ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «МЕТОДИКА АНАЛІЗУ ТА ОБРОБКИ ДАНИХ ЗАСОБАМИ МАШИННОГО НАВЧАННЯ»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Ігор ШОЛОМ*

Виконав:
здобувач вищої освіти Ігор ШОЛОМ
група ІСДМ-64

Керівник: Оксана ТКАЛЕНКО
науковий ступінь, вчене звання к.т.н., доцент

Рецензент:
науковий ступінь, вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій

Ступінь вищої освіти Магістр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ

_____Каміла СТОРЧАК
«____» _____20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ СТУДЕНТУ
Шолому Ігорю Вікторовичу
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: «Методика аналізу та обробки даних засобами машинного навчання»

керівник кваліфікаційної роботи Оксана ТКАЛЕНКО, к.т.н., доцент
(ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «15» жовтня 2024 року № 320.

2. Строк подання кваліфікаційної роботи: 27 грудня 2024 року.

3. Вихідні дані до кваліфікаційної роботи: датасет Iris;
бібліотеки Python: Sklearn, NumPy;
Random_state = 42;
тип ознак екземплярів датасету Iris – числові;
науково-технічна література з питань, пов'язаних з наукою про дані.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Методи машинного навчання для аналізу та обробки даних.

2. Дослідження напрямів машинного навчання.
3. Аналіз та обробка даних із використанням методів машинного навчання.

5. Перелік ілюстративного матеріалу: *презентація*
   1. Типи машинного навчання.
   2. Складові ML.
   3. Типи даних.
   4. Класичне навчання.
   5. Асоціативні правила.
   6. Відмінності різних типів ML.
   7. Ансамблеві методи.
   8. Структура штучної нейромережі.
   9. Модель ML для обробки даних.

6. Дата видачі завдання: 15 жовтня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	15.10 – 23.10.24	
2	Дослідження методів машинного навчання	24.10 – 01.11.24	
3	Дослідження особливостей глибокого навчання	02.11 – 07.11.24	
4	Дослідження особливостей побудови нейронних мереж	08.11 – 16.11.24	
5	Вивчення особливостей фреймворків TensorFlow, Keras, PyTorch, Theano	17.11 – 21.11.24	
6	Реалізація моделі ML для обробки даних	22.11 – 11.12.24	
7	Оформлення роботи: вступ, висновки, реферат	12.12 – 20.12.24	
8	Розробка демонстраційних матеріалів	21.12 – 25.12.24	

Здобувач вищої освіти _____ Ігор ШОЛОМ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник роботи
кваліфікаційної роботи _____ Оксана ТКАЛЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 73 стор., 28 рис., 7 табл., 24 джерел.

Мета роботи – розробка методики аналізу й обробки даних з використанням засобів машинного навчання для підвищення ефективності прийняття рішень та автоматизації процесів обробки інформації.

Об'єкт дослідження – процеси аналізу та обробки даних в інформаційних системах.

Предмет дослідження – методи, алгоритми та інструменти машинного навчання, що застосовуються для вирішення завдань аналізу, класифікації, прогнозування й оптимізації процесів обробки даних.

Короткий зміст роботи: Досліджені методи машинного навчання для аналізу та обробки даних, типи даних, алгоритми, відмінності різних типів ML. Досліджені особливості глибокого навчання, особливості побудови нейронних мереж, найпопулярніші бібліотеки та фреймворки для роботи з технологіями ML/DL. Здійснено обробку і аналіз даних з використанням засобів ML в інтерактивному середовищі Jupyter Notebook, яке є популярним інструментом серед дослідників, програмістів та аналітиків даних. Обгрунтовано вибір мови програмування та середовища розробки моделі ML. Визначені особливості роботи з датасетом. Здійснено вибір алгоритму навчання моделі ML. Виконано навчання та оцінку ML-моделі, здійснено перевірку точності моделі ML за допомогою тестових даних. Розроблено методику аналізу та обробки даних засобами машинного навчання.

КЛЮЧОВІ СЛОВА: ДАНІ, МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, МОВА ПРОГРАМУВАННЯ PYTHON, ВІЗУАЛІЗАЦІЯ, ТЕХНОЛОГІЯ, АЛГОРИТМ, СОКЕТ, ПРОТОКОЛ, ІНТЕРАКТИВНЕ СЕРЕДОВИЩЕ.

ABSTRACT

Text part of the master`s qualification work: 73 pages, 28 pictures, 7 table, 24 sources.

The purpose of the work is to develop a methodology for analyzing and processing data using machine learning tools to increase the efficiency of decision-making and automate information processing processes.

Object of research is the process of data analysis and processing in information systems.

Subject of research is machine learning methods, algorithms, and tools used to solve problems of analysis, classification, prediction, and optimization of data processing processes.

Summary of the work: Machine learning methods for data analysis and processing, data types, algorithms, differences between different types of ML were studied. The features of deep learning, the features of building neural networks, the most popular libraries and frameworks for working with ML/DL technologies were studied. Data processing and analysis were carried out using ML tools in the interactive environment Jupyter Notebook, which is a popular tool among researchers, programmers and data analysts. The choice of programming language and ML model development environment was justified. The features of working with the dataset were determined. The ML model training algorithm was chosen. The ML model was trained and evaluated, and the accuracy of the ML model was verified using test data. A method for analyzing and processing data using machine learning tools was developed.

KEYWORDS: DATA, MACHINE LEARNING, NEURAL NETWORK, PYTHON PROGRAMMING LANGUAGE, VISUALIZATION, TECHNOLOGY, ALGORITHM, SOCKET, PROTOCOL, INTERACTIVE ENVIRONMENT.

ЗМІСТ

ВСТУП……..8
РОЗДІЛ 1. МЕТОДИ МАШИННОГО НАВЧАННЯ ДЛЯ АНАЛІЗУ ТА ОБРОБКИ ДАНИХ……...11
1.1 Категорії методів машинного навчання для аналізу та обробки даних…….11
1.2 Визначення складових машинного навчання……...16
1.3 Характеристика типів даних та їх особливостей……..21
РОЗДІЛ 2. ДОСЛІДЖЕННЯ НАПРЯМІВ МАШИННОГО НАВЧАННЯ……...28
2.1 Класичне навчання……..28
2.2 Навчання з підкріпленням……..33
2.3 Ансамблеві методи……..39
2.4 Нейронні мережі та глибоке навчання……..45
РОЗДІЛ 3. АНАЛІЗ ТА ОБРОБКА ДАНИХ ІЗ ВИКОРИСТАННЯМ МЕТОДІВ МАШИННОГО НАВЧАННЯ……..54
3.1 Підготовка даних для обробки та аналізу, попередня обробка даних……...54
3.2 Розділення даних на тренувальну і тестову вибірки. Вибір алгоритму для навчання моделі ML……..61
3.3 Навчання та оцінка ML-моделі. Перевірка точності моделі за допомогою тестових даних……...64
ВИСНОВКИ……...69
ПЕРЕЛІК ПОСИЛАНЬ……71
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)……73

ВСТУП

Актуальність теми: У сучасному світі обсяги даних постійно зростають, створюючи виклики для їх ефективного аналізу та обробки. Традиційні методи обробки інформації часто не здатні забезпечити необхідну швидкість, точність та адаптивність для аналізу великих обсягів даних або роботи з неструктурованими даними. Машинне навчання, як ключова складова штучного інтелекту, пропонує потужні інструменти для автоматизації аналізу даних, виявлення закономірностей, прогнозування та оптимізації рішень у різних галузях. Розробка та впровадження методики аналізу й обробки даних із застосуванням алгоритмів машинного навчання є актуальним завданням, так як це дозволить підвищити ефективність використання інформаційних ресурсів, знизити витрати часу і ресурсів на обробку даних та отримати конкурентні переваги в умовах цифрової трансформації. Таким чином, дослідження у цій сфері є важливими як з наукової, так і з практичної точок зору.

Метою роботи є розробка методики аналізу й обробки даних з використанням засобів машинного навчання для підвищення ефективності прийняття рішень та автоматизації процесів обробки інформації.

Для досягнення поставленої мети необхідно виконати наступні *завдання*:

- дослідити методи ML для аналізу та обробки даних;
- дослідити бібліотеки та фреймворки для створення моделі ML;
- здійснити обробку та аналіз даних засобами ML в середовищі Jupyter Notebook;
- вибрати алгоритм для навчання моделі ML;
- виконати оцінку ML-моделі.

Об'єкт дослідження – процеси аналізу та обробки даних в інформаційних системах.

Предметом дослідження є методи, алгоритми та інструменти машинного навчання, що застосовуються для вирішення завдань аналізу, класифікації, прогнозування й оптимізації процесів обробки даних.

Наукова новизна отриманих результатів: запропоновано адаптивний підхід до вибору моделі машинного навчання та її параметрів в залежності від характеристик даних, що аналізуються (розмір, структура, тип даних).

Апробація результатів магістерської роботи:

1. Шолом І. В. «Методи та технології обробки даних в інформаційних системах». Тези доповіді на ІІ Всеукраїнській науково-технічній конференції «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу». – Київ, 18 листопада 2024 року.

2. Шолом І. В. «Використання алгоритмів пошуку на Python». Тези доповіді на ІІ Міжнародній науково-практичній конференції «Сучасні аспекти діджиталізації та інформатизації в програмній та комп’ютерній інженерії». – Київ, 25-27 грудня 2024 року.

1 МЕТОДИ МАШИННОГО НАВЧАННЯ ДЛЯ АНАЛІЗУ ТА ОБРОБКИ ДАНИХ

1.1 Категорії методів машинного навчання для аналізу та обробки даних

Аналіз та обробка даних є важливими кроками у процесі прийняття рішень, розробки моделей машинного навчання та вивчення інформації. Основними причинами для чого це необхідно, є: отримання цінних інсайтів (виявлення патернів та закономірностей, розуміння проблеми); покращення якості даних (виявлення та усунення помилок, нормалізація та масштабування); оптимізація процесу моделювання (зниження вимірності, вибір важливих характеристик); прийняття рішень на основі даних (підтримка бізнесу та стратегій, оптимізація процесів); передбачення та прогнозування (прогнозування майбутніх подій, оцінка ризиків); адаптація до змін (швидке реагування на зміни, визначення нових можливостей). Аналіз і обробка даних є ключовими для отримання корисної інформації, оптимізації процесів, створення ефективних моделей та прийняття обґрунтованих рішень.

Аналіз даних допомагає виявляти приховані закономірності та тенденції у даних, які можуть бути корисними для прийняття рішень або покращення продуктивності бізнесу. Обробка та аналіз допомагають краще зрозуміти дані, виявляти фактори, що впливають на процес або явище та створювати моделі для передбачення або класифікації.

У реальних даних часто зустрічаються помилки, пропуски або некоректні значення. Обробка даних дозволяє очистити їх від аномалій, заповнити пропущені значення та виправити невірні дані. Для багатьох алгоритмів машинного навчання важливо, щоб дані мали певний масштаб або структуру. Нормалізація даних дозволяє зробити їх придатними для подальшого аналізу.

Зменшення кількості змінних або параметрів у моделі може значно підвищити її ефективність та точність. Зниження вимірності також допомагає уникнути перенавчання моделі на невеликих або зашумлених наборах даних. Аналіз

допомагає виділити найбільш значущі фактори, які впливають на результат, що дозволяє створити більш прості та точні моделі.

Дані можуть дати відповіді на важливі питання, такі як потреби клієнтів, ефективність маркетингових компаній, оцінка ризиків тощо. Це дозволяє приймати більш точні та обґрунтовані рішення. Аналіз даних допомагає виявити вузькі місця в процесах та оптимізувати їх, що призводить до зниження витрат і покращення продуктивності.

Моделі машинного навчання, побудовані на основі аналізу даних, можуть допомогти передбачити майбутні події, наприклад, зміни ринкових умов або попит на продукцію. Аналіз даних дозволяє передбачити можливі ризики та підготувати плани для їх мінімізації.

Завдяки постійному аналізу даних компанії та організації можуть оперативно реагувати на зміни у поведінці користувачів, ринку або внутрішніх процесах. Обробка даних дозволяє виявляти нові можливості для розвитку продуктів, послуг або бізнесу, що можуть не бути очевидними без детального аналізу.

Аналіз та обробка даних – це фундаментальний процес у сучасному світі, особливо в епоху цифрової трансформації. Він дозволяє перетворювати великі обсяги інформації на цінні знання, які можуть бути використані для прийняття обґрунтованих рішень, оптимізації процесів та виявлення нових можливостей.

Аналіз даних допомагає виявити закономірності, тренди та кореляції, що дозволяє приймати рішення, які базуються на фактах, а не на здогадках. Виявлення неефективностей та вузьких місць у процесах дозволяє їх покращувати та підвищувати продуктивність. Аналіз даних може виявити нові ринки, сегменти клієнтів або продукти, які можуть принести компанії додатковий прибуток. Аналіз відгуків клієнтів та даних про використання продуктів дозволяє вносити необхідні зміни та покращувати їх якість. Аналіз даних може допомогти виявити потенційні ризики та розробити стратегії їх мінімізації.

Процес аналізу даних включає етапи збору даних, їх очищення, трансформації, аналізу, візуалізації, прийняття рішень.

Процес збору даних включає збір даних з різних джерел (бази даних, веб-сайти, соціальні мережі тощо). Очищення даних забезпечує видалення помилок, дублікатів та іншої непослідовної інформації. Трансформація даних – це перетворення даних у формат, придатний для аналізу. Аналіз даних передбачає використання статистичних методів, машинного навчання та інших інструментів для виявлення закономірностей та отримання відповідей на поставлені питання. Візуалізація даних – це представлення результатів аналізу у зручному для сприйняття вигляді (графіки, діаграми тощо). Прийняття рішень – це використання отриманих знань для прийняття обґрунтованих рішень.

Аналіз та обробка даних є ключовими факторами успіху у будь-якій сфері діяльності, дозволяє приймати більш обґрунтовані рішення, оптимізувати процеси та виявляти нові можливості. Завдяки розвитку технологій, аналіз даних стає все більш доступним і простим у використанні. Ці процеси корисні для застосування у всіх сферах життєдіяльності людини, особливо в таких як маркетинг (сегментація клієнтів, персоналізація маркетингових кампаній, прогнозування продажів), фінанси (виявлення шахрайських операцій, управління ризиками, прогнозування доходів), виробництво (оптимізація виробничих процесів, контроль якості продукції, прогнозування попиту), охорона здоров’я (розробка нових ліків, персоналізована медицина, аналіз медичних зображень).

Методи машинного навчання поділяються на декілька основних категорій залежно від типу даних та задачі, яку вони розв'язують. До основних методів машинного навчання належать: контрольоване навчання (Supervised Learning), неконтрольоване навчання (Unsupervised Learning), навчання з підкріпленням (Reinforcement Learning), напівконтрольоване навчання (Semi-supervised Learning), методи ансамблевого навчання (Ensemble Methods), глибоке навчання (Deep Learning) (рис. 1.1).

У контрольованому навчанні модель навчається на основі розмічених даних, де кожному вхідному значенню відповідає вихідне (цільове) значення. До задач, які розв'язуються таким методом відносяться *задачі класифікації* та *задачі регресії*. Задача ***класифікації*** забезпечує передбачення категорії або класу об'єкту,

наприклад виявлення спаму/не спаму). До алгоритмів, які використовуються для вирішення задачі класифікації, відносяться *логістична регресія, SVM (метод опорних векторів), дерева рішень, нейронні мережі.* Задача ***регресії*** забезпечує передбачення числового значення на основі вхідних даних, наприклад, передбачення ціни будинку. Прикладами алгоритмів для вирішення задачі регресії можуть бути: *лінійна регресія, регресійні дерева, градієнтний бустинг*.

Неконтрольоване навчання використовується для пошуку прихованих патернів або структури в даних без міток (цільових значень). Найпоширенішими задачами неконтрольованого навчання є: *задачі кластеризації*, *задачі зниження розмірності*. Задача ***кластеризації*** забезпечує групування схожих об'єктів, наприклад, сегментація клієнтів. Алгоритми, які застосовуються для вирішення задачі кластеризації: *k-середніх (k-means), ієрархічна кластеризація, DBSCAN*. ***Зниження розмірності*** – це спрощення складних наборів даних із великою кількістю змінних, зберігаючи важливу інформацію, наприклад, візуалізація даних. До алгоритмів вирішення задачі зниження розмірності належать *PCA (головні компоненти аналізу), t-SNE.*

У методі навчання з підкріпленням агент вчиться взаємодіяти з середовищем, отримуючи винагороди або покарання за свої дії, з метою максимізації сукупної винагороди. Використовується, наприклад, у грі в шахи або управлінні роботами. Прикладами алгоритмів для навчання з підкріпленнями є наступні: *Q-learning, Deep Q-Networks (DQN), SARSA.*

Напівконтрольоване навчання - це метод, де використовується велика кількість нерозмічених даних у поєднанні з невеликою кількістю розмічених даних. Цей метод ML ефективний тоді, коли розмічені дані складно або дорого отримати.

Методи ансамблевого навчання поєднують декілька моделей для досягнення кращої продуктивності, ніж одна модель. Приклади методів ансамблевого навчання: *Random Forest, AdaBoost, Gradient Boosting*.

Глибоке навчання є підмножиною машинного навчання, що використовує нейронні мережі з багатьма шарами для моделювання складних патернів у

великих наборах даних. Приклади моделей: *Convolutional Neural Networks (CNN) для зображень, Recurrent Neural Networks (RNN) для часових рядів і тексту.*

Всі ці методи можна застосовувати до різноманітних задач в залежності від типу даних та очікуваного результату.

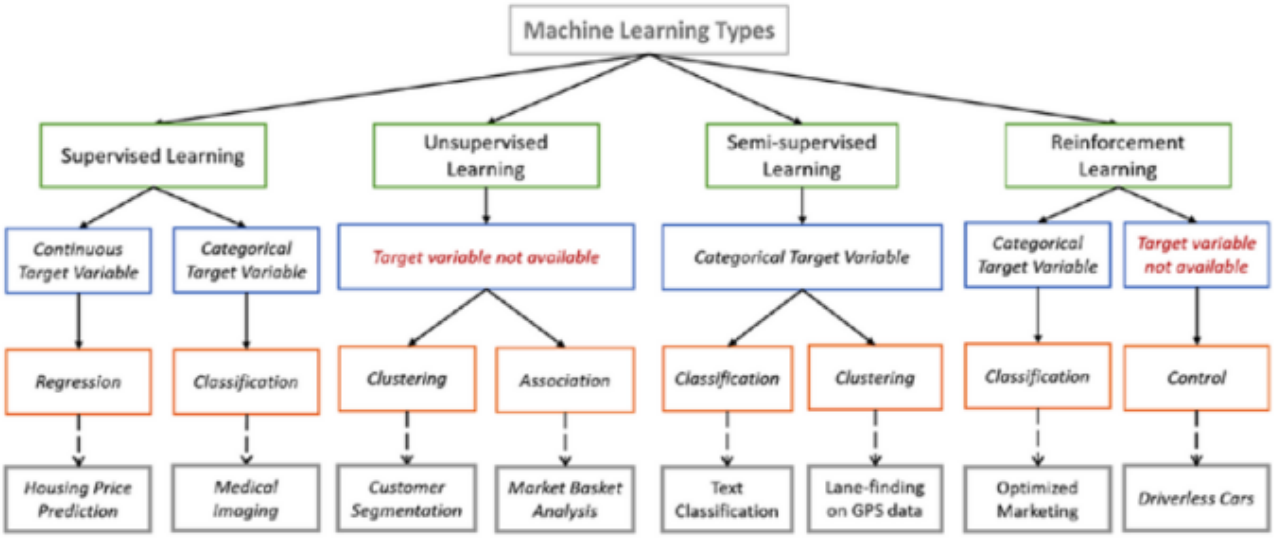

Рис. 1.1 Методи машинного навчання

Дослідження методів машинного навчання є важливим з причин, які мають значний вплив на науку, технології, бізнес і суспільство в цілому. Машинне навчання допомагає знаходити оптимальні рішення у багатьох сферах. Завдяки дослідженню різних методів ML можна вибрати найефективніші підходи для автоматизації складних процесів. Вивчення ML методів дозволяє покращити точність прогнозів і моделювання. Нові методи машинного навчання нададуть можливість відкривати нові закономірності в даних, які раніше були невидимі або складні для аналізу традиційними методами. З розвитком технологій та збільшенням обсягів даних виникають нові виклики, і методи машинного навчання повинні адаптуватися до нових умов. Дослідження допомагає розробляти моделі, які можуть навчатися з нових даних і адаптуватися до змін у середовищі. Бізнес-аналітика і прогнози, які основані на машинному навчанні, допомагають підприємствам приймати більш обґрунтовані рішення. Дослідження методів машинного навчання дозволить створювати моделі, які можуть аналізувати індивідуальні потреби та пропонувати нові, перспективні рішення.

Машинне навчання дозволяє уникнути людської суб'єктивності і помилок, що виникають через стереотипи або обмежені можливості сприйняття інформації. Моделі ML, навчені на великих наборах даних, можуть обробляти інформацію об'єктивно, покращуючи точність рішень.

Завдяки ML можна покращувати взаємодію користувачів з різними системами (наприклад, чат-боти, віртуальні асистенти). Ці моделі постійно самонавчаються на основі зворотного зв'язку, що дозволяє їм ставати все більш точними у відповіді на запити користувачів. У галузі безпеки машинне навчання використовується для виявлення кіберзагроз, шахрайства та інших ризиків у режимі реального часу. Дослідження нових методів допоможе покращити виявлення небезпечних активностей та загроз до того, як вони спричинять шкоду.

Дослідження методів машинного навчання також є важливим для вирішення етичних та соціальних питань, які виникають із впровадженням таких технологій. Наприклад, можна досліджувати методи для усунення упередженості в алгоритмах (bias) або забезпечення прозорості моделей для прийняття рішень [2, 24].

Загалом, дослідження машинного навчання — це основа інновацій, що рухають сучасний світ вперед у напрямку більш ефективних, розумних і адаптивних систем у багатьох галузях людської діяльності.

1.2 Визначення складових машинного навчання

Метою машинного навчання є створення алгоритмів і моделей, які можуть автоматично навчатися на даних і робити прогнози або приймати рішення без явного програмування для визначених завдань. Основна ідея полягає в тому, щоб комп'ютерні системи могли виявляти закономірності, адаптуватися до нових даних та покращувати свою продуктивність з часом.

Для того, щоб навчити машину, потрібно використовувати такі складові як *дані, ознаки, алгоритми*.

Дані у машинному навчанні — це фундаментальна складова, на основі якої алгоритми навчаються, приймають рішення та роблять прогнози. Вони є основним джерелом інформації, яку модель використовує для вивчення закономірностей, трендів або залежностей.

Дані є основою, на якій базується навчання будь-якої моделі. Якість і кількість даних безпосередньо впливають на результативність моделей машинного навчання. Чим більше даних і вони якісніші та різноманітніші, тим точніші та надійніші прогнози або рішення може приймати модель, тим краще модель може узагальнювати знання і застосовувати їх до нових ситуацій. Якість даних безпосередньо впливає на точність прогнозів. Помилкові або неповні дані можуть призводити до неправильних висновків.

Приклад розмітки даних приведений на рис. 1.2.

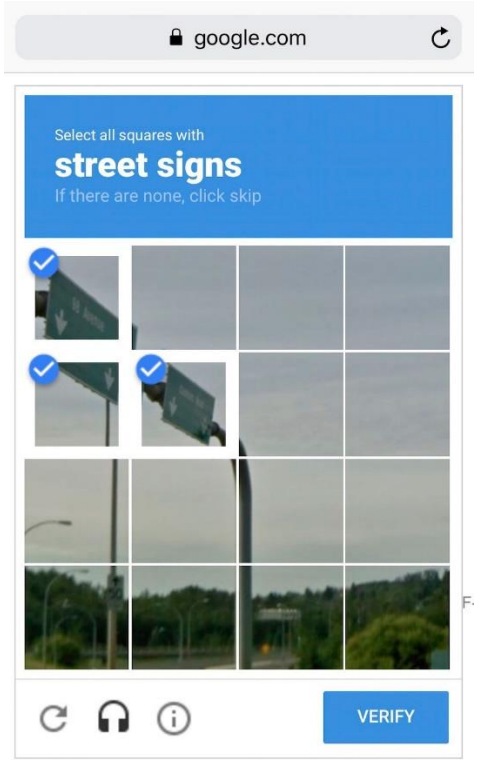

Рис. 1.2 Приклад розмітки даних

Досліджуючи типи даних для ML, можна зробити висновки, що дані повинні бути різноманітними за типом і форматом (табл. 1.1). Основними типами даних є: *числові, категоріальні, текстові, зображення та відеодані, часові ряди, аудіо*.

До числових даних відносяться дані у вигляді чисел (наприклад, вік, температура, прибуток, ціна). Категоріальні дані представляють собою категорії або класи (наприклад, стать: чоловік/жінка, тип продукту, колір, країна). Текстові дані представлені у вигляді тексту (наприклад, документи, повідомлення, коментарі у соціальних мережах). Зображення та відео – це візуальні дані, які можуть бути використані для задач, пов'язаних з комп'ютерним зором (наприклад, розпізнавання обличь, об'єктів). До часових рядів відносяться дані, що змінюються з часом (наприклад, ціни на акції, показники погодних умов). Аудіо – це звукові дані, наприклад, для розпізнавання мовлення.

Одним з варіантів класифікації типів даних є їх розподіл на *структуровані, неструктуровані, числові* та *категоріальні*.

Структуровані дані організовані у таблиці з чітко визначеними рядками і стовпцями (дані про клієнтів, результати експериментів). Неструктуровані дані не мають фіксованої структури (текст, зображення, аудіо, відео).

Таблиця 1.1

Типи даних

Табличні дані	**Зображення**	**Текстові дані**	**Звук**	**Числові дані**	**Категоріальні дані**	**Часові ряди**
Оцінка кредитоздатності	Виявлення дефектів на виробництві	Виявлення спаму	Ідентифікація по голосу	Вік	Визначення статі	Дані, що змінюються з часом
Ймовірність переходу на інший тарифний план	Самокеровані автомобілі	Перекладачі	Переведення голосу в текст	Температура	Тип продукту	Ціни на акції
Контекстна реклама	Ідентифікація по обличчю	Автодоповнення	Видалення шумів	Прибуток, ціна	Країни	Показники погодних умов

Ознаки (features) у машинному навчанні — це окремі характеристики або атрибути даних, які використовуються для навчання моделі. Кожна ознака представляє певний аспект інформації про дані та допомагає моделі виявляти закономірності і робити прогнози.

Ознака є одним із вхідних параметрів, який модель використовує для аналізу даних і прийняття рішень. Для кожного об'єкта (рядка даних) може бути кілька ознак, які описують його властивості. Наприклад, у випадку задачі прогнозування ціни на квартиру ознаками можуть бути: площа квартири, кількість кімнат, відстань до центру міста, рік будівництва. Для задачі класифікації електронної пошти як спам/не спам прикладами ознак можуть бути: кількість слів у листі, наявність певних ключових слів. Прикладами ознак у задачі розпізнавання зображень є: пікселі зображення, кольорові канали.

Типи ознак приведені у табл. 1.2.

Таблиця 1.2

Типи ознак

Числові ознаки	**Категоріальні ознаки**	**Бінарні ознаки**	**Часові ознаки**
Вік	Стать: чоловік/жінка	Є/немає	Дата події
Заробітна плата	Освітній рівень: середня освіта, вища освіта, аспірантура	1/0	Час події
Площа квартири	Тип продукту: електроніка, одяг, меблі	Наявність парковки або відсутність	Тимчасові ряди

Для числових ознак кількісні значення можуть бути безперервними або дискретними. У категоріальних ознаках якісні значення представляють різні категорії або класи. Бінарні ознаки - це ознаки, які можуть мати лише два можливих значення. Часові ознаки пов’язані з часом. Вибір правильних ознак є критично важливим для продуктивності моделі. Часто сирі дані можуть містити надто багато або надто мало інформації, тому інженерія ознак — це процес створення нових, більш інформативних ознак з наявних даних.

Алгоритм у машинному навчанні — це набір інструкцій або математичних правил, які використовуються для навчання моделей, що вивчають закономірності у даних і приймають рішення або роблять прогнози. Алгоритми обробляють вхідні дані, аналізують їх і створюють моделі, які можуть використовуватися для виконання конкретних завдань, таких як класифікація, регресія, кластеризація, зменшення розмірності тощо [3-5].

Наприклад, для задач класифікації алгоритми передбачають категорії або класи для даних (наприклад, класифікація електронної пошти на спам і не спам). У задачах регресії алгоритми передбачають числові значення (наприклад, прогнозування ціни на житло). Для задач кластеризації алгоритми групують дані на основі подібностей (наприклад, поділ клієнтів на сегменти). У задачах зменшення розмірності використовуються алгоритми, що зменшують кількість ознак, зберігаючи важливу інформацію (наприклад, для візуалізації або попередньої обробки даних).

Алгоритми використовують дані для навчання моделей на основі виявлення залежностей або правил у цих даних. Алгоритм обробляє вхідні дані, аналізує їх і створює модель. Алгоритми використовують функції втрат, щоб оцінити, наскільки добре модель працює. Наприклад, різниця між передбаченим і реальним значенням може використовуватися для корекції моделі. Модель поступово вдосконалюється через кілька ітерацій.

Алгоритм у машинному навчанні — це основний інструмент, який використовують для автоматизації процесів навчання з даних. Вибір правильного алгоритму залежить від типу задачі, наявних даних і бажаних результатів. Алгоритми можуть навчатися на розмічених даних (рис. 1.3), виявляти структуру в нерозмічених даних (рис. 1.4) або навіть взаємодіяти з середовищем і навчатися через винагороду (навчання з підкріпленням) (рис. 1.5).

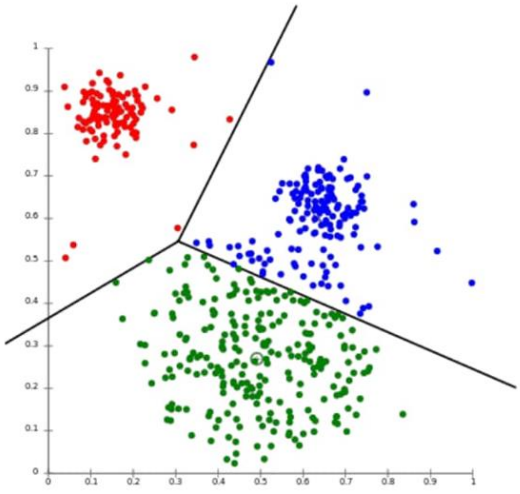

Рис. 1.3 Алгоритм k-найближих сусідів (k-NN)

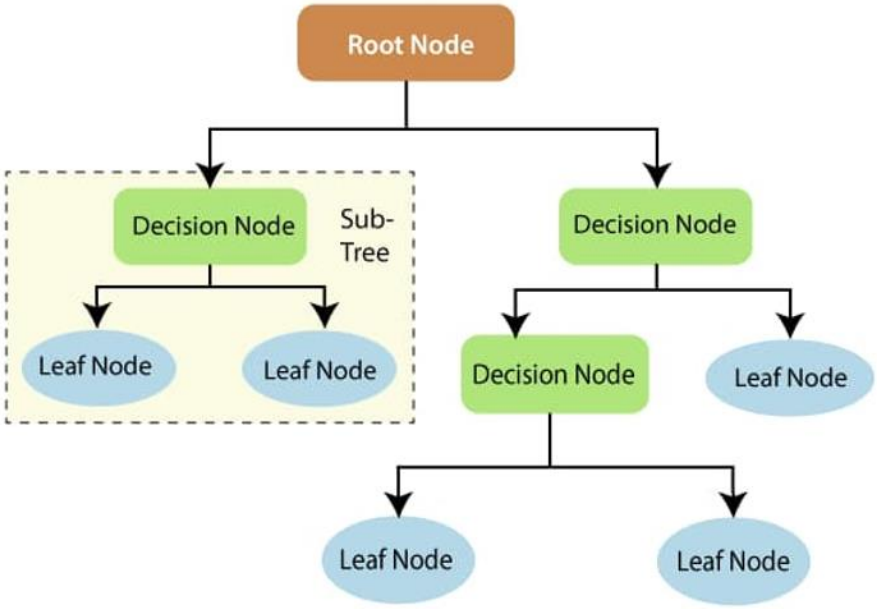

Рис. 1.4 Алгоритм Random Forest

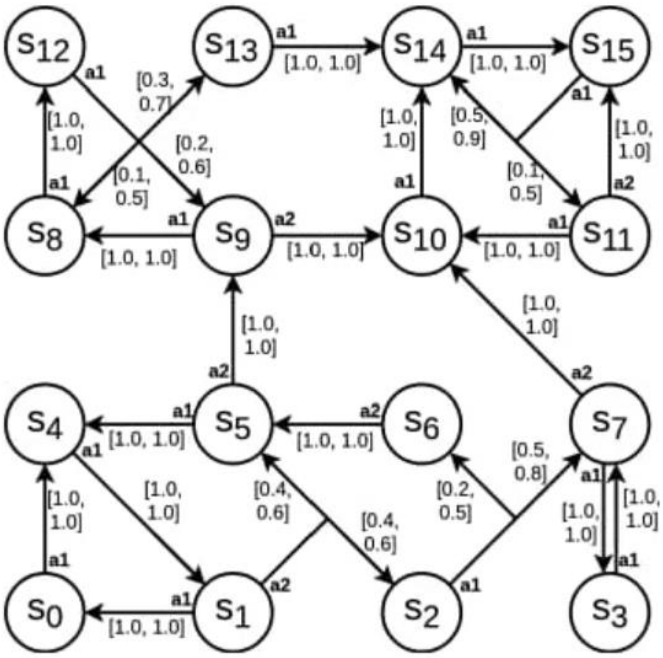

Рис. 1.5 Приклад алгоритму MDP

1.3 Характеристика типів даних та їх особливостей

У машинному навчанні *тип даних* визначає форму, структуру та зміст інформації, з якою працює модель. Від типу даних залежить вибір алгоритмів, способи підготовки даних та методи обробки.

Структурований тип даних – це тип даних, який поєднує кілька простіших типів (наприклад, числових, текстових або булевих) у єдину логічну структуру. Цей тип використовується для опису складних об'єктів, що мають кілька властивостей або атрибутів. Структуровані типи даних допомагають ефективно організовувати й оперувати інформацією, що складається з кількох пов'язаних значень.

Неструктурований тип даних – це тип даних, що не має чітко визначеної структури, схеми або організованого формату. Такі дані не вписуються у традиційні таблиці баз даних, оскільки вони не поділяються на окремі поля або стовпці і не мають єдиного формату. Неструктуровані дані можуть містити текст, зображення, відео, аудіо або інші мультимедійні формати, де інформація представлена у складному, варіативному або гібридному вигляді. Неструктуровані дані є цінним джерелом інформації, особливо для сучасних аналітичних систем і моделей машинного навчання, але через їх складність та варіативність потребують спеціальних методів обробки та зберігання.

Крім класифікації на структуровані та неструктуровані дані, є ще інші типи даних.

Текстові дані – це тип неструктурованих даних, що містить інформацію у вигляді тексту. Вони можуть включати слова, речення, параграфи або навіть цілі документи, написані природною мовою. Текстові дані зустрічаються у різноманітних джерелах, таких як електронні листи, соціальні мережі, статті, вебсторінки, чати, блоги та інші форми цифрового текстового контенту (рис. 1.6).

Для публікації текстових даних використовуються наступні формати: DOC, DOCX, PDF, TXT, PTF, ODT, CSV. Формат CSV (Comma-Separated Values) – це популярний текстовий формат для зберігання табличних даних, де кожен рядок у файлі представляє один запис, а кожен стовпець – значення визначеного поля. Значення полів у рядках розділені комами (або іншими роздільниками, такими як крапка з комою або табуляція), що й дало назву формату. Формат CSV є простим, легким і універсальним, але для складніших завдань або структур даних можуть знадобитися альтернативні формати, такі як JSON або XML.

Про затвердження Положення про набори даних, які підлягають оприлюдненню у формі відкритих даних

{Із змінами, внесеними згідно з Постановами КМ
№ 1035 від 28.12.2016
№ 354 від 24.05.2017
№ 1100 від 20.12.2017
№ 524 від 04.07.2018
№ 409 від 17.04.2019
№ 916 від 06.11.2019
№ 1065 від 04.12.2019
№ 1141 від 04.12.2019
№ 14 від 15.01.2020
№ 123 від 05.02.2020
№ 171 від 26.02.2020
№ 405 від 27.05.2020
№ 561 від 01.07.2020
№ 826 від 09.09.2020
№ 870 від 23.09.2020
№ 936 від 09.10.2020}

Відповідно до статті 10^1 Закону України "Про доступ до публічної інформації" Кабінет Міністрів України **п о с т а н о в л я є**:

1. Затвердити такі, що додаються:

Положення про набори даних, які підлягають оприлюдненню у формі відкритих даних.

Порядок щорічної оцінки стану оприлюднення та оновлення відкритих даних розпорядниками інформації на Єдиному державному веб-порталі відкритих даних (далі - Порядок).

{Пункт 1 в редакції Постанови КМ № 409 від 17.04.2019}

2. Розпорядникам інформації, визначеним Законом України "Про доступ до публічної інформації", забезпечити протягом шести місяців оприлюднення та подальше оновлення на своїх офіційних веб-сайтах наборів даних згідно з Положенням, затвердженим цією постановою, а також забезпечити надання інформації, необхідної для проведення оцінки стану оприлюднення та оновлення відкритих даних відповідно до Порядку.

Рис. 1.6 Приклад текстових даних

Текстові дані є надзвичайно важливими для сучасних технологій машинного навчання, аналітики даних і штучного інтелекту, оскільки містять цінну інформацію, яку можна використовувати для прийняття рішень, автоматизації та створення інноваційних продуктів і сервісів.

Табличні або структуровані дані – це дані, які організовані у вигляді таблиць з чітко визначеною структурою. Вони зберігаються у вигляді рядків і стовпців, де кожен рядок представляє один запис або об'єкт, а кожен стовпець – атрибут або властивість цього об'єкта. Такий формат зберігання дозволяє ефективно шукати, фільтрувати та маніпулювати даними.

Частіше всього структуровані дані представляються у форматі CSV, XLSX - формат електронних таблиць, розроблений компанією Microsoft для зберігання, обробки та аналізу табличних даних. Можна виконувати експорт даних у форматах XML, JSON. Приклад табличних або структурованих даних приведений у табл. 1.3. Формат XLSX є загальноприйнятим і дуже популярним завдяки своїй

функціональності, гнучкості та можливості обробки даних безпосередньо в середовищі Microsoft Excel та сумісних програмах, таких як Google Sheets і LibreOffice.

Таблиця 1.3

Приклад табличних або структурованих даних

Місто	Підприємство	Дата	Прибуток
Київ	ДП «Комунсервіс»	2024-05-01	15503148
Київ	ДП «Комунсервіс»	2024-03-30	1834126
Київ	ДП «Комунсервіс»	2024-02-15	1950456
Львів	ДП «Львівкартон»	2024-10-18	1675362
Львів	ДП «Львівкартон»	2024-07-08	946194
Харків	ДП «Льодова арена»	2024-02-23	2453173
Харків	ДП «Льодова арена»	2024-01-25	5382341

Графічні дані – це дані, які містять інформацію у формі зображень, візуальних об’єктів або графічних елементів. Вони охоплюють різні види візуального контенту, зокрема фотографії, ілюстрації, графіки, діаграми, карти, схеми та інші зображення, які передають візуальну інформацію (рис. 1.7).

Для представлення графічних даних використовуються такі формати як PNG, JPG, JPEG.

Технологія оптичного розпізнавання тексту (OCR, Optical Character Recognition) – це технологія, яка дозволяє перетворювати текст на зображеннях або у відсканованих документах у машинозчитуваний текст. За допомогою OCR можна автоматично "зчитувати" текст з різних джерел, таких як фотографії, відскановані документи, PDF-файли і конвертувати його у редагований текстовий формат.

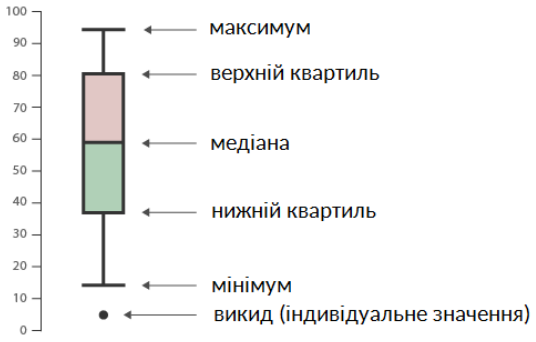

Рис. 1.7 Графічні дані

Геопросторові дані — це дані, що містять інформацію про об'єкти або явища з прив’язкою до певного місця на Землі. Вони включають координати, адреси, межі територій, топографічні дані, інформацію про рельєф, клімат, інфраструктуру та інші характеристики, які допомагають визначити розташування та властивості об'єктів у просторі. Для представлення геопросторових даних використовуються такі формати як SHP, GeoJSON, а також GeoTIFF, GPX.

Геопросторові дані включають координати (широту, довготу) або інші географічні параметри, які вказують точне розташування об'єктів. Дані такого типу можуть бути представлені у векторних (точки, лінії, полігони) та растрових (зображення, теплові карти) форматах. Геопросторові дані можуть містити атрибутивну інформацію про об'єкти, наприклад, назву об'єкта, тип місцевості, популяцію, площу, висоту над рівнем моря. Їх можна масштабувати для різних рівнів деталізації: від глобальних карт до детальних планів місцевості. Геопросторові дані є ключовим елементом у прийнятті рішень, що вимагають розуміння просторових взаємозв’язків і відстеження змін у реальному світі (рис. 1.8).

Рис. 1.8 Геопросторові дані

Архівні дані — це дані, які зберігаються для тривалого часу з метою їхнього захисту, збереження та можливого подальшого використання (рис. 1.9). Такі дані є результатом діяльності організацій, установ або приватних осіб і зазвичай мають історичну, культурну, правову або адміністративну цінність. Архівні файли

створюються за допомогою процесу стиснення даних та об’єднання кількох файлів або папок у єдиний файл для зберігання, передавання або зберігання інформації на довгий термін. Цей процес дозволяє оптимізувати використання пам’яті, полегшити управління файлами та спростити передавання даних.

Архівні файли створюються за допомогою спеціальних програм, відомих як архіватори (наприклад, WinRAR, 7-Zip, WinZip, tar у Unix-системах). Архіватори дозволяють об’єднувати декілька файлів або папок у один архівний файл, зменшувати їхній розмір за рахунок стиснення, а також додавати паролі для захисту доступу.

ZIP - один із найпоширеніших форматів архівів, який підтримується майже всіма операційними системами. ZIP дозволяє стискати дані без втрат та об’єднання декількох файлів в один. RAR - формат, який пропонує кращу ступінь стиснення, ніж ZIP, але потребує спеціального програмного забезпечення для розархівації. 7Z - формат із високим рівнем стиснення, створений програмою 7-Zip, підтримує різні алгоритми стиснення та шифрування. У Unix-подібних системах використовується формат TAR для об’єднання файлів і GZ (gzip) для стиснення, часто використовуються разом як TAR.GZ.

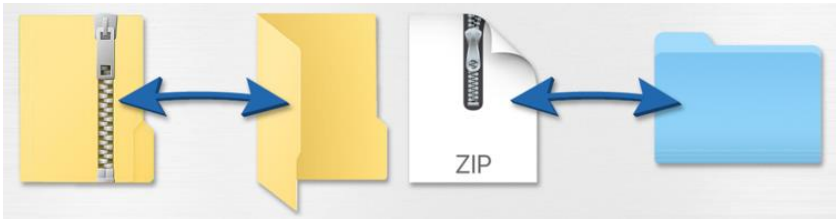

Рис. 1.9 Архівні дані

Впорядковані дані — це дані, які розташовані у визначеному порядку, зазвичай на основі певного критерію або правила, такого як числове значення, алфавітний порядок, дата тощо. Цей порядок полегшує аналіз, пошук і обробку інформації, оскільки впорядковані дані мають чітку структуру, яка зберігає логічний зв'язок між елементами (рис. 1.10). Впорядкування забезпечує легший та швидший доступ до певних значень або діапазонів даних.

Рис. 1.10 Приклад впорядкованих даних

Багато алгоритмів і методів аналізу даних працюють швидше з впорядкованими даними, що економить час і ресурси. У впорядкованих даних легше помітити пропущені, повторювані або аномальні значення, які можуть свідчити про помилки. Сферами застосування впорядкованих даних є бази даних та системи управління; аналітика та бізнес-інтелект; наукові дослідження [4-6].

Упорядковані дані в базах даних забезпечують швидкий доступ до інформації та оптимізують запити до бази; полегшують створення звітів, графіків, прогнозів і моделей для підтримки бізнес-рішень. У дослідженнях важливо зберігати дані у впорядкованому вигляді для аналізу закономірностей і виявлення тенденцій.

Впорядковані дані мають величезну цінність у багатьох галузях, оскільки вони роблять дані більш структурованими, зручними для обробки, аналізу і прийняття рішень.

2 ДОСЛІДЖЕННЯ НАПРЯМІВ МАШИННОГО НАВЧАННЯ

2.1 Класичне навчання

Досліджуючи напрями машинного навчання, які також можна синонімізувати з типами машинного навчання, станом на сьогоднішній день, таких можна виділити чотири: класичне навчання; навчання з підкріпленням; ансамблі; нейронні мережі і глибоке навчання. Класичне навчання передбачає використання простих даних та зрозумілих ознак. У навчанні з підкріпленням даних не має, але є середовище, з яким можна взаємодіяти. Ансамблеві методи навчання використовують, коли проблемою є якість. Нейромережі та глибоке навчання використовуються, коли дані мають складний характер, а ознаки є незрозумілими.

Класичне навчання поділяється на 2 категорії: навчання з вчителем (Supervised Learning) та навчання без вчителя (Unsupervised Learning). Для машинного навчання із вчителем використовуються числові дані або дані, які наперед категоризовані. Для навчання без вчителя використовуються дані без жодних міток. Прикладами задач з вчителем є задачі класифікації та регресії. У задачах класифікації, як приклад, потрібно передбачити якусь категорію, наприклад, розкласти речі за кольором (рис. 2.1). У задачах регресії, наприклад, є необхідність передбачити якесь значення (розкласти речі за довжиною).

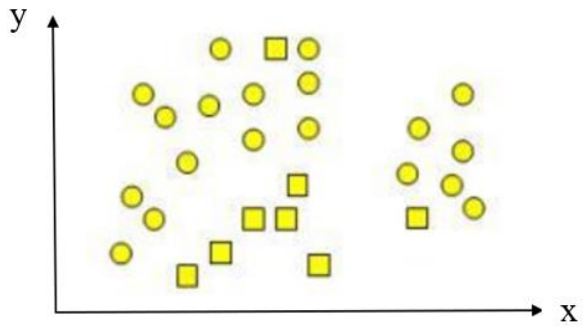

Рис. 2.1 Задача класифікації

Задача класифікації у машинному навчанні - це тип завдання, метою якого є визначення категорії або класу, до якого належить вхідний об'єкт на основі його

характеристик (ознак). Класифікація використовується тоді, коли вихідні дані мають дискретну природу, тобто складаються з обмеженого набору класів. Вхідними даними є набір ознак або характеристик, які описують об'єкт. До вхідних даних відносяться: числові, текстові, категорійні дані або зображення. Вихідні дані можуть мати дискретні значення (класи), наприклад: так/ні, клас А/клас В. Класів може бути два та більше. Якщо використовуються два класи – це задача бінарної класифікації. Якщо використовуються три та більше класів – це задача багатокласової класифікації. Метою задач класифікації є навчити модель правильно розпізнавати клас або класи, до якого/яких належить новий об'єкт, на основі наданих прикладів.

В якості прикладів задач бінарної класифікації можна привести наступні. Спам-фільтр: чи є електронний лист "спамом" або "не спамом"; для медичної діагностики це може бути виявленням захворювання: є пацієнт "хворий" або "здоровий"; для прогнозу відтоку клієнтів: залишиться клієнт або ні. Багатокласова класифікація: розпізнавання рукописних цифр (0-9); класифікація видів тварин за зображенням; визначення теми тексту (наприклад, спорт, політика, наука). Модель класифікації може визначати, чи буде транзакція в банку шахрайською, аналізуючи такі ознаки, як сума операції, час, місце.

Задача регресії у машинному навчанні - це тип завдання, метою якого є передбачення числового (неперервного) значення вихідної змінної на основі вхідних даних. Вона відрізняється від задачі класифікації тим, що результат є числовим і може набувати будь-якого значення у певному діапазоні, а не належати до дискретних категорій. Вхідні дані можуть бути числовими, категорійними (після кодування), часовими. Вихідні дані – це числове значення, яке модель має передбачити, наприклад: ціна, температура, прибуток, швидкість. Метою задач регресії є навчити модель прогнозувати значення залежної змінної на основі вхідних даних з мінімальною помилкою.

У машинному навчанні використовується декілька видів регресій, які застосовуються залежно від специфіки задачі, характеру даних і типу зв’язку між вхідними та вихідними змінними: лінійна (Linear Regression) (рис. 2.2),

поліноміальна (Polynomial Regression), логістична (Logistic Regression), ридж-регресія (Ridge Regression), лассо-регресія (Lasso Regression), еластична мережа (Elastic Net Regression), Байєсівська (Bayesian Regression), регресія методом опорних векторів (Support Vector Regression, SVR), регресія з деревами рішень (Decision Tree Regression), Ensemble-регресія, нейронні мережі для регресії.

Тип регресії залежить від характеру даних: для лінійних залежностей - лінійна або ридж-регресія, для нелінійних задач – поліноміальна регресія, дерева рішень або нейронні мережі, для великих наборів корельованих ознак доцільно використовувати лассо або еластичну мережу.

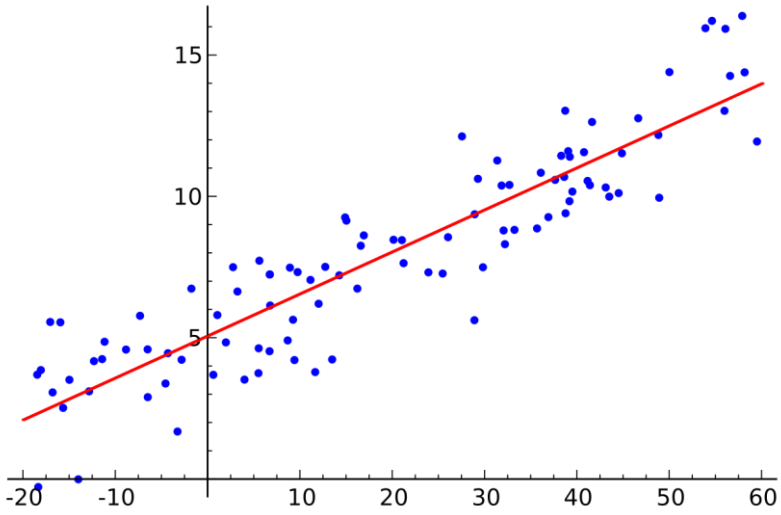

Рис. 2.2 Лінійна регресія

Навчання без учителя - це тип машинного навчання, в якому модель працює з немаркованими даними, тобто даними без заздалегідь визначених вихідних значень (міток або класів). Мета такого навчання полягає в пошуку прихованих закономірностей, груп або структур у даних без втручання людини. Прикладами задач без вчителя є задачі кластеризації, зменшення розмірності (узагальнення), асоціація. *Задача кластеризації* дозволяє поділяти елементи за подібністю, наприклад, розкдадувати схожі речі у купки (рис. 2.3, рис. 2.4). Задача зменшення розмірності (узагальнення) (Dimensionality Reduction) дозволяє знаходити залежності, наприклад, зібрати з речей найкращі наряди. Задача асоціації дозволяє виявляти послідовності елементів (наприклад, якщо й далі використовувати

задачу про речі, то асоціативні методи вирішення дозволяють знаходити речі, які людина часто носить разом).

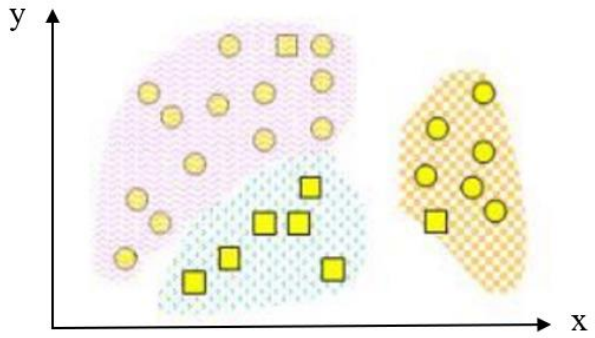

Рис. 2.3 Задача кластеризації

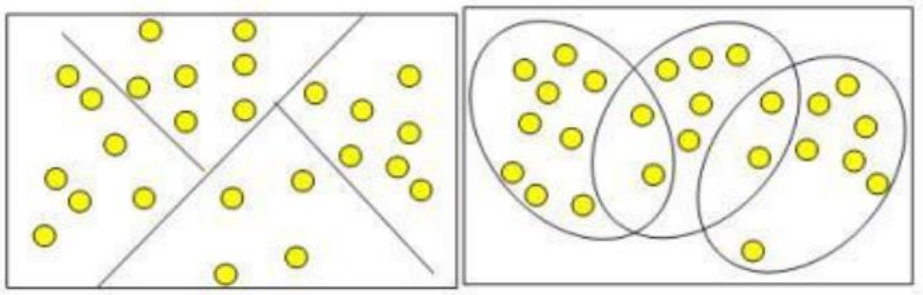

Рис. 2.4 Кластери, що не перетинаються і перетинаються

Зменшення розмірності - це задача машинного навчання, яка спрямована на скорочення кількості ознак (вимірів) у наборі даних при збереженні максимально важливої інформації. Цей процес спрощує аналіз даних, підвищує ефективність моделей і знижує ризик перенавчання (рис. 2.5).

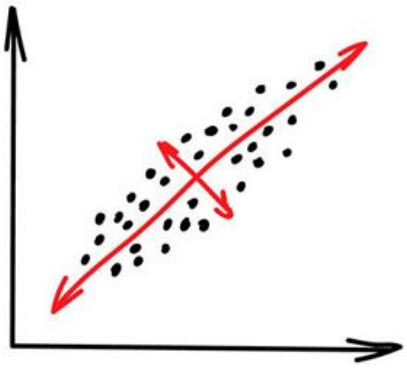

Рис. 2.5 Dimensionality Reduction

Переваги зменшення розмірності: полегшує візуалізацію складних даних, знижує ризик перенавчання через надмірну кількість ознак, покращує якість моделі шляхом усунення шуму. Це потужний інструмент для оптимізації даних і

підвищення ефективності роботи моделей машинного навчання. Зменшення розмірності використовується у задачах комп’ютерного бачення для обробки зображень або відео, у задачах біоінформатики для аналізу генетичних даних, при розпізнаванні мови для виділення важливих особливостей з аудіозаписів, у сфері маркетингу для сегментації клієнтів на основі великої кількості атрибутів.

Задача пошуку правил (асоціацій) у машинному навчанні - це процес виявлення закономірностей або зв'язків між об'єктами у наборі даних. Цей метод широко використовується для аналізу великих обсягів даних з метою знаходження правил, які описують залежності між різними змінними або подіями рис. 2.6.

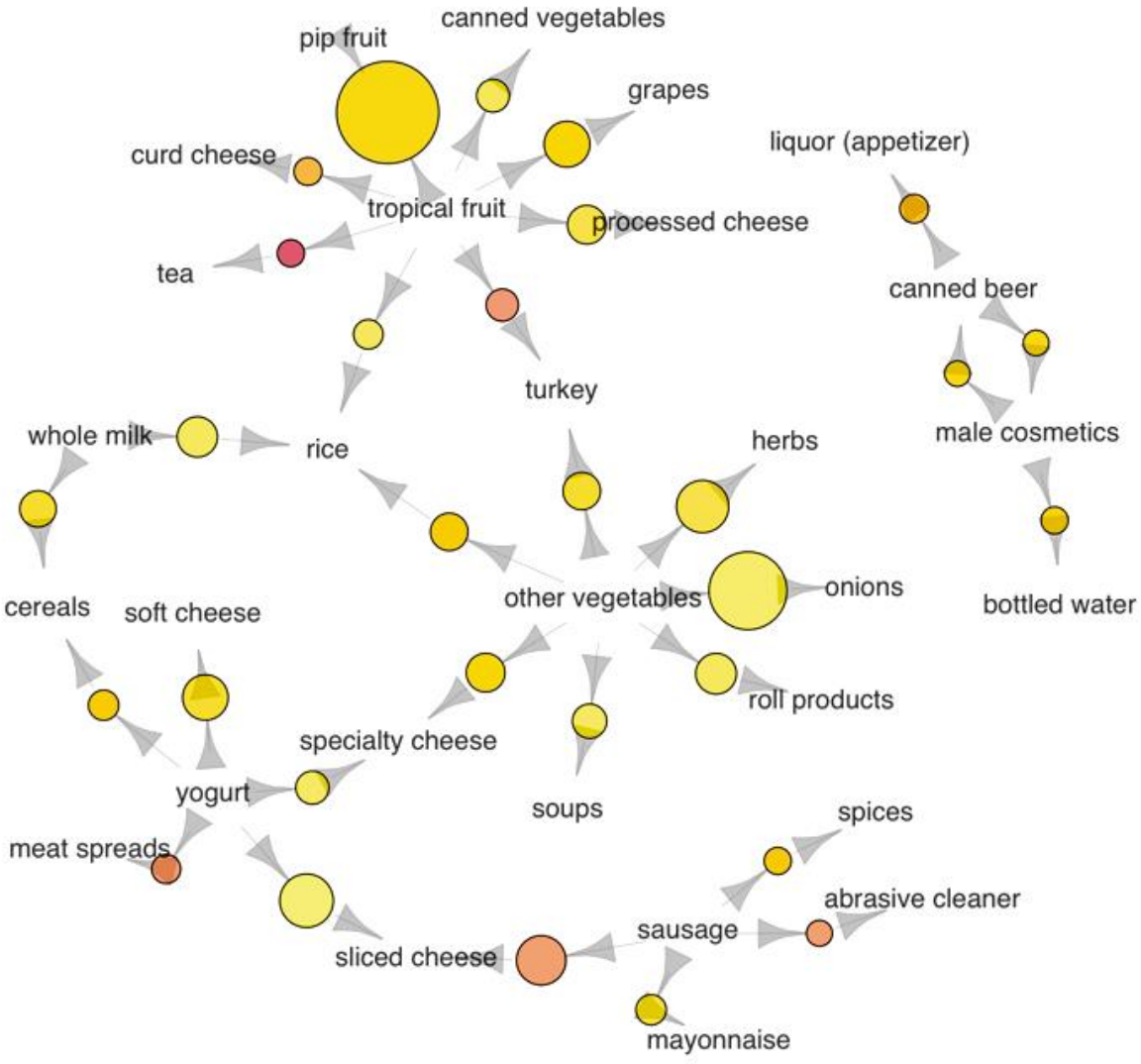

Рис. 2.6 Навчання на основі асоціативних правил

Для вимірювання асоціацій між багатьма елементами використовуються такі метрики як підтримка, достовірність, підвищення.

Підтримка (Support) – це параметр, який показує, на скільки популярний набір елементів, оскільки він використовується для визначення частоти появи визначеного набору елементів у наборі даних:

$$Support(A) = Frequency(A) \tag{2.1}$$

Достовірність (Confidence) - ймовірність того, що якщо *A* присутній, то *B* також буде присутній:

$$Confidence(A \to B) = \frac{Support(A \to B)}{Support(A)} \quad , \tag{2.2}$$

де *A* і *B* – набори елементів, а правило інтерпретується як «якщо *A* присутній у транзакції, то *B* також з певною ймовірністю».

Підвищення (Lift) – показник, який показує ймовірність того, що товар *A* буде придбаний, а також контролює популярність товару *B*:

$$Lift(A \to B) = \frac{Confidence(A \to B)}{Support(B)} \tag{2.3}$$

Якщо Lift > 1, правило вважається корисним.

2.2 Навчання з підкріпленням

Навчання з підкріпленням (Reinforcement Learning, RL) - це метод машинного навчання, в якому агент навчається приймати рішення шляхом взаємодії з середовищем і отримання зворотного зв'язку у вигляді нагород (або штрафів). Мета агента - знайти стратегію дій, яка максимізує сукупну винагороду протягом часу. Це не різновид навчання нейронної мережі. Це окремий напрямок стеку технологій, який на сьогоднішній день набуває широкої популярності.

Рольові комп’ютерні ігри, симулятори, шутери в основному супроводжуються контактними двобоями між людиною і ботами. Тобто це програма, яка діє максимально подібно до того, якби діяла у подібній ситуації людина. Це тільки один із прикладів застосування даної технології. На сьогоднішній день спектр її застосування розширюється. У період війми ми часто чуємо, що потрібно навчити дрон автономно літати у певному середовищі, не спираючись на систему GPS. Потрібно навчити безпілотний наземний апарат рухатися і орієнтуватися на місцевості і т.д. Все це стосується технології навчання з підкріпленням.

Навчання з підкріплення – це один із способів навчання, який пов’язаний із системою прийняття рішень.

Тобто у нашому випадку нейронна мережа, яку ми навчаємо, нічого не продукує, нічого не класифікує, а вона приймає рішення як діяти в даній конкретній обстановці, куди ми цю нейронну мережу занурюємо. Оскільки навчання з підкріпленням – це окремий стек технологій, то в ньому є вже певне професійне коло, яке сформувалося, і у цього професійного кола є своя певна термінологія.

У навчанні з підкріпленням модель, яка б вона не була, прийнято називати *агентом*. А віртуальний або реальний світ, де дана модель має використовуватися, називається *середовищем*.

Основною метою навчання з підкріпленням є прийти до розуміння того, як наша модель агента може досягати певної мети у певному середовищі. Наприклад, той самий контрстрайк. Там в якості моделі виступає бот, який потрапляє на віртуальну карту, основна задача якого вести себе свідомо, максимально подібно до людини у тому віртуальному середовищі, у тому віртуальному ландшафті, в якому перебуває гравець під час гри і в якому перебуває цей бот під час гри. У цьому й полягає мета будь-якої задачі, яка вирішується за допомогою навчання з підкріпленням.

Взаємодія агента з середовищем зводиться до виконання ним певних дій, наприклад, повороту вправо-вліво, вперед-назад.

В залежності від того, наскільки були правильні ці дії, агент отримує зміну своїх внутрішніх коефіцієнтів, які прийнято називати нагородою. Чим краще агент себе повів (на думку розробників) у даному середовищі, тим більшу винагороду він отримує. Саме на цьому і базується все навчання агента у конкретному віртуальному середовищі.

Вся взаємодія агента і середовища зводиться до наступних дій. Агент ініціалізує стан середовища, тобто він отримує первинну інформацію з приводу того, які дії він може здійснювати (наприклад, рух вперед, назад, вправо, вліво), куди він має рухатися у даній конкретній точці з урахуванням його попереднього досвіду. Коли агент обирає як йому діяти, він здійснює цю дію. На підставі здійснення цієї дії середовище генерує винагороду для даного агенту. Якщо він вчинив з точки зору середовища правильно (наприклад, не наткнувся на перешкоду, обійшов її), то він змінює свій стан, отримуючи більш високу винагороду. Якщо цього не сталося, він отримує більш низьку нагороду і після цього починає коригувати свою стратегію. Таким чином, відбувається взаємодія (рис. 2.7).

Рис. 2.7 Навчання з підкріпленням

Система підкріплення – будь-який набір правил, на основі якого можна змінювати з плином часу матрицю взаємодії (або стан пам’яті) перцептрона. Види системи підкріплення: альфа-система, гамма-система.

Альфа-система – система підкріплення, при якій ваги всіх активних зв'язків, які ведуть до елементу, змінюються на однакову величину, а ваги неактивних зв’ язків за цей час не змінюються. В альфа-системі, якщо агент у конкретному середовищі вчинив правильну дію, він отримав винагороду і його ваги змінилися. Якщо агент вчинив не правильну дію, його ваги просто не змінюються. Він не

отримує якийсь штраф, тобто зниження показника, а просто ваги, які визначають його винагороду, залишаються незмінними.

На сьогоднішній день популярності набуває гамма-система.

Гамма-система – це таке правило зміни вагових коефіцієнтів деякого елементу, при якому ваги всіх активних зв'язків спочатку змінюються на однакову величину. Потім з вагів цих зв'язків віднімається інша величина, рівна повній зміні вагів усіх активних зв'язків, поділеній на кількість усіх зв'язків.

Гамма-система передбачає не повну винагороду за правильну дію. Тобто, якщо агент, діючи у середовищі, вчинив правильний вчинок (повернув куди потрібно, пройшов в якомусь певному напрямку) – це зовсім не означає, що він отримає максимальну винагороду. Він може отримати не дуже високий коефіцієнт. Це робиться для того, щоб максимально гуманізувати поведінку агента в середовищі, тобто зробити ілюзію того, що агент може коливатися, вибираючи той чи інший маршрут, може помилятися і т.д.

Гамма-система використовується при створення реалістичних ігрових додатків. Що стосується навчання автономних навігаційних систем для робототехніки, досі використовується альфа-система і поки що від неї ніхто не хоче і не буде відмовлятися.

Як і в роботі з усіма іншими моделями у навчанні з підкріпленням є певні етапи застосування:

- підготовка даних;
- моделювання нейромережі;
- задання функції втрат у відповідності з винагородженням;
- ефективне визначення винагородження;
- тренування мережі;
- деплой додатку з навченою мережею.

Перший етап – це *підготовка даних*. Слід запам'ятати, що тут не має навчального набору у традиційному розумінні. Це не набір картинок, не набір цифр, не набір текстових фраз, звуків або чогось іншого. Тут в якості навчального

набору виступає певне віртуальне або реальне змодельоване середовище. От що таке навчальний набір у методиці навчання з підкріпленням.

Другий етап - це *моделювання нейромережі* під роботу у цьому середовищі.

Має бути певна метрика навчання нашої моделі у змодельованому середовищі, тому тут нам потрібно задати певну *функцію втрат* в залежності від винагородження. Тобто, чим більше винагородження, тим менші показники у нас повинна мати функція втрат.

Ефективне визначення винагородження. Як правило, коли ми моделюємо поведінку агента у конкретному середовищі, ми передбачаємо, що він один і той же маршрут може пройти не в один якийсь конкретний спосіб, а у декілька способів, при чому один із них буде найбільш ідеальний (найбільш раціональний), а інший може бути менш раціональним, більш довгим, більш затратним і т.д. В залежності від того, який шлях буде обирати наш агент, йому буде генеруватися винагородження. Якщо вибрав ідеальний шлях – все добре, якщо вибрав не ідеальний – він теж отримує певну винагороду, але меншу.

Наступний етап – *тренування*. Тут є певна особливість. Коли ми тренували попередні моделі глибокого навчання, то ми використовували таке поняття як епохи. Ми декілька разів пропускали через модель певний навчальний набір, але цих декількох разів могло бути 5, 10, 15, 20, 30 і більше, але все таки це була адекватна цифра. Коли ми тренуємо агента у певному середовищі, приходиться використовувати декілька сотень або й тисяч хопів (в залежності від того, на скільки складне середовище) навчання для того, щоб наш агент правильно навчився діяти у конкретному середовищі. Після того, як ми все виконали і коли ми побачили, що наша функція втрат нашого агенту не змінюється, ми можемо робити деплойт певного додатку. Робимо цей додаток бінарним за допомогою тих же самих методик, які ми з вами використовували раніше для звичайних моделей і вже як бінарний даний додаток може виступати як самостійний програмний модуль у тій самій комп’ютерній грі. Тобто, ми його навчити діяти на певній локації і він стає частиною програми нашого бота, який буде тепер правильно

бігати по карті, правильно стріляти, правильно ховатися і вести себе дуже людяно.

При застосуванні методу навчання з підкріпленням ми не можемо обійтися звичайними фреймворками до яких ми вже звикли. За допомогою PyTorch або TensorFlow ми можемо змоделювати модель, можемо завантажити навчальні набори, які нам не згодяться, бо це не віртуальні середовища. Розглянемо, де брати віртуальне середовище. Для цього є певні фреймворки. Ці фреймворки мають переваги і недоліки.

Дані фреймворки використовуються на експериментальному етапі, тому що жоден з цих фреймворків не імітує повністю тієї задачі, яку доведеться вирішувати нашому агенту моделі. Даний фреймворк ми можемо використати для того, щоб протестувати в ньому певну модель, щоб вона себе вела в плані навчання приблизно так, як ми розраховували і тоді ми можемо її тренувати на іншій моделі вже більш наближеній до того середовища, в якому ми створюємо власного агента.

У табл. 2.1 приведені основні бібліотеки для навчання з підкріпленням.

Таблиця 2.1

Основні бібліотеки для навчання з підкріпленням

OpenAI Gym (від OpenAI)	*Використовується для розробки та тестування алгоритмів навчання з підкріпленням. Забезпечує стандартизоване середовище для тренування агентів у таких задачах як симуляції фізики, ігри, навігація. Має простий і стандартизований інтерфейс, великий вибір середовищ, активну підтримку спільнотою та документованість*
MMLF (Maja Machine Learning Framework)	*Відкритий програмний фреймворк, який містить набір алгоритмів RL і тестових середовищ, що дозволяє досліджувати різні стратегії навчання з підкріпленням. Має модульну структуру, що дозволяє ефективно експериментувати з різними стратегіями RL у різноманітних середовищах,*
PyBrain	*Бібліотека для машинного навчання, що підтримує різні види навчання, включаючи навчання з підкріпленням, нейронні мережі та еволюційні алгоритми.*
RLPy	*Бібліотека, орієнтована на методи, що базуються на функціях вартості. Вона спеціалізується на використанні лінійного апроксимації функцій вартості в середовищах з дискретними діями.*

У табл. 2.2 приведені відмінності навчання з підкріпленням від інших типів навчання.

Таблиця 2.2

Відмінності різних типів машинного навчання

Характеристика	Навчання з учителем	Навчання без учителя	Навчання з підкріпленням
Тип даних	Дані з мітками	Немарковані дані	Нагороди й штрафи
Мета	Побудова моделі	Виявлення структури	Максимізація винагороди
Взаємодія із середовищем	Немає	Немає	Активна взаємодія

2.3 Ансамблеві методи

Ансамблеві методи машинного навчання - це техніки, які комбінують кілька окремих моделей для покращення загальної ефективності прогнозування. Ідея полягає в тому, щоб об’єднати слабкі моделі, які можуть бути не дуже точними поодинці, у потужну модель. Ці методи базуються на принципі, що комбінація багатьох різних моделей може дати кращі результати, ніж використання однієї моделі. Це допоможе зменшити помилки, підвищити стабільність та знизити ризик перенавчання. Ансамблеві методи є особливо корисними у задачах, де є багато шуму в даних або коли важливо досягти високої точності. Вони зазвичай застосовуються у задачах класифікації, регресії, а також у фінансових моделях, обробці зображень і в робототехніці.

Основні ансамблеві методи включають:

- багатократне голосування (Voting);
- багаторазове навчання з підсумовуванням (Bagging);
- послідовне навчання (Boosting);
- стекінг (Stacking).

У методі Voting кожен класифікатор у ансамблі робить свій прогноз, і остаточний прогноз формується на основі голосування між всіма класифікаторами. Існує два основних типи голосування:

- hard voting, коли кожен класифікатор просто голосує за певний клас і клас, який отримав найбільше голосів, обирається;
- soft voting: класифікатори не просто голосують, а видають ймовірності для кожного класу і прогноз формується на основі сумування цих ймовірностей.

Bagging - це техніка, яка передбачає використання декількох копій однієї й тієї ж моделі, навчених на різних підмножинах навчальних даних, вибраних випадковим чином (з повторенням). Найпоширеніший приклад — Random Forest, який використовує рішення дерев для класифікації або регресії, об'єднуючи результати від багатьох дерев (рис. 2.8).

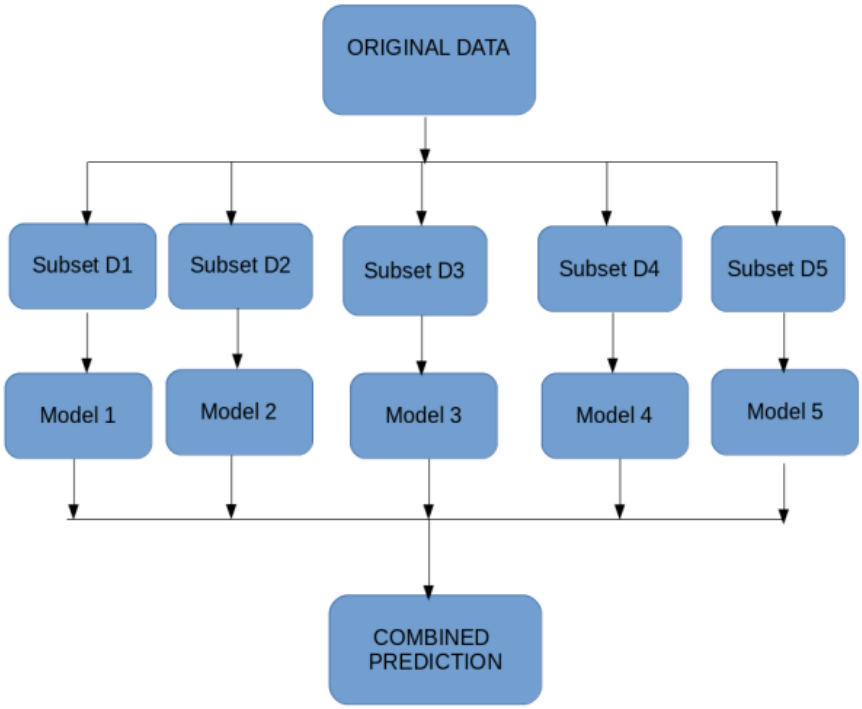

Рис. 2.8 Техніка Bagging

При застосуванні методу Boosting моделі тренуються по черзі (рис. 2.9). Кожен наступний класифікатор коригує помилки попереднього. Це дозволяє покращити загальну точність, концентруючи увагу на важких випадках. Найвідомішими методами boosting є:

- AdaBoost (Adaptive Boosting) підвищує вагу неправильно класифікованих прикладів, щоб наступний класифікатор приділив більше уваги саме цим прикладам (рис. 2.10);

- Gradient Boosting створює моделі по черзі, зменшуючи похибку за допомогою нових моделей, тренованих на залишках попередніх.

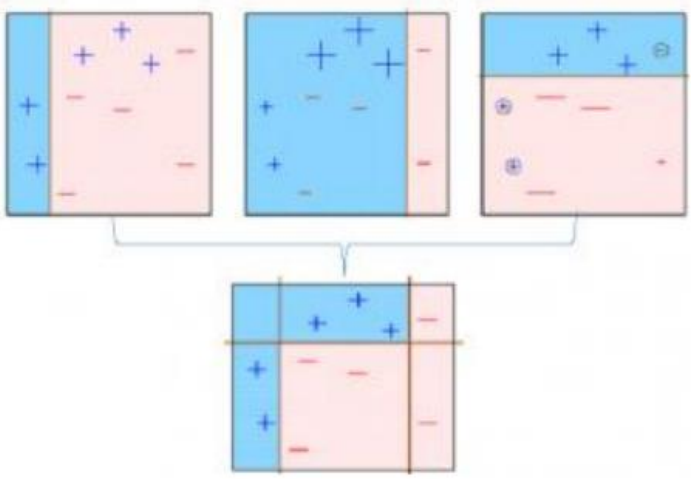

Рис. 2.9 Техніка Boosting

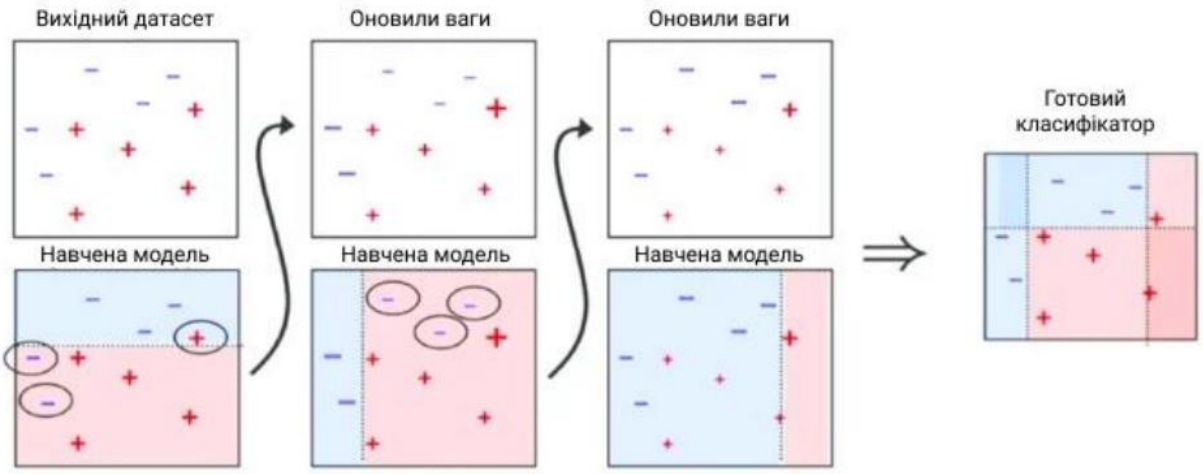

Рис. 2.10 Приклад алгоритму AdaBoost

У методі Stacking використовуються декілька різних моделей (наприклад, дерево рішень, логістична регресія, нейронні мережі) для створення прогнозу, а потім результати цих моделей комбінуються за допомогою метамоделі, яка робить остаточний прогноз. Стекінг дозволяє використовувати різні типи моделей для досягнення кращих результатів.

Завдяки своїй ефективності ансамблеві методи стали основою для багатьох популярних алгоритмів, таких як Random Forest, Gradient Boosting Machines (GBM), XGBoost, LightGBM.

Ансамблевий метод машинного навчання *Bagging (Bootstrap Aggregating)* покращує точність моделі, зменшуючи її дисперсію. Його ідея полягає в створенні кількох незалежних моделей, кожна з яких тренується на різних підмножинах даних, отриманих методом Bootstrap (випадкове вибіркове повторення). Потім результати цих моделей комбінуються для отримання кінцевого прогнозу.

Основні етапи роботи Bagging:

- створення підмножин даних;
- навчання моделей;
- комбінація прогнозів.

У процесі створення підмножини даних вибирається декілька підмножин із вихідного набору даних за допомогою техніки Bootstrap, яка дозволяє одному зразку потрапляти в декілька підмножин. Наприклад, якщо є 100 прикладів, то вибирається 100 прикладів із заміною, що може створити дублікати або залишити деякі приклади поза вибіркою.

У процесі навчання моделей кожна підмножина використовується для навчання окремої моделі (наприклад, дерева рішень або інших алгоритмів). Ці моделі працюють незалежно одна від одної.

При комбінації прогнозів, наприклад, у випадку класифікації використовується метод голосування, де фінальний клас обирається як найбільш розповсюджений серед передбачень окремих моделей. Для задач регресії використовується середнє арифметичне значення передбачень усіх моделей.

Метод Bagging усуває проблему перенавчання (overfitting) для нестійких моделей (наприклад, дерев рішень); більш стійкий до змін у даних; особливо корисний для алгоритмів із високою варіативністю, таких як дерева рішень.

Random Forest - це типовий приклад використання Bagging. У цьому методі:

- генерується багато дерев рішень (основних моделей);
- кожне дерево навчається на випадковій підмножині даних, вибраній із заміною (Bootstrap);
- кожне дерево використовує випадкову вибірку ознак для прийняття рішень, що додатково зменшує кореляцію між деревами;

- для класифікації обирається клас більшістю голосів серед усіх дерев, а для регресії береться середнє значення їхніх прогнозів.

Наприклад, для прогнозу чи клієнт банку відкриє депозит, набір даних повинен включати такі характеристики: вік, прибуток, чи має клієнт іпотеку, кількість банківських транзакцій. Підготовка включатиме застосовуємо Bootstrap, щоб створити 10 різних підмножин даних; навчання: на кожній підмножині тренуємо дерево рішень; прогноз: кожне дерево дає свій прогноз для клієнта (так/ні); голосування: фінальний прогноз обирається більшістю голосів усіх дерев.

Bagging ефективний для зменшення дисперсії та підвищення точності моделі. Random Forest є одним із найпопулярніших його реалізацій і широко використовується у задачах класифікації та регресії, включаючи аналіз фінансових даних, медичні діагнози й навіть розпізнавання облич.

Ансамблевий метод машинного навчання *Stacking* об'єднує прогнози кількох моделей (базових) за допомогою додаткової моделі (метамоделі). На відміну від Bagging і Boosting, які створюють ансамблі однорідних моделей (наприклад, дерев рішень), Stacking дозволяє комбінувати різні моделі, такі як дерева рішень, лінійна регресія, SVM, нейронні мережі тощо.

Етапи роботи Stacking:

- навчання базових моделей;
- генерація прогнозів;
- навчання метамоделі;
- фінальний прогноз.

Stacking дозволяє використовувати як лінійні, так і нелінійні моделі; можна комбінувати різні типи моделей (наприклад, дерево рішень, логістичну регресію, нейронну мережу); забезпечує кращі результати, ніж окремі моделі, завдяки використанню метамоделі. Разом з цим, техніка Stacking потребує додаткового налаштування та більше обчислювальних ресурсів; має ризик перенавчання, якщо використовувати занадто складні метамоделі; для уникнення витоку даних базові моделі повинні генерувати прогнози через крос-валідацію.

Прикладом застосування техніки Stacking є прогнозування цін на житло.

Техніка ансамблевого машинного навчання Boosting використовує послідовне навчання кількох слабких моделей для побудови однієї сильної моделі. Основна ідея Boosting полягає в тому, щоб кожна нова модель виправляла помилки попередніх, зосереджуючи увагу на складних для прогнозування даних.

Одним із прикладів використання техніки Boosting є прогнозування ймовірності дефолту клієнта банку, коли є дані про клієнтів банку (вік, прибуток, історія кредитів, зайнятість) і потрібно передбачити, чи є клієнт ризиковим для видачі кредиту. Boosting є потужною технікою для задач прогнозування, особливо якщо точність є ключовим фактором. Завдяки своїм варіаціям (AdaBoost, Gradient Boosting, XGBoost) Boosting широко застосовується у фінансах, медицині та сфері маркетингу.

Результати досліджень ансамблевих методів машинного навчання приведені у табл. 2.3.

Таблиця 2.3

Переваги та недоліки ансамблевих методів машинного навчання

Вид ансамблевого методу	*Переваги*	*Недоліки*
Bagging	1. Зменшення дисперсії 2. Підвищення стабільності 3. Ефективність	1. Не завжди покращує продуктивність для моделей із низькою варіативністю (наприклад, лінійної регресії) 2. Високі обчислювальні витрати через навчання кількох моделей
Stacking	1. Комбінування сильних сторін різних моделей 2. Гнучкість 3. Підвищення точності	1. Складність 2. Ризик перенавчання 3. Потреба у валідованих прогнозах
Boosting	1. Підходить як для задач класифікації, так і для регресії 2. Добре працює з нерівномірними даними та складними залежностями 3. Висока точність за рахунок концентрації на складних випадках	1. Чутливість до шуму в даних (оскільки він може бути "переоцінений") 2. Порівняно довгий час навчання 3. Висока складність налаштування параметрів

2.4 Нейронні мережі та глибоке навчання

Глибоке навчання (Deep Learning, DL) є одним із ключових напрямків сучасного штучного інтелекту. Це підрозділ машинного навчання, що використовує штучні нейронні мережі з великою кількістю шарів для аналізу даних та створення моделей, які можуть виконувати складні завдання. Воно базується на принципах ієрархічного навчання, де кожен шар мережі поступово виділяє все складніші ознаки із вхідних даних (рис. 2.11).

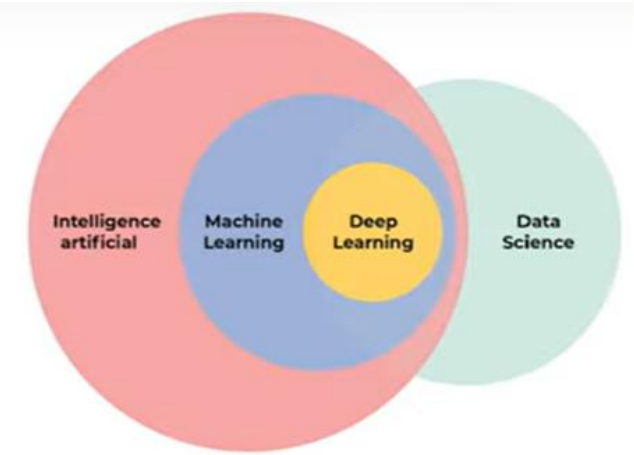

Рис. 2.11 Місце Deep Learning в системі Artificial Intelligence

Основними характеристиками глибокого навчання є: глибина мережі, автоматичне вилучення ознак, використання великих даних, потреба в обчислювальних ресурсах.

Нейронні мережі містять багато (іноді сотні або тисячі) шарів, кожен з яких виконує обчислення та передає результат далі. Глибокі моделі можуть самостійно навчатися витягати важливі характеристики з даних, на відміну від традиційного машинного навчання, де ознаки задаються вручну. Ефективність глибоких моделей значно зростає при роботі з великими обсягами даних. Глибоке навчання вимагає потужних апаратних засобів, таких як графічні процесори (GPU) або тензорні процесори (TPU).

Нейронна мережа - це обчислювальна модель, яка за побудовою та функціонуванням подібна біологічним нейронам у мозку людини. Вона використовується для обробки даних, виявлення закономірностей та вирішення

складних задач у різних галузях, таких як машинне навчання, глибоке навчання та штучний інтелект.

Глибокі нейронні мережі (рис. 2.12) складаються з:

- вхідного шару, який приймає дані (наприклад, пікселі зображення);
- прихованих шарів, які обчислюють проміжні ознаки (наприклад, контури або текстури);
- вихідного шару, який видає результат (наприклад, клас об'єкту або передбачення).

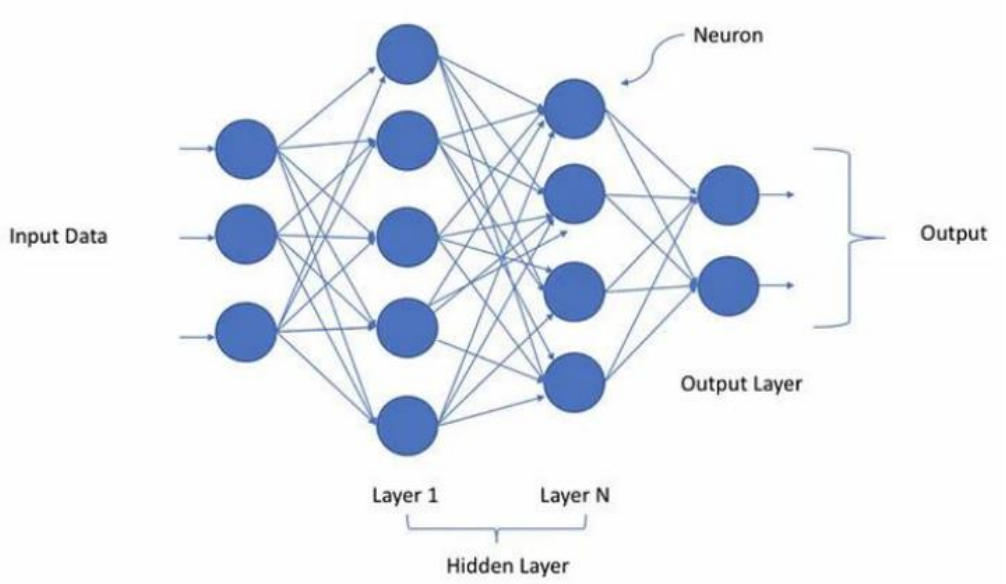

Рис. 2.12 Приклад глибокої нейронної мережі

Штучні нейрони - це базові елементи нейронної мережі. Кожен нейрон приймає кілька вхідних сигналів, обчислює зважену суму цих сигналів, додає зсув (bias) і передає результат через функцію активації.

Кожен нейрон з’єднаний із нейронами наступного шару через ваги (weights). Ваги визначають, наскільки сильний вплив має один нейрон на інший.

Функццція активації визначає чи буде активований нейрон, тобто чи передасть він сигнал далі.

Дані проходять через мережу від вхідного шару до вихідного, поступово обчислюючись у кожному шарі. На кожному етапі виконуються математичні операції, які перетворюють вхідні дані у більш корисні для задачі (рис. 2.13).

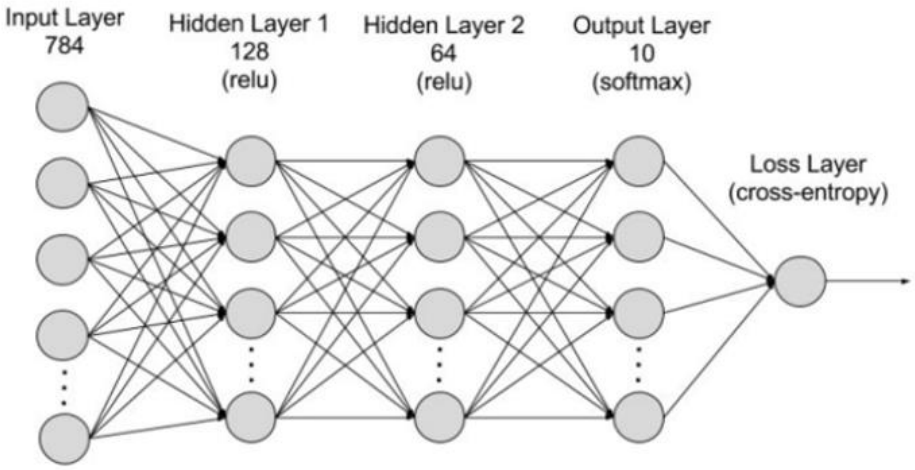

Рис. 2.13 – Структура штучної нейромережі

Для навчання нейронної мережі використовується набір даних для налаштування вагів і зсувів у мережі. Основний метод навчання - це зворотне поширення помилки (backpropagation), яке працює разом з алгоритмом оптимізації (наприклад, градієнтний спуск). Для визначення похибки (loss) обчислюється різниця між прогнозом мережі та реальними значеннями. Завданням навчання є мінімізувати цю похибку.

Спрощена модель біологічного нейрона, яка імітує процес прийняття рішень на основі вхідних даних, називається *прецептроном*. Він здатний виконувати прості завдання класифікації, наприклад, відокремлювати дані, які можна розділити прямою лінією.

До структури прецептрону входять:

- вхідні дані - набір значень x_1, x_2, ..., x_n, що відповідає характеристикам або ознакам об'єкту;
- ваги (w_1, w_2, ..., w_n), які відповідають важливості кожного вхідного сигналу;
- зсув (*bias*, b) додається для зсуву розділяючої гіперплощини, забезпечуючи більшу гнучкість.

Лінійна комбінація обчислюється за формулою:

$$z = \sum_{i=1}^{n} w_i x_i + b \tag{2.4}$$

Функція активації:

- якщо $z > 0$, вихід $y = 1$;
- якщо $z \leq 0$, вихід $y = 0$.

Основні принципи прецептрону інтегровані у багатошарові перцептрони (MLP), які здатні розв'язувати нелінійні задачі, інші типи нейронних мереж (згорткові, рекурентні тощо), де ідеї зважування вхідних сигналів і функцій активації є ключовими (рис. 2.14).

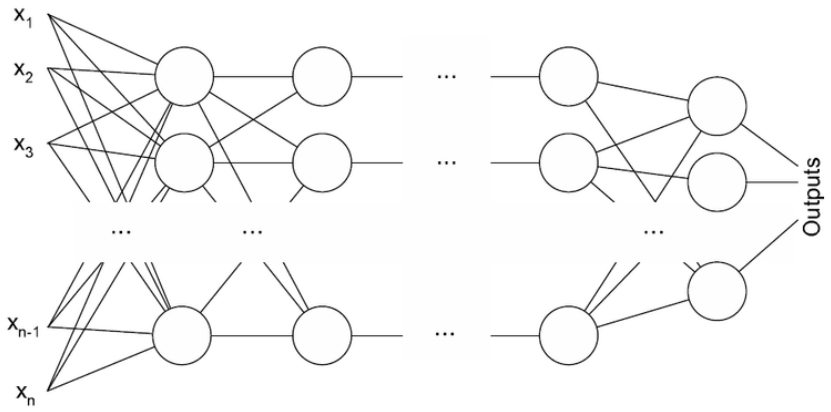

Рис. 2.14 Архітектура багатошарового прецептрону

Перцептрон — це основа, яка дала поштовх до розробки більш складних і потужних нейронних мереж. Його головна роль - забезпечення базових ідей навчання та обчислення, які були значно вдосконалені у сучасних технологіях глибокого навчання.

Згорткові нейронні мережі (Convolutional Neural Networks, CNNs) - це спеціалізований тип нейронних мереж, розроблений для роботи з даними, що мають просторову структуру, наприклад, зображення або відео. Їх основна особливість - використання операції згортки для автоматичного вилучення ознак з вхідних даних (рис. 2.15).

CNN складається з кількох типів шарів, кожен з яких виконує певну функцію. Основна операція згортковий шару (Convolutional Layer) – згортка. Вхідне зображення згортається із застосуванням фільтрів (ядра згортки). Фільтри виділяють локальні ознаки, наприклад, краї, текстури або форми. Кожен фільтр

переміщується по зображенню, створюючи активаційну карту (feature map), яка представляє виявлені ознаки.

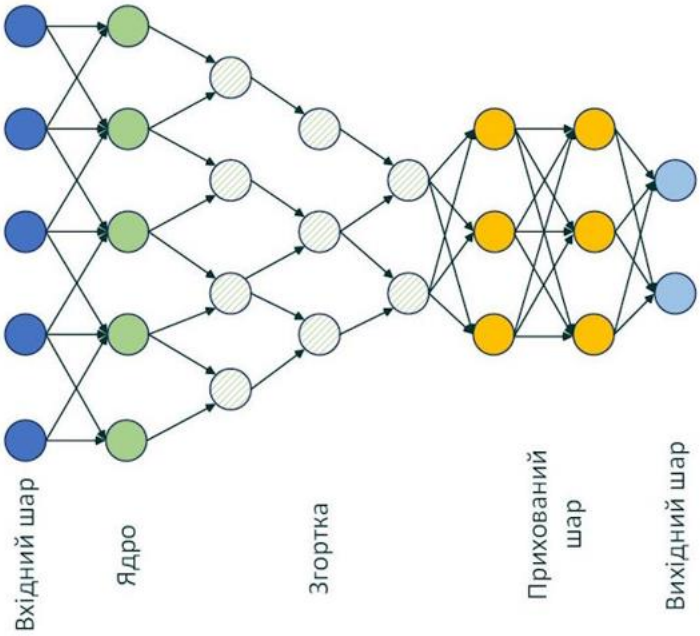

Рис. 2.15 Згорткова нейронна мережа

Шар підвибірки (Pooling Layer) використовується для зменшення розміру активаційної карти, що зменшує кількість параметрів і обчислювальну складність. Підвибірка зберігає важливі ознаки, водночас відкидаючи дрібні деталі.

Шар активації застосовує нелінійну функцію, наприклад, *ReLU* (Rectified Linear Unit), щоб зробити модель здатною працювати з нелінійними даними:

$$ReLU(x) = max(0, x) \tag{2.5}$$

Повнозв'язний шар (Fully Connected Layer) використовується наприкінці мережі для отримання кінцевого результату. Всі нейрони цього шару з'єднані з усіма нейронами попереднього шару. Видає результати класифікації або прогнозів.

Згорткові нейронні мережі є основою сучасного комп'ютерного зору та використовуються у багатьох додатках, де потрібно аналізувати візуальні дані.

Рекурентні нейронні мережі (*Recurrent Neural Networks, RNNs*) — це тип нейронних мереж, розроблений для роботи з послідовними даними. Їх основна особливість - здатність враховувати залежності між елементами послідовності,

зберігати контекст попередніх кроків і використовувати його для обробки поточного кроку (рис. 2.16).

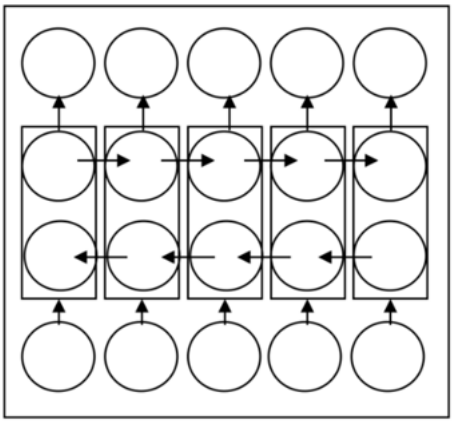

Рис. 2.16 Рекурентна нейронна мережа

Звичайні нейронні мережі працюють із незалежними входами, тобто кожен вхід обробляється окремо. RNN враховують, що дані можуть бути пов’язані між собою (наприклад, у тексті кожне слово залежить від попередніх). Для цього вони мають петлі (циклічні зв’язки), які дозволяють інформації передаватися від одного кроку до іншого.

На кожному часовому кроці *(t)* мережа отримує:

- поточний вхід x_t (елемент послідовності);
- стан мережі з попереднього кроку h_{t-1}, який зберігає інформацію про попередні елементи послідовності.

Математично це має наступний вигляд:

$$h_t = f(W_h h_{t-1} + W_x x_t + b) \quad , \tag{2.6}$$

де h_t - стан на поточному кроці;

W_h - матриця ваг для попереднього стану;

W_x - матриця ваг для поточного входу;

b – зсув;

f - функція активації (зазвичай *tanh* або *ReLU*).

На виході RNN може видавати:

- на кожному кроці (наприклад, прогнозування наступного слова);
- після обробки всієї послідовності (наприклад, класифікація тексту).

Автоенкодери (Autoencoders) - це тип штучних нейронних мереж, що використовуються для навчання ефективного представлення даних (зазвичай у зменшеному або стиснутому вигляді). Основна ідея автоенкодера - навчитись відновлювати вхідні дані після їх стискання, виділяючи найважливіші ознаки (2.17).

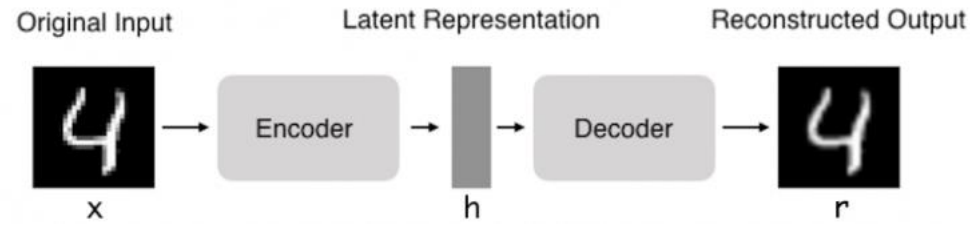

Рис. 2.17 Autoencoders

Автоенкодер складається з трьох основних компонентів: енкодер (Encoder), латентний простір (Latent Space), декодер (Decoder). Енкодер компресує вхідні дані до латентного простору. Латентний простір - простір зменшеного розміру, де зберігається інформація про головні ознаки вхідних даних. Декодер відновлює вхідні дані із стислого представлення.

Переваги автоенкодерів: автоматичне вилучення корисних ознак; гнучкість у порівнянні з класичними методами зменшення розмірності; можливість працювати з нелінійними залежностями. Ефективність автоенкодера залежить від правильного вибору гіперпараметрів та розмірів латентного простору. Автоенкодери - це потужний інструмент у машинному навчанні, який дозволяє ефективно працювати із стиснутими представленнями даних, виділяти їх основні ознаки та застосовувати їх у широкому спектрі завдань: від зменшення розмірності до генерації нових даних.

Генеративно-змагальні мережі (Generative Adversarial Networks, GANs) - це клас нейронних мереж, які використовуються для генерації нових даних, схожих на оригінальні. GAN складається з двох основних компонентів — генератора і дискримінатора, які "змагаються" між собою у процесі навчання (рис. 2.18). Мета GAN полягає в тому, щоб генератор створював дані, які настільки схожі на реальні, щоб дискримінатор не зміг їх відрізнити від справжніх.

Генератор створює нові дані на основі випадкового шуму. Мета генератора – “обдурити” дискримінатор, змушуючи його приймати згенеровані дані за справжні. Дискримінатор класифікує дані як "реальні" (з навчального набору) або "згенеровані" (створені генератором). Мета дискримінатора - правильно розпізнавати, які дані реальні, а які підроблені.

Рис. 2.18 Генеративно-змагальна мережа

Генератор отримує випадковий шум (зазвичай із нормального або рівномірного розподілу) як вхід, перетворює шум на згенеровані дані (наприклад, зображення, текст або звук). Дискримінатор отримує як реальні, так і згенеровані дані, видає оцінку (ймовірність) є дані реальними чи підробленими.

Генератор покращується, якщо дискримінатор неправильно класифікує згенеровані дані. Дискримінатор покращується, якщо він правильно класифікує реальні й згенеровані дані. Цикл повторюється, поки генератор не навчиться створювати дані, які дискримінатор майже не може відрізнити від реальних.

Типи генеративних нейронних мереж представлені у табл. 2.4.

Таблиця 2.4

Типи генеративних нейронних мереж

Тип	Ключове поняття	Переваги	Недоліки
Варіаційний автоенкодер	Структура кодувальника-декодера за допомогою імовірнісних графічних моделей, де нижня межа максимізується на логарифмічній імовірності даних.	Одночасно виконує генерацію і логічний висновок із моделюванням прихованих змінних	Зразки зображень, згенеровані VAE, мають тенденцію бути трохи розмитими
Генеративні змагальні мережі	Структура генератора-дискримінатора через змагальну навчальну гру, в якій зразки даних генеруються безпосередньо	Створює різкі зразки зображень	Складніше оптимізувати через нестабільну динаміку тренування
Авторегресивні мережі	Факторизується спільний розподіл даних в умовних розподілах, моделюючи кожний окремий вимір з урахуванням попередніх вимірювань.	Просте і стабільне навчання, що забезпечує найкращу логарифмічну ймовірність	NLM неефективні під час вибірки і не можуть легко забезпечити низькорозмірні функції

GAN використовуються для генерації зображень, для поліпшення роздільної здатності зображень, для генерації тексту, відео та анмацій. Можуть використовуватися для генерація медичних знімків для навчання моделей, коли доступ до реальних даних обмежений, для аудіо-генерації - створення музики або синтезу голосу.

Мережі GAN забезпечують генерацію високоякісних даних, які виглядають реалістично, мають широкий спектр застосувань. При цьому навчання GAN може бути складним і вимагати точного налаштування. Генератор може зосередитися на обмеженому наборі варіантів і не створювати різноманітні дані. GAN вимагають значних ресурсів для ефективного навчання.

Завдяки своїй здатності генерувати реалістичні дані, генеративно-змагальні мережі знайшли застосування у багатьох галузях, включаючи комп'ютерний зір, обробку тексту, аудіо та багато іншого. Їх постійний розвиток відкриває нові можливості для використання у науці та індустрії.

3 АНАЛІЗ ТА ОБРОБКА ДАНИХ ІЗ ВИКОРИСТАННЯМ МЕТОДІВ МАШИННОГО НАВЧАННЯ

3.1 Підготовка даних для обробки та аналізу, попередня обробка даних

Методика в аналізі та обробці даних засобами машинного навчання - це структурований підхід, що описує послідовність дій, методів і інструментів, які використовуються для ефективного опрацювання даних з метою вирішення конкретних задач за допомогою алгоритмів машинного навчання. Методика включає всі етапи роботи з даними - від збору і підготовки до побудови, тестування та оцінки моделі.

Ключовими складовими методики є збір даних, попередня обробка та аналіз даних, вибір алгоритму машинного навчання, навчання моделі, оцінка якості моделі, інтерпретація результатів, для більш складних проєктів - реалізація моделі на практиці.

Збір даних включає визначення джерел даних (баз даних, веб-сайтів, датчиків та інших джерел), Формування набору даних, необхідного для розв'язання задачі.

Попередня обробка даних складається з наступних етапів: очищення даних від пропусків, дублікатів або некоректних значень; масштабування числових змінних (нормалізація, стандартизація); кодування категоріальних змінних (наприклад, One-Hot Encoding); розділення даних на навчальну, тестову та валідаційну вибірки.

Аналіз даних полягає у дослідженні структури, розподілу та особливостей даних за допомогою статистичних методів. Потрібно виконати візуалізацію даних для пошуку патернів і трендів, а також виявити кореляції між ознаками.

У процесі вибору алгоритму машинного навчання необхідно підібрати модель, яка підходить для таких задач, наприклад, як задача класифікації, регресії, кластеризації та інші. У процесі навчання моделі необхідне використання навчальної вибірки для оптимізації параметрів моделі, Використання навчальної вибірки для оптимізації параметрів моделі.

Оцінка якості моделі полягає у визначенні точності, повноти, F1-міри, середньоквадратичної похибки тощо, у порівнянні результатів з іншими моделями або базовими метриками. Інтерпретація результатів полягає в аналізі роботи моделі, тобто у визначенні, які ознаки впливають на результат, а також у побудові звітів і візуалізацій, які допомагають зрозуміти отримані результати.

Для реалізації моделі на практиці необхідно виконати інтеграцію моделі ML у програмне забезпечення або бізнес-процеси. Потрібно, щоб підтримувалося постійне оновлення моделі при надходженні нових даних.

В якості прикладу для досліджень реалізуємо задачу класифікації [5]. Для реалізації моделі ML виберемо мову програмування Python, так як вона має величезний набір бібліотек та фреймворків, які значно полегшують роботу з машинним навчанням, найбільш популярними з яких є:

- *Scikit-learn;*
- *TensorFlow, Keras;*
- *PyTorch;*
- *Pandas;*
- *NumPy, SciPy;*
- *Matplotlib, Seaborn.*

Бібліотека Scikit-learn добре підходить для стандартних алгоритмів машинного навчання (класифікація, регресія, кластеризація) [6-9]. Фреймворки TensorFlow і Keras доцільно використовувати для глибокого навчання, побудови нейронних мереж. PyTorch також є потужним фреймворком для глибокого навчання. Бібліотеку Pandas доцільно використовувати для обробки даних. Бібліотеки NumPy, SciPy добре справляються з числовими обчисленнями і обробкою великих масивів даних. Бібліотеки Matplotlib, Seaborn актуально використовувати для візуалізації даних. Ці бібліотеки оптимізовані для швидкої та ефективної роботи, що значно спрощує виконання складних завдань.

В якості середовища розробки було вибрано Jupyter Notebook, яке є одним із найбільш популярних середовищ для розробки моделей машинного навчання

завдяки своїм унікальним перевагам, які забезпечують зручність, гнучкість і ефективність на різних етапах розробки моделей.

Jupyter Notebook дозволяє виконувати код по черзі, зберігаючи всі проміжні результати в одному середовищі. Це дуже корисно для машинного навчання, де можна поетапно перевіряти різні гіпотези, перевіряти дані, тренувати моделі та тестувати їх без необхідності запускати всю програму від початку.

На першому етапі завдання потрібно завантажити необхідні бібліотеки. Вибрано бібліотеки NumPy, Scikit-learn.

Бібліотека *NumPy* є однією з ключових інструментів у машинному навчанні та аналізі даних. Вона забезпечує ефективну роботу з багатовимірними масивами чисел та пропонує потужний набір математичних функцій, які полегшують реалізацію алгоритмів машинного навчання. Масиви NumPy (numpy.ndarray) оптимізовані для роботи з великими обсягами даних, що важливо для машинного навчання. NumPy дозволяє створювати та маніпулювати багатовимірними масивами (матрицями), що є основою для роботи з векторами та тензорами. Підтримка операцій додавання, множення, транспонування матриць, є ключовою для багатьох алгоритмів. Обробка та нормалізація даних: масштабування, перетворення та агрегація виконуються швидше та ефективніше. Використання векторизованих операцій дозволяє уникнути використання повільних циклів Python. NumPy - це основний інструмент для ефективної роботи з числовими даними у машинному навчанні. Без неї реалізація алгоритмів була б набагато складнішою та менш ефективною.

Бібліотека *Scikit-learn* є однією з найпопулярніших у сфері машинного навчання, оскільки забезпечує простий у використанні набір інструментів для побудови, тренування та оцінки моделей. Вона побудована на основі NumPy, SciPy та Matplotlib, і надає потужний функціонал для вирішення широкого спектру задач машинного навчання. Scikit-learn включає реалізації багатьох популярних алгоритмів машинного навчання, таких як:

- *класифікація*: SVM (метод опорних векторів), K-найближчих сусідів (KNN), Random Forest, логістична регресія;

- *регресія*: лінійна, поліноміальна регресії, Lasso, Ridge, ElasticNet;
- *кластеризація*: K-means, DBSCAN, агломеративна кластеризація;
- *зменшення розмірності*: PCA (метод головних компонент), t-SNE;
- *ансамблеві методи*: Random Forest, Gradient Boosting, AdaBoost.

Саме реалізація цих алгоритмів робить доцільним використання бібліотеки Scikit-learn у нашій задачі. Це дозволяє швидко експериментувати з різними підходами. Scikit-learn надає інтуїтивно зрозумілий інтерфейс із уніфікованим API для всіх моделей. Бібліотека Scikit-learn включає інструменти для підготовки даних: масштабування та нормалізацію, кодування категоріальних змінних, обробку пропущених значень, фільтрацію важливих ознак. Надає широкий набір метрик для оцінки якості моделей.

Scikit-learn включає вбудовані набори даних для тестування, такі як: Iris, Boston Housing, MNIST.

Виходячи з вищесказаного, можна зробити висновок, що Scikit-learn є універсальним інструментом для швидкого прототипування та впровадження моделей машинного навчання. Завдяки багатому функціоналу, простоті використання та інтеграції з іншими бібліотеками, він ідеально підходить для початківців і досвідчених фахівців.

На наступному етапі потрібно завантажити датасет, з яким будемо працювати. В якості датасету будемо використовувати датасет Iris.

Датасет Iris (або "Ірис Фішера") є одним із найпопулярніших навчальних наборів даних у машинному навчанні, який часто використовується для демонстрації та тестування алгоритмів класифікації. Цей набір даних є класичним прикладом для вирішення задач класифікації, оскільки він простий для розуміння та візуалізації, але водночас досить показовий. Датасет описує морфологічні характеристики квітів трьох різних видів ірису: iris-setosa, iris-versicolor, iris-virginica.

У кожному зразку представлено чотири числові ознаки (властивості), які характеризують фізичні розміри рослини:
- довжина чашолистка (sepal length, у сантиметрах);

- ширина чашолистка (sepal width, у сантиметрах);
- довжина пелюстки (petal length, у сантиметрах);
- ширина пелюстки (petal width, у сантиметрах).

Остання колонка є міткою (цільовим класом), яка визначає, до якого виду належить рослина.

Датасет Iris можна використовувати для класифікації, візуалізації, тестування моделей логістичної регресії, K-найближчих сусідів, SVM, дерева рішень, Random Forest.

Виконаємо характеристику датасету Iris:

- *кількість зразків*: 150 (по 50 для кожного виду);
- *кількість ознак*: 4;
- *кількість класів*: 3 (назви видів ірису);
- *тип ознак*: числові, всі значення у сантиметрах;
- *розподіл даних*: рівномірний (по 50 зразків для кожного класу).

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

Завантажуємо дані:

```
iris = load_iris()
```

Перегляд даних_1:

```
iris
```

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
        [4.9, 3. , 1.4, 0.2],
        [4.7, 3.2, 1.3, 0.2],
        [4.6, 3.1, 1.5, 0.2],
        [5. , 3.6, 1.4, 0.2],
        [5.4, 3.9, 1.7, 0.4],
        [4.6, 3.4, 1.4, 0.3],
        [5. , 3.4, 1.5, 0.2],
        [4.4, 2.9, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.1],
        [5.4, 3.7, 1.5, 0.2],
        [4.8, 3.4, 1.6, 0.2],
        [4.8, 3. , 1.4, 0.1],
        [4.3, 3. , 1.1, 0.1],
        [5.8, 4. , 1.2, 0.2],
```

Перегляд даних_2:

```
iris.data
```

```
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
```

```
iris.target
```

```
array([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, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

Iris.target - це атрибут об'єкта, який містить цільові значення (мітки класів) для кожного зразка у наборі даних.

```
iris.target_names
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
X = iris.data
y = iris.target
```

Команда *iris.target_names* повертає масив із назвами класів (видів рослин), представлених у наборі даних. Цей атрибут використовується для перетворення числових міток із *iris.target* у зрозумілі текстові назви. Числові мітки 0, 1 і 2 у *iris.target* відповідають таким видам ірису:

- 0: ‘setosa’;
- 1: ‘versicolor’;
- 2: ‘virginica’.

Крім чотирьох властивостей, які характеризують фізичні розміри рослини, до кожного рядку є відповідь – до якого класу відноситься екземпляр: 0, 1 або 2. Відповідь до першого рядка (5.1, 3.5, 1.4, 0.2) буде клас 0, тобто цей iris буде вигляду setosa.

Якщо ми в коді проскролимо нижче, візьмемо останній рядок з чотирьох чисел (5.9, 3, 5.1, 1.8), то такі параметри матимуть клас 2 і це вид iris, який називається

virginica. Параметри, які розміщуються по середині і мають значення 1 – це клас versicolor.

```
iris
        [6.9, 3.1, 5.4, 2.1],
        [6.7, 3.1, 5.6, 2.4],
        [6.9, 3.1, 5.1, 2.3],
        [5.8, 2.7, 5.1, 1.9],
        [6.8, 3.2, 5.9, 2.3],
        [6.7, 3.3, 5.7, 2.5],
        [6.7, 3. , 5.2, 2.3],
        [6.3, 2.5, 5. , 1.9],
        [6.5, 3. , 5.2, 2. ],
        [6.2, 3.4, 5.4, 2.3],
        [5.9, 3. , 5.1, 1.8]]),
 'target': array([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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]),
 'frame': None,
```

Задачею є написати таку модель, що якби у нас на вході були б також 4 числа (але ми не знаємо який саме це вид iris, так як у нас є тільки 4 числа), наша модель дала відповідь, а який же це клас: 0, 1 або 2. Це якраз і є *задачею класифікації*.

Потрібно написати модель, яка вловить закономірність і по новим чотирьом числам буде давати відповідь: це буде 0 або 1, або 2. Треба знайти такі спеціальні коефіцієнти до кожного класу, щоб при множенні цих коефіцієнтів на ці числа і записуванні їх у логістичну функцію ми отримали якомога більше значень. Тому що ці значення – це ймовірність, що наш зразок належить до конкретного класу. Треба для кожного класу підібрати такі коефіцієнти, щоб для правильних класів це були якомога вищі значення, а для неправильних класів – значення було близьке до нуля. І потім, коли нам випадає новий зразок (ми ще не знаємо до якого саме класу він належить), то, щоб знайти відповідь, який же саме це клас, ми маємо перемножити значення *x*-сів на наші коефіцієнти Θ і подивитися, яке значення ймовірності для кожного класу. Якщо:

$$\sigma(t) = \frac{1}{1+e^{-t}} \quad , \tag{2.7}$$

де *σ(t)* – логістична функція (сигмоїда);

t – це $x^T\Theta$, ми отримаємо велике число для якогось класу, значить ми говоримо, що цей зразок належить цьому класу. Якщо значення буде низьке, то будемо перевіряти якісь інші класи і виберемо той, де значення буде найвище. Тобто, треба для кожного класу визначити коефіцієнти, за допомогою цих коефіцієнтів ми можемо визначити, який це саме клас. А коефіцієнти ці шукаються за формулою (2.8):

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\log(\acute{p}^{(i)}) + (1 - y^{(i)})\log(1 - \acute{p}^{(i)})) \quad (2.8)$$

Потрібно визначити значення *X* та *y*:

```
X = iris.data
y = iris.target
```

3.2 Розділення даних на тренувальну і тестову вибірки. Вибір алгоритму навчання моделі ML

Потрібно розділити датасет на тренувальну і тестову вибірки:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

Розділяємо дані та *train* і *test* – для цього нам потрібно виконати модуль *train_test_split.*

Train_test_split – є дані, ми на частині тренуємося, а на частині перевіряємо як наша модель працює. Тобто це модуль, який дозволить розділити наші дані на навчальні та тренувальні.

X пишеться з великої літери, тому що це матриця, а *y* з маленької, тому що це просто вектор. І тепер ми використовуємо нашу функцію *train_test_split* і передаємо параметри. Ми вставляємо *X*, вставляємо *y* і маємо вказати, який ще ми хочемо *test_size*. Зазвичай ставимо десь 0.2, 0.3. На 80% даних ми будемо нашу модель навчати, на 20% даних ми будемо нашу модель перевіряти. *Random_state = 42* (прийнято проставляти значення 42, а можна й інше число, при цьому все буде працювати). Від цього *random_state* залежить, як саме наша функція

поділить наші дані. Тому що вона вибирає дані випадково, щоб позбутися всяких можливих залежностей (спочатку одні дані, потім інші). У нас це все поділиться випадково.

Виконуємо у коді вищевказану команду, після чого можна подивитися на *X_train*:

```
X_train
```

```
array([[4.6, 3.6, 1. , 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [6.7, 3.1, 4.4, 1.4],
       [4.8, 3.4, 1.6, 0.2],
       [4.4, 3.2, 1.3, 0.2],
       [6.3, 2.5, 5. , 1.9],
       [6.4, 3.2, 4.5, 1.5],
       [5.2, 3.5, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.2, 4.1, 1.5, 0.1],
       [5.8, 2.7, 5.1, 1.9],
```

Random_state – початкова змінна, на основі якої відбувається рандомізація на цьому етапі. Якщо *Random_state* змінити на *50* – нічого не поміняється, тільки рядки будуть різні. Можна прибрати *random_state* і все також буде працювати. Але якщо ми зафіксуємо це значення і потім наш код перевиконаємо, то воно поділиться так як і перший раз. Тому часто пишуть, щоб коли перезапустити, щоб там нічого не збилося, бо якщо збивається рандомом, значить там все не дуже добре працює.

```
len(X_train)
```

```
120
```

```
len(X_test)
```

```
30
```

Для вирішення задачі класифікації доцільно використовувати наступні алгоритми: Naive Bayes, Decision Trees, Logistic Regression.

Алгоритм логістичної регресії - це один з найбільш популярних методів для вирішення задач класифікації, зокрема для бінарної класифікації, де мета — передбачити категорію (класи) на основі вхідних даних. Логістична регресія використовується для передбачення ймовірностей належності до певного класу, а не конкретного класу, як, наприклад, у випадку лінійної регресії.

Логістична регресія базується на застосуванні логістичної (сигмоподібної) функції, яка трансформує лінійну комбінацію вхідних змінних у ймовірність. Модель використовує лінійне зважене поєднання вхідних ознак для обчислення ймовірності належності до одного з класів.

Логістична регресія починається з обчислення лінійної комбінації вхідних ознак (це основа для визначення того, до якого класу належить об'єкт):

$$z = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \ , \tag{2.9}$$

де w_0 - вільний параметр або зміщення (intercept);

$w_1, w_2, \ldots, w_n$ - коефіцієнти моделі, що вказують на важливість кожної ознаки x_i;

$x_1, x_2, \ldots, x_n$ - вхідні змінні або ознаки.

Оскільки ми хочемо отримати ймовірність того, що об'єкт належить до класу 1, ми використовуємо логістичну функцію (сигмоїду) для перетворення лінійної комбінації вхідних ознак у ймовірність:

$$P(y = 1|X) = \sigma(z) = \frac{1}{1 + e^{-z}} \ , \tag{2.10}$$

де $\sigma(z)$ - сигмоподібна функція, яка повертає значення між 0 і 1;

e^{-z} - експоненціальна функція для перетворення лінійної комбінації ознак.

Для того, щоб навчити модель логістичної регресії, потрібно мінімізувати функцію втрат, яка вимірює різницю між передбаченою ймовірністю $\acute{y}$ і фактичним значенням y. Для логістичної регресії зазвичай використовують логарифмічну функцію втрат (*log-loss* або крос-ентропію):

$$L(w) = -\frac{1}{m} \sum_{i=1}^{m} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \ , \tag{2.11}$$

де m - кількість об'єктів у навчальній вибірці;

y_i - фактичний клас для i-го зразка (0 або 1);

$\acute{y}_i$ - передбачена ймовірність для i-го зразка, обчислена за допомогою логістичної функції.

Переваги логістичної регресії:

- простота та швидкість виконання;
- легко інтерпретується (коефіцієнти моделі показують важливість ознак);
- добре працює для лінійно роздільних даних.

Недоліки логістичної регресії:

- погано справляється з нелінійними залежностями між ознаками;
- чутлива до сильно корельованих ознак.

Алгоритм логістичної регресії можна представити шістьома кроками.

1) На першому кроці відбувається ініціалізація параметрів. Початково задаємо параметри $w_0, w_1, \ldots, w_n$ (можна ініціалізувати їх випадковими значеннями або нулями).

2) Обчислення лінійної комбінації ознак. Обчислюємо $z = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$.

3) Застосування логістичної функції. Обчислюємо ймовірність належності до класу 1: $\acute{y} = \sigma(z)$.

4) Обчислення функції втрат. Оцінюємо, наскільки добре модель передбачає класи, використовуючи функцію втрат (*log-loss*).

5) Оновлення параметрів. Використовуємо метод градієнтного спуску для мінімізації функції втрат і оновлюємо параметри $w_0, w_1, \ldots, w_n$.

6) Перевірка на конвергенцію. Повторюємо кроки 2-5 до тих пір, поки зміни параметрів не стануть незначними (конвергенція).

3.3 Навчання та оцінка ML-моделі. Перевірка точності моделі за допомогою тестових даних

Тепер нам треба створити модель:

```
model = LogisticRegression()
```

Модель потрібно навчити. Щоб модель навчити, треба виконати *fit()* і передати *X_train* та *y_train*.

Метод *fit()* у Python є ключовим у бібліотеці Scikit-learn і використовується для навчання моделі. Коли ми викликаємо *fit()*, ми передаємо алгоритму дані, які він аналізує і підлаштовує свої параметри для подальшого використання (наприклад, передбачення). Метод *fit()* є основою для навчання моделей у Scikit-learn. Він адаптує алгоритм до ваших даних, після чого модель стає готовою до прогнозування або аналізу.

```
model.fit(X_train, y_train)
```

```
C:\Users\ASUS\anaconda3\Anakonda\lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to c
onverge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

Модель готова. Щоб поміряти метрики (точність, повнота), ми можемо порівняти їх з тим, що наша модель предиктить. Тобто ми створюємо нову змінну *y_predict* і тепер нам потрібно, щоб наша модель дала нам відповідь. Беремо *model.predict* і передаємо наші дані, які ми залишили для перевірки.

```
y_predict = model.predict(X_test)
```

```
y_predict
```

```
array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2,
       0, 2, 2, 2, 2, 2, 0, 0])
```

```
y_test
```

```
array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2,
       0, 2, 2, 2, 2, 2, 0, 0])
```

Розроблено модель, яка може повністю наші дані розбити – класифікувати. Дані співпадають, тому що це не складний датасет і навіть найпростіша модель може з точністю 100% робити класифікацію.

Передаємо на модель значення:

```
model.predict([[1,2,3,4]])
```

```
array([2])
```

Відповідь: 2. Якби у нас була рослина з такими показниками, вона б належала до класу 2.

Передаємо на модель інші значення:

```
model.predict([[4,7,3,4]])
```

```
array([0])
```

Відповідь: клас 0.

```
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Команда *classification_report* у Python використовується для створення текстового звіту з ключовими метриками оцінки якості моделі класифікації. Цей звіт включає такі показники, як точність (*precision*), повнота (*recall*), F1-оцінка і кількість зразків для кожного класу. Classification_report допомагає подивитися як добре працює наш класифікатор.

Функція *classification_report* є частиною модуля sklearn.metrics і дозволяє оцінити, наскільки добре модель класифікує дані на основі тестових міток та передбачених результатів.

Ключові показники у звіті:

1) *Precision* (точність) - частка правильно передбачених зразків певного класу від усіх зразків, передбачених як цей клас:

$$Precision = \frac{TP}{TP + FP} \quad , \tag{2.9}$$

де *TP* - істинно позитивні передбачення;

FP - хибно позитивні передбачення.

2) *Recall* (повнота) - частка правильно передбачених зразків певного класу від усіх реальних зразків цього класу:

$$Recall = \frac{TP}{TP + FN} \quad , \tag{2.10}$$

де *FN* – хибно негативні передбачення.

3) *F1-score* - гармонійне середнє між точністю і повнотою. Є важливим, якщо потрібно знайти баланс між вищевказаними двома метриками:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.11}$$

4) Support - кількість реальних зразків у кожному класі.

Результат роботи моделі можемо оцінити по класовим та загальним метрикам. Класові метрики:

- для класу setosa точність, повнота, та F1-оцінка дорівнюють 1.00, що означає, що всі передбачення правильні;

- для класу versicolor точність, повнота, та F1-оцінка також дорівнюють 1.00. Це означає, що всі передбачення правильні;

- аналогічно для класу virginica всі показники становлять 1.00 - передбачення правильні.

Загальні метрики:

- *accuracy* - загальна точність моделі (відношення правильних передбачень до загальної кількості зразків);

- *macro avg* - середнє значення показників для всіх класів без урахування кількості зразків;

- *weighted avg* - середнє значення показників, зважене за кількістю зразків у кожному класі.

У звіті *classification_report* значення *support* показують кількість зразків у реальних даних (тобто у тестовому наборі *y_test*) для кожного класу.

У наведеному коді для набору даних Iris, де є три класи квітів (setosa, versicolor, virginica), значення support вказують на кількість реальних зразків у тестовому наборі для кожного з цих класів.

Для класу setosa: значення support =10 означає, що у тестовому наборі є 10 реальних зразків класу setosa. Для класу versicolor: значення support =9 означає, що є 9 реальних зразків класу versicolor. Для класу virginica: значення support = 11 означає, що є 11 реальних зразків класу virginica.

Підсумкові значення support:

- у рядку accuracy: значення support = 30 означає, що у тестовому наборі загалом є 30 зразків;
- у рядках macro avg і weighted avg: значення support = 30 також вказує на загальну кількість зразків у тестовому наборі, які використовуються для розрахунку середніх метрик.

Метрики precision, recall, f1-score обчислюються для кожного класу, але support просто вказує на кількість зразків у реальних мітках (*y_test*), без відношення до передбачень.

Таким чином, support — це фіксоване значення, яке визначається реальними мітками (*y_test*) і не залежить від передбачень моделі (*y_pred*).

Метрика *classification_report* є потужним інструментом для аналізу роботи моделі класифікації, дозволяючи детально оцінити її продуктивність для кожного класу.

ВИСНОВКИ

Машинне навчання (ML) надає потужні інструменти для обробки та аналізу даних, що дозволяє вирішувати складні задачі з високою точністю та швидкістю. Алгоритми ML можуть самостійно виявляти закономірності та робити прогнози. Це особливо корисно для роботи з даними, які постійно оновлюються або змінюються. ML дозволяє аналізувати великі набори даних швидко та ефективно. Машинне навчання робить аналіз даних більш точним, ефективним та масштабованим, відкриваючи нові можливості у багатьох галузях.

Основними типами машинного навчання на сьогоднішній день є класичне навчання, навчання з підкріпленням, ансамблеві методи навчання, нейронні мережі і глибоке навчання. Кожен тип навчання досліджений в рамках магістерської кваліфікаційної роботи.

Python є однією з найпопулярніших мов програмування для машинного навчання через низку причин, які роблять її особливо зручною для роботи з даними, алгоритмами та моделями. Python має велику кількість готових бібліотек для машинного навчання та обробки даних, які значно спрощують розробку ML-проєктів. У межах даної роботи були використані бібліотеки Scikit-learn та NumPy.

Методика в аналізі та обробці даних засобами машинного навчання - це структурований підхід, що описує послідовність дій, методів і інструментів, які використовуються для ефективного опрацювання даних з метою вирішення конкретних задач за допомогою алгоритмів машинного навчання. Методика включає всі етапи роботи з даними - від збору і підготовки до побудови, тестування та оцінки моделі.

Процес розв’язання задачі класифікації у магістерській кваліфікаційній роботі базувався на наступній методиці: підготовка даних, розділення даних, вибір алгоритму, навчання моделі, оцінка моделі.

Підготовка даних включала збір датасету Iris, який містив приклади об'єктів з відомими класами, попередню обробку даних виконувати не потрібно було, так як дані у датасеті Iris були нормалізованими й у них не було пропусків.

Дані було розділено на тренувальну та тестувальну вибірки із здійсненням вибору алгоритму для моделі ML. Проаналізувавши основні алгоритми машинного навчання для задач класифікації, такі як алгоритм наївного Байєсу, дерева рішень, логістична регресії, в якості алгоритму було обрано логістичну регресію. Логістична регресія, наївний Байєс і дерева рішень є популярними алгоритмами машинного навчання для задач класифікації. Вибір між ними залежав від специфіки даних та задачі. Алгоритм логістичної регресії був вибраний, так як він безпосередньо моделює ймовірності приналежності до певного класу, що дає можливість отримувати інтуїтивно зрозумілі результати.

Логістична регресія забезпечує коефіцієнти, які дозволяють оцінити вплив кожної ознаки на результат. Це робить її зручною для задач, де важлива інтерпретація моделі. Логістична регресія працює добре навіть на невеликих наборах даних за умови, що немає високої кількості ознак, добре узагальнює результати на нових даних, якщо ознаки мають лінійний вплив.

Було здійснено навчання ML-моделі на тренувальних даних. Результати роботи моделі було оцінено по класовим метрикам. Для класів setosa, versicolor та virginica показники точності, повноти, та F1-оцінки становили 1.00, що означає, що всі передбачення були правильними.

Розроблена методика дозволить ефективно обробляти великі обсяги даних, що є особливо важливим значенням для сучасних задач Big Data та автоматизації процесів у бізнесі. Отримані результати підтверджують ефективність запропонованих підходів, що може слугувати основою для подальших досліджень.

ПЕРЕЛІК ПОСИЛАНЬ

1. http://www.learnpython.org/en/Welcome
2. http://realpython.com/python3-object-oriented-programming/
3. http://leetcode.com/problemset/all/
4. http://perso.limsi.fr/pointal/
5. http://media/python:cours:mementopython3-english.pdf
6. https://numpy.org/
7. https://pandas.pydata.org/
8. https://matplotlib.org/
9. https://scipy.org/
10. https://scikit-learn.org/stable/
11. https://keras.io/
12. https://pytorch.org/
13. https://www.tensorflow.org/
14. Mozaffari, M., Saad, W., Bennis, M., Nam, Y.-H., Debbah, M. A tutorial on UAVs for wireless networks: Applications challenges and open problems. IEEE Commun. Surveys Tuts., 21(3), 2019. — 2334-2360 с.
15. Wang, Y., Ru, Z.-Y., Wang, K., Huang, P.-Q. Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing. IEEE Trans. Cybern., 50(9), 2020. — 3984-3997 с.
16. Zhang, J., Zhou, L., Tang, Q., Ngai, E. C.-H., Hu, X., Zhao, H., et al. Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing. IEEE Internet Things J., 6(2), 2019. — 3688-3699 с.
17. Nguyen, V., Khanh, T. T., Nam, P. V., Thu, N. T., Hong, C. S., Huh, E.-N. Towards flying mobile edge computing. Proc. Int. Conf. Inf. Netw. (ICOIN), Jan. 2020. — 723-725 с.
18. Zhou, F., Hu, R. Q., Li, Z., Wang, Y. Mobile edge computing in unmanned aerial vehicle networks. IEEE Wireless Commun., 27(1), 2020. — 140-146 с.

19. Mishra, D., Natalizio, E. A survey on cellular-connected UAVs: Design challenges enabling 5G/B5G innovations and experimental advancements. Comput. Netw., 182, Dec. 2020.

20. W. Yu, W. Cheng, C. Aggarwal, K. Zhang, H. Chen, and Wei Wang. NetWalk: A flexible deep embedding approach for anomaly Detection in dynamic networks, ACM KDD Conference, 2018.

21. Jake VanderPlas Python Data Science Handbook: Essential Tools for Working with Data, 2022, 551p.

22. Heller, Nicholas et al. (2020). The KiTS19 Challenge Data: 300 Kidney Tumor Cases with Clinical Context, CT Semantic Segmentations, and Surgical Outcomes. arXiv: 1904. 00445 [q-bio.QM].

23. Hollo, Kaspar (2019). Exploring the Value of Weakly-Supervised Deep Learning Approaches for Artefact Segmentation in Brightfield Microscopy Images. URL: https://comserv. cs . ut . ee / home / files / hollo _ softwareengineering _ 2021.

24. Wang, Haofan et al. (2020). Score-CAM: Score-Weighted Visual Explanations for Convo-lutional Neural Networks. arXiv: 1910.01279 [cs.CV].

25. Yang, Guanyu et al. (2020). “Weakly-supervised convolutional neural networks ofrenal tumor segmentation in abdominal CTA images”. In: BMC Medical Imaging 20.1, p. 37. ISSN: 1471-2342. DOI: 10.1186/s12880- 020- 00435- w. URL: https://doi.org/10.1186/s12880-020-00435-w.

26. Selvaraju, Ramprasaath R. et al. (2019). “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. In: International Journal of ComputerVision 128.2, pp. 336–359. DOI: 10 . 1007 / s11263 - 019 - 01228 - 7. URL: https://doi.org/10.1007\%2Fs11263-019-01228-7.

27. Лосєв М.Ю. Програмування мовою Python: навчальний посібник / М. Ю. Лосєв, В. М. Федорченко. – Харків, - Львів: Видавництво ПП «Новий Світ - 2000», 2023. – 178 с.

28. Золотухіна О.А., Негоденко О.В., Резник С.Ю., Разіна С.Я. Якість та тестування інформаційних систем. Навчальний посібник підготовлено до друку

для самостійної роботи студентів вищих навчальних закладів. Київ: ННІІТ ДУТ, 2020. – 128 с.

29. Зінченко О.В., Фесенко М.А., Березівський М.Ю., Кисіль Т.М. Технології смарт-систем. – Навчальний посібник. – К.: ДУІКТ, 2023. – 162 с.

30. Нікольський Ю. В., Пасічник В. В., Щербина Ю. М. Системи штучного інтелекту: навчальний посібник. – Львів: «Магнолія-2006», 2024. – 279 с.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)