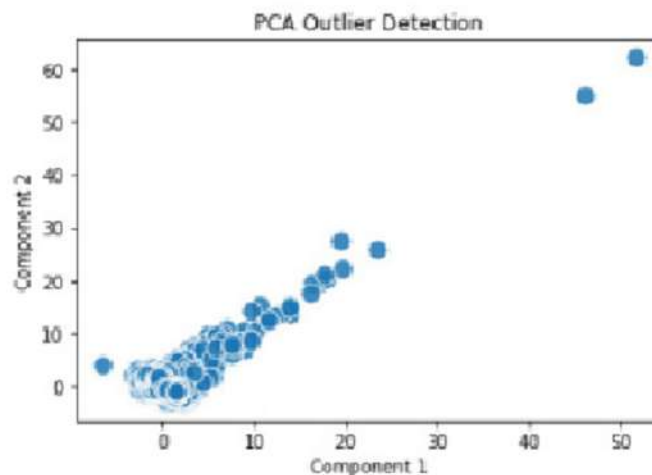


Рис. 3.17. Викиди відображаються у верхньому правому куті графіка

Щоб ефективно використовувати цей метод, потрібно спробувати різні порівняння, щоб знайти кореляції, що полегшить розпізнавання картини безпеки. Крім того, може виникнути ситуація, коли потрібно використовувати більш ніж дві функції, як це було зроблено в даному прикладі.

Можливо, існує зв'язок між недійсними логінами, IP-адресами та часом доби. Доки не буде зроблено модель для конкретної мережі, фахівці справді не знають, що саме аномалії, які вони виявляють, скажуть про картину безпеки.

Людська інтуїція надзвичайно важлива для визначення, які комбінації спробувати, і з'ясування того, які дані насправді є аномалією, а які є новинкою. Комп'ютер не може виконати це завдання. PCA (Principal component analysis) працює лише тоді, коли людина, яка його виконує, приймає рішення на основі попереднього досвіду, поточних тенденцій, унікальності мережі та очікуваної чи видимої поведінки.

Процес виявлення аномалій також включає людські спостереження і стратегії пом'якшення наслідків.

- *Відстежуйте зміни поведінки.* Співробітники служби безпеки повинні бути залучені до людей настільки, щоб вони могли бачити шаблони в тому, як люди діють, і вони повинні мати право ставити під сумнів зміни в установлених шаблонах.
- *Часто перевіряйте дані інформаційної панелі* на наявність змін у способі взаємодії користувачів, клієнтів, довірених третіх осіб тощо з вашою системою.
- *Переконайтеся, що перевірки безпеки дійсно виконуються* і що посилання справді перевіряються. Якщо не зроблено необхідні перевірки, невідомо, хто має доступ до мережі або що вони з нею роблять.
- *Виділіть час*, щоб справді шукати проблеми, а не покладатися на те, що вони вдарять в обличчя людей, яким ви довіряєте.
- *Попросіть усіх шукати потенційні проблеми.* Співробітники служби безпеки будуть шукати певний клас проблем, але вони не побачать їх усіх. Хтось, хто працює в бухгалтерії, може побачити аномалію, яку працівники служби безпеки можуть навіть не зрозуміти.

Запитання для самоперевірки



1. *Що називають пакетом у мережі?*
2. *Яка різниця між MAC та IP-адресами?*
3. *Що таке Wireshark?*
4. *Як фільтрувати пакети у мережі?*
5. *Як переформатувати лог у набір даних. Опишіть?*
6. *Які бібліотеки мови Python застосовуються для обробки даних в кібербезпеці?*
7. *Як перевірити репутацію IP-адреси?*

3.3. ШІ у кіберпросторі



Штучний інтелект (ШІ) надає передові можливості для виявлення, запобігання і реагування на загрози, що значно сприяє покращенню практики кібербезпеки. Це дає компаніям можливість захищатися від складніших онлайн-загроз.

3.3.1. Огляд застосування ШІ у кібербезпеці

Виявлення загроз. Системи штучного інтелекту здатні просіювати величезні обсяги даних і виявляти тенденції, які можуть бути ознаками онлайн-загроз. Аномалії, зловмисну поведінку та добре відомі моделі атак можна виявити за допомогою навчених моделей машинного навчання. Завдяки цьому служби безпеки можуть завчасно виявляти й усунути загрози. Щоб отримати докладну інформацію про виявлення шахрайства, зверніться до програми Data Science and AI, розробленої експертами галузі.

Аналіз поведінки. ШІ може ідентифікувати нормативну поведінку користувача, мережі та системи. ШІ відстежує відхилення від цих стандартів і розпізнає будь-яку незвичну поведінку, яка може вказувати на порушення. Системи на основі штучного інтелекту можуть успішно виявляти раніше невідомі вектори атак шляхом постійного навчання та адаптації до нових загроз.

Автоматизація та реагування. Штучний інтелект автоматизує низку процедур безпеки, включаючи керування вразливістю, полювання на загрози та реагування на інциденти. ШІ може швидко визначати пріоритети та аналізувати сигнали охоронної сигналізації, зменшуючи навантаження на людей-аналітиків і сприяючи швидшому часу реакції. Системи, керовані ШІ, також здатні автоматично поміщати в карантин або блокувати шкідливий трафік або дані.

Автентифікація користувачів. Використовуючи такі технології, як поведінкова біометрія, розпізнавання обличчя та розпізнавання голосу, штучний інтелект може покращити процедури автентифікації користувачів. Забезпечуючи ефективніші процеси автентифікації, ці технології зменшують можливість несанкціонованого доступу до систем і даних.

Виявлення зловмисного програмного забезпечення. На основі своїх поведінкових особливостей системи штучного інтелекту можуть розпізнавати свіжі та невиявлені штами шкідливого програмного забезпечення. Ще до того, як звичайні сигнатури стануть доступними, штучний інтелект може виявляти та зупиняти зловмисне програмне забезпечення в режимі реального часу, досліджуючи файлові структури, шаблони коду та поведінку мережі.

Розвідка загроз. Штучний інтелект здатний обробляти й аналізувати величезні обсяги даних розвідки про загрози з різноманітних джерел, таких як форуми, темна мережа та канали безпеки. Штучний інтелект допомагає командам безпеки залишатися в курсі та бути готовими зменшити нові ризики, отримуючи корисну інформацію та виявляючи нові загрози.

Аналітика безпеки. Платформи з можливостями аналітики на основі штучного інтелекту можуть отримувати й аналізувати дані безпеки з різних джерел, наприклад, журнали, мережевий трафік і поведінку користувачів. Це дає змогу групам безпеки знаходити вразливі місця, збирати корисну інформацію та покращувати загальну безпеку.

Забезпечення цілісності та надійності алгоритмів штучного інтелекту та забезпечення контролю з боку людини мають вирішальне значення для використання повного потенціалу штучного інтелекту в кібербезпеці.



Однак важливо зазначити, що ШІ в кібербезпеці не позбавлений проблем. Зловмисники потенційно також можуть використовувати системи штучного інтелекту чи уникати виявлення за допомогою методів боротьби з ним.

Штучний інтелект може спричинити проблеми у кібербезпеці кількома способами. Наприклад:

- *Створення нових кіберзагроз.* Хакери можуть використовувати штучний інтелект для створення нових чи складніших кібератак, таких як створення підробленого вмісту (наприклад, deepfakes), обхід заходів безпеки (таких як біометрія) або використання вразливостей (таких як експлойти нульового дня).
- *Компрометація систем штучного інтелекту.* Системи штучного інтелекту самі по собі можуть бути вразливими до кібератак, таких як підробка їхніх даних, алгоритмів або результатів. Це може призвести до пошкодження або упередженості систем штучного інтелекту, що може спричинити шкоду чи помилки.
- *Порушення етичних і соціальних питань.* Системи ШІ можуть порушувати етичні та соціальні питання, пов'язані з кібербезпекою, наприклад, конфіденційність, підзвітність, прозорість, надійність.
- Надійність, справедливість, людська гідність тощо. Ці проблеми можуть вплинути на безпеку і добробут окремих людей та суспільства.

3.3.2. Генеративні технології штучного інтелекту



Генеративні попередньо навчені трансформатори (GPT), змінюють сферу безпеки. GPT означає:

1. «Трансформери» є основними будівельними блоками цих моделей, здатними зрозуміти зв'язок між даними в послідовності в межах важливого контексту.
 2. «Попередньо навчені» означає, що ці моделі тренуються на трильйонах точок даних, що дозволяє їм зрозуміти зв'язки між різними поняттями, сигналами, словами чи зображенням.
 3. «Генеративність» означає, що коли ці моделі отримують точку входу, вони можуть генерувати нові дані на основі початкових вхідних даних.
- Запровадження GPT значно скоротило час і ресурси, необхідні раніше. Замість того щоб збирати мільярди точок даних і витратити шість місяців на ресурсомістку роботу, користувачі тепер можуть створювати прості програми, використовуючи вільно доступні основні моделі GPT. Те, що раніше займало від 6 до 12 місяців, тепер можна зробити за лічені години.

Агенти GPT у сфері кібербезпеки охоплюють різні сфери, як зазначено в репозиторії Awesome GPT-Agents, зокрема:

- *Threat Intelligence*. Такі агенти, як Threat Intel Bot спеціалізуються на наданні найновішої інформації про удосконалені постійні загрози (APT).
- *Тестування на проникнення*. Такі інструменти, як GP(en)T(ester) пропонують вказівки щодо тестування на проникнення, допомагаючи користувачам визначати та використовувати вразливості системи.
- *Методи ухилення від зловмисного програмного забезпечення*. MagicUnprotect взаємодіє з базами даних, щоб надати інформацію про ухилення від зловмисного програмного забезпечення.
- *Аналіз вихідного коду*. Агенти можуть аналізувати вихідний код, допомагаючи користувачам зрозуміти та виправити потенційні проблеми безпеки в їхніх проектах кодування.
- *Веббезпека*. Web Hacking Wizard зосереджується на темах веббезпеки, надаючи інтерактивні вказівки та роз'яснення.
- *Кар'єрна орієнтація*. Такі агенти, як Cyber Security Career Mentor і Cyber Mentor направляють кар'єрні шляхи в розвивають навички у галузі кібербезпеки.
- *Оцінка ризиків і консультування з питань кібербезпеки*. CyberGuard — це консультант для домашнього та малого бізнесу, який допомагає у загальних питаннях кібербезпеки та оцінки ризиків.

У табл. 3.2 наведено підсумок аналізу технічних та організаційних проблем використання штучного інтелекту, разом із підходами до їх вирішення.

Проблеми генеративного ШІ та потенційні рішення

Виклики	Потенційні рішення
Необхідність спеціальних знань	Демонстрації та prompts для конкретних завдань, пошук і закріплення знань, а також розширення контексту
Упереджені результати	Алгоритми пом'якшення упередженості, збалансовані навчальні дані, аудит, інклюзивне навчання та міждисциплінарні дослідження
Створення шкідливого контенту	Системи модерації, переривання та корекція, а також методи кондиціонування, такі як RLHF
Логічні невідповідності	Гібридні архітектури, бази знань і розширення пошуку
Фактичні неточності	Розширення пошуку, бази знань і послідовне оновлення бази знань
Ризики конфіденційності	Диференційна конфіденційність, інтегроване навчання, шифрування та анонімізація
Висока затримка та витрати на обчислення	Оптимізоване апаратне забезпечення та ефективний дизайн моделей
Ліцензійні обмеження	Відкриті / синтетичні дані, користувальницькі дані та справедливі ліцензійні угоди
Безпека/вразливості	Стійкість до суперництва та найкращі практики кібербезпеки
Управління	Фреймворки компласнсу та етичного розвитку управління

Програми на основі LLM відрізняються від класичних програм інтеграцією великих мовних моделей (LLM) та їхніх ключових компонентів, таких як навчальні дані, моделі, стан, політика, запитання та дії. У той час як функціональні компоненти класичних програм, такі як інтерфейс, серверна частина, інфраструктура, база даних і стандартні функції, залишаються невід'ємною частиною програм на основі LLM, які будуть модифіковані для інтеграції концепцій і можливостей програм на основі LLM.



Нова архітектура буде набагато елегантнішою, чотирикомпонентною структурою, заснованою на GPT: *стан, політика, питання та дії*.

Програмні додатки на базі схем мають чітку та жорстку структуру. Вхідні та вихідні дані мають бути явно створені, маршрутизовані та підтримувані. Будь-яке відхилення від цієї структури веде до помилок, а додавання нових функцій вимагає лінійних зусиль з боку розробників організації.

Нове програмне забезпечення буде засноване на розумінні. Такі програми матимуть майже необмежений вхід, оскільки вони засновані на природній мові, що надсилається системі, яка насправді розуміє, що людина запитує. Додати нові функції буде так само просто, як задавати різні запитання та/або надавати різні команди. Додатки на основі розуміння, пристосовуються до запитань, які їм задають. Щоб додати нову функціональність, ставите різні запитання та просите виконати різні дії з результатами

За змістом це перехід від архітектури на основі схемотехніки до архітектури на основі розуміння (табл. 3.3).

Таблиця 3.3

Порівняльний аналіз архітектури на основі схемотехніки та архітектури на основі розуміння (III)

Елементи структури	На основі схем	На основі розуміння
Вхід	Жорсткість — виклики програми повинні бути явно запрограмовані в програмі, як запити, командою розробників, використовуючи явні запити до структурованої бази даних	Необмеженість — звернення до програм здійснюються за допомогою природної мови, тож не потрібно явно кодувати нові запити на сервері, коли потрібно задати інші запитання
Дані	Структурованість — сховища даних програм мають чітко визначені схеми, які вимагають певних способів додавання, отримання, зміни та видалення даних із системи	Динамічність — бекенд системи є набором моделей, які розуміють світ і ваш бізнес. Щоб оновити сховище даних, отримуєте більші/кращі дані і зберігаєте свої моделі
Вихід	Статичність . Вихід із програми повертає конкретну інформацію, яку запитує конкретний запит. Щоб зробити щось інше, треба написати окремий код для цього	Гнучкість — вихідні дані програми такі ж гнучкі, як і запитання, які ставлять моделям. А запитання та відповідь можуть безпосередньо пов'язуватись із діями
Дія	Відокремленість — дії відокремлені від результатів запиту. Наприклад, зробивши запит про ціни на житло до бази даних, треба виконати одну або кілька додаткових дій, щоб зробити їх корисними	Зв'язаність — Дії можна пов'язати з результатом, оскільки моделі розуміють, як вони пов'язані. Це означає, що не потрібно створювати явні мости від кожного результату до нових дій
Розуміння	Відсутнє — програма не розуміє ні дані, ні запити, що виконуються проти неї, ні результат, який вона повернула. Це простий механізм для зберігання та вилучення даних.	Глибоке — моделі розуміють матеріал, з яким вони мають справу, що означає, що вдосконалення моделей одночасно покращує більшість аспектів програми.
Гнучкість	Лінійна — щоб додати більше функцій до програми, потрібно явно розширити кількість закодованих вхідних даних, кількість вихідних дій, структуру даних або всі разом	Надзвичайна — головною концепцією стає: 1) забезпечення якості та свіжості моделей; 2) забезпечення правильності запитань

Розвідка — це термін, що означає збір, аналіз і обмін інформацією про ворога для прийняття вищих рішень.

- *Дані*: необроблені факти чи цифри без контексту.
- *Інформація*: Дані, які були оброблені, упорядковані або структуровані.
- *Знання*: інформація, яка була об'єднана, щоб стати корисною.
- *Розуміння*: здатність застосовувати знання в нових ситуаціях і контекстах.

Чи в вузькому розумінні:

- дані необроблені;
- інформація впорядкована;
- знання корисні;
- розуміння застосовується.

Отже, остаточне визначення терміна стане таким: це здатність застосовувати знання в нових ситуаціях і контекстах. Ми не вимагаємо від систем GPT досвіду. Ми не питаємо, чи вони щось відчують. Питання полягає в тому, чи можуть вони узагальнювати концепції, використовуючи нову інформацію, тобто застосовувати знання до нових ситуацій і контекстів. Це і є розуміння. Відповідно, побудувати на їх основі систему кібербезпеки є можливим.

Алгоритм побудови системи безпеки з використанням LLM.

1. Виберіть базову модель — починаєте з останньої та найкращої загальної моделі GPT від OpenAI, Google, Meta чи будь-кого іншого. Багато компаній матимуть, назовемо це OpenAI GPT-6. Він уже знає настільки неймовірно багато про безпеку, біотехнології, управління проєктами, планування, зустрічі, бюджети, реагування на інциденти та готовність до аудиту, що компанії могли б вижити лише з ним, проте потрібен більш персоналізований контекст.
2. Навчіть свою спеціальну модель — навчаєте спеціальну модель, яка базується на власних даних компанії, які додаються поверх GPT-6. Це все в розділі STATE. Це телеметрія та контекст компанії, журнали брандмауерів, фінансові документи, чати, електронні листи, стенограми зустрічей. Тренуєте власну модель на цьому наборі даних.
3. Навчіть свою політичну модель — тепер тренуєте іншу модель, яка відповідає бажанням вашої компанії. Місія, цілі, ваші антицілі, ваші виклики, ваші стратегії. Це вказівки, отримані від людей, яких ми використовуємо, щоб керувати частиною архітектури ДІА. Коли ми просимо його зробити щось для нас і побудувати наші плани, він зробить це за допомогою огорож, зображених тут у політиці.
4. Скажіть системі виконати такі дії — тепер моделі об'єднано. У нас є GPT-6, поєднаний з нашою моделлю STATE, також з нашою моделлю POLICY, і разом вони знають нас краще, ніж ми самі.

Тож надаємо ІІІ той самий список робіт, який виконуємо по кібербезпеці компанії:

- дайте мені список наших найбільш важливих програм з позицій бізнесу та ризику;
- створіть пріоритетний список наших головних загроз для них і співвіднесіть його з тим, на що наша команда безпеки витрачає свій час і гроші;
- надайте рекомендації щодо того, як скоригувати наш бюджет, чисельність персоналу, окр і список проектів, щоб належним чином відповідати нашим фактичним загрозам;
- давайте напишемо скориговану стратегію безпеки, використовуючи цей новий підхід;
- визначте 5 найкращих крі, які ми будемо відстежувати, щоб показати прогрес у досягненні наших цілей;
- побудуйте вкладену структуру окр, яка впливає з цієї стратегії з урахуванням нашої організаційної структури;
- створіть оновлену презентацію для дошки з описом нового підходу;
- створіть список способів, яких нам не вистачає з точки зору відповідності, враховуючи правила, під які ми підпадаємо;
- потім створіть повний план реалізації з розбивкою на наступні чотири квартали;
- нарешті, напишіть наш перший квартальний звіт про безпеку та оновлюйте цей документ.

У найближчому майбутньому доведеться ще раз перевірити вихідні дані моделей, оскільки галюцинації ІІІ є реальною річчю на такому ранньому етапі розвитку.

Єдине, що йому потрібно, це 1) актуальні моделі з використанням останніх даних і 2) правильні запитання, які надходять від керівників організації. У цьому випадку ми вже маємо ці запитання у списку вище.

Отже, ми говоримо про те, щоб перейти від тисяч годин роботи на квартал, розподілених між десятками людей, до, можливо, 1–5% від цього. У новій моделі робота буде зосереджена на забезпеченні актуальності ПОЛІТИКИ та правильності ЗАПИТАНЬ, які ми ставимо.

3.3.2. Трансформація програмних вертикалей

У статичному аналізі приймають вхідні дані та запитують про дві речі: Що не так? Як це виправити?

Розглянемо за його допомогою трансформацію програмного забезпечення (рис. 3.18).

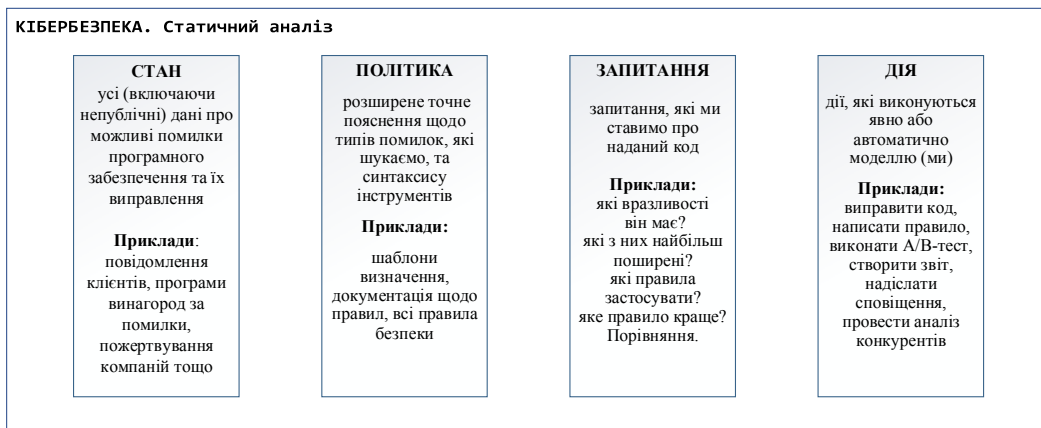


Рис. 3.8. Статичний аналіз у LLM

ШІ зруйнує все існуюче програмне забезпечення, яке це робить, оскільки воно базується на розумінні. Отже, як тільки він достатньою мірою виявить проблему через ваш STATE і зрозуміє, що ви намагаєтеся зробити через свою POLICY, він зможе зробити набагато більше, ніж просто знайти проблеми з кодом і виправити їх. Він зможе виконувати такі дії:

- знайдіть проблему;
- покажіть, як це виправити будь-якою мовою (кодуванням або людиною);
- напишіть оперативний посібник, щоб уникнути цих помилок;
- напишіть правило в технології вашого інструменту, яке б це виявляло;
- дайте вам фіксований код;
- переконайтеся, що код працює.

Крім того, ви зможете створювати кілька версій коду, щоб побачити, як усі вони реагуватимуть на найпоширеніші атаки, а потім давати рекомендації на основі цих результатів.



Програмне забезпечення безпеки в цілому — це:

- виявлення та реагування;
- хто тут справжні нападники?
- хто закопався в очікуванні активації?
- знайдіть останні ttp у нашій організації;
- напишіть правила в нашому програмному забезпеченні виявлення, яке їх знайде;
- поділіться цими правилами з нашими однолітками;
- перегляньте їхні правила та перевірте їх;
- створіть фальшиву паралельну інфраструктуру, яка виглядає точно так само, як наша, але призначена для відлову зловмисників за такими критеріями;
- автоматично вимикайте облікові записи, надсилайте сповіщення, скидайте маркери тощо, коли бачите успішні атаки;

- слідкуйте за підозрілими пов'язаними подіями, такими як невідомі телефонні дзвінки з подальшими віддаленими сеансами та переглядом документації.

Керування поверхнею атаки (ASM) — практика визначення того, які активи компанії мають зовнішній слід та застосування до цих знань аналізу загроз.

- Витягніть усі дані про компанію.
- Знайдіть усі його злиття та приєднання.
- Знайдіть всю документацію, пов'язану з цими речами.
- Складіть список усіх доменів.
- Постійно запускайте інструменти, щоб знайти всі субдомени.
- Відкриті порти.
- Додатки на портах.
- Постійно переглядайте ці сайти за допомогою автоматизації.
- Надсилайте дані в модель SPQA, щоб знайти найбільш вразливі місця.
- Запустіть автоматизацію проти цих плям.
- Постійно відкривайте нашу нову поверхню.
- Постійний моніторинг / сканування та скидання в озеро даних (відро S3 або еквівалент).
- Постійно повторюйте модель STATE.
- Підключіться до системи оповіщення та інструментів для створення звітів.
- Нехай система сама оптимізується.

Корпоративна безпека

- Відстежуйте всю активність на наявність підозрілих дій і акторів.
- Автоматично виявляти, блокувати та повідомляти про ці дії.
- Переконайтеся, що безпека SaaS повністю синхронізована з корпоративними політиками безпеки.

Безпека постачальників і ланцюгів поставок. Безпека постачальників і ланцюгів постачання стане одним із найсерйозніших і найпотужніших порушень від SPQA лише через те, наскільки важкою зараз є ця проблема.

- Складіть список усіх наших продавців.
- Використовуйте кожну отриману анкету.
- Знаходьте кожне місце в нашій інфраструктурі, до якого торкається програмне забезпечення постачальника.
- Знайдіть уразливі компоненти в цих місцях.
- Складіть пріоритетний список найвищих ризиків для різних аспектів нашої компанії.
- Рекомендуйте пом'якшення, щоб знизити ризик, дивлячись на найсуворіші.
- Створіть список альтернативних постачальників, які мають подібні можливості, але не мають таких ризиків.
- Створіть план міграції до трьох найкращих варіантів.

Сьогодні в будь-якій великій організації це майже неможливо. Програма на основі моделей штучного інтелекту зробить це за лічені хвилини. І те саме щоразу, коли модель(і) оновлюються.

Але саме природна мова є основним засобом атаки для систем штучного інтелекту на базі LLM та її можна використовувати для атаки на компоненти систем на базі штучного інтелекту в усьому стеку (рис. 3.18).

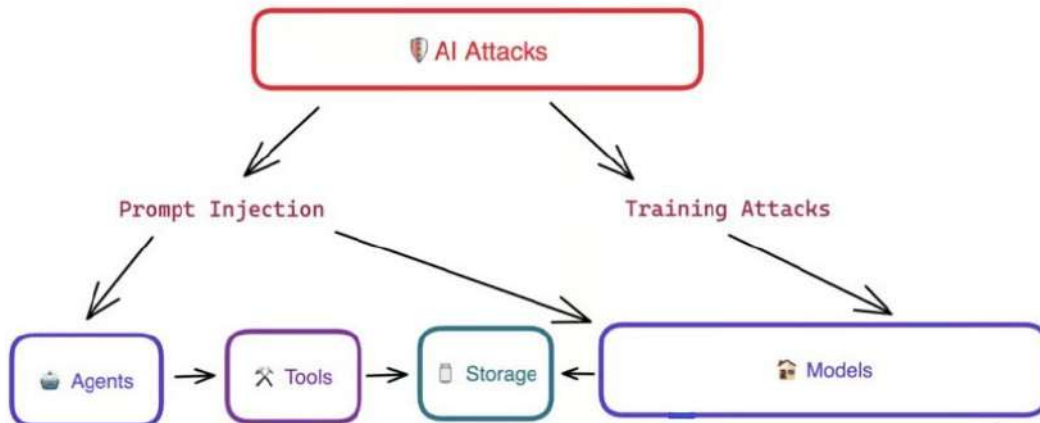


Рис. 3.18. Основні компоненти для поверхні атаки штучного інтелекту

Агенти на мові Langchain — це суб'єкти на базі штучного інтелекту, який мають мету та набір інструментів для її реалізації. Агенти є головним компонентом майбутнього ШІ.

Атака на агентів матиме великий вплив. Щоб агенти штучного інтелекту були корисними, вони повинні мати відповідні повноваження, тобто їм потрібно 1) знати величезну кількість про вас, включаючи дуже особисту та конфіденційну інформацію для найвищої ефективності; і 2) вони повинні вміти поводитися як ви. Це означає надсилання грошей, публікацію в соціальних мережах, написання вмісту, надсилання повідомлень тощо. Зловмисник, який отримає ці знання та доступ, матиме значний важіль впливу на ціль.

Інструменти — це інструменти, до яких Агенти мають доступ для виконання своєї роботи. Для академічного дослідницького агента він може мати інструмент веб-пошуку, інструмент аналізу паперу, підсумовувач, детектор плагіату та будь-що інше.

Моделі — це найдосконаліша річ у сфері безпеки ШІ. Вчені роками впроваджували машинне навчання з великим успіхом. Основним фокусом цих типів атак було б те, щоб спонукати моделі поводитися погано, тобто бути менш надійними, більш токсичними, більш упередженими.

Впровадження штучного інтелекту пов'язане з кількома проблемами, зокрема, з підготовкою даних, створенням моделей, масштабованістю та етичними проблемами. Компаніям потрібна систематична методологія для використання кількох базових можливостей у бізнесі та технологіях для впровадження технологій штучного інтелекту та машинного навчання на підприємстві.

Основні типи атак на системи штучного інтелекту

Компоненти систем ШІ	Типи атак
Агенти	• Змінити маршрутизацію агента; • надсилати команди невизначеним системам
Інструменти	• Виконувати довільні команди; • проходити через ін'єкцію на підключених системах інструментів; • виконання коду в системі агента
Сховища даних	• Атакувати вбудовані бази даних; • витягувати конфіденційні дані; • змінювати дані, що призводить до підроблених результатів роботи моделей
Моделі	• Обхід моделі захисту; • змусити модель виявляти упередженість; • вилучення даних інших користувачів і/або серверних даних; • змушувати модель виявляти нетерпиме поведінку; • отруювати результати інших користувачів • порушити довіру / надійність моделі; • доступ до неопублікованих моделей

Наприклад, уявіть чат-бота охорони здоров'я, який може створювати медичні записи. Чи існує ймовірність того, що користувач може маніпулювати моделлю, щоб отримати чийсь запис? Якщо програма використовує вихід LLM для підсумовування електронних листів або статей в Інтернеті, чи може електронний лист або стаття змінити спосіб обробки інших електронних листів або статей? Якщо LLM використовують для курування дописів у блогах, чи можуть зловмисні дописи вводити інструкції для штучного маніпулювання їхнім рейтингом на сторінках пошуку, незалежно від їхньої фактичної якості?

Інтеграція LLM вимагає ретельного розгляду дизайну системи та заходів безпеки для вирішення потенційних проблем безпеки. Можливо, також варто вивчити концепцію пентестування LLM, коли спеціалісти з безпеки тестують додатки, щоб виявити вразливі місця щодо своїх моделей LLM.

Підсумовуючи, інтеграція додатків на основі LLM у кібербезпеку є значним зрушенням, яке потенційно може революціонізувати галузь. Як можна побачити, додатки на базі LLM пропонують широкі можливості для виявлення загроз, оцінки ризиків і загального збільшення інтелекту. Однак вони також мають потенційні проблеми з кібербезпекою, які необхідно вирішити.

Запитання для самоперевірки



1. Сфера застосування штучного інтелекту в безпеці?
2. Назвіть проблеми генеративного ШІ та потенційні рішення?
3. У чому полягає різниця між архітектурою на основі схематехніки та архітектурою на основі розуміння?
4. Алгоритм побудови системи безпеки з використанням LLM. Опишіть.
5. Що таке керування поверхнею атаки (ASM) у випадку ШІ?
6. Які ви знаєте основні типи атак на системи штучного інтелекту?
7. Як відбувається атака на агента ШІ?

3.4. Машинне навчання у захисті інформації



Кібербезпека — це практика захисту ваших цифрових пристроїв, мереж, даних і особистих даних від несанкціонованого доступу, крадіжки, пошкодження чи неналежного використання. Це одна з найважливіших і актуальних тем цифрової ери, оскільки ми все більше покладаємося на технології в особистому та професійному житті. Кібербезпека існує у чотирьох великих категоріях: безпека даних, інформаційна безпека, безпека програм і безпека користувачів.

Машинне навчання — це кардинальна зміна філософії «вхід — процес — вихід». Традиційний підхід визначив обчислення як отримання виходу шляхом застосування процесу до вхідних даних. У машинному навчанні нам даються вхідні та вихідні дані, а завдання полягає в тому, щоб отримати процес, який поєднує ці два. Продовжуючи аналогію із системою виявлення шахрайства з кредитними картками, надано набір даних, який має вхідні функції та вихідні позначки.



Наше завдання — дізнатися, як можна обчислити прапор на основі вхідних даних. Вивчаючи процес, ми можемо узагальнити його на нові дані, які надходять на наш шлях традиційним способом «введення — процес — виведення».

Більшість питань, які пов'язані із безпекою, — це проблеми класифікації.

Класифікація — це завдання, у якому точкам даних присвоюються окремі категоричні мітки (шахрайство / нешахрайство або низький ризик / середній ризик / високий ризик).

Сьогодні машинне навчання застосовується до великої кількості галузей:

- у сфері продажів і маркетингу, щоб визначити сегмент клієнтів, які, ймовірно, купують певний продукт;
- в онлайн-рекламі для передбачення кліків і відповідного відображення реклами;
- у прогнозуванні клімату та погоди, щоб передбачати тенденції на основі даних, накопичених за століття;
- у системах рекомендацій, щоб знайти найкращі елементи (фільми, пісні, публікації та людей), які стосуються користувача.

Незважаючи на те що сьогодні машинне навчання застосовується в будь-якій галузі, особливості його використання у кібербезпеці відрізняється від інших сфер.

3.4.1. Проблеми застосування смарт-аналітики у кібербезпеці



Більшість добре вивчених методів машинного навчання мають наглядову природу. Це означає, що вони залежать від міток базової істинності для вивчення моделі. У проблемах безпеки основна правда не завжди доступна, на відміну від інших областей. Наприклад, у таких завданнях, як орієнтація на клієнтів або прогноз погоди, історична інформація може сказати, чи дійсно клієнт придбав продукт, чи дійсно йшов дощ. Це можна використовувати як основну правду.

Однак така очевидна основна істина недоступна в проблемах безпеки, таких як шахрайство або виявлення ботнетів. Ми покладемося на експертні анотації та дослідження вручну, щоб позначити дані, що вимагає ресурсів. Крім того, мітки базуються на людських знаннях і евристиках (тобто чи здається це шахрайством?), а не на абсолютній істині, як ті, які ми описали для націлювання на клієнтів або прогнозування погоди. Через відсутність міток високої надійності нам часто доводиться покладатися на неконтрольовані або напівконтрольовані методи для програм безпеки.

Також останніми роками спостерігається поштовх до збереження конфіденційності користувачів, а з підвищенням конфіденційності знижується корисність. Це виявляється особливо складним у завданнях безпеки. Неможливість відстежувати користувачів на різних вебсайтах ускладнює здатність виявляти шахрайські кліки. Відсутність дозволів на збір інформації про місцезнаходження точно зменшить сигнали, які ми маємо доступ для виявлення шахрайства з кредитними картками. Усі ці заходи зберігають конфіденційність користувачів, уникаючи надмірного відстеження та профілювання; однак вони також погіршують продуктивність моделей безпеки, що насправді на користь користувача. Що менше доступних сигналів, то нижча точність передбачення моделей.

Більшість потужних моделей ML (нейронні мережі, трансформатори та моделі навчання на основі графів) працюють як чорний ящик. Прогнози, зроблені моделлю, не піддаються інтерпретації. Оскільки ці моделі вивчають ознаки вищого порядку, людина не може зрозуміти та пояснити, чому певний приклад класифікується саме так.

Однак у додатках безпеки пояснюваність важлива. Потрібно виправдати кожен прогноз, хоча б на позитивний клас. Якщо транзакцію позначено як шахрайську, банк або компанія, яка видає кредитні картки, повинні зрозуміти, що ввійшло в прогноз, щоб з'ясувати правду. Якщо користувачів потрібно заблокувати або заборонити, платформі може знадобитися адекватне обґрунтування, більш переконливе, ніж модельне передбачення.

Класичні алгоритми машинного навчання, такі як дерева рішень і логістична регресія, забезпечують деяку інтерпретацію на основі структури дерева та коефіцієнтів відповідно. Ця потреба в інтерпретованості моделей являє собою перешкоду для використання найсучасніших методів глибокого навчання в кібербезпеці.

Природа зловмисників і вектори атак постійно змінюються. У результаті також відбувається зміна в функціях, які очікує модель, якщо дані під час висновку значно відрізняються за своєю природою від тих, на яких навчалася модель. Наприклад, абсолютно новий варіант зловмисного програмного забезпечення може не бути виявлений моделлю, оскільки в навчальних даних не було прикладів цього варіанта. Модель виявлення фейкових новин, навчена в 2019 р., може бути не в змозі розпізнати дезінформацію, пов'язану з covid-19, оскільки ніколи не навчалася на цих даних. Цей дрейф даних ускладнює створення моделей, які мають стабільну продуктивність у природних умовах.

Виявлення підозрілої активності. Багато проблем у сфері кібербезпеки створюються як завдання виявлення аномалій, оскільки поведінка зловмисника зазвичай відхиляється від належної поведінки актора.

Аномалія — це все, що виходить за рамки звичайного — подія, яка не відповідає нормальній поведінці й тому вважається підозрілою. Наприклад, якщо людина постійно використовує свою кредитну картку в Бангалорі, транзакція з використанням тієї самої картки в Парижі може бути аномалією. Якщо вебсайт отримує приблизно 10 тис. відвідувань щодня, день, коли він отримує 2 млн відвідувань, може виявитися аномальним. Аномалії нечисленні та рідкісні та вказують на дивну та підозрілу поведінку. Алгоритми виявлення аномалій не контролюються; немає позначених даних для навчання моделі. Визначається, якою є нормальна очікувана поведінка, і виявляється все, що відхиляється від неї, як ненормальне. Оскільки дані з мітками дуже рідко доступні в сферах, пов'язаних із безпекою, методи виявлення аномалій мають вирішальне значення для виявлення атак, шахрайства та вторгнень.



Слово «аномалія» означає щось, що відхиляється від того, що є стандартним, нормальним або очікуваним.

Аномалії — це події або точки даних, які не збігаються з рештою даних. Вони представляють відхилення від очікуваної тенденції даних. Аномалії трапляються рідко, тому їх небагато.

Наприклад, розглянемо бота або модель виявлення шахрайства, яка використовується на вебсайті соціальних мереж, таких як X. Якщо перевіримо кількість запитів на підписку, надісланих користувачеві за день, зможемо отримати загальне уявлення про тенденцію та побудувати ці дані. Скажімо, ми побудували ці дані за місяць і отримали таку тенденцію (рис. 3.19).

Здається, користувач має приблизно 30–40 запитів на підписку на день, проте на 8-й і 18-й дні бачимо сплеск, який чітко виділяється з денної тенденції. Ці два дні й будуть аномаліями. Аномалії також можна візуально спостерігати в двовимірному просторі. Якщо зобразити всі точки в наборі даних, аномалії мають виділятися як такі, що відрізняються від інших.

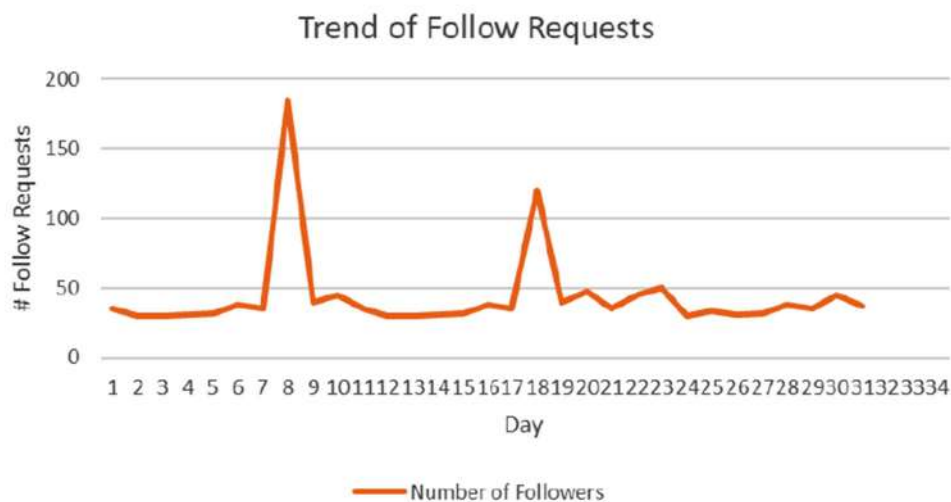


Рис. 3.19. Тенденція щодо кількості запитів на підписку протягом місяця

Наприклад, у нас є ряд метрик, таких як кількість надісланих повідомлень, уподобайок, ретвітів і тощо користувачем. Використовуючи всі ознаки разом, можемо побудувати n -вимірний вектор ознак для користувача. Застосовуючи такий алгоритм зменшення розмірності, як аналіз головних компонентів (РСА) (на високому рівні цей алгоритм може конвертувати дані в нижчі розміри та зберегти властивості), можемо зменшити їх до двох вимірів і побудувати дані. Скажімо, створюємо такий графік, як на рис. 3.20, де кожна точка представляє користувача, а розміри — основні компоненти вихідних даних. Точки, пофарбовані червоним кольором, чітко виділяються серед решти даних — це викиди.

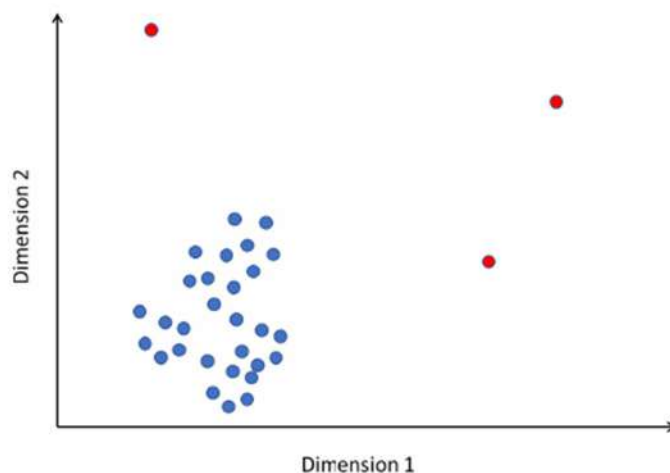


Рис. 3.20. Двовимірне представлення даних з аномаліями червоного кольору

Зауважте, що аномалії необов'язково є зловмисною подією — вони просто вказують на те, що тенденція відхиляється від очікуваного. Наприклад, коли користувач раптово отримує більшу кількість запитів у друзі, це аномально, але це могло бути тому, що він опублікував дуже цікавий вміст.

Аномалії треба досліджувати, щоб визначити, чи є вони шкідливими. Виявлення аномалій вважається важливою проблемою у сфері кібербезпеки. Незвичайні або аномальні події часто можуть вказувати на порушення безпеки або атаки. Крім того, для виявлення аномалій не потрібні мічені дані, які важко отримати в разі проблем безпеки.

Розглянемо реальний набір даних, який допоможе спостерігати та виявляти аномалії в дії, для чого використаємо Дослідницький аналіз даних (EDA).

Набір даних, який широко використовується для завдань виявлення аномалій і вторгнень, це набір даних Лабораторії безпеки мережі — виявлення знань у базах даних (NSL-KDD) (<https://www.unb.ca/cic/datasets/nsf.html>). Спочатку його було створено в 1999 р. для використання в конкурсі на 5-й Міжнародній конференції з виявлення знань та аналізу даних (KDD). Завдання конкурсу полягала в тому, щоб розробити детектор мережевого вторгнення, який є прогностичною моделлю, яка може розрізняти погані з'єднання, які називаються вторгненнями або атаками, від безпечних нормальних з'єднань. Ця база даних містить стандартний набір даних, які підлягають перевірці, включаючи широкий спектр вторгнень, змодельованих у військовому мережевому середовищі.

Спочатку імпортуємо потрібні бібліотеки Python:

```
import pandas as pd
import numpy as np
import os
from requests import get
```

Встановлюємо шляхи для розташування навчальних і тестових даних, а також шляхи до файлу мітки, який містить заголовок (імена функцій) для даних:

```
train_data_page = "http://kdd.ics.uci.edu/databases/kddcup99/kddcup.
data_10_percent.gz"
test_data_page = "http://kdd.ics.uci.edu/databases/kddcup99/kddcup.
testdata.unlabeled_10_percent.gz"
labels = "http://kdd.ics.uci.edu/databases/kddcup99/kddcup.names"
datadir = "data"
```

Далі завантажуюмо дані та назви стовпців за допомогою команди `wget` через Python. Оскільки ці файли заархівовані / стиснуті, треба спочатку розпакувати вміст за допомогою команди `gunzip`. Наступний фрагмент коду Python робить це:

```
# Завантажити навчальні дані
print("Завантаження")
os.system("wget " + train_data_page)
```

```

training_file_name = train_data_page.split("/")[-1].replace(".gz", "")
os.system("gunzip " + training_file_name )
with open(training_file_name, "r+") as ff:
lines = [i.strip().split(",") for i in ff.readlines()]
ff.close()
# Завантажте мітки навчальних стовпців
print("Завантаження")
response = get(labels)
labels = response.text
labels = [i.split(",")[0].split(":") for i in labels.split("\n")]
labels = [i for i in labels if i[0]!='']
final_labels = labels[1:]

```

Створюємо DataFrame із завантажених потоків:

```

data = pd.DataFrame(lines)
labels = final_labels
data.columns = [i[0] for i in labels]+'target'
for i in range(len(labels)):
if labels[i][1] == ' continuous.':
data.iloc[:,i] = data.iloc[:,i].astype(float)

```

На цьому етапі із завантаження даних і створення з них DataFrame завершується. DataFrame — це таблична структура даних, яка дозволяє маніпулювати, нарізати та фільтрувати дані за потреби.

Після завантаження даних з'являється можливість переглянути DataFrame, просто надрукувавши п'ять верхніх рядків: `data.head()` (рис. 3.21).

	duration	protocol	type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_srv_port_rate	dst_host_diff_host_rate	dst_host_server_rate	dst_host_srv_server_rate	dst_host_renew_rate	dst_host_srv_renew_rate	target	
0	0.0	tcp	http	SF	131.0	5460.0	0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	normal
1	0.0	tcp	http	SF	236.0	406.0	0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	normal
2	0.0	tcp	http	SF	236.0	1337.0	0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	normal
3	0.0	tcp	http	SF	216.0	1337.0	0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	normal
4	0.0	tcp	http	SF	217.0	2032.0	0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	normal

5 rows x 22 columns

5 rows x 42 columns

Рис. 3.21. П'ять перших рядків із набору даних NSL-KDD

Як бачите, відображаються п'ять верхніх рядків даних. Набір даних містить 42 стовпці. Останній стовпець під назвою `target` визначає тип мережевої атаки для кожного рядка даних. Щоб перевірити розподіл мережевих атак (тобто скільки прикладів кожного виду атак існує), можемо виконати наступний оператор: `data['target'].value_counts()`

Буде перераховано всі мережеві атаки та кількість (кількість рядків) для кожної атаки, як показано на рис. 3.22.

Бачимо, що в даних присутні різні типи атак, причому найбільшу частину становлять типи `smurf` і `neptune`. Розглянемо, як моделювати ці дані за допомогою статистичних алгоритмів.

```

smurf.          280790
neptune.        107201
normal.         97278
back.           2203
satan.          1589
ipsweep.        1247
portsweep.      1040
warezclient.    1020
teardrop.       979
pod.            264
nmap.           231
guess_passwd.   53
buffer_overflow. 30
land.           21
warezmaster.    20
imap.           12
rootkit.        10
loadmodule.     9
ftp_write.      8
multihop.       7
phf.            4
perl.           3
spy.            2
Name: target, dtype: int64

```

Рис. 3.22. Розподіл даних за мітками (тип атаки)

3.4.2. Виявлення однофакторних викидів

Розглянемо базові статистичні алгоритми, які можуть допомогти виділити аномалії та таким чином визначити вторгнення.



У найосновнішій формі виявлення аномалій, відомої як однофакторне виявлення аномалій, створюємо модель, яка враховує тенденції та виявляє аномалії на основі однієї функції за раз. Можемо побудувати кілька таких моделей, кожна з яких працює на одній функції даних.

z-оцінка. Це найбільш фундаментальний метод виявлення викидів і наріжний камінь статистичного виявлення аномалій. Заснований на центральній граничній теоремі, яка свідчить, що в більшості спостережуваних розподілів дані згруповані навколо середнього значення. Для кожної точки даних обчислюємо z-показник, який вказує, наскільки він далекий від середнього. Оскільки абсолютні відстані залежатимуть від масштабу та характеру даних, вимірюємо, на скільки стандартних відхилень від середнього значення падає точка. Якщо середнє значення та стандартне відхилення ознаки дорівнюють μ і σ відповідно, z-оцінка для точки x обчислюється у такий спосіб:

$$z = \frac{x - \mu}{\sigma}$$

Значення z — це кількість стандартних відхилень від середнього значення, на яке x падає. Центральна гранична теорема каже, що більшість даних (99%) потрапляє в межах двох стандартних відхилень (по обидва боки) від середнього. Отже, чим вище значення z , тим вище ймовірність того, що точка є аномалією.

Аномалії рідкісні та їх небагато. Щоб імітувати це налаштування, беремо вибірку з набору даних. Вибираємо тільки ті ряди, для яких ціль або звичайна, або краплеподібна. У цьому новому наборі даних приклади, позначені краплею, є аномаліями. Призначаємо мітку 0 для нормальних точок даних і 1 для аномальних, код:

```
data_resampled = data.loc[data["target"].
isin(["normal.", "teardrop."])]
def map_label(target):
    if target == "normal.":
        return 0
    return 1
data_resampled["Label"] = data_resampled["target"].apply(map_label)
```

Оскільки однофакторне виявлення викидів працює лише на одній функції одночасно, давайте виберемо `wrong_fragment` як функцію для демонстрації. Щоб обчислити z -показник для кожної точки даних, спочатку обчислюємо середнє значення та стандартне відхилення `wrong_fragment`. Потім віднімаємо середнє значення всієї групи від значення `wrong_fragment` у кожному рядку та ділимо його на стандартне відхилення, код:

```
mu = data_resampled["wrong_fragment"].mean()
sigma = data_resampled["wrong_fragment"].std()
data_resampled["Z"] = (data_resampled["wrong_fragment"] - mu) / sigma
```

Можемо побудувати графік розподілу z -показника, щоб візуально визначити природу розподілу. Наступний рядок коду може створити графік щільності (рис. 3.23):

```
data_resampled["Z"].plot.density()
```

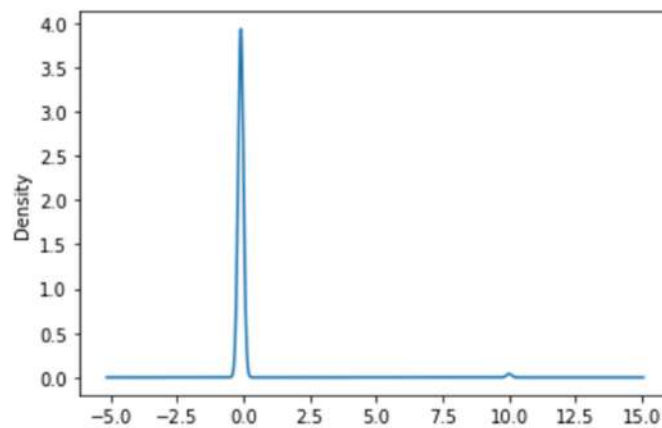


Рис. 3.23. Графік щільності z -показника

Бачимо різкий сплеск щільності навколо 0, який вказує на те, що більшість точок даних мають z -показник близько 0. Крім того, зверніть увагу на дуже маленьке спалах навколо 10; це невелика кількість викидних точок, які мають високі z -показники.

Тепер все, що потрібно зробити, це відфільтрувати ті рядки, які мають z -показник більше 2 або менше -2. Потрібно призначити мітку 1 (передбачувана аномалія) цим рядкам і 0 (передбачувана норма) іншим, код:

```
def map_z_to_label(z):
    if z > 2 or z < -2:
        return 1
    return 0
data_resampled["Predicted_Label"] = data_resampled["Z"].apply(map_z_to_label)
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt
import seaborn as sns
confusion = confusion_matrix(data_resampled["Label"],
                             data_resampled["Predicted_Label"])
plt.figure(figsize = (10,8))
sns.heatmap(confusion, annot = True, fmt = 'd', cmap="YlGnBu")
```

Тепер маємо фактичні та прогнозовані мітки для кожного рядка, можемо оцінити продуктивність нашої моделі за допомогою матриці плутанини.

Матриця плутанини дає швидкий спосіб візуально перевірити продуктивність моделі (рис. 3.24). Хороша модель покаже найвищі справжні позитивні та негативні сторони, тоді як дві інші матимуть менші значення. Матриця плутанини дозволяє зручно обчислювати показники, що стосуються класифікації, а саме точність, прецизійність, запам'ятовування та оцінку F-1.

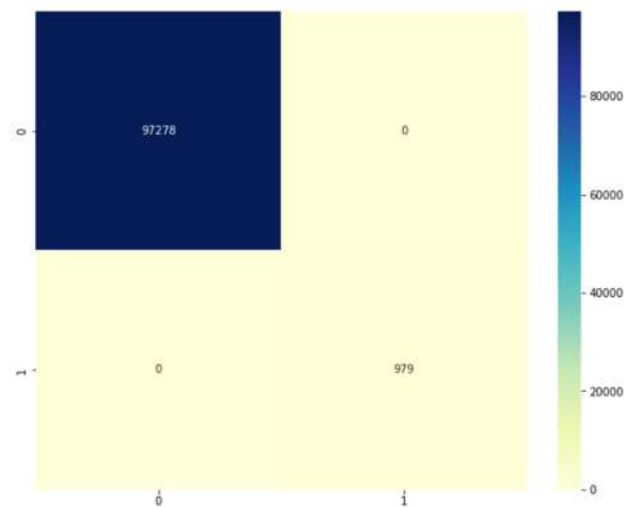


Рис. 3.24. Матриця плутанини

Пакет `scikit-learn` у `Python` надає вбудований метод, який дозволяє обчислювати матрицю, а інший пакет під назвою `seaborn` дозволяє швидко побудувати її візуалізацію (див. рис. 3.19). Код проілюстровано в наступному фрагменті:

Матриця показує, що модель працює дуже добре; усі точки даних класифіковано правильно. Ми маємо лише справжні позитивні та негативні результати, а не помилкові позитивні чи помилкові негативні результати.

Також ми можемо для виявлення аномалій застосовувати наступні методи:

1. Еліптична огинаюча — це алгоритм для виявлення аномалій у гаусових даних. На високому рівні алгоритм моделює дані як високо вимірний розподіл Гауса.
2. Фактор локального викиду (також відомий як LOF) — це алгоритм виявлення аномалій на основі щільності. Він перевіряє точки в локальному оточенні точки, щоб виявити, чи ця точка є аномальною.
3. Машина опорних векторів (SVM) — це алгоритм, який широко використовується для задач класифікації. Однокласова SVM (OC-SVM) — це версія цього алгоритму, яка використовується для виявлення аномалій.
4. Алгоритм ізольованого лісу — це неконтрольований алгоритм виявлення аномалій. Він базується на базовому визначенні аномалій; аномалії є рідкісними явищами та значно відрізняються від нормальних точок даних.

Використовуючи методи, описані в цьому розділі, ви зможете перевірити набір даних і виявити аномальні точки даних. Виявлення аномалій є ключовим у багатьох проблемах кібербезпеки, таких як виявлення вторгнень в мережу або шахрайства.

Запитання для самоперевірки



1. Що таке аномалія?
2. Які є проблеми застосування смарт-аналітики у кібербезпеці?
3. Як виявити однофакторні викиди?
4. Процедура пошуку аномалій в даних. Опишіть.
5. Що таке матриця плутанини?
6. Які бібліотеки мови `Python` застосовуються для пошуку аномалій?

3.5. Аналітика етичного хакерства



Загальна картина, що характеризує руйнівників і злочинців в галузі комп'ютерної безпеки, змінюється з часом, втім базові категорії загроз і небезпек лишаються незмінними. Для того щоб створити плани атакуючих злочинців, проводяться спеціальні дослідження, і тому завжди важливо правильно класифікувати різні типи реально існуючих атак.



Зростаючі виклики в кібербезпеці включають:

- загрози, що постійно змінюються: швидкий розвиток цифрових технологій призводить до зростання складності та кількості кібератак;
- проблеми захисту даних: забезпечення безпеки чутливих даних залишається одним з основних викликів у сфері кібербезпеки;
- розвиток систем штучного інтелекту;
- обробка великих обсягів даних.

3.5.1. Безпека даних та інформаційна безпека



Дані в необробленому форматі зберігаються на жорстких дисках в офісах або «хмарі», які згодом зберігаються на фізичних машинах у центрах обробки даних. На рівні даних роль кібербезпеки полягає в забезпеченні їх безпеки. Отже, основна увага приділяється підтримувannya конфіденційності, цілісності та доступності даних (три цілі тріади ЦРУ). Проблеми кібербезпеки на цьому рівні зосереджені на нових криптографічних схемах, легких алгоритмах шифрування, системах відмовостійкості та дотриманні правил зберігання даних.

Інформаційна безпека. Дані з центрів обробки даних і хмари перетворюються на інформацію, яка використовується компаніями, для створення різноманітних сервісів. На інформаційному рівні роль кібербезпеки полягає в тому, щоб співробітники мали доступ до інформації та її правильного використання. Проблеми на цьому рівні фокусуються на адмініструванні безпеки мереж, виявленні порушень політики та внутрішніх загроз, а також забезпечення відсутності випадкового витоку особистих даних.

Безпека додатків. Інформація перетворюється на форму, зручну для споживачів за допомогою різноманітних сервісів. Прикладом цього слугує інформація про користувачів Facebook, яка перетворюється на список найкращих рекомендацій друзів. На рівні програм роль кібербезпеки полягає в тому, щоб гарантувати, що програма не може бути скомпрометована. Проблеми на цьому рівні зосереджені на виявленні зловмисного програмного забезпечення, атак на ланцюг

поставок, аномалій, ботів і автоматизованих облікових записів, а також на позначенні фішингових електронних листів і шкідливих URL-адрес.

Безпека користувачів. Нарешті програми використовуються кінцевими користувачами, і роль кібербезпеки тут полягає в тому, щоб забезпечити безпеку цих користувачів. Проблеми на цьому рівні включають виявлення ненависті, модерування вмісту, позначення шахрайських транзакцій, характеристику образливої поведінки та захист користувачів від цифрових злочинів (викрадення особистих даних, шахрайства та вимагання). Зауважте, що цей аспект виходить за рамки технічних елементів кібербезпеки і входить до сфери гуманітарних, юридичних та соціальних наук.



Метою мережевої безпеки є захист корпоративної мережі від несанкціонованого доступу. Мережна безпека перевіряє дані, що проходять мережею підприємства, щоб виявити вторгнення в мережу та підключені до неї комп'ютери. Крім того, архітектура мережі та її засоби захисту можуть бути використані для спрямування активності користувачів і зловмисників, спрямовуючи її до датчиків і захисних механізмів, а також від слабких місць і вразливостей.

Безпеку мережі слід розглядати з позицій засобів контролю безпеки, які включають наступне:

- профілактичні засоби керування, такі як брандмауери, які блокують дії зловмисників і відокремлюють ділянки мережі один від одного;
- аналітика, такі як виявлення вторгнень, які виявляють дії зловмисників, які неможливо заблокувати;
- елементи керування моніторингом, які фіксують активність, введену в механізми кореляції, які підтримують криміналістику, розслідування та більш складне виявлення атак, що враховує кілька змінних і джерел даних.

Більшість цілеспрямованих атак певним чином використовують мережу та покладаються на мережу для продовження атаки, поки вона триває. Серед векторів загроз безпеці мережі є такі:

- зловмисники проникають на підприємство через вихідні мережеві підключення із серверів або клієнтів у внутрішній мережі;
- зловмисники проникають на підприємство через мережеві підключення серверів, що виходять в Інтернет;
- зловмисники використовують внутрішні мережі для переміщення між комп'ютерами всередині підприємства;
- зловмисники використовують корпоративні мережі, щоб отримати дані та видалити їх із підприємства;
- зловмисники беруть під контроль компоненти мережевої інфраструктури, а потім використовують їх, щоб проникнути на підприємство чи обійти інші заходи безпеки.



Безпека програмних додатків передбачає надання можливостей захисту, які є специфічними для додатків, що використовуються на підприємстві. Програми, які найбільше потребують додаткового захисту, це ті, які спілкуються через мережу і доступні з Інтернету.

За цим простим визначенням, технології та можливості безпеки додатків включають захист електронної пошти, функції брандмауера з підтримкою додатків, шлюзи баз даних і вебпроксі-сервери пересилання.

Безпека програм захищає програму від кібератак, розуміючи внутрішню роботу програми та її протоколів, як-от мова гіпертекстової розмітки (HTML), яка використовується для створення вебсторінок, або простий протокол передачі пошти (SMTP), який використовується для передачі електронної пошти. Безпека програми допомагає запобігти атакам, які використовують вразливі місця програми або протоколи зв'язку програми. Простіше кажучи, безпека програм залежить від вимог різних протоколів програми та відповідних типів даних, які обробляють протоколи.

Серед векторів загроз безпеці програм є такі.

- Багато хакерів проникають на підприємство, використовуючи електронну пошту для надсилання шкідливих повідомлень користувачам. Потім ці повідомлення можуть містити вкладення або посилання, які використовують уразливості в інших програмах (наприклад, офісна продуктивність, програми для перегляду зображень або документів), щоб отримати контроль над кінцевими обчислювальними пристроями (такими як персональні комп'ютери, сервери та мобільні пристрої).
- Інші хакери використовують уразливості у веббраузерах і вебплагінах (додаткових функціях), щоб отримати контроль над користувачами, які відвідують шкідливі сайти. Ці вектори загроз можуть бути особливо підступними, коли зловмисники скомпрометують законний вебсайт і використовують його для надання зловмисного програмного забезпечення відвідувачам, які нічого не підозрюють.
- Також вектори атак передбачають використання вразливостей у серверних програмах підприємства (таких як сервери вебдодатків), щоб взяти під контроль ці сервери та використовувати їх для проникнення на підприємство.
- Потрапляючи в середину мережі, зловмисники використовують програми або для використання їх даних та компрометації моделей машинного навчання, або для використання самих програм для зловмисних цілей.

Безпека даних. Безпека для Big Data — це важливий аспект, який вимагає особливої уваги. Добре збалансовані набори даних трапляються рідко, а використання незбалансованих наборів даних може призвести до необ'єктивності результатів, які важко виявити. Ось декілька ключових пунктів:

- *Єдина модель безпеки в Дельта VI.* Багато компаній сьогодні можуть зібрати й проаналізувати більше даних завдяки розвитку технологій і зниженню вартості системи зберігання. Однак у процесі впровадження системи для аналізу великих даних можуть виникнути складності, пов'язані з безпекою.
- *Aleo.* Безпека в епоху Big Data: Проєкт Aleo, що розвивається у сфері криптографії та блокчейну, змінює ключ у забезпеченні рольової безпеки в епоху Big Data. Aleo використовує технологію блокчейну для створення приватних, безпечних та масштабованих додатків.
- *Розподілений доступ.* Компанії — власники Big Data використовують розподілений доступ: у різних груп співробітників різний рівень доступу і тільки до визначених сегментів бази даних. Ці аспекти допомагають забезпечити безпеку даних в епоху Big Data.



Однак важливо пам'ятати, що безпека — це постійний процес, який вимагає постійного моніторингу, оновлення та адаптації до нових загроз.

Блокчейн пропонує децентралізоване рішення, що може знизити ризики централізованих кібератак. Блокчейн гарантує незмінність записів, що сприяє забезпеченню цілісності даних. Децентралізація через блокчейн Aleo знижує ризики, пов'язані з централізованими базами даних та дозволяє користувачам самостійно контролювати свою цифрову безпеку.



Найімовірніший спосіб, яким атакуючі можуть проникнути у вашу інфраструктуру, — це шлях через мережеве середовище. Забезпечення безпеки мережі (network security) являє собою велику практичну сферу діяльності із захисту комп'ютерних мереж і кінцевих пунктів мережного доступу від шкідливих дій, некоректного використання та критичних відмов (аварій). Мережеві захисні екрани (firewalls) є, ймовірно, найбільш широко відомими системами захисту мереж, що застосовують стратегії доступу та фільтрацію неавторизованого трафіку між об'єктами в мережевому середовищі. Втім забезпечення захисту мережі не обмежується лише одними мережевими екранами.

3.5.2. Модельна задача



Більшість сучасних пристроїв ведуть журнал активності. Там зберігається інформація, щодо пристроїв, з якими спілкуються комп'ютер, завантажених файлів і типу протоколу зв'язку. Ця корисна інформація відіграє життєво важливу роль у визначенні джерела загрози та потенційної шкоди від неї.

Оперативне перехоплення даних — це головний спосіб фіксації мережної активності для онлайнного та офлайнного аналізу. Подібно до відеореєстраторів

спостереження за дорожнім трафіком на перехресті, аналізатори пакетів (packet analyzers) (також відомі під назвою сніфери (sniffers) пакетів / мереж / протоколів) перехоплюють і реєструють у журналах трафік у мережі. Мережеві журнали реєстрації (logs) корисні не лише для дослідження інцидентів, пов'язаних з безпекою, але також для налагодження, спостереження за продуктивністю та моніторингу мережевих операцій. У разі розміщення в належних пунктах мереж і при правильно налаштованих мережних комутаторах та інтерфейсах аналізатор пакетів може стати корисним інструментом для генерації докладних наборів даних, що дають повну картину того, що відбувається в мережі.

Наприклад, якщо сервер став жертвою комп'ютерного вірусу, вивчення його активності може допомогти точно визначити дію, яка призвела до завантаження шкідливої програми / файлу.

Для аналізу потрібно буде використовувати кілька бібліотек Python:

Pandas як `pd` → читати дані та зберігати у наборі даних

Matplotlib як `plt` → дані та графіка

Networkx як `nx` → побудова графіків даних як вузлів, якщо вони спілкувалися

Далі завантажуюмо дані на основі шляху до розміщення файлу. Дивлячись на набір даних, бачимо наступні стовпці: «No.», «Time», «Source», «Destination», «Protocol», «Length», «Info», як показано на рис. 3.25.

```

wireshark_url = "../input/wireshark-lab-data/Wireshark_python.csv"

wireshark_data = pd.read_csv(wireshark_url)

wireshark_data.head(10)

```

	No.	Time	Source	Destination	Protocol	Length	Info
0	1	0.000000	192.168.159.129	10.10.20.129	TCP	66	1030 > 80 [SYN] Seq=0 Win=64240 Len=0 MSS=14...
1	2	0.000248	192.168.159.129	10.10.20.129	TCP	66	1031 > 80 [SYN] Seq=0 Win=64240 Len=0 MSS=14...
2	3	0.001128	10.10.20.129	192.168.159.129	TCP	60	80 > 1030 [SYN, ACK] Seq=0 Ack=1 Win=64240 L...
3	4	0.001193	192.168.159.129	10.10.20.129	TCP	54	1030 > 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
4	5	0.001364	10.10.20.129	192.168.159.129	TCP	60	80 > 1031 [SYN, ACK] Seq=0 Ack=1 Win=64240 L...
5	6	0.001515	192.168.159.129	10.10.20.129	TCP	54	1031 > 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
6	7	0.010413	192.168.159.129	10.10.20.129	HTTP	516	GET / HTTP/1.1
7	8	0.011105	10.10.20.129	192.168.159.129	TCP	60	80 > 1031 [ACK] Seq=1 Ack=463 Win=64240 Len=0
8	9	0.012930	192.168.159.129	192.168.159.2	DNS	76	Standard query 0x9590 A wpad.localdomain

Рис. 3.25. Перегляд набору даних рядок за рядком

Тепер, коли дані розташовано в акуратному наборі даних, можемо використувати функції:

- `groupby()` → виберіть стовпець для ідентифікації та групування за унікальними значеннями;

- `count()` → підрахувати, скільки разів значення з'являється у кадрі даних як значення;
- `sort_values` → відображати значення на основі розміру.

Будемо використовувати дані функції, щоб проаналізувати пристрої, які ініціювали розмови, приймали повідомлення та типи їх зв'язку.

Перегляд джерел показує, які пристрої мали найменшу / найбільшу кількість зв'язків (рис. 3.26).

```
sources = wireshark_data.groupby("Source").Source.count()
sources.sort_values()
```

```
Source
23.15.55.245          2
192.168.159.2        11
204.79.197.219       15
204.79.197.200       17
fe80::19b3:a678:9d18:ca3 19
fe80::8406:feb5:36e:aab6 30
192.168.159.1        34
104.42.196.205       135
192.168.159.129      370
10.10.20.129         538
Name: Source, dtype: int64
```

Рис. 3.26. Кількість зв'язків

Також можете записати ту саму інформацію з пунктів призначення (рис. 3.27).

```
destinations = wireshark_data.groupby("Destination").Source.count()
destinations.sort_values()
```

```
Destination
ff02::1:2          1
23.15.55.245       2
204.79.197.219     11
192.168.159.255    12
204.79.197.200     13
224.0.0.252       16
ff02::1:3         16
192.168.159.2     18
ff02::fb         32
224.0.0.251       33
104.42.196.205    91
10.10.20.129     208
192.168.159.129  718
Name: Source, dtype: int64
```

Рис. 3.27. Пункти призначення, зв'язки

Цей тип інформації варто враховувати в аналізі мережевого трафіку, оскільки деякі IP-адреси можуть бути з невідомих пристроїв (що може означати, що вони є зловми-сними), або якщо деякі IP-адреси демонструють ненормальний рівень активності.

Якщо проаналізувати протоколи, буде видно кілька типів зв'язку HTTP. Фа-хівець із безпеки знає, що інформація, яка передається, не шифрується. Це озна-чає, що будь-хто може прочитати це, і треба сподіватися, що нічого конфіденцій-ного там не було! (рис. 3.28).

```
protocols = wireshark_data.groupby("Protocol").Source.count()
protocols.sort_values()
```

Protocol	
DHCPv6	1
ICMP	1
HTTP	14
NBNS	18
DNS	22
LLMNR	32
MDNS	65
TLSv1.2	92
TCP	926

Name: Source, dtype: int64

Рис. 3.28. Який тип зв'язку має аномальну частоту на основі звичайної діяльності

Застосуємо теорію графів. Використовуючи бібліотеку networkx у Python, мо-жемо будувати графіки вузлів, які представляють тип зв'язку.

Оскільки дані розташовані у фреймі даних pandas, можемо використовувати networkx безпосередньо на них і вибрати такі вузли, як «Джерело» і «Ціль». Для аналізу є цікавими вузли «Джерело» і «Призначення». Вибір цих вузлів допо-може побачити, як пристрої в цій мережі обмінюються даними (рис. 3.29).

```
import networkx as nx
network = nx.from_pandas_edgelist(wireshark_data, source='Source', target='Destination', edge_
network.nodes()
```

NodeView((('192.168.159.129', '10.10.20.129', '192.168.159.2', '224.0.0.251', '192.168.159.1', 'fe80::8406:feb5:36e:aab6', 'ff02::fb', 'fe80::19b3:a678:9d18:ca3', 'ff02::1:3', '224.0.0.252', '204.79.197.200', '192.168.159.255', '104.42.196.205', 'ff02::1:2', '23.15.55.245', '204.79.197.219')))

```
network.edges()
```

EdgeView(((('192.168.159.129', '10.10.20.129'), ('192.168.159.129', '192.168.159.2'), ('192.168.159.129', '224.0.0.251'), ('192.168.159.129', '224.0.0.252'), ('192.168.159.129', '204.79.197.200'), ('192.168.159.129', '192.168.159.255'), ('192.168.159.129', '104.42.196.205'), ('192.168.159.129', '23.15.55.245'), ('192.168.159.129', '204.79.197.219'), ('224.0.0.251', '192.168.159.1'), ('192.168.159.1', '224.0.0.252'), ('192.168.159.1', '192.168.159.255'), ('fe80::8406:feb5:36e:aab6', 'ff02::fb'), ('fe80::8406:feb5:36e:aab6', 'ff02::1:3'), ('ff02::fb', 'fe80::19b3:a678:9d18:ca3'), ('fe80::19b3:a678:9d18:ca3', 'ff02::1:3'), ('fe80::19b3:a678:9d18:ca3', 'ff02::1:2'))))

Рис. 3.29. Фрагмент коду, вирізаний у першій комірці

Із вибраними вузлами використаємо бібліотеку `networkx` із функцією `draw_circular` для відображення карти мережі на основі трафіку (рис. 3.30).

```
nx.draw_circular(network, with_labels = True)
```

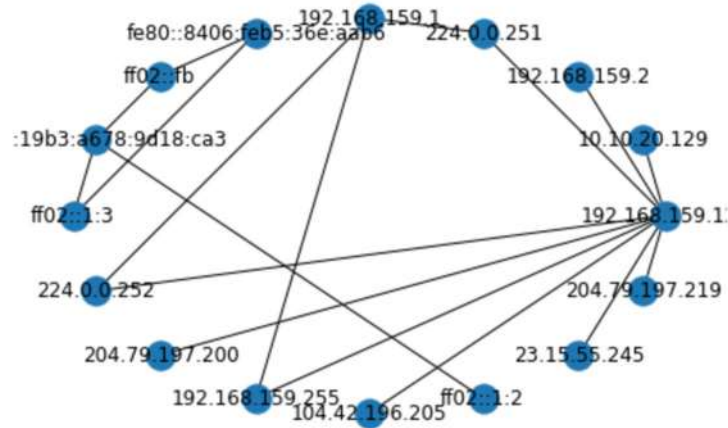


Рис. 3.30. Карта мережі

Припустімо, що позначили пристрій червоним прапорцем. Ми не знаємо його власника, тому його дії не повинні відображатися у приватному мережевому трафіку.

Викликаємо цей підозрілий пристрій і за допомогою `networkx` створимо кольорову карту, яка показуватиме, з якими пристроями зломисник спілкувався (рис. 3.31).

```
suspect = "192.168.159.2"
pos = nx.spring_layout(network)
nx.draw(network, pos, node_color = "green",
        node_size = 300, with_labels = True)

options = {"node_size": 1000, "node_color": "r"}
nx.draw_networkx_nodes(network, pos, nodelist=[suspect], **options)

plt.show()
```

Рис. 3.31. Код програми

Спостерігаючи за результатами, можемо побачити, з якими пристроями зломисник обмінювався даними, і виявити інші пристрої, до яких він потенційно міг дістатися (рис. 3.32).

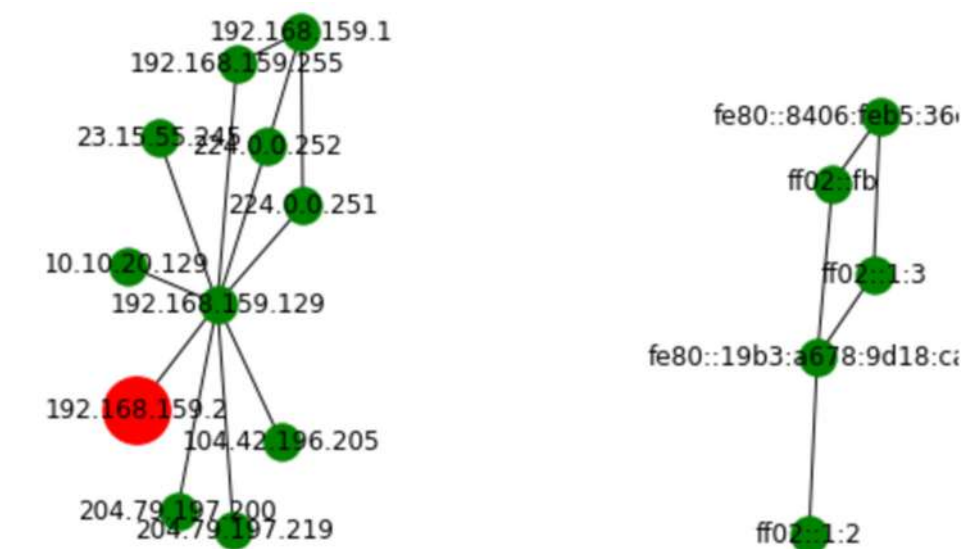


Рис. 3.32. Кольорова карта мережі

Розглянувши шкідливий пристрій глибше, можемо створити новий набір даних, який включає інші пристрої, з якими він спілкувався, тип зв'язку, час, коли це сталося, та інформацією про нього (рис. 3.33).

```

wireshark_data.loc[wireshark_data['Source'] == '192.168.159.2']

```

No.		Time	Source	Destination	Protocol	Length	Info
28	29	1.017454	192.168.159.2	192.168.159.129	DNS	76	Standard query response 0x9590 No such name A ...
51	52	2.040600	192.168.159.2	192.168.159.129	DNS	220	Standard query response 0xcee1 A www.bing.com ...
96	97	5.766607	192.168.159.2	192.168.159.129	DNS	208	Standard query response 0x390f A nav.smartscre...
470	471	13.807402	192.168.159.2	192.168.159.129	DNS	208	Standard query response 0x396c A nav.smartscre...
542	543	22.182801	192.168.159.2	192.168.159.129	DNS	208	Standard query response 0xa868 A nav.smartscre...
681	682	26.330684	192.168.159.2	192.168.159.129	DNS	76	Standard query response 0x86fd No such name A ...
688	689	27.331617	192.168.159.2	192.168.159.129	DNS	76	Standard query response 0x86fd No such name A ...
691	692	29.167490	192.168.159.2	192.168.159.129	DNS	208	Standard query response 0x6248 A nav.smartscre...
1016	1017	33.878308	192.168.159.2	192.168.159.129	DNS	76	Standard query response 0x35a5 No such name A ...
1025	1026	34.050231	192.168.159.2	192.168.159.129	DNS	160	Standard query response 0xce3f A edge.microsof...
1054	1055	35.776693	192.168.159.2	192.168.159.129	DNS	208	Standard query response 0x300e A nav.smartscre...

Рис. 3.33. Новий набір даних

Це можна зробити для стовпця «Протокол», щоб створити різні кадри даних залежно від типу зв'язку.

Забезпечення секретності в програмних системах, побудованих на основі машинного навчання та штучного інтелекту повинно входити в список найважливіших вимог, оскільки порушення приватності і вразливість секретних даних зазвичай призводять до серйозних і дорогих наслідків.

Виробничі системи повинні забезпечувати засоби захисту та гарантувати збереження секретності, засновані на надійних і перевірених теоретичних і технічних робочих програмних середовищах, і обмежувати збитки, які можуть спричинити атакуючі, отримавши секретну інформацію.

Запитання для самоперевірки



1. *Що є метою мережевої безпеки?*
2. *Що є метою безпеки даних?*
3. *Що таке оперативне перехоплення даних?*
4. *Які бібліотеки Python необхідні для аналізу проникнення в мережу?*
5. *Як в кібербезпеці застосовується теорія графів?*
6. *Як виявити, з якими пристроями зломисник спілкувався в мережі?*
7. *Які складові безпеки для BigData ви знаєте?*

3.6. Смарт-безпека у HR



Застосування автоматизованих систем на основі штучного інтелекту в управлінні персоналом і професійними ризиками на корпоративному рівні з кожним роком дедалі більше зростає. Штучний інтелект впроваджується для підвищення ефективності та точності процесів управління персоналом. ШІ використовують для автоматизації повсякденних завдань з управління персоналом, покращення процесів найму й відбору та оптимізації продуктивності співробітників.



ШІ також використовується для покращення залучення й утримання співробітників. Чат-боти на основі штучного інтелекту можна використовувати, щоб відповідати на запитання співробітників, надавати персоналізовані рекомендації та навіть допомагати працівникам знаходити правильні вакансії. Водночас аналітику, керовану штучним інтелектом, можна використовувати для виявлення моделей поведінки співробітників і для розуміння того, як краще залучати й утримувати співробітників.

3.6.1. Основні напрями використання штучного інтелекту в управлінні персоналом

Використання ШІ в управлінні персоналом допомагає зменшити упередженість на робочому місці. Системи, керовані штучним інтелектом, можна використовувати для усунення несвідомих упереджень у процесі найму і відбору, а також для управління продуктивністю. Аналітику на основі штучного інтелекту можна застосовувати й для виявлення моделей дискримінації на робочому місці та для вироблення заходів протидії цій дискримінації.



За останні роки зростає використання автоматизованих систем, заснованих на штучному інтелекті, у корпоративному управлінні персоналом та професійних ризиках. Штучний інтелект впроваджується з метою підвищення ефективності та точності процесів управління персоналом. Ці системи автоматизують повсякденні завдання, такі як найм, відбір та оптимізація продуктивності співробітників.

3.6.2. Основні переваги використання штучного інтелекту у кадровому рекрутингу включають

Таблиця 1.

Вплив систем на базі штучного інтелекту на кадрову справу

Функції кадрового рекрутингу	Вплив штучного інтелекту (ШІ)
Підвищення швидкості та збільшення обсягу пошуку кандидатів.	Штучний інтелект допомагає одночасно шукати кандидатів на посаду в різних джерелах. Він аналізує великі обсяги даних з професійних мереж, резюме, профілів у соціальних мережах та інших джерел, щоб знайти найбільш відповідних претендентів для вакансій.
Аналіз ринку праці.	Штучний інтелект може автоматично аналізувати різноманітні джерела інформації про зарплати, що дозволяє визначити адекватний рівень заробітної плати для конкретної вакансії. Це може значно підвищити шанси залучити співробітників необхідного рівня.
Створення автоматизованих вакансій.	Одним з базових інструментів штучного інтелекту для HR-працівників (менеджерів з персоналу) є програми, які автоматично розміщують вакансії на різних робочих платформах і сайтах. Ці програми аналізують ринок праці та допомагають підібрати найефективніші ключові слова, що привернуть увагу кандидатів на посаду.
Оптимізація відбору кандидатів.	Інструменти штучного інтелекту дозволяють аналізувати професійні навички, освіту, досвід та інші критерії для відбору кандидатів. Це допомагає знизити людський фактор та підвищити об'єктивність при відборі співробітників на посади.
Автоматизовані співбесіди.	Інтерв'ювання кандидатів — це процес, який вимагає значних зусиль від співробітників відділу кадрів. Використання штучного інтелекту дозволяє автоматизувати перші етапи співбесід, використовуючи чат-ботів. Ці боти можуть ставити запитання, аналізувати відповіді, оцінювати реакцію кандидата і визначати його сумісність з вакансією. Такий підхід дозволяє зекономити час і зусилля, витрачені на співбесіди з кандидатами на посади.
Прогнозування успішності кандидатів.	Застосування аналітики на основі ШІ та машинного навчання дозволяє створити моделі, що допомагають передбачити, наскільки успішним може бути кандидат на конкретній посаді.

За даними [3] можна виділити такі основні ШІ-інструменти, які спрощують та автоматизують процеси рекрутингу, роблячи ці процеси для HR-відділів більш ефективними й оптимальними:

- *ATS (Applicant Tracking System)*, система управління кандидатами — це інструмент, який допомагає зберігати, відстежувати та аналізувати дані про кандидатів на посади на різних етапах рекрутингового процесу. ATS дозволяє автоматизувати завдання, такі як публікація вакансій, управління резюме, планування співбесід і звітність.
- *Розпізнавання мови й аналіз текстів*. Інструменти, які використовуються для аналізу та розуміння контексту резюме, супровідних листів та інших текстових документів, можуть автоматично виділяти ключові слова або

фрази, що вказують на відповідність кандидата вимогам вакансії. Це допомагає зрозуміти, чи відповідає кандидат необхідним критеріям для конкретної робочої посади

- *Чат-боти та віртуальні асистенти.* Це перший етап комунікації з кандидатами. Вони дозволяють розбиратись зі стандартними запитаннями, надавати інформацію про компанію та збирати основні дані про кандидатів.
- *Аналітика та прогнозування.* Штучний інтелект (ШІ) вміє аналізувати великі обсяги даних про кандидатів, які стосуються внутрішніх та зовнішніх ринків праці. Це допомагає ідентифікувати ключові тенденції на ринку праці та передбачити попит на певні професійні навички.
- *Відеоінтерв'ювання.* Відеоінтерв'ювання дозволяють проводити онлайн-зустрічі з кандидатами. Інструменти штучного інтелекту можуть допомогти аналізувати звернення кандидатів, їхні жести та поведінку, щоб отримати додаткові інсайти щодо професійної придатності



Системи управління продуктивністю на основі штучного інтелекту розроблені, щоб надавати компаніям дані про продуктивність співробітників у реальному часі. Системи на основі штучного інтелекту можна використовувати для автоматизації певних аспектів управління продуктивністю, таких як встановлення цілей і відстеження прогресу. Це може допомогти зменшити кількість часу та ресурсів, необхідних для управління продуктивністю співробітників.



Окрім надання даних у режимі реального часу, системи керування продуктивністю на основі штучного інтелекту також можна використовувати для **виявлення потенційних проблем та надання рекомендацій щодо покращення.** Наприклад, системи на основі штучного інтелекту можуть аналізувати дані про продуктивність співробітників і визначати закономірності, які можуть вказувати на потребу в додатковому навчанні або ресурсах. Це може допомогти компаніям вирішити потенційні проблеми з персоналом, перш ніж вони стануть більш серйозними, тобто ШІ за змістом сприяє безпеці в управлінні кадрами.



Використання штучного інтелекту для підвищення залучення й утримання співробітників. Наприклад, аналіз настроїв у колективі за допомогою ШІ можна використовувати для моніторингу робочої атмосфери. Потім на основі цих даних визначаються сфери удосконалення, такі як, наприклад, забезпечення кращого навчання персоналу. ШІ також можна використовувати для персоналізації досвіду співробітників. Використавши дані, керовані штучним інтелектом, компанії можуть створювати *індивідуальні плани залучення, адаптовані до кожного окремого працівника.*

Це може включати надання індивідуальних відгуків, встановлення індивідуальних цілей, а також пропонування винагород і визнання. Нарешті ШІ можна використовувати для покращення спілкування між співробітниками та керівництвом. Чат-боти на основі штучного інтелекту можна використовувати, щоб відповідати на запитання співробітників і надавати своєчасні відгуки. Це може допомогти створити більш відкритий та прозорий робочий клімат, що сприятиме більшому залученню й утриманню персоналу.



Автоматизація кадрових завдань за допомогою штучного інтелекту та зменшення адміністративного тягаря у відділах кадрів (HR-відділах). Штучний інтелект використовується для автоматизації різноманітних завдань з управління персоналом, що дозволяє відділам кадрів (HR-відділам) зменшити адміністративний тягар і зосередитися на більш стратегічних ініціативах. ШІ використовується для спрощення процесу найму. Чат-боти на основі штучного інтелекту можуть перевіряти потенційних кандидатів, допомагаючи визначити найбільш відповідних претендентів на певну посаду.



ШІ використовується для *автоматизації процесу адаптації персоналу*. Зокрема інструменти на основі штучного інтелекту можна використовувати для створення персоналізованих планів адаптації для нових співробітників, гарантуючи, що вони отримають необхідне навчання та ресурси, щоб успішно працювати на своїй посаді.



У жовтні 2020 р. експерти Всесвітнього економічного форуму дійшли висновку, що, хоча штучний інтелект забере 85 млн робочих місць у всьому світі до 2025 р., він також створить 97 мільйонів нових робочих місць у різних сферах: від великих даних і машинного навчання до інформаційної безпеки та цифрового маркетингу. Наприклад, на сьогодні Україна — одна з лідерів за кількістю розробників ШІ.

3.6.3. Основні ризики ШІ в НР

Проте, крім низки переваг, штучний інтелект, як і будь-яка проривна технологія, несе суспільству нові ризики. Зокрема, ШІ веде до трансформації ринку праці, змінивши природу давно усталених ролей персоналу.



Financial Times провів огляд [2] дослідження Goldman Sachs, результати якого свідчать, що 300 мільйонів штатних співробітників великих економік піддаються впливу технологій штучного інтелекту. Дві третини професій у США та Європі потраплять під певний рівень впливу автоматизації, а юристи й адміністративні співробітники ризикують стати непотрібними. Майже 7% американських працівників займають посади, принаймні 50% завдань яких може виконувати генеративний ШІ, тому вони вразливі і можуть втратити роботу.



Інструменти, такі як ChatGPT (який генерує тексти на задані теми та відповідає на питання зрозумілою мовою), DALL-E (який створює високоякісні зображення, виходячи з текстових описів) і MusicLM (який генерує музику), можуть призвести до того, що люди, які працюють у креативних професіях, залишаться без роботи. Ілюстратори, комерційні графічні дизайнери та редактори зображень можуть втратити актуальність у світі, де технології штучного інтелекту (ШІ) стають все більш поширеними. Ризики для ринку праці вимагають уваги від парламентів та урядів провідних країн. При розробці державних стратегій розвитку ШІ важливо захищати ключові інтереси людини та суспільства.



Системи управління персоналом (HR), побудовані на базі штучного інтелекту, мають вражаючі можливості. Вони здатні швидко та точно обробляти великі обсяги даних, що дозволяє спеціалістам з управління персоналом зосередитися на стратегічних завданнях. Процеси найму та відбору, побудовані на основі штучного інтелекту, допомагають виявляти найкращих кандидатів на роботу. Такі системи можуть аналізувати резюме, відбирати потенційних співробітників та забезпечувати більш об'єктивний підхід до відбору.

Крім того, системи управління продуктивністю співробітників, побудовані на базі штучного інтелекту, можуть допомогти визначити сфери для вдосконалення. Вони аналізують продуктивність, виявляють слабкі місця та пропонують індивідуальні програми навчання для співробітників. Такий підхід сприяє підвищенню ефективності та розвитку команди. Отже, впровадження штучного інтелекту в рекрутинг дозволяє автоматизувати й оптимізувати певні бізнес-процеси, зробити їх більш точними й ефективними.

3.6.4. Модельна задача щодо мінімізації витрат та ефективну класифікацію кандидатів з точки зору кадрової безпеки



Компанія, яка працює в галузі великих даних і науки про дані, хоче визначити науковців з даних серед тих, хто закінчив курси. Щоб досягти цього, планується розробити модель машинного навчання, яка передбачає, чи буде кандидат шукати нову роботу чи залишиться в компанії. Ця модель спрямована на мінімізацію витрат, економію часу, підвищення якості навчання та ефективну класифікацію кандидатів з точки зору безпеки. Розуміння факторів, що впливають на рішення співробітників, допоможе досягти цих цілей і приймати обґрунтовані рішення.

Аналітичний процес включає такі етапи.

Збір даних. Збір даних є життєво важливим етапом у процесі смарт-аналізу, оскільки він значно впливає на корисність і точність. Якість даних, отриманих під час збору, має першорядне значення. Збір даних має бути релевантним, не мати дублікатів і відсутньої інформації та точно відображати всі відповідні класифікації та підкатегорії. Забезпечення високоякісних даних створює міцну основу для успішних проєктів машинного навчання.

Попередня обробка даних. Після отримання даних дуже важливо їх попередньо обробити. Попередня обробка передбачає підготовку даних шляхом очищення, перевірки та перетворення їх у придатний для використання набір даних. Належна попередня обробка гарантує, що дані мають відповідний формат для ефективного аналізу та моделювання.

Побудова моделі. Вибір остаточної моделі для вирішення зазначеної бізнес-проблеми є критичним кроком, який потребує пильної уваги до кількох ключових елементів. Ці елементи включають тип вхідних даних, тип вихідних даних, а також додаткові фактори, такі як точність, складність, масштабованість та можливість інтерпретації. На цьому етапі модель навчається розуміти бажані результати, а потім її перевіряють за допомогою того самого набору методів. Результати тренувальних і тестових запусків порівнюються, оцінюючи продуктивність шляхом порівняння фактичних значень із прогнозованими. Цей процес допомагає вибрати найбільш прийнятну модель, яка ефективно вирішує поточну бізнес-проблему.

Оцінка моделі. Оцінка моделі має вирішальне значення для вибору найкращої моделі та точного прогнозування значень. Він оцінює відповідність між моделлю та даними, а також порівнює кілька моделей.

Розгортання моделі. Після виконання всіх необхідних кроків моделі машинного навчання розгортаються у виробничих середовищах, подібно до інших програм. Дані постійно надходять у модель із внутрішніх або зовнішніх джерел, що дозволяє проводити аналіз у реальному часі. Продуктивність моделі відстежується, звіти та візуалізації генеруються за допомогою різних інструментів. Це

допомагає зацікавленим сторонам зрозуміти ефективність фірми, визначити сфери для вдосконалення та прийняти обґрунтовані бізнес-рішення.

Джерело даних. Проаналізуємо набір даних <https://www.kaggle.com/datasets/arashnic/hr-analytics-job-change-of-data-scientists> для цієї проблеми.

Розвідувальний аналіз даних: Усі дані розділені на навчання та тестування. Ціль не включено до тесту, але файл даних цільових значень тесту доступний для відповідних завдань. Зразок подання відповідає `enrollee_id` тестового набору, який також містить стовпці: `enrollee_id`, `target`

Примітка:

- набір даних незбалансований;
- більшість функцій є категоріальними (номінальні, порядкові, двійкові), а деякі мають високу кількість елементів.

Особливості:

- `enrollee_id`: унікальний ідентифікатор кандидата;
- `city`: код міста;
- `city_development_index`: індекс розвитку міста (масштабований);
- `gender`: стать кандидата;
- `relevent_experience`: відповідний досвід кандидата;
- `enrolled_university`: тип університетського курсу, якщо такий є;
- `Education_level`: рівень освіти кандидата;
- `major_discipline`: основна дисципліна навчання кандидата;
- `experience`: загальний стаж кандидата в роках;
- `company_size`: кількість працівників у компанії поточного роботодавця;
- `company_type`: тип поточного роботодавця;
- `lastnewjob`: різниця в роках між попередньою та поточною роботою;
- `training_hours`: навчальні години завершені;
- `target`: 0 — не шукаю нову роботу, 1 — шукаю нову роботу.

Візуалізація даних. Графік на рис. 3.34 показує, що в наборі даних більше представників чоловічої статі, і близько 30% з них, швидше за все, шукатимуть зміни на роботі. Більшість жінок-кандидатів, швидше за все, не шукають зміни роботи. Крім того, близько 75% кандидатів, які мають відповідний досвід, не шукають зміни, тоді як близько 48% кандидатів без відповідного досвіду шукають зміни.

На графік з рис. 3.35 показано більшу кількість кандидатів, не зарахованих до університету, які, імовірно, змінять роботу. Крім того, більшість кандидатів у наборі даних мають середній освітній рівень і з меншою ймовірністю змінять роботу після навчання.

Ліва діаграма на рис. 3.36 показує, що більшість кандидатів у дисципліні STEM не шукають зміни роботи. З іншого боку, правильна коробкова діаграма показує, що досвідчені кандидати з меншою ймовірністю прагнуть змінити роботу.

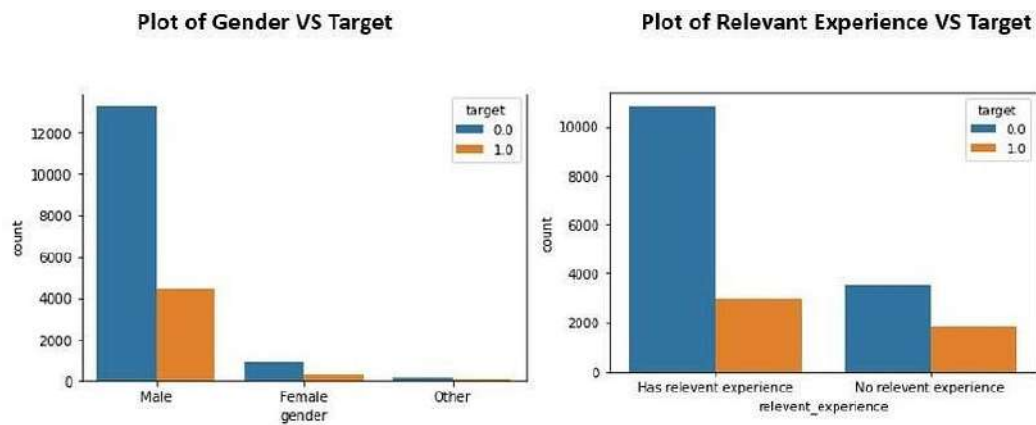


Рис. 3.34. Статистичний аналіз набору даних

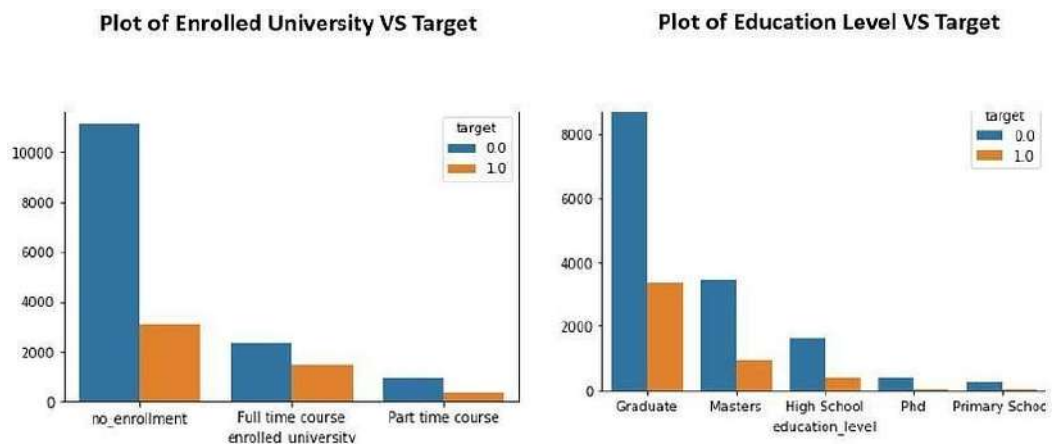


Рис. 3.35. Аналіз по рівню освіти

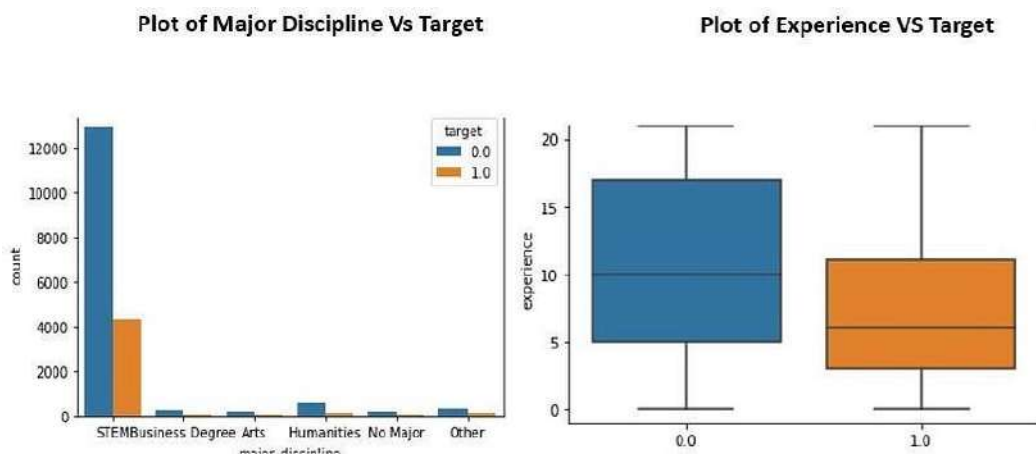


Рис. 3.36. Аналіз по напрямам освіти та по досвіду

На лівому графіку рис. 3.37 визначено список компаній, класифікованих за кількістю співробітників. Компанії з приблизно 50–99 співробітниками мають співвідношення 2:1, що вказує на те, що вдвічі більше кандидатів, які навряд чи змінять роботу, порівняно з тими, хто, ймовірно, змінить роботу. На правому графіку наведено список типів компаній, де «PVT LTD» має найбільшу кількість кандидатів, які навряд чи змінять роботу.

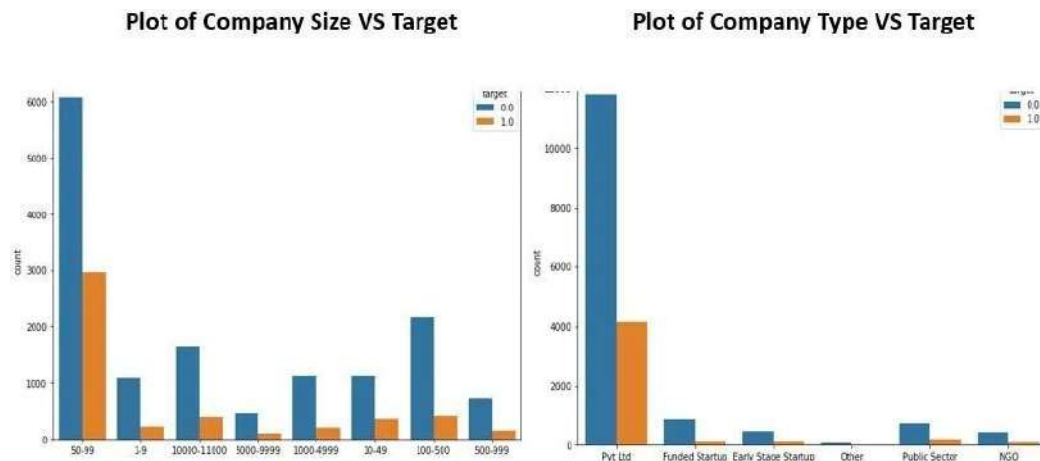


Рис. 3.37. Аналіз по списку компаній

Навчання моделі, оцінка та перевірка. На діаграмі Heatmap (рис. 3.38) вище виявлено п'ять атрибутів із найвищою кореляцією, а саме:

- 1) досвід;
- 2) вступив до університету;
- 3) відповідний досвід;
- 4) розмір компанії;
- 5) остання нова робота.

Цільова змінна в наборі даних показала дисбаланс: кількість 14 381 для цілі 0 (не шукає зміни роботи) і кількість 4777 для цілі 1 (шукає зміна роботи). Щоб усунути цей дисбаланс, ми застосували SMOTE (технологія передискретизації синтетичної меншості) для підвищення вибірки класу меншості. Оскільки набір даних містив категоричні змінні, застосуємо Label Encoder, щоб перетворити їх у числові змінні. Це дозволить ефективно працювати з категоріальними даними.

Використовуючи збалансований набір даних, розділимо його на навчальний і тестовий набори для створення моделі. Створено шість моделей: лінійна регресія, наївна Баєсівська модель, дерево рішень, опорна векторна машина, класифікатор випадкового лісу, класифікатор XGB. Серед цих моделей Random Forest Classifier показав найкращі результати, отримавши оцінку 0,67.



Рис. 3.38. Діаграма аналізу кореляцій

Висновок



За допомогою візуалізації даних визначено, що більшість дослідників даних у вибірці:

- чоловіки;
- мають відповідний досвід;
- зараз не навчаються в університеті;
- є випускником у галузі Stem;
- мають стаж роботи понад 20 років;
- мають роботу в приватній компанії;
- мають роботу в компанії середнього розміру;
- минулого року отримали нову роботу;
- набір даних незбалансований лише 24,9% дослідників даних у вибірці хочуть змінити роботу.

Цей аналіз ілюструє потенціал використання смарт-аналітики для прогнозування зміни роботи на основі таких факторів, як досвід кандидата, навчання в університеті, відповідний досвід і попередня історія роботи. Важливо зазначити, що точність результатів значною мірою залежить від якості набору даних, який використовується для навчання моделі.

Ключем до успіху в бізнесі є здатність залучати та утримувати найкращі таланти. Для відділу кадрів (HR) життєво важливо визначити фактори, які утримують співробітників, і ті, які спонукають їх піти. Організації могли б зробити більше, щоб запобігти втраті хороших людей.

3.6.5. Модельна задача щодо розуміння та визначення факторів пов'язаних з втратою персоналу



Вихідні дані. Маємо гіпотетичний набір даних, завантажений з IBM HR Analytics Employee Attrition & Performance <https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset>. Він містить 1470 точок даних (рядків) і 35 характеристик (стовпців), що описують досвід і характеристики кожного співробітника; і позначені (навчання під наглядом) із зазначенням того, чи вони все ще в компанії, чи вони пішли працювати в інше місце. Моделі машинного навчання можуть допомогти зрозуміти та визначити, як ці фактори пов'язані з відтоком робочої сили.

Розвідувальний аналіз даних (EDA). Імпортуємо відповідні бібліотеки Python і прочитаємо файл даних (рис. 3.39).

```
df = pd.read_csv('WA_Fn-UseC_-HR-Employee-Attrition.csv')
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical

Рис. 3.39. Структура набору даних

Набір даних добре організовано без пропущених значень. Цільовий клас — незбалансований з вибуттям 16%, як показано на рис. 3.40.

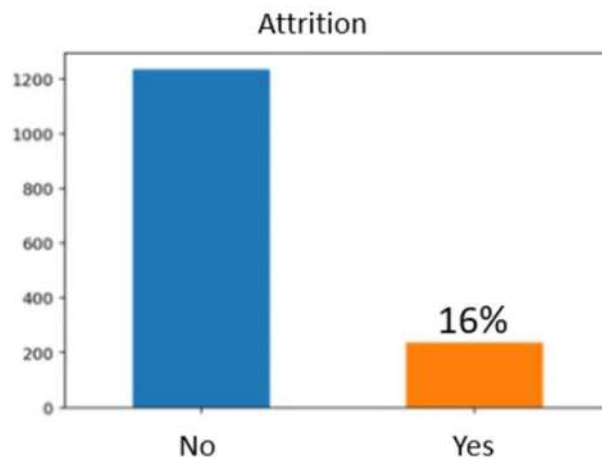


Рис. 3.40. Баланс цільового класу

Працівники отримують погодинну оплату від 30 до 100 дол. США, і, схоже, відставання відбувається на кожному рівні, незалежно від погодинної ставки працівника. Можливо, співробітники звільняються через низьку оплату праці. Це можна буде підтвердити пізніше на важливості функції.

Здається, що понаднормова робота є одним із ключових факторів виснаження, оскільки більша частка понаднормових працівників пішла (рис. 3.41).

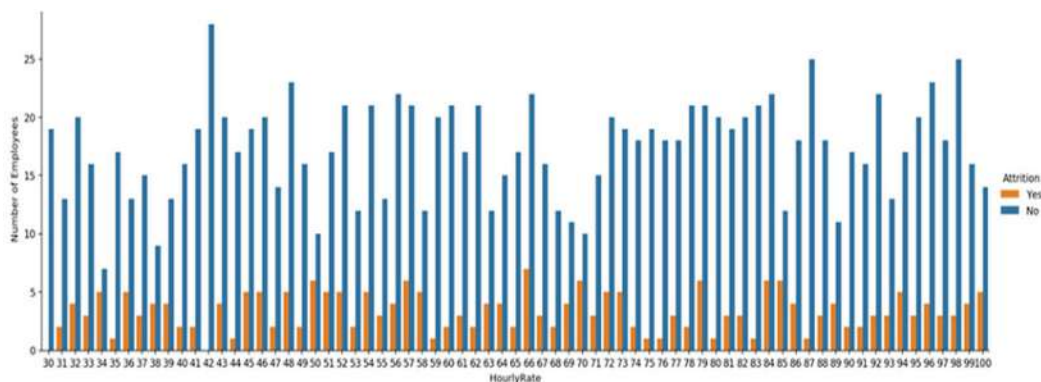


Рис. 3.41. Показники понаднормової роботи

Для цього аналізу включено лише три відділи (рис. 3.42).

```
for dpmt in df['Department'].unique():
    print('\n', dpmt, ':')
    print(df[df['Department']==dpmt]['JobRole'].value_counts())
```

```

Sales :
Sales Executive      326
Sales Representative  83
Manager             37

Research & Development :
Research Scientist   292
Laboratory Technician 259
Manufacturing Director 145
Healthcare Representative 131
Research Director    80
Manager             54

Human Resources :
Human Resources      52
Manager             11

```

Рис. 3.42. Кількісна структура

Попередня обробка даних. Дані потрібно попередньо обробити, оскільки моделі машинного навчання краще читають числа, ніж слова. Використовуючи кодування міток, категоричні дані можна замінити числами. Код нижче призначений для відображення всіх категорійних даних.

```

for col in df.select_dtypes(['object']).columns:
    print(col, ': ', sorted(df[col].unique()))

```

Продуктивність базової моделі. Необхідно розділити дані у співвідношенні 80:20. Обов'язково вказати у коді параметр «stratify=y», щоб пропорція класів у вихідному наборі даних була такою ж, як у наданому стовпці stratify.

Оскільки target має двійкові категоріальні класи з 84% «0» і 16% «1», «stratify=y» гарантує, що розподіл 80:20 містить 84% «0» і 16% «1» в обох вихідних наборах даних.

Оскільки набір даних є незбалансованим, треба використати «StratifiedKFold» у перехресній перевірці під час навчання моделей, і продуктивність кожної базової моделі можна зобразити в таблиці.

```

X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=.2, random_state=SEED, stratify=y)

```

```

kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=SEED)

```

Результати продуктивності базової моделі низькі: показники F1 коливаються від 20% до 40% для більшості моделей. Після налаштування гіпер параметрів і порогового значення логістична регресія досягає показника F1 50,5% і показника Recall 48,9%.

Вибір функції. Дані функції відхилено, оскільки вони не містять корисної інформації: «Over18», «EmployeeCount», «EmployeeNumber», «StandardHours».

Крім того, якщо функції тісно пов'язані одна з одною (присутня мультиколінеарність), одну з них слід видалити, щоб запобігти хибним результатам для лінійних моделей, таких як логістична регресія. Незважаючи на те, що моделі на основі дерева не зазнають спрямованого впливу, вони також можуть призвести до надмірного підгонки. Наведений нижче код призначено щоб знайти найбільш корельовані функції, і прибрати ту, яка має меншу важливість (рис. 3.43).

```
df_top10 = df.loc[:,top10_features.index].join(df['Attrition'])
sns.set(style="white")
mask = np.zeros_like(df_top10.corr(), dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
fig, ax = plt.subplots(figsize=(9,8))
cmap = sns.diverging_palette(255, 10, as_cmap=True)
sns.heatmap(df_top10.corr().round(2), mask=mask, annot=True,
cmap=cmap, vmin=-1, vmax=1, ax=ax)
```

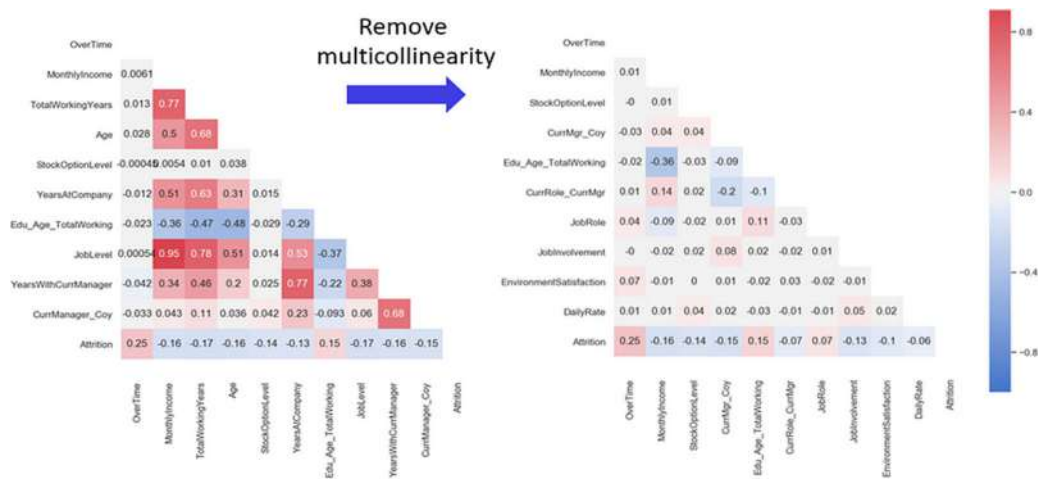


Рис. 3.43. Теплова карта кореляції до та після видалення мультиколінеарності

Розробка функцій. Створемо нову функцію «EduField_Dept», яка визначає, чи пов'язана «JobRole» з «EducationField», з мітками «0» = не пов'язана, «1» = пов'язана, «2» = певною мірою пов'язана. Однак HR, здається, чудово впорався з підбором вакансій, оскільки ця функція виявилася некорисною.

Інший набір нових функцій також створено на основі моїх власних спостережень та інтуїції:

```
Job_Coy = JobLevel / (YearsAtCompany + 1)
Edu_Age_TotalWorking = Education / (Age + TotalWorkingYears)
CurrMgr_Coy = YearsWithCurrManager / (YearsAtCompany + 1)
CurrRole_CurrMgr = YearsInCurrentRole / (YearsWithCurrManager + 1)
```

Додайте +1 до знаменника, щоб уникнути ділення на нуль.

З цими новими функціями знову запусимо коди продуктивності моделі, включаючи налаштування гіперпараметрів (пошукайте довгу групу коду). Цього разу Random Forest Classifier виявився з показником F1 49,6% і Recall 61,7%.

Цей набір даних із двох класів є незбалансованим (84% проти 16%). Як наслідок, існує ймовірність того, що побудована модель може бути упереджена до більшості та надмірно представленого класу. Після застосування методу синтетичної передискретизації меншості (SMOTE) для надмірної вибірки класу меншості можна спостерігати деяке покращення як у показниках F1, так і в Recall (рис. 3.44).

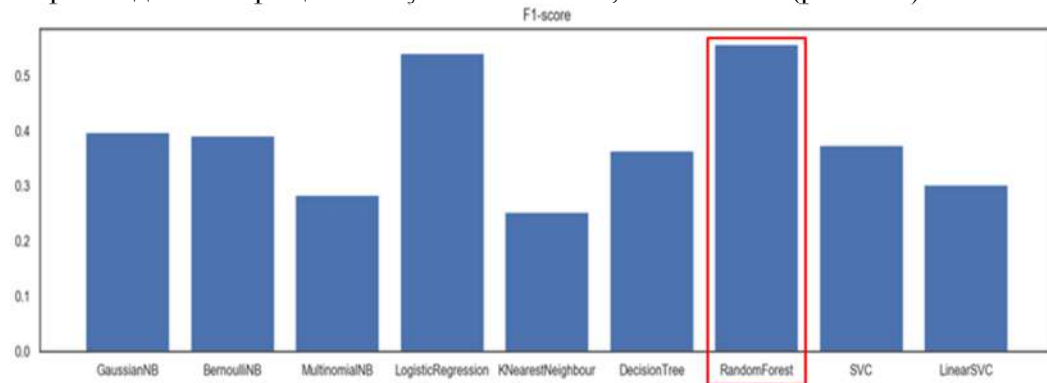


Рис. 3.44. Порівняння моделей

Random Forest Classifier став остаточною моделлю-переможцем з показником F1 55,7% і найвищим показником запам'ятовування 68,1%. Це може бути найвища можлива оцінка, досягнута з властивими обмеженнями в наборі даних (рис. 3.45–3.46).



Рис. 3.45. Оцінка моделі

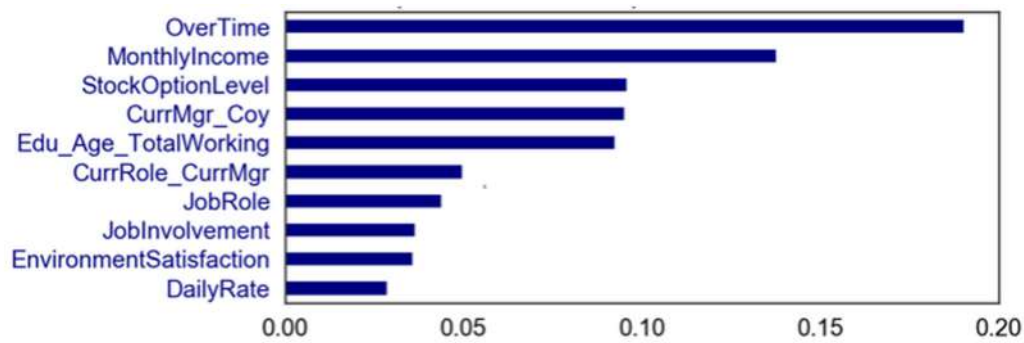


Рис. 3.46. Головна функція, отримана від Random Forest Classifier



Основним чинником звільнення працівників у цій гіпотетичній організації є грошовий фактор, оскільки «Понаднормовий дохід» і «Щомісячний дохід» опинилися на вершині. Це може бути наслідком поганого процесу компенсації понаднормової праці або причиною поганого балансу часу між роботою та особистим життям. Наступним важливим фактором є особисті стосунки з колегами, де нинішній керівник і його роль можуть бути основними причинами звільнення. Нарешті, залучення працівників до успіху компанії є критично важливим фактором задоволення, і організація повинна підтримувати постійну участь і мотивацію працівників.



Моделі машинного навчання настільки гарні, як і набори даних, що приймають, і більш якісні дані посилять модель. Якщо включити повний спектр оцінок ефективності, можна здобути більше інформації. У реальній життєвій ситуації дістати правильні дані часто складніше, ніж сам процес аналізу. HR аналітика набирає популярності в організаціях, які проходять цифрову трансформацію. Сфера її застосування розширилася від аналізу ефективності роботи співробітників до надання розуміння, як саме можна покращити організаційні процеси.

ЗАВДАННЯ ДЛЯ МОЗКОВОГО ШТУРМУ



1. Чи можна створити чітку картину викрадання або компрометації даних, використовуючи журнали серверів та програм? Якщо так, то яким чином?
2. Збільшення використання штучного інтелекту та додатків на основі LLM у медичному секторі може мати серйозні правові наслідки. Обговоріть дану проблему.
3. Деякі дослідники кажуть, що роботи і чат-боти можуть збільшити страхові ризики та підвищити комісії. Дослідите це питання і напишіть звіт.
4. Якщо ви використовуєте онлайн соціальну мережу, таку як Facebook, Instagram або LinkedIn, щоб переглянути свої зв'язки в цій мережі: скільки сильних зв'язків і скільки слабких зв'язків?
5. Деякі вчені кажуть, що аналітика та методи машинного навчання взагалі дегуманізують управлінську діяльність, а інші кажуть, що ні. Обговоріть аргументи обох точок зору. Обговоріть потенційний вплив штучного інтелекту на конфіденційність, безпеку та дискримінацію.

LAB 3:

АНАЛІЗ АТАК НА СИСТЕМУ КІБЕРБЕЗПЕКИ



Мета. Дослідити набір даних атак на систему кібербезпеки, який складається з 25 різноманітних метрик і 40 000 записів. Лабораторна робота має на меті розшифрувати суть цих кібернападів, виявити повторювані риси та розпізнати закономірності, які виявлені протягом спостережуваного періоду.

Формат даних. Набір даних доступний у структурованому форматі `cybersecurity_attacks.csv`, тип файлу CSV. Доступний за посиланням: https://github.com/incribo-inc/cybersecurity_attacks

Опис набору даних по стовпцях:

- Timestamp
- Source IP Address
- Destination IP Address
- Source Port
- Destination Port
- Protocol
- Packet Length
- Packet Type
- Traffic Type
- Payload Data
- Malware Indicators
- Anomaly Scores
- Alerts/Warnings
- Attack Type
- Attack Signature
- Action Taken
- Severity Level
- User Information
- Device Information
- Network Segment
- Geo-location Data
- Proxy Information
- Firewall Logs
- IDS/IPS Alerts
- Log Source

Завдання. Виконайте дослідницький аналіз даних (EDA) з даним набором, та побудуйте відповідну візуалізацію.

Ключові дані для аналізу

- Позначка часу (39 997 записів): діючи як часовий хребет, це дозволяє зрозуміти тенденції в усьому часовому спектрі.
- IP-адреси: за допомогою записів із детальним описом джерела (40 000) і призначення (40 000) дає змогу нанести на карту географічні епіцентри кіберактивності.
- Порти: розділення портів джерела (29 761) і призначення (29 895) висвітлює які служби під прицілом.
- Протокол (3 типи): палітра протоколів, що охоплює TCP, UDP або ICMP, пропонує підказки щодо способів атаки.
- Деталі пакета: тонкощі довжини пакета (1437 типів) і жанру пакета (2 типи) проливають світло на особливості передачі даних.
- Дані про трафік і корисне навантаження: диференціація типів трафіку (3 типи) у поєднанні з інформацією про корисне навантаження (40 000 записів) розкриває ймовірні кібернаміри.
- Індикатори атак: такі показники, як індекси зловмисного програмного забезпечення, оцінки аномалій (9826 різновидів), сповіщення, модальності атак (3 типи) та унікальні сигнатури, відображають сферу кіберзагроз.
- Реакція та вплив: такі індикатори, як вжиті контрзаходи, серйозність загрози та виявлення/сповіщення про вторгнення, окреслюють план контрреагування та потенційний вплив.
- Дані користувачів та інфраструктури: мозаїка даних, що включає профілі користувачів (32 389 записів), показники пристроїв (32 104 записи), розділи мережі, географічні маркери (8 723 типи), журнали проксі (20 148 записів), описи брандмауера та джерела журналів, створює цілісний погляд на кіберекосистему.

Порядок виконання роботи

1. *Завантаження даних.* Почніть із завантаження набору даних у Python за допомогою таких бібліотек, як Pandas або NumPy.
2. *Очищення даних.* Визначте відсутні значення у своєму наборі даних і вирішіть, як з ними впоратися.
3. *Детальне дослідження трьох типів векторів атаки.* Виявити переважні методи, які використовують зловмисники. Розуміння цього дозволить керувати майбутніми оборонними стратегіями.

4. *Виявлення аномалій.* Виявіть та обробіть аномалії за допомогою статистичних методів або методів візуалізації. Маючи близько 10 000 унікальних оцінок аномалій, проаналізуйте розподіл та інтенсивність цих аномалій. Мета полягає в тому, щоб визначити шаблони чи повторювані аномалії, які можуть вказувати на постійні вразливості або цілеспрямовані атаки.
5. *Кореляційний аналіз.* Дослідите зв'язки між змінними, обчисліть коефіцієнти кореляції та побудуйте теплові карти кореляції. Визначте високорельовані метрики і визначте їхній вплив на дані, зокрема, кількість атак за мережевими протоколами, сигнатури, пов'язані з виявленими атаками, мають бути проаналізовані. Розпізнавання їх та класифікація може надати цінну інформацію про суб'єктів загрози або інструменти атак, які зазвичай використовуються.
6. *Візуалізація даних.* Використайте Matplotlib або Seaborn для створення значущих візуалізацій. Оберіть такі методи, як гістограми, діаграми розсіювання, прямокутні діаграми та кореляційні матриці, щоб отримати уявлення про розподіл, зв'язки та аномалії у даному наборі даних.
7. *Висновки.* Зробіть висновки, класифікуйте загрози з точки зору потенційного впливу та надайте гіпотези щодо протидії ним. Важливий аспект аналізу — це розуміння методів негайної відповіді на загрози та оцінювання ефективності поточних контрзаходів і, можливо, підготовка майбутніх протоколів реагування.

Розділ 4. КЕЙСИ

Усі моделі неправильні, але деякі корисні.
Джордж Е. П. Бокс

4.1. Data-driven підхід до прийняття рішень у бізнес-розвідці



Data-driven підхід — це підхід, коли рішення ухвалюються, спираючись на аналіз цифр / метрик, а не на інтуїцію та особистий досвід.



Проблема. Ольга — власниця невеликого кафе і продавчиня смачних тортів. Її кіоск працює вже шість років без розвитку. У чому справа? Застосувати data-driven підхід для вирішення проблеми.

4.1.1. Основні метрики



Метрики оточують вас усюди. Це кількісно вимірні показники, які компанія відстежує й аналізує для оцінки успіху свого продукту. Зазвичай кожна метрика має назву, значення і розмірність. *Наприклад:* метрика — швидкість, значення — 50, розмірність — км/год; метрика — довжина, значення — 20, розмірність — см. У нашому випадку метрика — кількість проданих тортів, значення — 100, розмірність — штука.

Щоб дізнатись, що ти точно щось покращив, потрібно з'ясувати, як було до цього, і порівняти нові результати зі старими.

Бізнес-модель у Ольги: виробництво і продаж без посередників. Найпростіші метрики для Ольги: кількість проданих тортів; дохід; собівартість тортів; прибуток.

Ольга за день продає 100 маленьких і 50 великих тортів, маленький коштує 100 грн, великий — 160 грн, собівартість маленького — 60 грн, великого — 80 грн. Отже, чистий прибуток від маленького торта становитиме 40 грн, від великого — 80 грн.

4.1.2. Робота з гіпотезами в аналітиці

Зараз Ольга заробляє:

— від маленьких тортів: $100 \text{ шт.} \times 40 \text{ грн (прибуток)} = 4000 \text{ грн};$

— від великих тортів: $50 \text{ шт.} \times 80 \text{ грн (прибуток)} = 4000 \text{ грн}.$

Отже, Ольга заробляє у загальному підсумку 8000 грн, тобто продаючи вдвічі менше великих тортів, прибуток від них такий самий.

Ольга хоче зробити акцію у своєму кіоску і дарувати безкоштовну каву усім, хто придбав великий торт.

Це — ідея!

Така акція сприятиме підвищенню продажу великих тортів настільки, щоб як мінімум перекрити витрати на безкоштовну каву.

Це — гіпотеза.

Чи це призведе до збільшення прибутку? — покаже *експеримент*.

Не завжди спрацювають всі гіпотези. Для цього і потрібно їх перевіряти експериментами.

Розрахунок. Нічого не змінюється:

1. Якщо зараз Ольга буде дарувати усім, хто купив великий торт (і ми подарували каву), а з продажами нічого не зміниться, то вона буде витрачати 10 грн на каву на кожному проданому великому торті.

Загалом 50 шт. (великих тортів продано за день) $\times$ 10 грн (собівартість кави).

Виходить, що Ольга буде витрачати 500 грн щодня під час цього експерименту.

Звідси загальний прибуток знизиться на 500 грн, тобто становитиме 7500 грн прибутку на день.

2. Якщо припустити, що 20 людей замість маленького торта будуть вибирати великий, щоб дістати каву у подарунок:

— продамо 80 маленьких тортів замість 100:

80 шт. $\times$ 40 грн (прибуток) = 3200 грн;

— продамо на 20 шт. більше великих тортів, тобто:

70 великих 70 шт. $\times$ 80 грн (прибуток) = 5600 грн.

отже, Ольга буде заробляти не 8000, а 8800 грн за один день.

Висновок:

- На експеримент ми будемо витрачати 500 грн на день. для того щоб зрозуміти, чи працює це, чи ні, нам потрібно, наприклад, 10 днів, звідси ціна всього експерименту — 5000 грн.

- У разі якщо гіпотеза підтвердиться, тобто якщо 20 людей на день буде вибирати великий торт замість маленького, ми будемо заробляти на 800 грн більше щодня.

І звідси Ольга тепер буде заробляти на 24 тис. грн більше на місяць.

Експериментувати — завжди ризик, проте ви отримуєте досвід і знання. Зазвичай потрібно кілька експериментів, щоб знайти той, що спрацює.

Головне — не боятися експериментувати.

Які ще метрики може аналізувати і рахувати Ольга?

- кількість покупців (**users**);
- кількість проданих одиниць товару;
- скільки коштує залучення кожного покупця;
- який відсоток покупців повертаються (**retention rate**);
- який відсоток покупців підходять і не купують (**bounce rate**);
- скільки покупців прийшли по рекомендації тих, хто вже придбав торт раніше.

Як Ользі з багатьох метрик вибрати потрібні?

1. Почати цікавитись темою продуктової та бізнес-аналітики.
2. Визначати метрики, рахувати, генерувати гіпотези, експериментувати і накопичувати знання про своїх клієнтів, конкурентів, власний бізнес. З кожним новим експериментом Ольга буде дізнаватись дедалі більше, іноді втрачаючи гроші на невдалі експерименти, але в довготривалій перспективі — це вигідно. Знання про те, як працює її бізнес, будуть ставати більш ґрунтовними.

4.1.3. Основні етапи та інструменти

Збір даних (збираємо і зберігаємо дані, які ми можемо використовувати в майбутньому для покращень), аналіз (сортуємо дані, об'єднуємо у потрібні нам метрики і аналізуємо для прийняття рішень), рішення (приймаємо рішення з певним ступенем ризику для досягнення головної мети).

Почати збирати дані про ваш продукт можна з опитувань. Це найпростіший спосіб, який не потребує цифрових інструментів.

Зберігати результати можна почати у Google-таблицях (рис. 4.1).

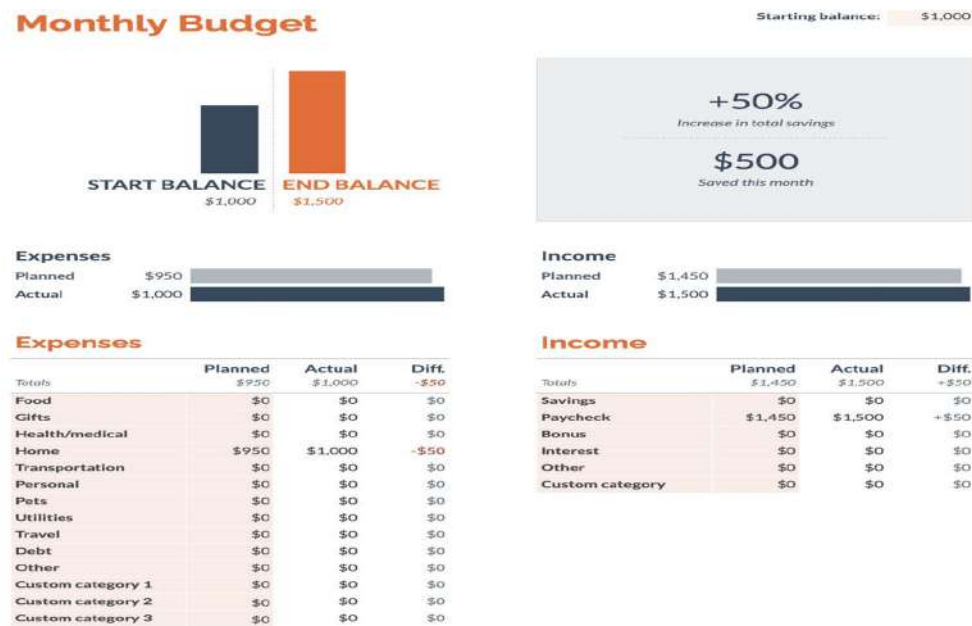


Рис. 4.1. Застосування Google-таблиць

Або створити онлайн-дошку (рис. 4.2), де будуть розташовані основні метрики продукту і змінюватись автоматично відповідно до даних, які будуть у таблицях.

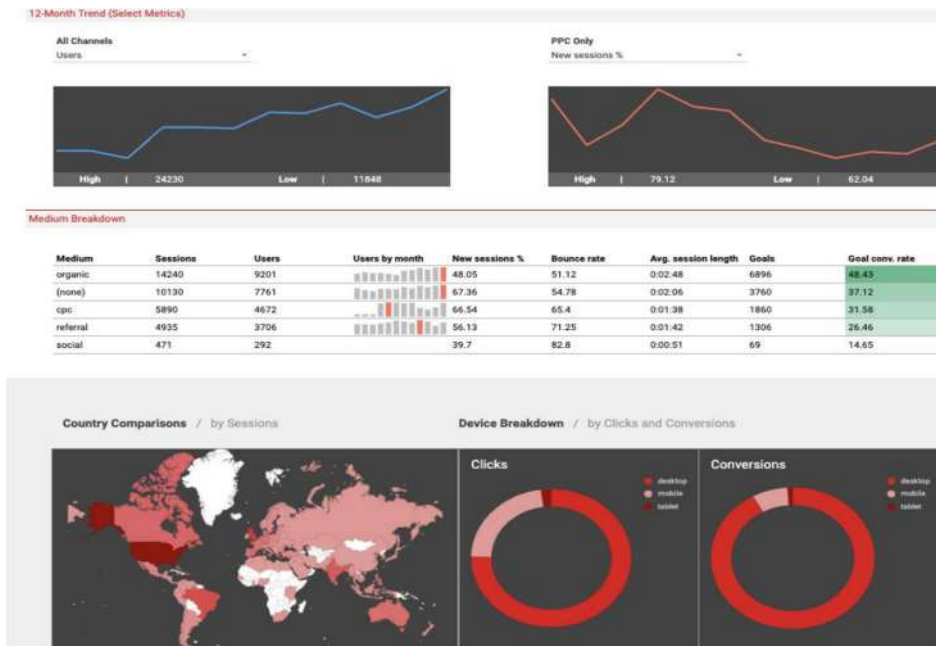


Рис. 4.2. Online-dashboard

Програмний продукт Looker studio є трохи складнішим за Google-таблиці, проте дає змогу завантажувати дані з багатьох джерел (рис. 4.3).

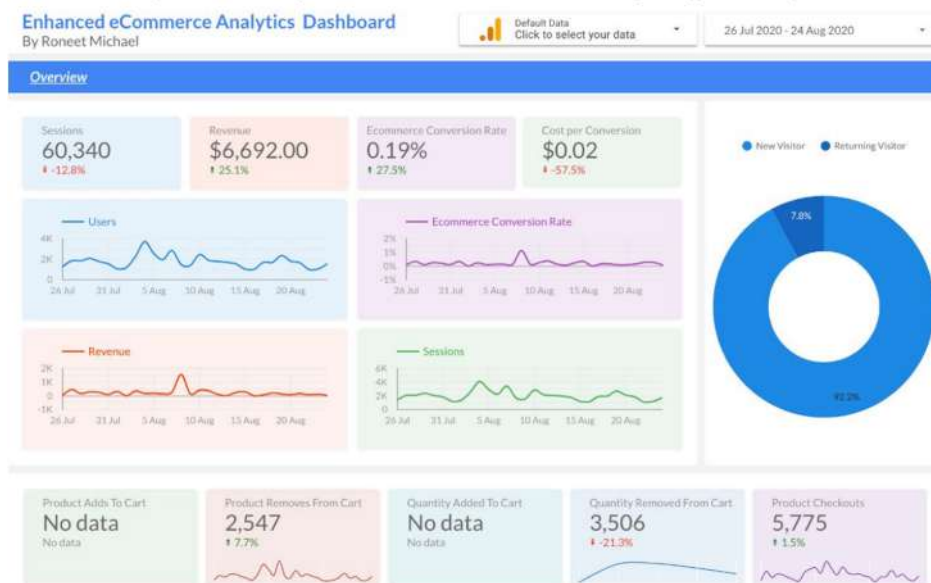


Рис. 4.3. Приклад звіту Looker studio

Google analytics — найпоширеніший інструмент для збору і аналізу даних у мережі.

Нагадаємо, що генеративний AI — це тип штучного інтелекту, який здатний генерувати зображення, тексти, та інші типи контенту у відповідь на спеціальні запити користувача — prompts*, які показано на рис. 4.4.

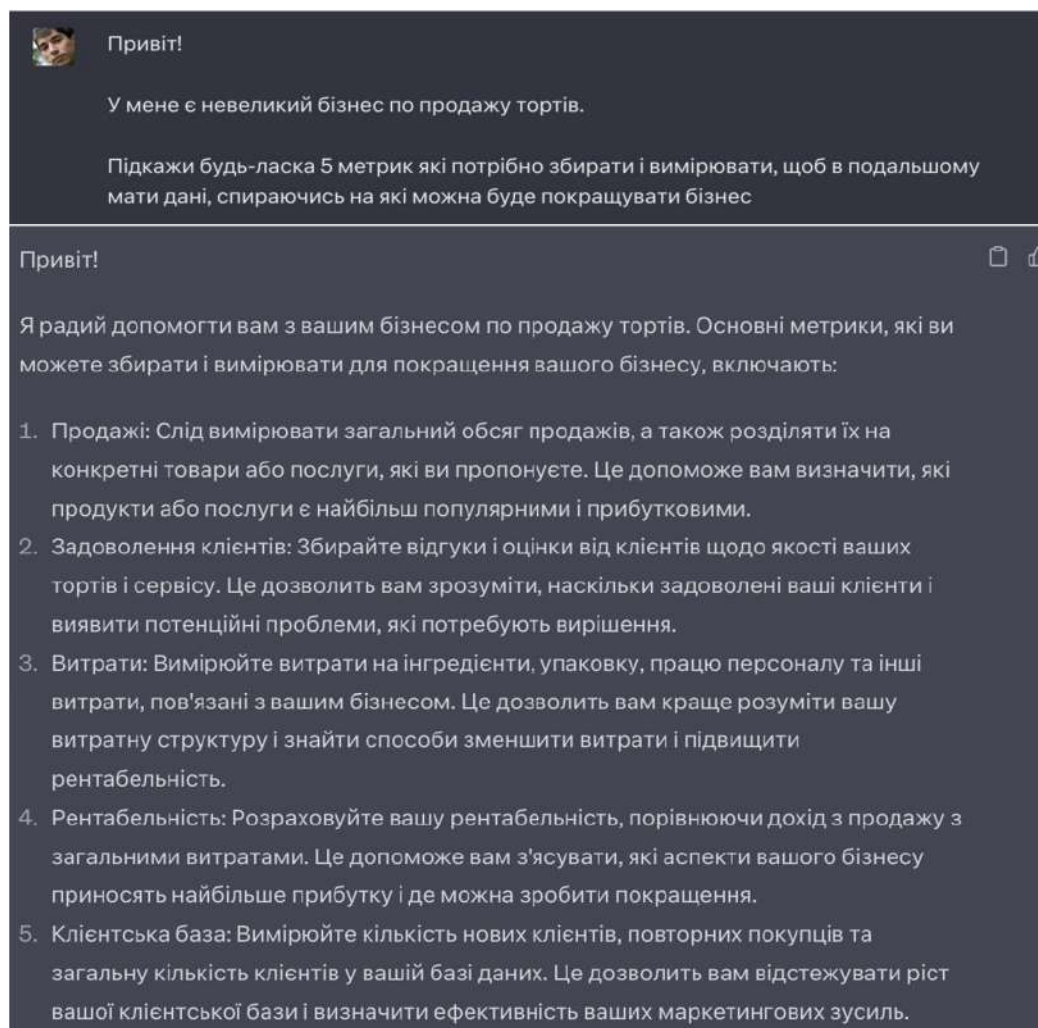


Рис. 4.4. Спеціальний запит користувача, *prompt*

Популярність генеративного AI:

- у листопаді 2022 р. був випущений ChatGPT від OpenAI — чат-бот на базі GPT-тримовної моделі;
- у листопаді 2022 р. була випущена 4 версія моделі Midjourney для генерації зображень;
- листопад став стартом для популярності генеративного AI, оскільки контент, згенерований цими моделями, був дуже якісним.

4.2. Аналіз та сегментація осіб за лояльністю у кіберрозвідці



RFM-модельовання — це техніка маркетингового аналізу, яка використовується для оцінки цінності клієнта.

Пам'ятаємо, що ваші клієнти дуже важливі для вашого бізнесу. Зрештою, без них бізнесу не було б. Проте багато малих компаній не вміють оцінити своїх клієнтів, щоб визначити, які з них є найважливішими для їхнього успіху. Ви можете виміряти свої витрати на залучення та сегментувати клієнтів на основі їхньої поведінки на вашому сайті. Однак ці дані не говорять вам, як купівельна поведінка допомагає вашому бізнесу досягти успіху.

Існує кілька способів оцінити своїх клієнтів, втім один із найефективніших і найпростіших у застосуванні — аналіз RFM.

4.2.1. Модель RFM (Recency, Frequency, Monetary)

Модель RFM (Recency, Frequency, Monetary) базується на трьох факторах:

- 1) *давність*: коли нещодавно клієнт зробив покупку;
- 2) *частота*: як часто клієнт робить покупку;
- 3) *грошова вартість*: скільки грошей клієнт витрачає на покупки.

Модель RFM пропонує числові значення для трьох наведених вище параметрів. Ці цінності допомагають компаніям краще зрозуміти потенціал клієнтів.

Наприклад. Якщо клієнт щодня робив покупки у MacDonalds протягом минулого року і нічого не купував протягом останнього місяця, він міг перейти до бренду-конкурента. Можливо, зараз клієнт перейшов до «Львівських круасанів» або «Сільпо» через кращі пропозиції чи зручність.

Отже, після аналізу MacDonalds може націлитися на цих клієнтів і розробити маркетингову стратегію, щоб повернути їх.



Проблема. Створити модель RFM за допомогою Python для роздрібного онлайн-магазину з використанням набору даних, які містять понад 4 тис. унікальних ідентифікаторів клієнтів, і призначитимемо RFM кожному з цих клієнтів. Це транснаціональний набір даних, який містить усі транзакції, здійснені з 01/12/2020 по 09/12/2021 для роздрібної торгівлі в онлайн-магазині, що базується у Великобританії.

4.2.2. Етапи створення моделі

1. Аналіз набору даних. Прочитаємо завантажений набір даних за допомогою таких рядків коду:

```
import pandas as pd
df2 = pd.read_csv('data.csv', encoding='unicode_escape')
```


Тепер давайте подивимося на заголовок нашого набору даних даних (рис. 4.5):

```
df.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom

Рис. 4.5. Заголовок *DataFrame*

Для цього аналізу ми будемо використовувати лише чотири стовпці: Quantity, InvoiceDate, UnitPrice, CustomerID.

2. Розрахунок вартості. Почнемо з розрахунку значення М — грошової вартості. Це найпростіше значення для розрахунку. Усе, що нам потрібно зробити, це підрахувати загальну суму, витрачену кожним клієнтом.

Для цього нам потрібно використовувати стовпці UnitPrice і Quantity. Спочатку ми помножимо ці значення, щоб отримати загальну суму, витрачену кожним клієнтом на кожну транзакцію.

Код для цього такий:

```
df['Total'] = df['Quantity']*df['UnitPrice']
```

Перевіримо заголовок набору даних (рис. 4.6):

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Total
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom	15.30
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom	22.00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom	20.34

Рис. 4.6. Доповнення *DataFrame*

```
m = df.groupby('CustomerID')['Total'].sum()
m = pd.DataFrame(m).reset_index()
```

Дивлячись на заголовок нового набору даних, бачимо наведене на рис. 4.7:

	CustomerID	MonetaryValue
0	12346.0	0.00
1	12347.0	4310.00
2	12348.0	1797.24
3	12349.0	1757.55
4	12350.0	334.40

Рис. 4.7. Новий *DataFrame*

Отже, ми успішно розрахували грошову вартість кожного клієнта у наборі даних.

3. Розрахунок частоти. Тепер розрахуємо частоту. Ми хочемо дізнатися, скільки разів кожен клієнт зробив покупку. Давайте ще раз подивимося на фрейм даних, щоб побачити, як ми можемо це зробити. Щоб дізнатися, скільки разів кожен клієнт здійснював покупку, нам потрібно використовувати стовпці CustomerID і InvoiceDate. Нам потрібно підрахувати кількість унікальних дат, коли кожен клієнт бачив здійснення покупки. Для цього запустіть такі рядки коду:

```
freq = df.groupby('CustomerID')['InvoiceDate'].count()
f = pd.DataFrame(freq).reset_index()
```

Поглянувши на заголовок кадру даних, побачимо таке (рис. 4.8):

	CustomerID	InvoiceDate
0	12346.0	2
1	12347.0	182
2	12348.0	31
3	12349.0	73
4	12350.0	17

Рис. 4.8. DataFrame з показником частоти

Отже, успішно розробили кількісну міру частоти для кожного клієнта в наборі даних.

4. Обчислення давності. Нарешті ми можемо обчислити давність кожного клієнта у наборі даних. Щоб обчислити давність, нам потрібно дізнатися, коли людину востаннє бачили за покупкою.

Щоб знайти це значення, потрібно використовувати стовпці CustomerID і InvoiceDate. Спочатку треба знайти останню дату, коли кожен клієнт бачив здійснення покупки. Потім призначити цій даті певне кількісне значення. Для цього нам спочатку треба перетворити стовпець InvoiceDate на об'єкт datetime і виконати такі рядки коду:

```
df['Date'] = pd.to_datetime(df['InvoiceDate'])
```

Знову подивимося на заголовок кадру даних (рис. 4.9):

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Date
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom	2010-12-01 08:26:00
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom	2010-12-01 08:26:00
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom	2010-12-01 08:26:00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom	2010-12-01 08:26:00
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom	2010-12-01 08:26:00

Рис. 4.9. Доповнення DataFrame

Зверніть увагу, що тепер у нас є новий стовпець «Дата». Тепер слід відшукати останню дату, коли кожен клієнт бачив здійснення покупки. Для цього нам потрібно призначити ранг усім датам для кожного CustomerID. Остання дата буде оцінена як 1, друга за останньою дата як 2 і т. д.

Виконайте такі рядки коду:

```
df['rank'] =
df.sort_values(['CustomerID', 'Date']).groupby(['CustomerID'])['Date'].rank(method='min').astype(int)
```

Тепер маємо різні ранги залежно від часу, коли клієнта бачили під час покупки. Остання покупка має ранг 1. Відфільтруємо фрейм даних і позбудемося всіх інших покупок. Треба зберегти лише найновіші:

```
recent = df[df['rank']==1]
```

Тепер усе, що варто зробити, це придумати кількісне значення актуальності. Це означає, що особа, яку бачили день тому, набуває вищу актуальність порівняно з особою, яку бачили тиждень тому. Для цього давайте лише обчислимо різницю між кожною датою у наборі даних і найбільш ранішньою датою. Отже, новіші дати матимуть вищу цінність. Виконайте такі рядки коду:

```
recent['recency'] = recent['Date'] - pd.to_datetime('2010-12-01 08:26:00')
```

Тепер подивимося на заголовок набору даних (рис. 4.10):

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Date	rank	recency
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom	2010-12-01 08:26:00	1	0 days
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom	2010-12-01 08:26:00	1	0 days
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom	2010-12-01 08:26:00	1	0 days
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom	2010-12-01 08:26:00	1	0 days
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom	2010-12-01 08:26:00	1	0 days

Рис. 4.10. Дововнення DataFrame

Зверніть увагу, що існує новий стовпець із позначкою «Останній час», і кількість днів від найдавнішої дати в наборі даних обчислено. Значення 0 днів вказує на найнижчу давність. Тепер можемо перетворити значення актуальності в числові. Для цього запустіть такі рядки коду:

```
def recency(recency):
    res = str(recency).split(' ')[0]
    return(int(res))

recent['recency'] = recent['recency'].apply(recency)
```

Зауважимо, що наведений вище фрейм даних містить багато повторюваних значень для кожного CustomerID. Це пов'язано з тим, що розбивка здійснюється за продуктами, і ті самі клієнти купували кілька продуктів одночасно. Давайте виберемо лише стовпці CustomerID і Recency і видалимо дублікати:

```
recent = recent[['CustomerID', 'recency']]
recent = recent.drop_duplicates()
```

Поглянувши на заголовок кадру даних, ви повинні побачити зображене на рис. 4.11:

	CustomerID	recency
0	17850.0	0
9	13047.0	0
26	12583.0	0
46	13748.0	0
65	15100.0	0

Рис. 4.11. Новий DataFrame з показником давності

5. Інтеграція даних. Результати зберігаються в окремих фреймах даних, тому давайте об'єднаємо їх разом:

```
finaldf = f.merge(m, on='CustomerID').merge(recent, on='CustomerID')
```

Тепер подивимося на заголовок кінцевого кадру даних (рис. 4.12):

	CustomerID	InvoiceDate	MonetaryValue	recency
0	12346.0	2	0.00	48
1	12347.0	182	4310.00	6
2	12348.0	31	1797.24	15
3	12349.0	73	1757.55	355
4	12350.0	17	334.40	63

Рис. 4.12. Об'єднаний DataFrame для моделі RFM

Вдалося додати значення RFM до кожного ідентифікатора клієнта в наборі даних. На основі цих розрахунків можна отримати інформацію про клієнтів або побудувати *модель кластеризації*, щоб згрупувати подібних клієнтів.

Основні інструменти. Перед початком треба переконатися, що на вашому пристрої встановлено Python IDE разом із бібліотекою Pandas.

Якщо вдалося успішно виконати наведені вище коди, спробуйте піти далі та нормалізувати ці значення. Значення RFM представлені за шкалою від 1 до 5, теж можна створити контейнери для кожного з цих значень і згрупувати їх.

4.3. Ідентифікація осіб методом «K-means»



Сегментація клієнтів важлива для бізнесу, щоб зрозуміти його цільову аудиторію. Різні рекламні оголошення, політичні реклами тощо, можна підбирати та надсилати різним сегментам аудиторії на основі їх демографічного профілю, інтересів і рівня достатку. Існує багато неконтрольованих алгоритмів машинного навчання, які можуть допомогти компаніям визначити свою базу користувачів і створити споживчі сегменти. K-Means — популярна техніка неконтрольованого навчання для кластеризації. Мета K-Means полягає в тому, щоб згрупувати всі доступні дані в підгрупи, що не перекриваються і відрізняються одна від одної. Цей алгоритм може приймати немарковані дані клієнта та призначати кожному точку даних кластерам.

Це означає, що кожна підгрупа / кластер складається з особливостей, які відрізнятимуть їх від інших кластерів.

Кластеризація K-Means — це техніка, яка широко використовується аналітиками, щоб допомогти компаніям із сегментацією клієнтів. Це важлива навичка, яку потрібно мати, і більшість компаній перевіряють ваше розуміння цього алгоритму / вашу здатність застосовувати його у реальних життєвих сценаріях.



Проблема. Ви володієте торговим центром або супермаркетом, і через членські картки ви маєте деякі основні дані про своїх клієнтів, якот: ідентифікатор клієнта, вік, стать, річний дохід і показник витрат. **Показник витрат** — це показник, що ви призначаєте клієнту на основі визначених вами параметрів, таких як поведінка клієнта та дані про покупки. Ви бажаєте виявити групи клієнтів, яких можна легко об'єднати [Цільові клієнти], щоб компанія могла належно спланувати стратегію роботи з ними.

Набір даних для кейсу ми можемо одержати тут: <https://www.kaggle.com/datasets/vjchoudhary7/customer-segmentation-tutorial-in-python> Установіть бібліотеки: pandas, numpy, matplotlib, seaborn, scikit-learn, kneed.

Завдання.

- Попередня обробка даних для кластеризації K-Means.
- Побудова алгоритму кластеризації K-Means з нуля.
- Показники, які використовуються для оцінки ефективності моделі кластеризації.
- Візуалізація побудованих кластерів.
- Інтерпретація та аналіз побудованих кластерів.

4.3.1. Імпорт і читання набору даних

Виконайте такі рядки коду, щоб імпортувати необхідні бібліотеки Python і прочитати набір даних:

```
# Імпорт
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sea
from kneed import KneeLocator
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from mpl_toolkits.mplot3d import Axes3D

# зчитуємо data frame
df = pd.read_csv('Mall_Customers.csv').
```

Отримуємо наступний заголовок нашого набору даних даних (рис. 4.13):

```
df.head()
```

	CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

Рис. 4.13. Набір даних

У наборі даних є п'ять змінних. CustomerID — це унікальний ідентифікатор кожного клієнта в наборі даних, і ми можемо видалити цю змінну, тому що вона не надає нам жодної корисної інформації про кластери. Оскільки стать є категоріальною змінною, її потрібно закодувати і перетворити на числову.

Усі інші змінні будуть масштабовані відповідно до нормального розподілу перед подачею до моделі. Ми стандартизуємо ці змінні з середнім значенням 0 і стандартним відхиленням 1.

4.3.2. Стандартизація змінних

Необхідно стандартизувати всі змінні в наборі даних, щоб отримати їх приблизно в тому самому масштабі.

У світі даних стандарт належить до формату або представлення, якому має відповідати кожне значення певної області. Отже, стандартизація даних означає процес перетворення неправильного або неприйняттого представлення даних у прийнятну форму.

Наприклад. Номер телефону клієнта зберігається як текстовий рядок в одній системі, тоді як в іншій системі допускається лише номер із 8 цифр, що призводить до невідповідності типу даних.

```
col_names = ['Annual Income (k$)', 'Age', 'Spending Score (1-100)']
features = df[col_names]
scaler = StandardScaler().fit(features.values)
features = scaler.transform(features.values)
scaled_features = pd.DataFrame(features, columns = col_names)
scaled_features.head()
```

Отримуємо заголовок набору даних (рис. 4.14):

	Annual Income (k\$)	Age	Spending Score (1-100)
0	-1.738999	-1.424569	-0.434801
1	-1.738999	-1.281035	1.195704
2	-1.700830	-1.352802	-1.715913
3	-1.700830	-1.137502	1.040418
4	-1.662660	-0.563369	-0.395980

Рис. 4.14. *Доповнений DataFrame*

Бачимо, що усі змінні були перетворені і тепер зосереджені навколо нуля. Змінна «Стать» є категоричною, і нам потрібно перетворити її на числову змінну. Це означає, що нам треба замінити числа для кожної категорії. Ми можемо зробити це з Pandas за допомогою `pd.get_dummies()`.

```
gender = df['Gender']
newdf = scaled_features.join(gender)
newdf = pd.get_dummies(newdf, prefix=None, prefix_sep='_',
dummy_na=False, columns=None, sparse=False, drop_first=False,
dtype=None)
newdf = newdf.drop(['Gender_Male'],axis=1)
newdf.head()
```

	Annual Income (k\$)	Age	Spending Score (1-100)	Gender_Female
0	-1.738999	-1.424569	-0.434801	0
1	-1.738999	-1.281035	1.195704	0
2	-1.700830	-1.352802	-1.715913	1
3	-1.700830	-1.137502	1.040418	1
4	-1.662660	-0.563369	-0.395980	1

Рис. 4.15. *Стандартизований DataFrame*

На рис. 4.15 бачимо, що гендерна змінна була трансформована. Також ми вилучили Gender_Male із кадру даних. Це тому, що більше немає потреби зберігати цю змінну. Значення для Gender_Male можна вивести з Gender_Female (тобто якщо Gender_Female дорівнює 0, тоді Gender_Male матиме значення 1, і навпаки).

4.3.3. Побудова моделі кластеризації

Найкраще у sklearn полягає в тому, що він надає стандартний і елегантний API для кожної реалізації моделі ML:

- Навчіть модель за допомогою model.fit().
- Спрогнозуйте результат за допомогою model.predict().
- Обчисліть точність за допомогою model.score().

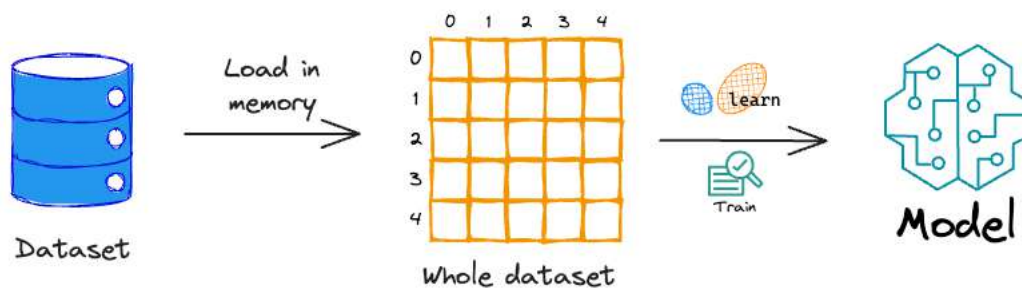


Рис. 4.16. Побудова моделі

```
SSE = []
for cluster in range(1,10):
    kmeans = KMeans(n_jobs = -1, n_clusters = cluster, init='k-means++')
    kmeans.fit(newdf)
    SSE.append(kmeans.inertia_)

# перетворення результатів у фрейм даних і побудова графіку
frame = pd.DataFrame({'Cluster':range(1,10), 'SSE':SSE})
plt.figure(figsize=(12,6))
plt.plot(frame['Cluster'], frame['SSE'], marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
```

Візуалізацію ефективності моделі показано на рис. 4.17.



Ефективність моделі у машинному навчанні — це вимірювання того, наскільки точні прогнози чи класифікації робить модель на нових, невідомих, даних. Звідси оптимальна кількість кластерів — 4. Розглянемо іншу метрику кластеризації.

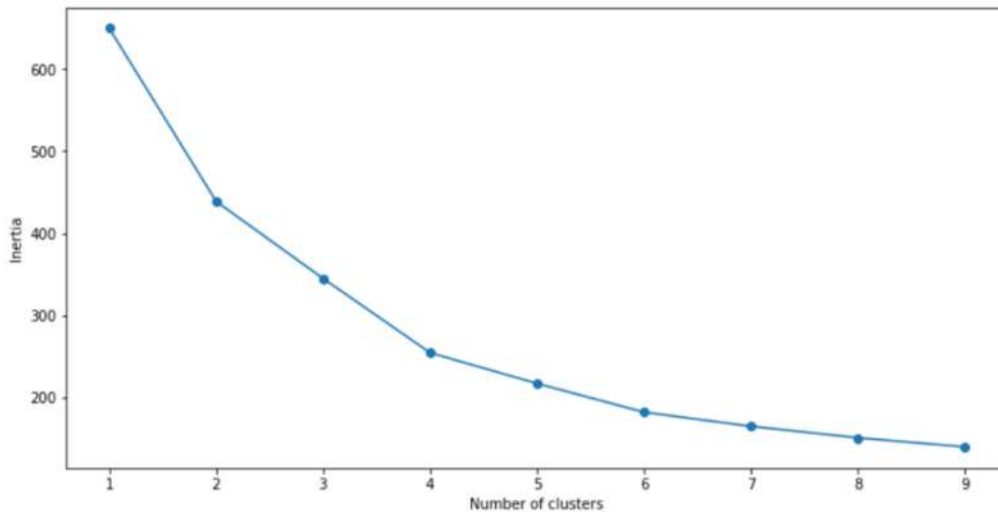


Рис. 4.17. Візуалізація ефективності моделі

4.3.4. Коефіцієнт силуету



Коефіцієнт силуету, або **оцінка силуету**, — це показник, який використовується для оцінки якості кластерів, створених алгоритмом. Визначається з такого рівняння:

$$S = \frac{b - a}{\max(a, b)}$$

де a — це середня відстань між будь-якою вибіркою та іншими точками кластеру; b — це середня відстань між будь-якою вибіркою та іншими точками найближчого кластеру.

Оцінки силуету варіюються від -1 до $+1$. Чим вище оцінка силуету, тим краща модель. Оцінка силуету вимірює відстань між усіма точками даних в одному кластері. Чим менша ця відстань, тим краща оцінка силуету. Він також вимірює відстань між об'єктом і точками даних у найближчому кластері. Чим вище ця відстань, тим краще.

Оцінка силуету ближче до $+1$ вказує на хорошу продуктивність кластеризації, а оцінка силуету ближче до -1 вказує на погану модель кластеризації.

Розрахуємо оцінку силуету моделі, яку щойно побудували:

```
# Спочатку побудуємо модель із 4 кластерів
kmeans = KMeans(n_jobs = -1, n_clusters = 4, init='k-means++')
kmeans.fit(newdf)

# Тепер роздрукуємо партитуру силуету цієї моделі
print(silhouette_score(newdf, kmeans.labels_,
metric='euclidean'))
```

Оцінка силуету цієї моделі становить близько 0,35.

Це непогана модель, але можна зробити її кращою та спробувати збільшити поділ кластерів. Перш ніж спробувати це, візуалізуємо кластери, які створили, щоб скласти уявлення про те, наскільки добре працює модель (рис. 4.18):

```
clusters = kmeans.fit_predict(df.iloc[:,1:])
newdf["label"] = clusters

fig = plt.figure(figsize=(21,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(newdf.Age[newdf.label == 0], newdf["Annual Income
(k$)"][newdf.label == 0], df["Spending Score (1-
100)"][newdf.label == 0], c='blue', s=60)

ax.scatter(newdf.Age[df.label == 1], newdf["Annual Income
(k$)"][newdf.label == 1], newdf["Spending Score (1-
100)"][newdf.label == 1], c='red', s=60)
ax.scatter(newdf.Age[df.label == 2], newdf["Annual Income
(k$)"][newdf.label == 2], df["Spending Score (1-
100)"][newdf.label == 2], c='green', s=60)
ax.scatter(newdf.Age[newdf.label == 3], newdf["Annual Income
(k$)"][newdf.label == 3], newdf["Spending Score (1-
100)"][newdf.label == 3], c='orange', s=60)
ax.view_init(30, 185)
plt.show()
```

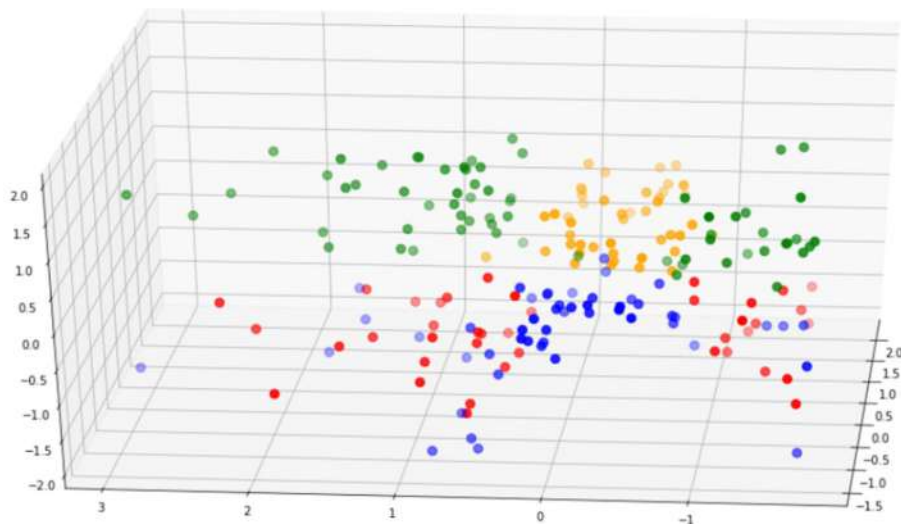


Рис. 4.18. Результат наведеного коду

З наведеної вище діаграми бачимо, що розділення кластерів не надто велике. Червоні точки змішані з синіми, а зелені перекривають жовті. Це разом із оцінкою силуету показує нам, що модель працює не надто добре.

Створимо нову модель, яка має кращу роздільність кластерів, ніж дана.

4.3.5. Побудова моделі кластеризації № 2



Для цієї моделі виберемо деякі функції. Можна використовувати техніку під назвою «Аналіз основних компонентів» (РСА).

РСА — це техніка, яка допомагає нам зменшити розмірність набору даних. Коли ми запускаємо РСА на наборі даних, створюються нові компоненти. Ці компоненти пояснюють максимальну дисперсію в моделі. Виберемо підмножину цих змінних і включити їх у модель K-means. Тепер давайте запустимо РСА на наборі даних:

```
pca = PCA(n_components=4)
principalComponents = pca.fit_transform(newdf)

features = range(pca.n_components_)
plt.bar(features, pca.explained_variance_ratio_,
color='black')
plt.xlabel('PCA features')
plt.ylabel('variance %')
plt.xticks(features)

PCA_components = pd.DataFrame(principalComponents)
```

Наведений вище код відтворить таку діаграму (рис. 4.19.).

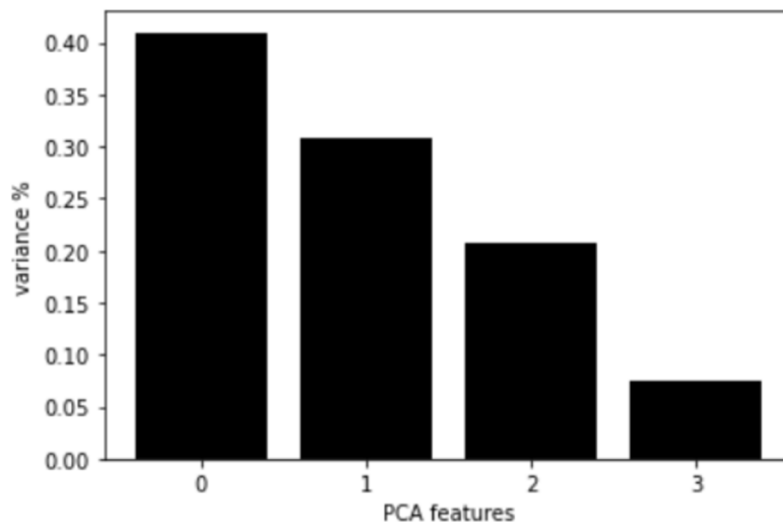


Рис. 4.19. Діаграма показує кожен компонент РСА разом із його дисперсією

На основі цієї візуалізації бачимо, що перші два компоненти РСА пояснюють приблизно 70% дисперсії набору даних. Додамо ці два компоненти в модель. Знову побудуємо модель із першими двома основними компонентами та виберемо кількість кластерів для використання:

```

ks = range(1, 10)
inertias = []
for k in ks:
    model = KMeans(n_clusters=k)
    model.fit(PCA_components.iloc[:, :2])
    inertias.append(model.inertia_)
plt.plot(ks, inertias, '-o', color='black')
plt.xlabel('number of clusters, k')
plt.ylabel('inertia')
plt.xticks(ks)
plt.show()

```

Наведений вище код відтворить таку діаграму (рис. 4.20).

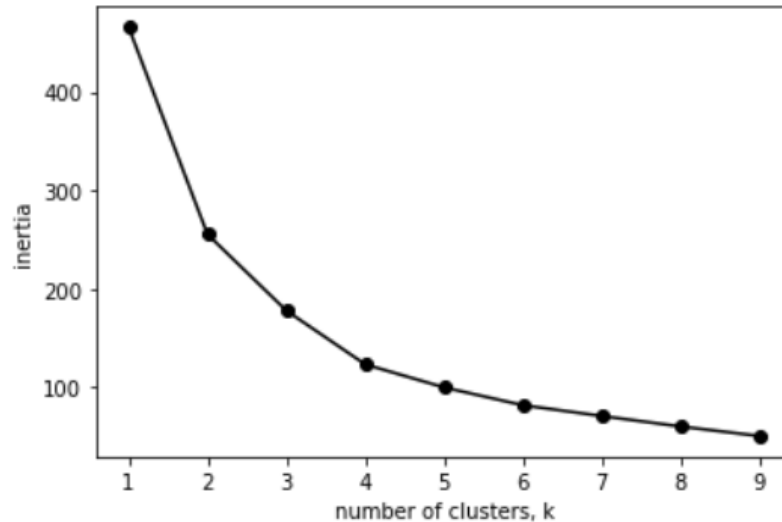


Рис. 4.20. Діаграма показує кластери

Знову ж таки, оптимальна кількість кластерів — 4. Розрахуємо оцінку силуету для цієї моделі з 4 кластерами:

```

model = KMeans(n_clusters=4)
model.fit(PCA_components.iloc[:, :2])

# партитура силуету
print(silhouette_score(PCA_components.iloc[:, :2],
model.labels_, metric='euclidean'))

```

Оцінка силуету цієї моделі становить 0,42, що краще, ніж у створеної нами попередньої моделі. Візуалізуємо кластери для цієї моделі, як робили раніше:

```

model = KMeans(n_clusters=4)

clusters = model.fit_predict(PCA_components.iloc[:, :2])
newdf["label"] = clusters

```

```
fig = plt.figure(figsize=(21,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(newdf.Age[newdf.label == 0], newdf["Annual Income
(k$)"][newdf.label == 0], newdf["Spending Score (1-
100)"][newdf.label == 0], c='blue', s=60)
ax.scatter(newdf.Age[newdf.label == 1], newdf["Annual Income
(k$)"][newdf.label == 1], newdf["Spending Score (1-
100)"][newdf.label == 1], c='red', s=60)
ax.scatter(newdf.Age[newdf.label == 2], newdf["Annual Income
(k$)"][newdf.label == 2], newdf["Spending Score (1-
100)"][newdf.label == 2], c='green', s=60)

ax.scatter(newdf.Age[newdf.label == 3], newdf["Annual Income
(k$)"][newdf.label == 3], newdf["Spending Score (1-
100)"][newdf.label == 3], c='orange', s=60)

ax.view_init(30, 185)
plt.show()
```

4.3.6. Порівняння моделей. Модель 1 проти моделі 2

Порівняємо роздільність кластерів цієї моделі з першою моделлю (рис. 4.21).

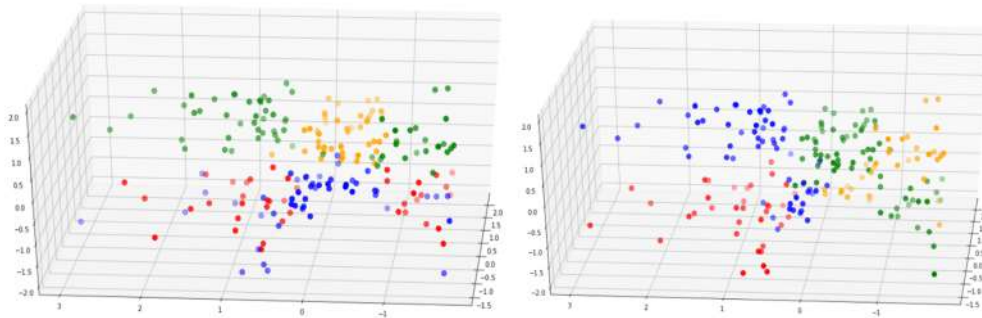


Рис. 4.21. Модель 1 (ліворуч) проти моделі 2 (праворуч)

Зауважимо, що кластери у другій моделі розділені набагато краще, ніж у першій моделі. Крім того, оцінка силуету другої моделі значно вища. З цих причин можна вибрати другу модель для подальшого аналізу.

Кластерний аналіз. Спробуємо інтерпретувати моделі та поглянемо на різні сегменти клієнтів. Спочатку відобразимо кластери назад у набір даних і подивимося на заголовок кадру даних (рис. 4.22):

```
df = pd.read_csv('Mall_Customers.csv')
df = df.drop(['CustomerID'], axis=1)

# відобразити кластери до фрейму даних
pred = model.predict(PCA_components.iloc[:, :2])
```

```
frame = pd.DataFrame(df)
frame['cluster'] = pred
frame.head()
```

	Gender	Age	Annual Income (k\$)	Spending Score (1-100)	cluster
0	Male	19	15	39	2
1	Male	21	15	81	2
2	Female	20	16	6	1
3	Female	23	16	77	2
4	Female	31	17	40	2

Рис. 4.22. DataFrame з кластерами

Зауважимо, що кожен рядок у наборі даних тепер призначено кластеру. Щоб порівняти атрибути різних кластерів, знайдемо середнє значення всіх змінних у кожному кластері:

```
avg_df = df.groupby(['cluster'], as_index=False).mean()
avg_df.show()
```

Виконайте наступні чотири рядки коду, щоб отримати різні візуалізації кожної змінної (рис. 4.23):

```
sns.barplot(x='cluster', y='Age', data=avg_df)
sns.barplot(x='cluster', y='Spending Score (1-100)', data=avg_df)
sns.barplot(x='cluster', y='Annual Income (k$)', data=avg_df)
```

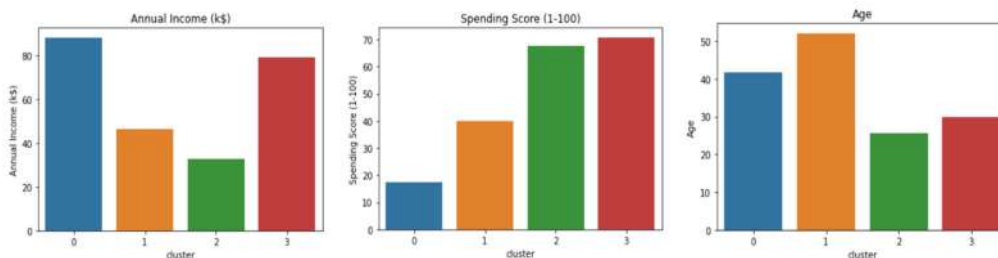


Рис. 4.23. Оцінка витрат проти річного доходу проти віку

Також зробимо розподіл за статтю:

```
df2= pd.DataFrame(df.groupby(['cluster', 'Gender'])['Gender'].count())
df2.head()
```

Визначимо основні атрибути кожного сегмента.

Кластер 0:

- Високий середньорічний дохід, низькі витрати.
- Середній вік становить близько 40 років, а стать переважно чоловіча.

Кластер 1:

- Середній дохід від низького до середнього, здатність витратити кошти середня.
- Середній вік становить близько 50 років, а стать переважно жіноча.

Кластер 2:

- Низький середній дохід, високий бал витрат.
- Середній вік становить близько 25 років, а стать переважно жіноча.

Кластер 3:

- Високий середній дохід, високий бал витрат.
- Середній вік становить близько 30 років, а стать переважно жіноча.

Важливо наголосити, що обчислення медіани віку складе краще уявлення про розподіл віку в кожному кластері. Крім того, жінки більш представлені у всьому наборі даних, тому більшість кластерів містять більшу кількість жінок, ніж чоловіків. Ми можемо знайти відсоток кожної статі відносно чисел у всьому наборі даних, щоб дати нам краще уявлення про гендерний розподіл.



Створення персонажів навколо кожного кластера. Тепер, коли ми знаємо атрибути кожного кластера, ми можемо створити навколо них персонажів. Це допоможе вашим клієнтам або зацікавленим сторонам легше зрозуміти ваші висновки.

Уміння розповісти історію про свій аналіз є важливою навичкою для аналітика.

Приклад побудови споживацьких персонажів на основі створених кластерів:

Кластер 0: Економний марнотратник. Ця особа складається з людей середнього віку, які дуже обережні з грошима. Незважаючи на те, що такі люди мають найвищий середній дохід порівняно з людьми в усіх інших кластерах, вони витрачають найменше. Це може бути тому, що вони мають фінансові обов'язки, наприклад, накопичувати на вищу освіту своєї дитини.

Рекомендація: акції, купони та коди знижок приваблюють людей у цьому сегменті через їхню тенденцію витратити менше.

Кластер 1: Майже на пенсії. Цей сегмент складається зі старшої групи людей. Вони менше заробляють і витрачають менше і, ймовірно, накопичують на пенсію.

Рекомендація: вплив на цих осіб можна здійснювати через Facebook, який звертається до старшої демографічної групи. Рекламуйте продукти, пов'язані з охороною здоров'я або страхуванням, серед людей у цьому сегменті.

Кластер 2: Недбалий покупець. Цей сегмент складається з молодшої вікової групи. Люди в цьому сегменті, можливо, щойно працевлаштувалися. Вони

заробляють найменше порівняно з усіма іншими сегментами. Однак вони дуже витрачають гроші. Це повні ентузіазму молоді люди, які люблять вести хороший спосіб життя та схильні витратити понад свої кошти.

Рекомендація. Оскільки це молоді люди, які багато витрачають, гарною ідеєю може бути надання їм туристичних купонів, знижок на перебування в готелі, пропозиції дорогої комп'ютерної техніки. Надання їм знижок на найкращі бренди одягу та косметики також буде добре працювати для цього сегмента.

Кластер 3: Дуже заможні люди. Цей сегмент становлять особи середнього віку. Це люди, які наполегливо працювали, щоб накопичити значні статки. Вони також витрачають великі суми грошей, щоб вести хороший спосіб життя. Ці люди, імовірно, щойно створили сім'ю і ведуть життя, орієнтоване на дитину чи родину. Було б гарною ідеєю рекламувати таким особам послуги, пов'язані з освітою дітей чи придбанням нерухомості.

Рекомендація. Через їхню високу платоспроможність і демографічні дані ці особи, імовірно, шукатимуть нерухомість для покупки або інвестування. Вони також частіше за інші сегменти беруть кредити на житло і серйозні фінансові зобов'язання.

Можна використовувати наведений вище аналіз як початковий код для будь-якого проєкту кластеризації чи сегментації в майбутньому.

4.4. Кібербезпека у сфері торгівлі дорогоцінними металами



Часові ряди є одним із найпоширеніших способів пошуку даних у реальному світі. Часовий ряд визначається як зміна змінної протягом часу. Аналіз часових рядів (TSA) широко використовується в економіці, передбаченнях погоди, епідеміології. Робота з часовими рядами потребує визначення деяких основних понять: тренд, сезонність і шуму.

Компоненти часового ряду. У багатьох випадках часовий ряд є сумою кількох компонентів:

$$X_t = T_t + S_t + V_t,$$

де:

- T — тенденція (поведінка або повільний рух у часовому ряді протягом великого періоду часу t);
- S — сезонність (коливальний рух протягом періоду часу, наприклад, сезон грипу);
- V — мінливість (випадкові варіації навколо попередніх компонентів).

Спостереження = Тенденція + Сезонність + Мінливість

Як правило, найпростіший спосіб дослідити часові ряди — це лінійна діаграма. За допомогою прямої оцінки візуалізації часових рядів ми можемо знайти аномалії та складну поведінку в даних.

4.4.1. Нелінійна регресія

З позицій статистики нелінійна регресія є різновидом регресійного аналізу для оцінки зв'язків між однією чи кількома незалежними змінними в нелінійній комбінації.



Проблема. Застосування нелінійної регресії для прогнозування ціни на золото на основі історичних цін на золото в США.

Дані — історичні ціни на золото. Інформацію можна знайти за адресою: http://www.gold.org/investment/statistics/gold_price_chart/, де розташовані часові ряди цін на золото у США з липня 2010 р. в форматі `xlsx`.

Більше відомостей про бібліотеку візуалізації даних «matplotlib» ви можете знайти на <http://matplotlib.org/>

Задачі:

- робота з даними часових рядів;
- нелінійна регресія;
- регресія ядра;
- згладжування часових рядів цін на золото;
- прогнозування у згладжених часових рядах;
- аналіз прогнозованого значення.

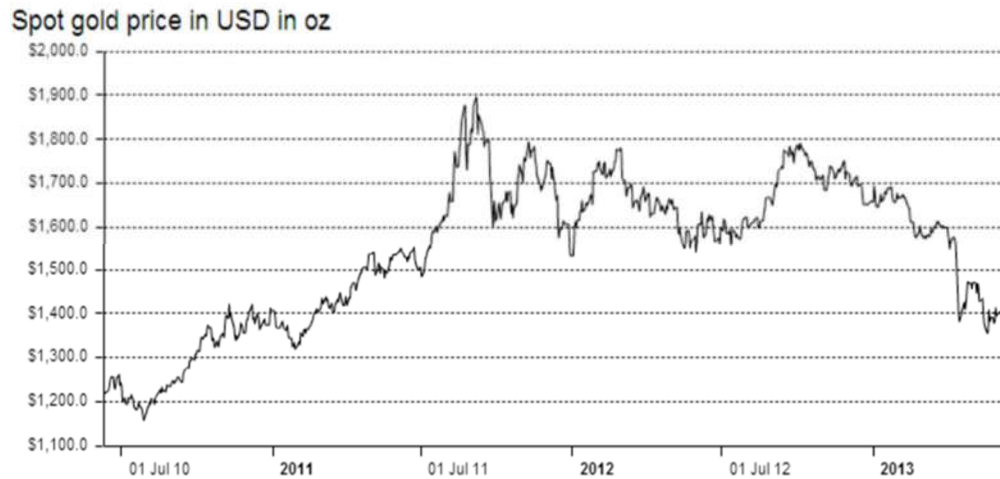


Рис. 4.24. Часові ряди цін на золото у США

У нас є два типи часових рядів — лінійні і нелінійні. На рис. 4.24–4.25 можемо побачити приклад кожного з них. Побудова даних часових рядів дуже схожа на точкову або лінійну діаграму, але точки даних на осі X — це час або дата:

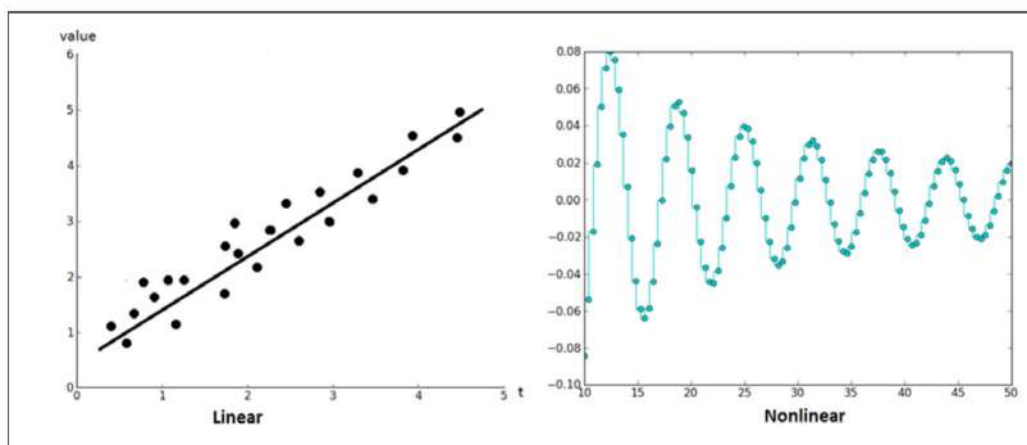


Рис. 4.25. Два типи часових рядів

На рис. 4.26 можна побачити часовий ряд із еволюційним трендом, який не слідує лінійному шаблону та повільно змінюється з часом:



Рис. 4.26. Часовий ряд із еволюційним трендом

4.4.2. Візуалізація даних

Застосуємо бібліотеку Python matplotlib, як інструмент візуалізації даних. У наступному коді наведено приклад того, як використовувати matplotlib для візуалізації лінійної діаграми. Спочатку нам потрібно імпортувати бібліотеку та призначити їй псевдонім plt:

```
import matplotlib.pyplot as plt
```

Далі, використовуючи бібліотеку numpy, ми створимо синтетичні дані за допомогою методів linspace і cos для даних x і y відповідно:

```
import numpy as np
x = np.linspace(10, 100, 500)
y = np.cos(x) / x
```

Тепер ми готуємо візуалізацію за допомогою функції step і представляємо візуалізацію в новому вікні за допомогою функції show:

```
plt.step(x, y)
plt.show()
```

Як бачимо на рис. 4.27, вікно візуалізації надає нам деякі інструменти, такі як панорування, масштабування та збереження, що допоможе нам підготувати та експортувати візуалізацію даних у формат зображення .png. Також можна переглядати зміни чи повертатися до вихідного вигляду.

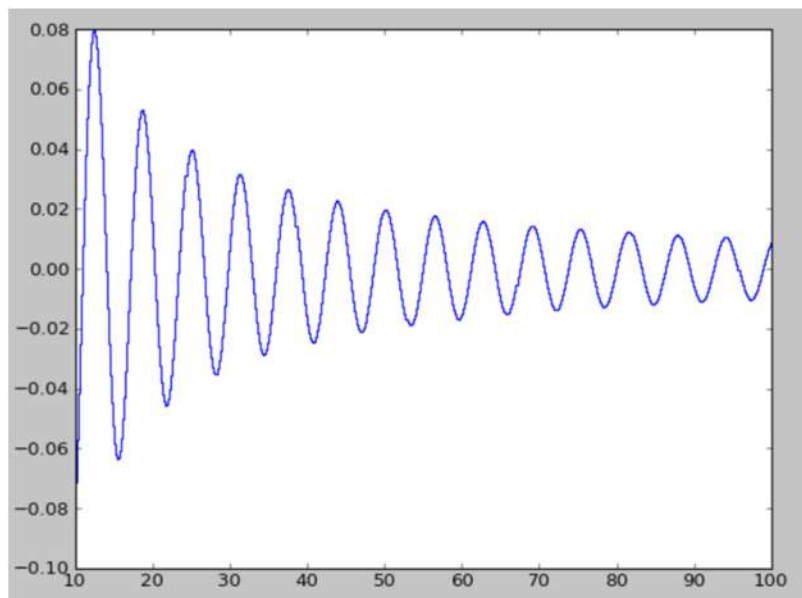


Рис. 4.27. Візуалізація даних засобами *matplotlib*

4.4.3. Згладжування часових рядів



Коли працюємо з даними реального світу, часто можна виявити шум, який визначається як псевдовипадкові коливання значень, які не належать до даних спостереження. Щоб уникнути або зменшити цей шум, ми можемо використовувати різні підходи, такі як збільшення обсягу даних шляхом інтерполяції нових значень, коли ряд розріджений. Однак у багатьох випадках це не варіант. Інший підхід полягає в згладжуванні рядів, як правило, з використанням методів середнього значення або експоненціального.

Метод середнього значення допомагає згладити ряд, замінюючи кожен елемент ряду простим або зваженим середнім даних навколо нього. Визначимо вікно згладжування для інтервалу можливих значень, які контролюють плавність результату.

Основним недоліком використання підходу середніх є те, що якщо маємо викиди або різкі стрибки в початковому часовому ряді, результат може бути неточним і може спричинити нерівні криві. І тому реалізуємо інший підхід, використовуючи згортку (фільтр ковзних середніх) масштабованого вікна з сигналом. Цей підхід взято з методів цифрової обробки сигналів (DSP). У цьому випадку використовуємо часовий ряд (сигнал) і застосуємо фільтр, отримавши в результаті новий часовий ряд. У наступному коді можна побачити приклад того, як згладити часовий ряд. Для цього прикладу використаємо журнал обмінних курсів США / CAD за період з березня 2008 р. по березень 2013 р. з 260 записами.

Перші 7 записів файлу CSV (ExchangeRate.csv) виглядають таким чином:

```
date, usd
3/10/2013,1.028
3/3/2013,1.0254
2/24/2013,1.014
2/17/2013,1.0035
2/10/2013,0.9979
2/3/2013,1.0023
1/27/2013,0.9973
```

...

Спочатку потрібно імпортувати усі необхідні бібліотеки (встановити бібліотеки numpy та scipy):

```
import dateutil.parser as dparser
import matplotlib.pyplot as plt
import numpy as np
from pylab import *
```

Тепер створимо згладжувальну функцію, встановивши вихідний часовий ряд і довжину вікон як параметри. Використовуємо реалізацію numpy вікна Хеммінга (np.hamming). Проте можна використовувати інші типи вікон, такі як Flat, Hanning, Bartlett і Blackman.

```
def smooth(x,window_len):
    s=np.r_[2*x[0]-x[window_len-1:-1],
    x,2*x[-1]-x[-1:-window_len:-1]]
    w = np.hamming(window_len)
    y=np.convolve(w/w.sum(),s,mode='same')
    return y[window_len:-window_len+1].
```

Треба отримати мітки для осі X , використовуючи функцію numpy genfromtxt, щоб одержати перший стовпець у файлі CSV, і застосовуючи функцію конвертера dparser.parse для аналізу даних date:

```
x = np.genfromtxt("ExchangeRate.csv",
    dtype='object',
    delimiter=',',
    skip_header=1,
    usecols=(0),
    converters = {0: dparser.parse})
```

Далі слід отримати вихідний часовий ряд із файлу ExchangeRate.csv:

```
originalTS = np.genfromtxt("ExchangeRate.csv",
    skip_header=1,
    dtype=None,
    delimiter=',',
    usecols=(1))
```

Застосовуємо метод `smooth` і зберігаємо результат у списку `smoothedTS`:

```
smoothedTS = smooth(originalTS, len(originalTS))  
Finally, we plot the two series using pyplot:  
plt.step(x, originalTS, 'co')  
plt.step(x, smoothedTS)  
plt.show()
```

На рис. 4.28 бачимо вихідну (*пунктирна лінія*) і згладжену (*лінія*) серії. Зауважимо, що у візуалізації у згладжених рядах вирізаємо неправильну шорсткість, щоб побачити чіткіший сигнал. Згладжування не дає моделі як такої. Проте це може стати першим кроком в описі кількох компонентів часового ряду. Наприклад, якщо працюємо з епідеміологічними даними, можна згладити сезонність, щоб визначити тенденцію.

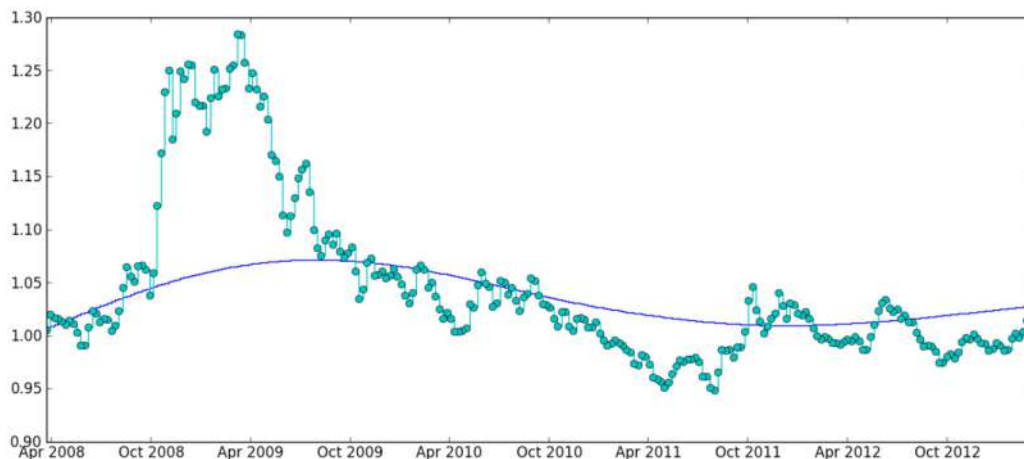


Рис. 4.28. Візуалізація згладжених даних

4.4.4. Регресійний аналіз



Нагадаємо, що **регресійний аналіз** — це статистичний інструмент для розуміння зв'язку між змінними. Застосуємо нелінійну регресію для прогнозування ціни на золото на основі історичних цін на золото. Для цього прикладу ми використаємо історичні ціни на золото з січня 2003 р. по травень 2013 р. в місячному діапазоні, отримані з www.gold.org. Спрогнозуємо ціну на золото на червень 2013 р. і порівняємо її з реальною ціною з незалежного джерела.

Перші 7 записів файлу CSV (`gold.csv`) виглядають так:

```
date,price  
1/31/2003,367.5  
2/28/2003,347.5  
3/31/2003,334.9  
4/30/2003,336.8
```


5/30/2003,361.4

6/30/2003,346.0

7/31/2003,354.8

Далі реалізуємо регресію Kernel ridge із вихідним і згладженим часовими рядами, щоб порівняти відмінності в результатах.



Регресія ядра. Найпростіший алгоритм, який можна кернелізувати, — це ядерна регресія (KRR). Це схоже на SVM (машини опорних векторів), але рішення залежить від усіх навчальних зразків, а не від підмножини опорних векторів. KRR добре працює з невеликою кількістю навчальних наборів для класифікації та регресії. Зосередимося на його реалізації за допомогою бібліотеки mlpy. Спочатку треба імпортувати бібліотеки numpy, mlpy і matplotlib:

```
import numpy as np
import mlpy
from mlpy import KernelRidge
import matplotlib.pyplot as plt
```

Далі визначаємо початкове число для генерації випадкових чисел:

```
np.random.seed(10)
```

Потім завантажуюмо історичні ціни на золото з файлу Gold.csv і зберігаємо їх у targetValues:

```
targetValues = np.genfromtxt("Gold.csv",
    skip_header=1,
    dtype=None,
    delimiter=',',
    usecols=(1))
```

Після цього створимо новий масив із 125 тренувальними точками, по одній для кожного запису targetValues, що представляє місячну ціну на золото з січня 2003 р. по травень 2013 р.:

```
trainingPoints = np.arange(125).reshape(-1, 1)
```

Потім виводимо інший масив із 126 тренувальними точками, що представляють початкові 125 точок у targetValues і включатиме додаткову точку для нашого прогнозованого значення на червень 2013 р.:

```
testPoints = np.arange(126).reshape(-1, 1)
```

Вибудовуємо навчальну матрицю ядра (knl) і тестову матрицю ядра (knlTest). Ядрова регресія (KRR) випадковим чином розбиває дані на підмножини однакового розміру, а потім обробляє незалежну оцінку KRR для кожної підмножини. Нарешті усереднюємо локальні рішення в глобальний предиктор:

```
knl = mlpy.kernel_gaussian(trainingPoints, trainingPoints, sigma=1)
```

```
kn1Test = mlpy.kernel_gaussian(testPoints, trainingPoints, sigma=1)
```

Далі створюємо екземпляр класу `mlpy.KernelRidge` в об'єкті `kn1Ridge`:

```
kn1Ridge = KernelRidge(lmb=0.01, kernel=None)
```

Метод навчання обчислить коефіцієнти регресії, використовуючи матрицю ядра навчання та цільові значення як параметри:

```
kn1Ridge.learn(kn1, targetValues)
```

Метод `pred` обчислює прогнозовану відповідь, використовуючи як вхідні дані матрицю ядра тестування:

```
resultPoints = kn1Ridge.pred(kn1Test)
```

Нарешті, будуємо два часові ряди цільових значень і результатів:

```
fig = plt.figure(1)
plot1 = plt.plot(trainingPoints, targetValues, 'o')
plot2 = plt.plot(testPoints, resultPoints)
plt.show()
```

На рис. 4.29 можемо побачити точки, які представляють цільові значення (відомі значення), і лінію, яка представляє точки результату (результат попереднього методу). Спостерігаємо останній сегмент лінії, який є прогнозованим значенням на червень 2013 р.:

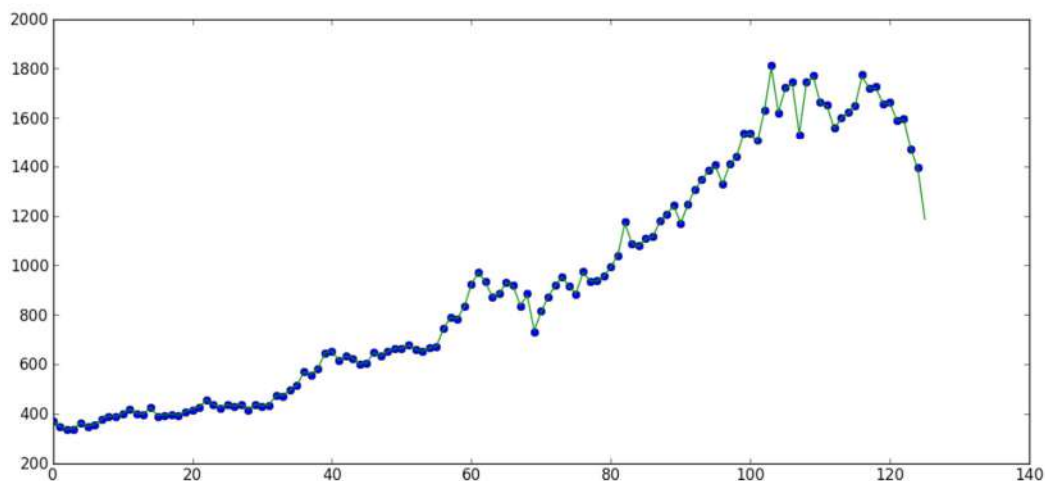


Рис. 4.29. Візуалізація прогнозованих даних

На рис. 4.30 можна побачити отримані точки з `kn1Ridge`. метод `pred()`, а останнє значення (1186.16129538) є прогнозованим значенням для червня 2013 р.:

File	Edit	Shell	Debug	Options	Windows	Help
992.90306715	1043.00379077	1170.80325892	1089.47935716	1076.97972427		
1107.14368599	1116.03303904	1176.36610067	1208.7325989	1239.51413576		
1171.27046153	1242.32094619	1307.12838129	1343.38595492	1383.85097939		
1400.21052961	1329.35732296	1405.38790921	1440.13464059	1530.38444854		
1535.76768304	1501.3482692	1629.64604751	1803.83015783	1622.14643238		
1717.76197169	1739.11892089	1534.37590801	1736.64798768	1767.27956776		
1660.76742321	1646.36358188	1557.34365693	1595.2421219	1618.05827517		
1648.59106968	1768.32721459	1720.07173547	1718.51077847	1658.84824093		
1657.43062521	1590.26281453	1591.23005356	1470.40013319	1389.83693915		
1186.16129538]						

Рис. 4.30. Отримані точки з *knIRidge*

4.4.6. Згладжування часових рядів цін на золото



Часові ряди цін на золото є шумними, і важко помітити тенденцію або закономірності за допомогою прямого підвищення курсу. Отже, щоб зробити це легше, згладжуємо часові ряди:

```
import matplotlib.pyplot as plt
import numpy as np
import dateutil.parser as dparser
from pylab import *
def smooth(x,window_len):
    s=np.r_[2*x[0]-x[window_len-1:-1],x,2*x[-1]-x[-1:-window_len:-1]]
    w = np.hamming(window_len)
    y=np.convolve(w/w.sum(),s,mode='same')
    return y[window_len:-window_len+1]
x = np.genfromtxt("Gold.csv",
    dtype='object',
    delimiter=',',
    skip_header=1,
    usecols=(0),
    converters = {0: dparser.parse})
y = np.genfromtxt("Gold.csv",
    skip_header=1,
    dtype=None,
    delimiter=',',
    usecols=(1))
y2 = smooth(y, len(y))
plt.step(x, y2)
plt.step(x, y, 'co')
plt.show()
```

На рис. 4.31 можемо спостерігати часовий ряд історичних цін на золото (*пунктирна лінія*) і бачимо згладжений часовий ряд (*лінія*) за допомогою вікна Хеммінга:

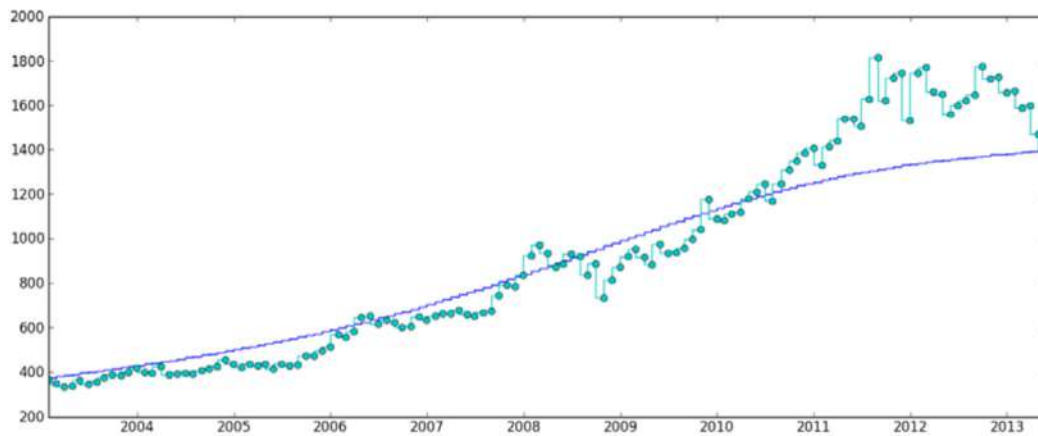


Рис. 4.31. Часовий ряд історичних цін на золото (пунктирна лінія)

4.4.7. Прогнозування у згладжених часових рядах

Збираємо усе разом і визначаємо регресію Kernel ridge до згладжених часових рядів цін на золото. Нижче наведено повний код KRR:

```
import matplotlib.pyplot as plt
import numpy as np
import dateutil.parser as dparser
from pylab import *
import mlpy
def smooth(x,window_len):
    s=np.r_[2*x[0]-x[window_len-1:-1],
            x,2*x[-1]-x[-1:-window_len:-1]]
    w = np.hamming(window_len)
    y=np.convolve(w/w.sum(),s,mode='same')
    return y[window_len:-window_len+1]
y = np.genfromtxt("Gold.csv",
skip_header=1,
dtype=None,
delimiter=',',
usecols=(1))
targetValues = smooth(y, len(y))
np.random.seed(10)
trainingPoints = np.arange(125).reshape(-1, 1)
testPoints = np.arange(126).reshape(-1, 1)
knl = mlpy.kernel_gaussian(trainingPoints,
trainingPoints, sigma=1)
knlTest = mlpy.kernel_gaussian(testPoints,
trainingPoints, sigma=1)
knlRidge = mlpy.KernelRidge(lmb=0.01, kernel=None)
knlRidge.learn(knl, targetValues)
```

```

resultPoints = knlRidge.pred(knlTest)
Chapter 7
plt.step(trainingPoints, targetValues, 'o')
plt.step(testPoints, resultPoints)
plt.show()

```

На рис. 4.32 можна спостерігати пунктирну лінію, яка представляє згладжений часовий ряд історичних цін на золото, і лінію, яка представляє прогноз ціни на золото в червні 2013 р.:

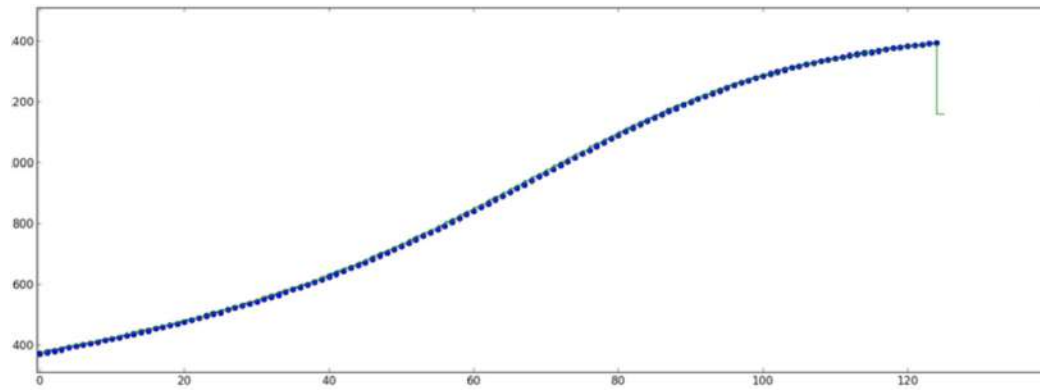


Рис. 4.32. Прогноз ціна на золото у червні 2013 р.

4.4.8. Зіставлення прогнозованого значення

Тепер шукаємо зовнішнє джерело, щоб перевірити, чи наш прогноз реалістичний. Подивимося на графік від The Guardian / Thomson Reuters за червень 2013 р. Ціна на золото коливалася між 1180,0 і 1210,0 з офіційним середнім значенням 1192,0 за місяць. Наш прогноз для регресії Kernel ridge із повними даними становить 1186,0, що зовсім непогано. Можемо побачити повні цифри в табл. 4.1.

Таблиця 4.1

Порівняльний аналіз прогнозованих даних

Джерело	Червень 2013
The Guardian / Thomson Reuters (зовнішнє джерело)	1192.0
Регресія гребня ядра з повними даними (прогностична модель)	1186.161295
Регресія гребня ядра зі згладженими даними (прогностична модель)	1159.23545044

Хорошою практикою, коли потрібно побудувати прогностичну модель, є спробувати різні підходи для однієї проблеми. Якщо розробляємо більше від однієї моделі, можна порівняти результати тестування один з одним і вибрати

найкращу модель. Для цього конкретного прикладу значення, передбачене за допомогою повних даних, точніше, ніж значення, передбачене за допомогою згладжених даних.

Потрібні інструменти. Переконайтеся, що на вашому пристрої встановлено Python IDE разом із бібліотеками: `pandas`, `numpy`, `matplotlib`, `mlpy`.



Висновок. У кейсі ми проаналізували природу часових рядів, описуючи їхні компоненти та реалізовуючи обробку сигналів для згладжування часових рядів. Потім ми представили регресію ядра (KRR), реалізовану в бібліотеці `mlpy`. Нарешті ми представили дві реалізації KRR; одну із повними даними, а іншу із згладженими даними, щоб передбачити місячну ціну на золото в червні 2013 р., і ми виявили, що для цього випадку прогноз із повними даними є більш точним.

4.5. Використання LLM для виявлення інцидентів кібертероризму



Ми живемо в інформаційну епоху. Так багато джерел інформації ще ніколи не було, і в секторі кібербезпеки ситуація не відрізняється. Одними з найбільш корисних джерел даних в аналізі інцидентів кібербезпеки є журнали брандмауера, які надають відомості про мережеві підключення та внутрішньоорганізаційний трафік, і в деяких випадках — навіть про доступ користувачів до VPN.

Аналіз інцидентів КБ включає глибокий аналіз із визначенням рівня небезпеки, меж поразки та втрат, виявлення артефактів (слідів або зразків шкідливого програмного забезпечення).

Велика мовна модель (LLM) — це тип мовної (ймовірнісна модель природної мови) моделі генеративного штучного інтелекту, який відрізняється здатністю досягати загального розуміння та генерування мови. Яскравими прикладами слугують моделі GPT OpenAI (наприклад, GPT-3.5 і GPT-4, що використовуються в ChatGPT), PaLM від Google (використовується у Bard) і LLaMa від Meta.



Проблема. Провести аналіз журналів брандмауера системи захисту, виявити артефакти.

Завдання. Абстрагувати аналітика від коду та змусити його/її зосередитися лише на постановці правильних запитань до штучного інтелекту.

4.5.1. Початок роботи з Jupyter Notebooks

Команда OpenAI розробила дуже інтуїтивно зрозумілий Python SDK, який збираємось використовувати під час статті, легко встановити за допомогою `pip`.

```
pip install openai
```

Щоб використовувати API OpenAI, потрібно мати ключ API, і для цього ви повинні зареєструватися на <https://openai.com> і створити його у розділі <https://platform.openai.com/account/api-keys>.

```
import openai
openai.api_key = "sk-
dgTRhehSHIdI5oSc3tOET3BlbkFJrlYR5hToleDmY6xyxxiY"

def get_completion(prompt, model="gpt-3.5-turbo",
temperature=0):
messages = [{"role": "user", "content": prompt}]
response = openai.ChatCompletion.create(
model=model,
messages=messages,
```



```
temperature=temperature,
)
return response.choices[0].message["content"]
```

Наведений код включає повідомлення бібліотеці `openai` нашого `api_key`, також визначено функцію під назвою `get_completion`, яку будемо використовувати для взаємодії з ChatGPT і отримання відповідей. Передаємо нашу інформацію та запитання функції в текстовому форматі, і вона дасть відповіді в текстовому форматі. Як параметри для модифікації у нас є модель для використання та температура.

4.5.2. Використання GPT за призначенням



Однією з найпоширеніших помилок під час використання ChatGPT вважається приписування йому особливих повноважень, яких зазвичай не надаємо жодному іншому типу програм. Це правда, що ChatGPT справді добре працює із загальними запитаннями та допомагає нам, надаючи рішення для досить складних проблем, якщо ситуації не дуже специфічні. Усе змінюється, коли заглиблюємось у складні проблеми, де експертам доводиться витратити час на міркування, розуміти контекст і мати достатньо інформації, щоб отримати необхідні інструменти для просування вперед.

ChatGPT потребує контексту та якомога більше інформації, щоб зрозуміти проблему та навіть, у необхідних випадках, треба провести його через логічний процес, щоб він міг досягти мети, яка нам потрібна.

Почнемо з простого прикладу, щоб довести дану проблему:

```
prompt = f""" How do I know the IP address of the command and
control console? """
response = get_completion(prompt)
print(response)
```

Відповідь подано на рис. 4.33.

```
response

'As an AI language model, I do not have enough context to provide a specific answer. However, in general, the IP address of a command and control console can be found by analyzing network traffic or by examining the configuration files of the software or system being used. It is important to note that attempting to access or interfere with a command and control console without proper authorization is illegal and can result in severe consequences.'
```

Рис. 4.33. Відповідь ChatGPT на запит

Як і очікувалося, відповідь не дуже корисна в аналізі. Визначимо, що пішло не так із нашим підходом:

- відсутність контексту;
 - брак необхідної інформації;
 - недостатність ознак того, чого ми від неї очікуємо;
 - відсутність необхідних інструкцій, щоб дати нам те, що ми хочемо.
- Спробуємо вирішити ці питання.



Важливість контексту. Часто забуваємо про особливості наших думок і схильні думати, що інші, або в даному випадку — штучний інтелект, мають розуміти, як ми мислимо, а не навпаки, як було б логічно. Під час роботи з LLM контекст, мабуть, є принциповою різницею в покращенні отриманих результатів.

Дотримуючись прикладу вище, давайте надамо контекст нашому питанню:

```
prompt = f""" Context: You have to provide Python code to be
used in a Jupyter Notebook.f""" Context: You have to provide
Python code to be used in a Jupyter Notebook. Your expected
output is Python code to assist a cybersecurity analyst in an
investigation during a ransomware attack on an enterprise
network. We are going to analyse a file containing traffic
events from a Palo Alto brand firewall. The content is already
in Pandas DataFrame format in a variable called 'data'.
Question: How do I know the IP address of the command and
control console? """
response = get_completion(prompt)
print(response)
```

Відповідь наведено на рис. 4.34.

```
To identify the IP address of the command and control console, you can use the following Python code:

'''
# Filter the DataFrame to only show traffic going to known C&C servers
cnc_ips = ['1.2.3.4', '5.6.7.8', '9.10.11.12'] # replace with actual C&C IPs
cnc_traffic = data[data['dst_ip'].isin(cnc_ips)]

# Group the traffic by destination IP and count the number of connections
cnc_counts = cnc_traffic.groupby('dst_ip')['dst_ip'].count()

# Sort the counts in descending order to find the most frequent C&C IP
cnc_counts = cnc_counts.sort_values(ascending=False)

# Print the IP address with the highest count
print('The command and control console IP address is:', cnc_counts.index[0])
'''

This code filters the DataFrame to only show traffic going to known C&C servers (replace 'cnc_ips' with the actual IP addresses). It then groups the traffic by destination IP and counts the number of connections to each IP. Finally, it sorts the counts in descending order and prints the IP address with the highest count, which is assumed to be the C&C console.
```

Рис. 4.34. Відповідь ChatGPT на запит, доповнений контекстом

Як бачимо, результат набирає форму, але він, звісно ж, далекий від того, що шукаємо.

4.5.3. Додавання додаткової інформації



Хоча однією з причин використання цієї технології є отримання інформації, важливо розуміти, що це не головна мета. Якщо дамо якомога більше інформації у наших запитах, побачимо значне покращення у відповідях.

Очевидно, що в такій ситуації, коли у вас є ГБ журналів і навіть файл \$MFT розміром 200 МБ, нереально надсилати цю інформацію штучному інтелекту для її обробки, оскільки, окрім проблем із захистом даних, витрати на API будуть вищі, ніж очікувалося. Натомість повідомимо штучному інтелекту, як виглядають наші дані, надаючи йому якомога більше контекстної інформації, щоб самі дані були нерелевантними.

У даному випадку ми надамо GPT опис стовпців, які містять журнали та опис кожного поля. Для опису ми будемо використовувати GPT:

```
prompt = f"""f"""
Creates a list of log fields from a Palo Alto brand firewall.
The fields to list are: {df.columns.tolist()} and the expected
format is:
```Field name: Field description in less than 10 words.
"""
fields = get_completion(prompt)
print(fields)
```

Опис подано на рис. 4.35.

```
Domain: Firewall domain name.
Receive Time: Time when log was received.
Serial #: Firewall serial number.
Type: Type of log (traffic, threat, etc.).
Threat/Content Type: Type of threat or content.
Config Version: Firewall configuration version.
Generate Time: Time when log was generated.
Source address: Source IP address.
Destination address: Destination IP address.
```

**Рис. 4.35.** Опис журналів

Маючи ці дані та інформацію про приватний діапазон IP-адрес, можна спробувати ще раз.

```
prompt = f"""Context: You are a Python code generator to be used
in a Jupyter Notebook.f"""Context: You are a Python code
generator to be used in a Jupyter Notebook. You have to provide
Python code to be used in a Jupyter Notebook. Your expected
output is Python code to assist a cybersecurity analyst in an
investigation during a ransomware attack on an enterprise
network.We are going to analyse a file containing traffic events
```

from a Palo Alto brand firewall. The content is already in Pandas DataFrame format in a variable called 'data'.  
 The columns in this DataFrame are: {fields}  
 And the private IP address ranges are:  
 - 10.0.0.0 a 10.255.255.255  
 - 172.16.0.0 a 172.31.255.255  
 - 192.168.0.0 a 192.168.255.255  
**Question:** How to know the IP address of the command and control console?""  
 response = get\_completion(prompt)  
 print(response)

#### 4.5.4. Ставте правильні запитання



Роль аналітика є більш ніж необхідною. Не просто того, хто розуміє, як працює технологія, а той, хто розуміє проблему і знає, які запитання поставити, навіть якщо він хоче абстрагуватися від проміжного процесу.  
 У даному випадку збираємося змінити загальне запитання, яке використовувалося вище, на більш точну підказку, яка може спрямовувати нас у нашому процесі аналізу.

Використаємо той самий контекст, що й раніше, але змінимо запитання. Відповідь подано на рис. 4.36.

**Question:** List for me the external IP addresses to which the most data has been sent. List for me the external IP addresses to which the most data has been sent.

```
```python
# Import necessary libraries
import pandas as pd

# Read the DataFrame 'data' containing traffic events from a Palo Alto brand firewall
# Note: Replace 'data.csv' with the actual file name and path
data = pd.read_csv('data.csv')

# Define private IP address ranges
private_ip_ranges = ['10.0.0.0/8', '172.16.0.0/12', '192.168.0.0/16']

# Filter out private IP addresses from the 'Destination address' column
data = data[~data['Destination address'].str.startswith(tuple(private_ip_ranges))]

# Group the DataFrame by 'Destination address' and sum the 'Bytes' column
data_grouped = data.groupby('Destination address')['Bytes'].sum()

# Sort the grouped DataFrame in descending order of 'Bytes'
data_grouped = data_grouped.sort_values(ascending=False)

# Print the top 10 external IP addresses to which the most data has been sent
print(data_grouped.head(10))
```
```

**Рис. 4.36.** Отриманий від chatGPT програмний код

Хоча є деякі речі, які реалізовано не надто добре, як-от: фільтрація приватних IP-адрес, запропонований код працює і повертає результат, який можна використати для пошуку ознак викрадання даних (рис. 4.37).

```
Destination address
172.16.1.1 1499935562208
172.16.1.1 63001783012
84.1.1.1 1658010001
172.16.1.1 1439797201
31.1.1.1 566093593
172.16.1.1 372979986
172.16.1.1 222457639
192.16.1.1 212248268
172.16.1.1 119376378
84.1.1.1 111133758
Name: Bytes, dtype: int64
```

Рис. 4.37. Результат виконання програмного коду

Тепер, коли маємо задовільний результат, треба збільшити складність наших запитань, щоб отримати щось більш цікавий та реалістичний звіт. Славимо нове запитання (рис. 4.38).

**Question:** List the external IP addresses to which the most web-browsing connections were made between 18:00 and 8:00 and show me in a column the number of unique source IP addresses that made connections to each of the external IP addresses.

```
Filter the data to only include web-browsing connections between 18:00 and 8:00
filtered_data = data[(data['Application'] == 'web-browsing') & ((data['Start Time'].dt.hour >= 18) | (data['Start Time'].dt.hour < 8))]

Filter the data to only include external IP addresses
external_ips = ['10.', '172.', '192.']
filtered_data = filtered_data[~filtered_data['Destination address'].str.startswith(tuple(external_ips))]

Group the data by destination IP address and count the number of unique source IP addresses
grouped_data = filtered_data.groupby('Destination address')['Source address'].nunique().reset_index()

Sort the data by the number of unique source IP addresses in descending order
sorted_data = grouped_data.sort_values(by='Source address', ascending=False)

Display the top 10 external IP addresses with the most web-browsing connections and the number of unique source IP addresses for each
top_external_ips = sorted_data.head(10)
print(top_external_ips[['Destination address', 'Source address']])
```

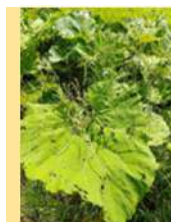
| Destination address | Source address |
|---------------------|----------------|
| 84.1.1.1            | 2              |

Рис. 4.38. Отриманий від chatGPT програмний код



У цьому випадку отримали лише одну IP-адресу, і це досить підозріло. У розслідуванні інцидентів такого типу нам потрібно детально проаналізувати той факт, що лише дві вихідні IP-адреси встановлюють HTTP-з'єднання в неробочий час.

#### 4.5.6. Спрямувати штучний інтелект у правильному напрямі



Можна вказати штучному інтелекту правильний шлях досягнення бажаного результату. У деяких випадках чітко розуміємо шлях, яким слід йти, щоб досягти правильної мети, але не хочемо витрачати час на пошуки функцій Python, які приведуть нас до відповідного результату, тобто знаємо що, але не як, і для цього можна використовувати chatGPT, щоб допомогти нам.

**Question: We want to find external IP addresses suspected to be command and control consoles, so we are going to look for repetitive connections throughout the day between two IP addresses, follow these steps:**

- Filter for connections to external IP addresses.
- Extract the time of the connection without taking into account the minutes.
- Group the IP connections by source and destination.
- During the above grouping it counts how many unique values exist in the connection times, the sum of the bits sent to the destination IP and counts the number of connections made between these two IP addresses.

Результат запиту наведено на рис. 4.39.

```
Filter for connections to external IP addresses
external_ips = data[(~data['Destination address'].str.startswith('10.')) & (~data['Destination address'].str.startswith('172.')) & (~data['Destination address'].str.startswith('192.'))]

Extract the time of the connection without taking into account the minutes
data['Connection Time'] = pd.to_datetime(data['Receive Time']).dt.floor('H')

Group the IP connections by source and destination
grouped_data = data.groupby(['Source address', 'Destination address', 'Connection Time']).agg({'Bytes Sent': 'sum', 'Repeat Count': 'count'}).reset_index()

Count how many unique values exist in the connection times, the sum of the bits sent to the destination IP and counts the number of connections made between these two IP addresses
grouped_data = grouped_data.groupby(['Source address', 'Destination address']).agg({'Connection Time': 'nunique', 'Bytes Sent': 'sum', 'Repeat Count': 'sum'}).reset_index()

Filter for repetitive connections throughout the day between two IP addresses
grouped_data = grouped_data[grouped_data['Connection Time'] > 1]

Display the results in a table sorted in descending order by the number of connection hours
grouped_data = grouped_data.sort_values(by=['Connection Time'], ascending=False)
display(grouped_data)
```

**Рис. 4.39.** Отриманий від chatGPT програмний код

Коли виконуємо цей код, отримуємо такий результат (рис. 4.40).

|     | Source address | Destination address | Connection Time | Bytes Sent | Repeat Count |
|-----|----------------|---------------------|-----------------|------------|--------------|
| 356 | 172.16.202.123 | 84.8                | 13              | 237042     | 3039         |
| 372 | 172.16.202.14  | 84.8                | 13              | 237198     | 3041         |
| 378 | 172.16.202.2   | 84.8                | 13              | 228696     | 2932         |
| 377 | 172.16.202.19  | 84.8                | 13              | 234312     | 3004         |
| 376 | 172.16.202.18  | 84.8                | 13              | 228852     | 2934         |
| *** | ***            | ***                 | ***             | ***        | ***          |
| 589 | 172.16.212.46  | 84.8                | 2               | 9380       | 134          |
| 587 | 172.16.212.45  | 84.8                | 2               | 15960      | 228          |
| 586 | 172.16.212.45  | 84.8                | 2               | 115398     | 26           |
| 576 | 172.16.212.31  | 84.8                | 2               | 10671      | 2            |
| 1   | 172.16.10.13   | 84.8                | 2               | 21549      | 21           |

**Рис. 4.40.** Результат виконання програмного коду

Отримали IP-адреси, які повинні проаналізувати, оскільки факт наявності тисяч з'єднань між двома IP-адресами у 13-ти різних періодів дня (які є в журналі) достатньо підозріло, і це може бути потенційно небезпечним. Шукаємо довгі з'єднання в трафіку. Цей тип з'єднань зазвичай треба аналізувати, оскільки все більше зломисників використовують інструменти віддаленої допомоги, такі як TeamViewer, щоб уникнути виявлення та зберегти доступ до мережі організації. Для цього скористаємося таким запитанням:

**Question: We want to create a graph of long connections between an internal IP address and an external IP address by following the steps below.**

- Filter the communications to external IP addresses.
- Sum the connection time to each destination IP and store it in a separate column called 'sum\_time'.
- Add the number of connections made to each destination IP and store it in a separate column 'sum\_conn'.
- Filter and keep the results with 'sum\_conn' greater than 10.
- Keep only one row for each destination IP address.
- Divide the number of connections by the sum of connections and store it in a column called 'avg\_conn'.
- Filter out the ten results with the highest 'avg\_conn'.
- Create a graph with matplotlib where on the 'x' axis is the number of connections and on the 'y' axis is the total connection time.



Це відповідь chatGPT (рис. 4.41):

```
import pandas as pd
import matplotlib.pyplot as plt

Припустимо, що у вас є DataFrame df з колонками 'external_ip', 'internal_ip',
'connection_time'

Фільтрація повідомлень за зовнішніми IP-адресами
external_connections = df[df['external_ip'].notna()]

Сума часу підключення до кожної IP-адреси призначення
external_connections['sum_time'] = external_connections.groupby('external_ip')
['connection_time'].transform('sum')

Кількість підключень до кожної IP-адреси призначення
external_connections['sum_conn'] = external_connections.groupby('external_ip')
['connection_time'].transform('count')

Фільтрація результатів, де 'sum_conn' більше 10
filtered_connections = external_connections[external_connections['sum_conn'] > 10]

Збереження лише одного рядка для кожної IP-адреси призначення
unique_connections = filtered_connections.drop_duplicates(subset='external_ip')

Розділення кількості з'єднань на суму з'єднань
unique_connections['avg_conn'] = unique_connections['sum_conn'] /
unique_connections['sum_time']

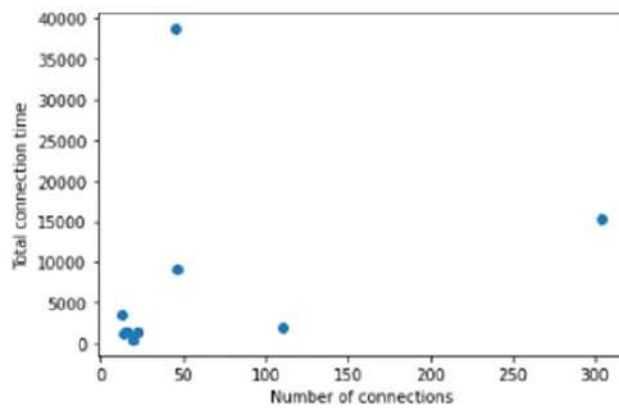
Відфільтрування десяти результатів з найвищим 'avg_conn'
top_connections = unique_connections.nlargest(10, 'avg_conn')

Створення графіка
plt.figure(figsize=(10, 6))
plt.bar(top_connections['external_ip'], top_connections['sum_time'])
plt.xlabel('IP-адреса призначення')
plt.ylabel('Загальний час з\'єднання')
plt.title('Графік довгих з\'єднань')
plt.xticks(rotation=45)
plt.show()
```

**Рис. 4.41.** Отриманий від chatGPT програмний код

Після виконання коду, матимемо такий результат, як показано на рис. 4.42.

Як можна побачити на рис. 4.42, перший результат із середньою тривалістю 840 секунд для кожного з'єднання — це той артефакт, що обов'язково треба проаналізувати.



```
top_results[['Destination address', 'sum_time', 'sum_conn', 'avg_conn']]
```

|         | Destination address | sum_time | sum_conn | avg_conn   |
|---------|---------------------|----------|----------|------------|
| 293528  | 31                  | 38663    | 46       | 840.500000 |
| 1364052 | 216.5               | 3504     | 13       | 269.538462 |
| 167133  | 3                   | 9039     | 47       | 192.319149 |
| 1311770 | 142.250             | 1412     | 15       | 94.133333  |
| 2509298 | 31                  | 1415     | 16       | 88.437500  |
| 2380559 | 142.250             | 1200     | 14       | 85.714286  |
| 2172240 | 3                   | 1336     | 23       | 58.086957  |
| 112505  | 31                  | 15286    | 304      | 50.282895  |
| 2166816 | 31                  | 411      | 20       | 20.550000  |
| 115592  | 3                   | 2026     | 111      | 18.252252  |

**Рис. 4.42.** Результат виконання програмного коду

**Основні інструменти.** Jupyter Notebook — додаток із відкритим вихідним кодом, в якому можна відразу побачити результат виконання коду (у цьому плані це не звичне для всіх IDE). Основна мова Jupyter Notebook — Python. <https://jupyter.org/>. ChatGPT — чат-бот зі штучним інтелектом, розроблений лабораторією OpenAI: <https://chat.openai.com/>



**Висновок.** Хоча факт використання Jupyter Notebooks полегшує працю аналітика кібербезпеки і робить її дуже зручною. Існує можливість створити невеликий програмний інтерфейс, щоб ще більше абстрагувати аналітика від коду і зробити його більш інтуїтивно зрозумілим.

---

## ТЕРМІНОЛОГІЧНИЙ СЛОВНИК

---

**Агломеративна кластеризація** — методологія кластеризації, яка спочатку призначає кожен точку окремому кластеру, а потім поступово об'єднує кластери на основі відстані між кластерами.

**Аналіз даних** — це збір, перетворення та організація даних для висновків і прогнозів. Простіше кажучи, аналітика даних перетворює необроблені дані на корисну інформацію, яка сприяє прийняттю обґрунтованих галузевих рішень.

**Аналіз трафіку** — аналіз сукупності повідомлень шифрованих, які передані по системі зв'язку, що не приводить до дешифрування, але дозволяє противнику і / або порушнику отримати непряму інформацію про повідомлення відкритих і в цілому про функціонування системи зв'язку, яка досліджується. Використовує особливості оформлення повідомлень шифрованих, їх довжину, час передачі, дані про відправника та одержувача тощо.

**Архітектура даних** — схема для керування даними та організація життєвого циклу даних від збору до видалення. Фінансові установи покладаються на надійну архітектуру даних, щоб безпечно керувати конфіденційною інформацією клієнтів, забезпечуючи дотримання правил захисту даних.

**А/В-тестування** — спосіб порівняння двох версій чогось, щоб дізнатися, яка версія працює краще.

**Атрибут** — це дескриптор для стовпця в електронній таблиці або базі даних, що використовується в DNA. Атрибути мають життєво важливе значення для фінансового аналізу, оскільки вони дають змогу професіоналам аналізувати та порівнювати різні аспекти ефективності компанії шляхом маркування та класифікації фінансових даних.

**Аутентифікація джерела даних** — перевірка і підтвердження того, що набір даних (повідомлення, документ) був створений саме заявленим джерелом. Не треба плутати з аутентифікацією відправника, так як він міг передати документ, створений і підписаний іншою особою. Передбачає перевірку цілісності, так як якщо дані зазнали модифікації, то вони вже мають інше джерело.

**Аутентифікація повідомлення** — перевірка того, що повідомлення було отримано неушкодженим, незмінним (з моменту відправлення), тобто перевірка цілісності. Для сторін, які не довіряють одна одній, необхідно використовувати систему цифрового підпису.

**База даних** — зберігає дані в електронному структурованому вигляді, полегшуючи ефективне сортування, пошук і оновлення.

**Байєсовська статистика** — потужний метод аналізу та інтерпретації даних у сучасній аналітиці даних; також називають байєсівським висновком.

**Биграмма** — пара букв (символів), блоків тексту.

**Безперервні змінні** — змінні, які можуть приймати нескінченний і незліченний набір значень.

**Бізнес-аналітика** — аналізує дані за допомогою статистичних і кількісних методів для прийняття рішень і покращення бізнес-результатів. Телекомунікаційні компанії використовують бізнес-аналітику для аналізу даних про виклики клієнтів, визначення тенденцій і покращення якості послуг.

**Business Intelligence (BI)** забезпечує практичну інформацію, об'єднуючи дані, програмне забезпечення, інфраструктуру та людську інтуїцію. Це дозволяє організаціям задавати запитання простою мовою та отримувати зрозумілі відповіді.

**Блок тексту** — мультиграма тексту (тексту відкритого, тексту шифрованого або проміжного), складена зі знаків розташованих поспіль. Зазвичай текст розбивається на блоки однакової довжини.

**Бойові дії в кіберпросторі** — кібератаки, що проводяться державами (групами держав, організованими політичними групами), проти кіберінфраструктур і є частиною військової кампанії.

**Брандмауер (міжмережевий екран)** — комплекс апаратних і програмних засобів, що захищають комп'ютерну мережу від певних атак ззовні.

**Вага функції булевої** — число двійкових наборів в табличному завданні булевої функції, на яких функція приймає значення «1».

**Відкриті дані** — це тип загальнодоступних даних, що сприяє прозорості та заохочує спільний аналіз. Технологічні новатори використовують відкриті дані для розробки додатків, які вирішують суспільні виклики, від оптимізації транспорту до рішень у сфері охорони здоров'я.

**Вірогідність** — ймовірність спостереження фактичних даних за певним набором бета-параметрів.

**Вибірка** — сегмент сукупності, який є репрезентативним для всієї сукупності.

**Видобуток даних** — інтелектуальний аналіз даних ретельно вивчає дані, щоб виявити приховані ідеї та зв'язки. Платформи електронної комерції використовують інтелектуальний аналіз даних, щоб рекомендувати продукти на основі поведінки користувачів, покращуючи клієнтський досвід покупки.

**Випадкове число** — генератор випадкових (псевдовипадкових) чисел є складовою частиною багатьох криптосистем. З його допомогою можна створювати унікальні ключі, які неможливо обчислити заздалегідь. Найбільш досконалі генератори випадкових чисел засновані на даних, отриманих з аналогових джерел, і, як правило, вимагають використання спеціального обладнання.

**Викиди** — спостереження, які мають ненормальну відстань від значень інших даних або загальної моделі сукупності даних.

**Використання переваг в кіберпросторі** — використання в своїх інтересах наявних можливостей в кіберпросторі для досягнення поставленої мети.

**Візуалізація даних** — перетворення складних наборів даних у діаграми та графіки для кращого розуміння. Наприклад, службовці охорони здоров'я використовували візуалізацію даних, щоб повідомляти про тенденції щодо COVID-19, допомагаючи громадськості розуміти складні дані та приймати обґрунтовані рішення.

**Гіпотеза** — теорія або пояснення, засноване на доказах, яке ще не доведено.

**Глобальні викиди** — значення, які повністю відрізняються від загальної групи даних і не пов'язані з будь-якими іншими викидами.

**Data Science** — наука про дані поєднує статистику, інформатику та математику, щоб отримати розуміння тенденцій. Наприклад, організації використовують науку про дані для аналізу екологічних даних, керуючи зусиллями щодо збереження та розробки політики.

**DataFrame** — двовимірна позначена структура даних із рядками та стовпцями.

**Діагностична аналітика** — це виявлення даних і дослідження, «чому щось сталося», розкриваючи причини, що стоять за спостережуваними тенденціями. Відділи кадрів використовують діагностичну аналітику для виявлення та усунення факторів, що впливають на утримання співробітників, що сприяє більш ефективному управлінню талантами.

**Дисперсійний аналіз (ANOVA)** — група статистичних методів, які перевіряють різницю середніх між трьома або більше групами.

**Групування** — групування безперервних значень у меншу кількість категорій або інтервалів.

**Журнал змін** — це хронологічний запис кроків, зроблених під час маніпулювання даними, і це особиста історія даних, яка зберігає кожну внесену зміну. Він є незамінним в аудиті та комплаєнсі, забезпечуючи прозорість і відстежуваність фінансових операцій.

**Запит** — це запит конкретної інформації з бази даних, який забезпечує цілеспрямований пошук даних. Команди обслуговування клієнтів використовують запити для швидкого доступу до відповідної інформації про клієнтів, підвищуючи ефективність вирішення проблем.

**Збагачення даних** — процес, що передбачає додавання відповідних даних до набору для підвищення глибини та якості аналізу. Маркетингові команди використовують збагачення даних, щоб персоналізувати кампанії, забезпечуючи цільову та релевантну комунікацію з різними сегментами клієнтів.

**Динамічна система** — математична абстракція, призначена для опису і вивчення систем, що еволюціонують з часом. Наприклад, будинок з сонячним підігрівом.

- Довірчий інтервал** — діапазон значень, що описує невизначеність оцінки.
- Достовірність** — показує рівень конфіденційності, яку має ключ конкретного користувача.
- Екстраполяція** — здатність моделі передбачати нові значення, які виходять за межі діапазону значень у навчальних даних.
- Ентропія** — теоретико-інформаційна характеристика розподілу випадкової величини. Ентропія (по К.Шеннона) дискретної випадкової величини  $S$  з розподілом  $(p_1, \dots, p_n, \dots)$  дорівнює  $H(S) = - \sum p_i \cdot \log(p_i)$ .
- Ентропія алгоритмічна** — введена А. Н. Колмогоровим міра кількості інформації, необхідної для опису кінцевого об'єкта. Під е. а. двійкового слова розуміють складність цього слова відносно оптимального способу його опису. Поняття алгоритмічної ентропії пов'язане з поняттям ентропії випадкової величини (по К.Шеннону).
- Етика даних** — добре обґрунтовані стандарти правильного та неправильного, які визначають, як дані збираються, передаються та використовуються.
- Фільтрування** — процес вибору меншої частини набору даних на основі заданих значень і використання її для перегляду чи аналізу.
- Файл JSON** — файл для зберігання даних, який зберігається у форматі JavaScript
- Ідентифікація** — це процес побудови математичних моделей за експериментальними даними, а також на основі деякої апріорної інформації з точністю, яка задовольняє вибраному критерію ідентифікації.
- Ідентифікація користувача** — процедура встановлення присвоєного цій стороні унікального системного імені — ідентифікатора, яке дозволяє відрізнити від інших сторін. Зазвичай процедура ідентифікації полягає в пред'явленні цього імені і передуює процедурі аутентифікації, тобто підтвердження правильності ідентифікації. Термін часто для стислості використовують для позначення загальної процедури ідентифікації / аутентифікації сторін.
- Інженерія даних** — це архітектура доступності, створення систем і перетворення необроблених даних у корисну інформацію. Транспортні компанії використовують інженерію даних для створення систем оптимізації маршрутів, зменшення споживання палива та підвищення ефективності доставки.
- Інференційна аналітика** — вибирає репрезентативну вибірку, застосовує статистичні методи та робить прогнози щодо ширшого набору даних. На відміну від описової аналітики, яка просто описує закономірності в межах наявних даних, інференційна аналітика використовує теорію ймовірностей для здійснення обґрунтованих екстраполяцій.
- Інформаційна панель** — це візуальний командний центр, який динамічно представляє реальні дані за допомогою інтерактивних візуалізацій. Керів-

ники покладаються на інформаційні панелі для прийняття стратегічних рішень, моніторингу ключових показників ефективності та коригування маркетингових стратегій.

**Інформаційна перевага** — володіння інформацією кращої якості, або в більшому обсязі.

**Інформаційний простір** — будь-яке середовище, в якому інформація створюється, через яке передається, приймається, в якому зберігається, обробляється та знищується.

**Інформаційна операція** — організована діяльність із збирання і накопичення, підготовки, поширення, обмеження в доступі, або обробці інформації для досягнення поставленої мети.

**Інформаційна війна** — це найвищий ступінь інформаційного конфлікту між державами, коли інформаційні операції проводяться державними структурами для досягнення військово-політичних цілей.

**Інформаційний конфлікт** — напружена ситуація між державами або організованими групами, в якій проведення інформаційних операцій призводить до дій у відповідь.

**Інформаційна безпека** — властивість інформаційного простору протистояти загрозам, реагувати на них і відновлюватися (після нанесення збитку).

**K-means** — неконтрольований алгоритм поділу, який використовується для організації немаркованих даних у групи або кластери.

**Категоричні дані** — дані, які розділені на обмежену кількість якісних груп.

**Категоріальні змінні** — змінні, які містять кінцеву кількість груп або категорій.

**Кіберпростір** — середовище (віртуальний простір), яке надає можливості для здійснення комунікацій та/або реалізації суспільних відносин, утворене в результаті функціонування сумісних (з'єднаних) комунікаційних систем та забезпечення електронних комунікацій з використанням мережі Інтернет та/або інших глобальних мереж передачі даних (або електронне середовище, в якому інформація створюється, передається, приймається, зберігається, обробляється та знищується).

**Кіберінфраструктура** — сукупність людей, процесів (в тому числі керуючих), та систем, які створюють кіберпростір.

**Кіберсервіси** (послуги, служби) — різні види обміну даними в кіберпросторі для прямої чи уявної користі людині.

**Кібероб'єкт** — будь-якій індивідуальний об'єкт або суб'єкт, що існує в кіберструктурі.

**Кіберактив** — кібероб'єкт (кіберсуб'єкт), що має цінність.

**Кіберсили** — кіберактиви, створені для проведення операцій.



**Кібербосць** — людина, яка має спеціальні навички і безпосередньо втягнутий в кібервійну.

**Кіберзлочин** — суспільно небезпечне винне діяння у кіберпросторі та/або з його використанням, відповідальність за яке передбачена законом України про кримінальну відповідальність та/або яке визнано злочином міжнародними договорами України (або використання кіберпростору із злочинними намірами, які визначаються в якості таких національним або міжнародним законодавством).

**Кібертероризм** — терористична діяльність, що здійснюється у кіберпросторі або з його використанням (або використання кіберпростору в терористичних цілях, які визначаються в якості таких національним або міжнародним законодавством).

**Кіберконфлікт** — напружена ситуація між та/або серед держав та/або політично організованих груп, при якій ворожі (небажані) кібератаки провокують (призводять) до відповідних дій.

**Кібервійна** — найвища ступінь кіберконфлікту між або серед держав, під час якої держави вживають кібератаки проти кіберінфраструктур противника, як частина військової компанії: може бути об'явлена формально однією (усіма) конфліктуючими сторонами, або не об'являтися формально і *de facto*.

**Кібербезпека** — захищеність життєво важливих інтересів людини і громадянина, суспільства та держави під час використання кіберпростору, за якої забезпечуються сталий розвиток інформаційного суспільства та цифрового комунікативного середовища, своєчасне виявлення, запобігання і нейтралізація реальних і потенційних загроз національній безпеці України у кіберпросторі (або властивість кіберпростору (кіберсистеми) протистояти навмисним та/або ненавмисним загрозам, а також реагувати на них та відновлюватися після дії цих загроз).

**Кіберпроникнення** — неавторизований доступ до кібероб'єкту (кіберсуб'єкту).

**Кіберзагроза** — наявні та потенційно можливі явища і чинники, що створюють небезпеку життєво важливим національним інтересам України у кіберпросторі, справляють негативний вплив на стан кібербезпеки держави, кібербезпеку та кіберзахист її об'єктів (або виявлена або встановлена загроза використання кібервразливості).

**Кіберексфільтрація** — тип кібероперацій, пов'язаний з копіюванням або вилученням будь-яких даних.

**Кібершпionаж** — кібероперація по отриманню несанкціонованого доступу до чутливої інформації прихованими методами.

**Кібероперація** — організована діяльність в кіберпросторі зі збору та накопичення, підготовці, поширенню, обмеженню в доступі, обробці інформації для досягнення поставленої мети.

**Кібератака** — спрямовані (навмисні) дії в кіберпросторі, які здійснюються за допомогою засобів електронних комунікацій (включаючи інформаційно-комунікаційні технології, програмні, програмно-апаратні засоби, інші технічні та технологічні засоби і обладнання) та спрямовані на досягнення однієї або сукупності таких цілей: порушення конфіденційності, цілісності, доступності електронних інформаційних ресурсів, що обробляються (передаються, зберігаються) в комунікаційних та/або технологічних системах, отримання несанкціонованого доступу до таких ресурсів; порушення безпеки, сталого, надійного та штатного режиму функціонування комунікаційних та/або технологічних систем; використання комунікаційної системи, її ресурсів та засобів електронних комунікацій для здійснення кібератак на інші об'єкти кіберзахисту (або наступальне використання кіберзброї з метою нанесення шкоди певної мети).

**Кіберконтратака** — використання кіберзброї з метою нанесення шкоди певної мети у відповідь на атаку.

**Кібероборона** — сукупність політичних, економічних, соціальних, військових, наукових, науково-технічних, інформаційних, правових, організаційних та інших заходів, які здійснюються в кіберпросторі та спрямовані на забезпечення захисту суверенітету та обороноздатності держави, запобігання виникненню збройного конфлікту та відсіч збройній агресії (або організована сукупність засобів і дій для захисту, пом'якшення та ефективного відновлення від ворожих впливів кібератак).

**Кіберзброя** — програмне, апаратне забезпечення, або прошивки мікросхем, які розроблені або застосовуються для нанесення шкоди в кіберсфері.

**Кібервразливість** — властивість кібероб'єкта, яка потенційно може бути використана для проведення кібероперацій.

**Кіберрозвідка** — діяльність, що здійснюється розвідувальними органами у кіберпросторі або з його використанням; збір і обробка цінної інформації з використанням кібероперацій; збір цінної інформації про кіберактиви іншого суб'єкта / об'єкта.

**Кількісні дані** — такі, що забезпечують вимірні значення, що дозволяє проводити статистичний аналіз і об'єктивні порівняння. Економічні аналітики використовують кількісні дані для вимірювання рівня інфляції, безробіття та інших ключових економічних показників.

**Кореляція** — вимірює те, як дві змінні мають тенденцію змінюватися разом.

**Лінійна регресія** — метод, який оцінює лінійний зв'язок між безперервною залежною змінною та однією або кількома незалежними змінними.

**Логістична регресія** — техніка, яка моделює категоріальну залежну змінну (Y) на основі однієї або кількох незалежних змінних (X).

**Логічні дані** — тип даних, який має лише два можливі значення, як правило, істинне або хибне.

- Математична модель** — сукупність математичних об'єктів (рівнянь, систем рівнянь і нерівностей, алгебраїчних виразів і т. д.), математичних символів, що описують мовою, досліджуваний об'єкт і його відносини з навколишнім світом.
- Машинне навчання** — використання та розробка алгоритмів і статистичних моделей для навчання комп'ютерних систем аналізу та виявлення закономірностей у даних.
- Markdown** — мова розмітки, яка дозволяє користувачеві писати форматований текст у середовищі кодування або редакторі звичайного тексту
- Matplotlib** — бібліотека для створення статичних, анімованих та інтерактивних візуалізацій у Python.
- Медіана** — середнє значення в наборі даних.
- Метрики** — методи та критерії, що використовуються для оцінки даних.
- Міри** — числові значення, які можна агрегувати або використовувати в обчисленнях.
- Менеджер аналітичної групи** — фахівець із обробки даних, який контролює аналітичну стратегію організації, часто керуючи кількома групами.
- Метадані** — надають додаткову інформацію про дані, наприклад їх походження, структуру або дату створення. Бібліотеки використовують метадані для класифікації книг, допомагаючи відвідувачам ефективно знаходити певні назви.
- Множинна лінійна регресія** — метод, який оцінює зв'язок між однією безперервною залежною змінною та двома чи більше незалежними змінними.
- Модель відкритого тексту** — математична модель, що описує властивості реальних відкритих текстів, які виробляються певними джерелами, або природними (осмислений текст на якійсь мові), або штучними (міжмашинний обмін, телеметрія та ін.). Найпростішими є послідовність незалежних випробувань і ланцюг Маркова.
- Наївний Байєс** — методика контрольованої класифікації, яка базується на теоремі Байєса з припущенням незалежності провісників.
- Неструктуровані дані** — це тип даних, який потребує більш чіткої організації та додаткових зусиль для ефективного аналізу. Фахівці з права орієнтуються в неструктурованих даних у юридичних документах, витягаючи відповідну інформацію для аналізу справи та формулювання стратегії.
- NumPy** — бібліотека Python, яка містить багатовимірні масиви та матричні структури даних і функції для роботи з ними.
- Описова аналітика** — є описом історичних даних і відповідає на питання «те, що сталося», перетворюючи необроблені дані на доступні візуальні оповіді. Підприємства використовують описову аналітику у звітах про продажі,

щоб зрозуміти минулу ефективність, визначити успішні стратегії та області для вдосконалення.

**Очищення даних** — забезпечує точність даних шляхом прискіпливої перевірки надійності даних і відсутності в них суперечностей. Платформи електронної комерції залежать від очищення даних, щоб підтримувати точні записи про запаси, запобігаючи помилкам у виконанні замовлень і невдоволенню клієнтів.

**Оцінка силуету** — середнє значення коефіцієнтів силуету всіх спостережень у моделі.

**Пакетна обробка** — означає обробку великих обсягів даних одночасно в періоди низької потреби в обчислювальних ресурсах. Платформи електронної комерції використовують пакетну обробку для виконання замовлень, оновлення рівня запасів і масової обробки транзакцій для оптимізації операцій.

**Перевірка гіпотези** — статистична процедура, яка використовує вибірккові дані для оцінки припущення щодо параметра сукупності.

**Прескриптивна аналітика** — рекомендує, «як діяти» на основі використання моделей машинного навчання, моделювання та інших методів. Наприклад, постачальники медичних послуг використовують її, щоб оптимізувати плани лікування, забезпечуючи персоналізований та ефективний догляд за пацієнтами.

**Прогностична аналітика** — передбачає майбутні тенденції за допомогою статистичних алгоритмів і машинного навчання. Фінансові установи використовують прогностичну аналітику, щоб оцінити кредитний ризик, передбачити потенційні дефолти та прийняти рішення про кредитування.

**Причинно-наслідковий зв'язок** — описує зв'язок, коли одна змінна безпосередньо спричиняє певні зміни іншої.

**Приватна інформація** — інформація, яка дозволяє встановити особу особи прямими чи непрямыми засобами.

**Pandas** — бібліотека Python, побудована на основі NumPy, яка використовується для обробки та аналізу табличних даних.

**Python** — мова програмування загального призначення.

**Рекомендаційні системи** — методи неконтрольованого навчання, які використовують немарковані дані, щоб запропонувати відповідні пропозиції користувачам.

**Реляційна база даних** — база даних, яка зберігає пов'язану інформацію в окремих таблицях, та забезпечує ефективний пошук даних за допомогою взаємопов'язаних запитів.

**Розвідувальний аналіз даних (EDA)** — процес дослідження, організації та аналізу наборів даних і узагальнення їхніх основних характеристик, часто

за допомогою методів суперечки та візуалізації даних; шість основних практик EDA: виявлення, структурування, очищення, приєднання, перевірка та представлення.

**Словник** — структура даних, яка складається з набору пар ключ-значення.

**Спорідненість** — метрика, яка використовується для обчислення відстані між точками/кластерами

**Структуровані дані** — такі, що впорядковують інформацію в рядки та стовпці, полегшуючи простий аналіз. Фінансові аналітики покладаються на структуровані дані для порівняння квартальних показників продажів, забезпечуючи чітку основу для оцінки ефективності.

**Структурована мова запитів (SQL)** — це мова програмування для керування реляційними базами даних і забезпечення безперебійної взаємодії. Адміністратори баз даних використовують SQL для керування та оптимізації продуктивності бази даних, забезпечуючи ефективне зберігання та пошук даних.

**Структура даних** — сукупність значень даних або об'єктів, які містять різні типи даних.

**Силуетний аналіз** — порівняння балів силуету різних моделей.

**Суперечність даних** — обробка даних охоплює етапи виявлення, перетворення, перевірки та публікації, гарантуючи готовність необроблених даних до аналізу. Дослідники ринку витрачають багато часу на обговорення даних, щоб підготувати дані опитування для аналізу, витягаючи значущі ідеї з різноманітних відповідей.

**Seaborn** — бібліотека візуалізації на основі matplotlib, яка забезпечує простіший інтерфейс для роботи зі звичайними діаграмами та графіками.

**Табличні дані** — дані у формі таблиці з рядками та стовпцями.

**Теорема Байєса** — рівняння, яке можна використовувати для обчислення ймовірності результату або класу, враховуючи значення змінних предиктора.

**Тип даних** — атрибут, який описує фрагмент даних на основі його значень, мови програмування або операцій, які він може виконувати.

**Файл CSV (значення, розділені комами)** — розділяє значення комами та забезпечує сумісність із різними програмними засобами. Він оптимізує обмін даними у фінансовому аналізі, забезпечуючи плавний імпорт даних в інструменти бюджетування або економічні моделі.

**Чисті дані** — золотий стандарт точності та повноти, а також канва, на якій аналітики даних малюють глибокі оповіді. У наборі даних на основі опитувань, чисті дані гарантують, що відповіді є послідовними, безпомилковими та готовими для змістовного аналізу.

**Управління даними** — встановлює правила доступу до даних, використання та підзвітності. Організації охорони здоров'я використовують управління

даними, щоб забезпечити дотримання правил конфіденційності, захищаючи дані пацієнтів від несанкціонованого доступу.

**Цілісність даних** — захист даних від несанкціонованих змін і забезпечення узгодженості даних з часом. Урядові установи покладаються на цілісність даних у базах даних перепису, забезпечуючи точний підрахунок населення для розподілу ресурсів і планування політики.

**Штучний інтелект (ШІ)** — відноситься до комп'ютерних систем, здатних виконувати завдання, які зазвичай потребують людського інтелекту.

**Якісні дані** — такі, що описують нечислові якості, часто охоплюючи суб'єктивні елементи, думки чи вподобання. Дослідники в галузі освіти збирають якісні дані за допомогою інтерв'ю та фокус-груп, щоб дослідити сприйняття та досвід студентів.

**Ящик чорний** — кінцевий автомат, у якого відомі тільки вхідний і вихідний алфавіти та доступні для спостереження вихідні послідовності при довільних вхідних послідовностях. Якщо додатково відома оцінка числа станів автомата, то говорять про ящик чорний відносний.

---

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

---

1. Ainsworth, M. B. (2017). "Artificial Intelligence for Executives." SAS White Paper, ai20for20executives.pdf, October 2018.
2. Amri Aymen «OpenAI GPT for Python Developers», published by FAUN, 2023, 219 pp.
3. Andronic, S. (2017). "5 Ways to Use Artificial Intelligence as a Competitive Advantage." Moonoia.com. Anon. (2017, February 20).
4. AHIMA Foundation. (2010). A study of health care fraud and abuse: Implications for professional managing health information. Retrieved September 15, 2018, from <https://www.ahimafoundation.org/downloads/pdfs/Fraud%20and%20Abuse%20-%20final%2011-4-10.pdf>
5. Bagde, P. R., & Chaudhary, M. S. (2016). Analysis of fraud detection mechanism in health insurance using statistical data mining techniques. International Journal of Computer Science and Information Technologies, 7(2), 925–927.
6. "Big Data and Data Sharing: Ethical Issues." UK Data Service. [ukdataservice.ac.uk/media/604711/big-data-and-data-sharing\\_ethical-issues.pdf](http://ukdataservice.ac.uk/media/604711/big-data-and-data-sharing_ethical-issues.pdf).
7. Caelen Olivier, Blete Marie-Alice «Developing Apps with GPT-4 and ChatGPT», 2023, USA, Published by O'Reilly Media, Inc. — 183 pp.
8. Chen, Y., Kang, Y., Chen, Y., & Wang, Z. (2020). "Probabilistic forecasting with temporal convolutional neural network", Neurocomputing, 399: 491-501.
9. Coldrena, G. M., Koppelman, F. S., Kasturirangana, K., & Mukherjee, A. (2003). Modeling aggregate air-travel itinerary shares—Logit model development at a major US airline. Journal of Air Transport Management, 9, 361–369.
10. De Prado, M. L. 2018. Advances in Financial Machine Learning. John Wiley & Sons
11. Delen, D., D. Cogdell, & N. Kasap. (2012). "A Comparative Analysis of Data Mining Methods in Predicting NCAA Bowl Outcomes." International Journal of Forecasting, 28, 543–552.
12. Gallicchio, C., Micheli, A., & Pedrelli, L. 2018. "Design of deep echo state networks", Neural Networks, 108: 33-47.
13. Gallatin Kyle, Albon Chris «Machine Learning with Python Cookbook», 2023, Published by O'Reilly Media, Inc — 414 pp.
14. Goerg, M. (2014). Estimating reach curves from one data point. Google Inc. from <https://ai.google/research/pubs/pub43218>.
15. Hilpisch Yves J. «Derivatives analytics with Python: data analysis, models, simulation, calibration and hedging» / John Wiley & Sons Ltd, 2015, 377 pp.
16. Fabio Nelli «Python Data Analytics: With Pandas, NumPy, and Matplotlib» 2023, USA, Apress Berkeley, CA, — 445 pp.
17. Farhaoui Yousef, Moussaid Laila «Big Data and Smart Digital Environment»/ Springer Nature Switzerland AG 2019, 415 pp.



18. Kazemi, S. M., Goel, R., Eghbali, S., Ramanan, J., Sahota, J., Thakur, S., ... & Brubaker, M. 2019. Time2vec: Learning a vector representation of time. arXiv preprint arXiv:1907.05321.
19. Lea, C., Flynn, M. D., Vidal, R., Reiter, A., & Hager, G. D. 2017. Temporal convolutional networks for action segmentation and detection. In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 156-165.
20. LexisNexis. (2011). Bending the cost curve: Analytics driven enterprise fraud control. from <http://lexisnexis.com/risk/downloads/idm/bending-the-cost-curve-analytics-driven-enterprise-fraud-control.pdf>.
21. Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. 2021. “Temporal fusion transformers for interpretable multi-horizon time series forecasting”, International Journal of Forecasting, 37(4): 1748-1764.
22. Ledoit, O., & Wolf, M. 2003 “Improved estimation of the covariance matrix of stock returns with an application to portfolio selection,” Journal of Empirical Finance, 10(5): 603-621
23. Mittal, V. “Top 15 Deep Learning Applications That Will Rule the World in 2018 and Beyond.” Medium.com, October 3, 2017.
24. Model Persistence (from the scikit-learn user guide): [https://scikit-learn.org/stable/modules/model\\_persistence.html](https://scikit-learn.org/stable/modules/model_persistence.html)
25. Novelty and Outlier Detection (from the scikit-learn user guide): [https://scikit-learn.org/stable/modules/outlier\\_detection.html](https://scikit-learn.org/stable/modules/outlier_detection.html)
26. Naive Bayes (from the scikit-learn user guide): [https://scikit-learn.org/stable/modules/naive\\_bayes.html](https://scikit-learn.org/stable/modules/naive_bayes.html)
27. Narayan, K. “Leverage Artificial Intelligence to Build your Sales Pipeline.” LinkedIn, February 14, 2018.
28. Outlier Detection with Isolation Forest: <https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e>
29. Oreshkin, B. N., Carpow, D., Chapados, N., & Bengio, Y. 2019. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. arXiv preprint arXiv:1905.10437.
30. Passive Aggressive Algorithm (video explanation): <https://www.youtube.com/watch?v=uxGDwyPWNkU>
31. Python Context Managers and the "with" Statement: <https://blog.ramosly.com/python-context-managers-and-the-with-statement-8f53d4d9f87>
32. Seeing Theory — Chapter 5, Bayesian Inference: <https://seeing-theory.brown.edu/index.html#secondPage/chapter5>
33. SQLAlchemy — Python Tutorial: <https://towardsdatascience.com/sqlalchemy-python-tutorial-79a577141a91>
34. Stochastic Gradient Descent (from the scikit-learn user guide): <https://scikit-learn.org/stable/modules/sgd.html>
35. Strategies to scale computationally: bigger data (from the scikit-learn user guide): [https://scikit-learn.org/stable/computing/scaling\\_strategies.html](https://scikit-learn.org/stable/computing/scaling_strategies.html)

36. Simple Beginner's Guide to Reinforcement Learning & its implementation: <https://www.analyticsvidhya.com/blog/2017/01/introduction-to-reinforcement-learning-implementation/>
37. Singhal, S., Finn, P., Schneider, T., Schaudel, F., Bruce, D., & Dash, P. (2016). Global private payors: A trillion-euro growth Industry. New York: McKinsey and Company Retrieved September 15, 2018, from <http://healthcare.mckinsey.com/sites/default/files/global%20private%20payors%20%28updated%29.pdf>.
38. Support Vector Machine — Introduction to Machine Learning Algorithms: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>
39. The 5 Clustering Algorithms Data Scientists Need to Know: <https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68>
40. IPython: Built-in magic commands: <https://ipython.readthedocs.io/en/stable/interactive/magics.html>
41. The Importance of Integrity: How Plot Parameters Influence Interpretation: <https://www.t4g.com/insights/plot-parameters-influence-interpretation/>
42. «5 Python Libraries for Creating Interactive Plots»: <https://mode.com/blog/python-interactive-plot-libraries/>
43. Лук'яненко Д.Г. Парадигма креативного менеджменту в глобальній економіці / за заг. ред. д.е.н., проф. Д. Г. Лук'яненка. — К.: ДВНЗ "КНЕУ", 2016. — 231 с.
44. Лук'яненко Д.Г. Світові фінансові кризи та глобальне регулювання / Управління міжнародною конкурентоспроможністю в умовах глобалізації економічного розвитку: Монографія: У 2 т. — К.: КНЕУ, 2006. — 816 с.
45. Колот А. М., Герасименко О. О. «Праця 4.0» як інститут та платформа розвитку неоекономіки. Неоекономіка та імперативи розвитку підприємництва: Монографія. К.: КНЕУ, 2021. С.122-145.URL: [https://ir.kneu.edu.ua/bitstream/handle/2010/36871/Piprn\\_2021.pdf?sequence=](https://ir.kneu.edu.ua/bitstream/handle/2010/36871/Piprn_2021.pdf?sequence=)
46. Колот А. М., Герасименко О. О. Праця XXI: філософія змін, виклики, вектори розвитку: монографія. Київ : КНЕУ імені Вадима Гетьмана, 2021. 487 с. URL: [https://ir.kneu.edu.ua/bitstream/handle/2010/36870/Pfzvvr\\_2021.pdf?sequence=3&isAllowed=y](https://ir.kneu.edu.ua/bitstream/handle/2010/36870/Pfzvvr_2021.pdf?sequence=3&isAllowed=y)
47. Когут Ю.І. Кібервійни, кібертероризм, кіберзлочинність (концепції, стратегії, технології), К.: Дакор, 2022. 224 с.
48. Irada Dzhalladova and Miroslava Růžicková: Dynamical Systems with Random Structure and their Applications, 10th volume of the International book series Stability, Oscillations and Optimization of Systems, Cambridge Scientific Publishers, UK, 2020. ISBN 978-1-908106-66-7
49. I. Dzhalladova. *Stochastic Mathematics*, K.: KNEU, 2020.

50. DZHALLADOVA, I., RUZICKOVA M., Dynamic system with random structure for modeling security and risk management in cyberspace, Opuscula Math. 39, no. 1 (2019).
51. DZHALLADOVA, I., SKAPA S., NOVOTNA V, Design and analysis of a model for detection of information attacks in computer networks, International Journal of Economics, Finance and Management Sciences ISSUE 3/2019; Vol. 53 .
52. DZHALLADOVA, I., GLADKA Y., Mathematics, K.: KNEU, 2020
53. DZHALLADOVA, I., BATECHKO N., GLADKA Y. System Approach to the Formation of the Information Culture Levels in Cyberspace, Advances in Intelligent Systems and Computing, 2020, 1114 AISC
54. Камінський О.Є. Хмарні технології в парадигмі інформаційної економіки: монографія. Київ: КНЕУ, 2018. 230 с.

*Навчальне видання*

**ДЖАЛЛАДОВА Ірада  
КАМІНСЬКИЙ Олег  
КОЛОТ Анатолій  
КОГУТ Юрій  
ЛУК'ЯНЕНКО Дмитро**

**СМАРТ-АНАЛІТИКА  
ТА БЕЗПЕКА**

Навчальний посібник

Редактор *В. Македон*  
Художник обкладинки *Т. Зябліцева*  
Верстка *Т. Мальчевської*

Підписано до друку 11.07.2024. Формат 70×100/16. Папір офсет.  
Гарнітура Тип Таймс. Друк офсет. Ум.-друк. арк. 21,04.  
Обл.-вид. арк. 22,63. Зам. 24-5831.

Київський національний економічний університет імені Вадима Гетьмана  
03680, м. Київ, Берестейський проспект, 54/1  
E-mail: [litera@kneu.edu.ua](mailto:litera@kneu.edu.ua)

*Для нотаток*

*Для нотаток*