ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «МОДЕЛЬ МАШИННОГО НАВЧАННЯ НА ОСНОВІ SUPERVISED LEARNING»

на здобуття освітнього ступеня бакалавра
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

_____ *Олександр КОРКУШКО*

Виконав:
здобувач вищої освіти
група ІСД-42
Олександр КОРКУШКО

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

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

Київ – 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій

Ступінь вищої освіти Бакалавр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ

_____Каміла СТОРЧАК
«____» _____20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ СТУДЕНТУ
Коркушко Олександру Олександровичу
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: «Модель машинного навчання на основі Supervised Learning»

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

затверджені наказом вищого навчального закладу від «24» лютого 2025 року № 56.

2. Строк подання кваліфікаційної роботи: 30 травня 2025 року.

3. Вихідні дані до кваліфікаційної роботи: датасет Iris;
бібліотеки Python: Sklearn, NumPy;
Random_state = 42;
типи 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. Дослідження особливостей машинного навчання.

2. Дослідження підходу Supervised Learning у машинному навчанні.
3. Розробка моделі машинного навчання на основі Supervised Learning.

5. Перелік ілюстративного матеріалу: *презентація*

1. Складові машинного навчання. Типи даних.
2. Типи машинного навчання.
3. Приклад Supervised learning.
4. Алгоритми класифікації.
5. Порівняння мов програмування Python та R у машинному навчанні.
6. Вибір середовища розробки моделі ML.
7. Бібліотеки Python для машинного навчання.
8. Розробка моделі машинного навчання на основі Supervised Learning.
9. Навчання та оцінка якості ML-моделі.

6. Дата видачі завдання: 24 лютого 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	24.02 – 28.02.25	
2	Вивчення класифікації методами ML/DL	03.03 – 07.03.25	
3	Дослідження особливостей машинного навчання	10.03 – 14.03.25	
4	Дослідження алгоритмів класифікації	17.03 – 28.03.25	
5	Особливосі навчання з учителем (Supervised Learning)	31.03 – 10.04.25	
6	Оцінка якості роботи моделі ML	11.04 – 24.04.25	
7	Оформлення роботи: вступ, висновки, реферат	25.04 – 14.05.25	
8	Розробка демонстраційних матеріалів	15.05 – 30.05.25	

Здобувач вищої освіти _____ Олександр КОРКУШКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

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

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 60 стор., 22 рис., 11 табл., 20 джерел.

Мета роботи - розробка моделі машинного навчання на основі Supervised Learning.

Об'єкт дослідження – процес розробки та навчання моделі ML на основі Supervised Learning.

Предмет дослідження – методи, алгоритми та підходи до побудови моделі машинного навчання з учителем.

Короткий зміст роботи: Досліджені методи машинного навчання для аналізу та обробки даних, типи даних, алгоритми, відмінності різних типів ML, методи ML для вирішення задач класифікації в мережі. Досліджені особливості машинного навчання, популярні бібліотеки та фреймворки для роботи з технологіями ML. Обгрунтовано вибір мови програмування. Здійснено обробку і аналіз даних з використанням засобів ML, розроблено модель ML для вирішення задачі класифікації в інтерактивному середовищі Jupyter Notebook. Здійснено вибір алгоритму навчання моделі ML. Виконано навчання та оцінку ML-моделі, здійснено перевірку точності моделі ML за допомогою тестових даних.

КЛЮЧОВІ СЛОВА: МАШИННЕ НАВЧАННЯ, МОВА ПРОГРАМУВАННЯ PYTHON, ВІЗУАЛІЗАЦІЯ, ТЕХНОЛОГІЯ, АЛГОРИТМ, СОКЕТ, ПРОТОКОЛ, ІНТЕРАКТИВНЕ СЕРЕДОВИЩЕ, ІНФОРМАЦІЙНА СИСТЕМА, МОДЕЛЬ.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі машинне навчання (ML) є однією з ключових технологій, що активно використовується у фінансовому аналізі, медицині, кібербезпеці, автоматизації бізнес-процесів, розпізнаванні образів та багатьох інших сферах. Серед методів машинного навчання підхід Supervised Learning (навчання з учителем) є одним із найпоширеніших і ефективних, оскільки дозволяє створювати високоточні моделі для класифікації, регресії та прогнозування на основі розмічених даних.

Однак, розробка ефективної моделі Supervised Learning вимагає правильного вибору алгоритмів, налаштування гіперпараметрів, обробки даних та оцінки якості навчання. Актуальність дослідження полягає в аналізі сучасних підходів до побудови таких моделей, оптимізації їхньої продуктивності та впровадженні найкращих практик машинного навчання. Результати роботи можуть бути використані для покращення точності прогнозування у різних сферах, що підтверджує практичну значущість дослідження.

Метою роботи є розробка моделі машинного навчання на основі Supervised Learning.

Для досягнення поставленої мети необхідно виконати наступні *завдання*:

- дослідити особливості машинного навчання (ML);
- дослідити особливості класичного машинного навчання із вчителем (Supervised Learning);
- дослідити інструменти для аналізу та обробки даних, фреймворки для роботи з технологіями ML;
- здійснити вибір середовища розробки моделі ML;
- розробити модель ML на основі Supervised Learning;
- виконати оцінку якості роботи моделі машинного навчання.

Об'єкт дослідження – процес розробки та навчання моделі ML на основі Supervised Learning.

Предметом дослідження є методи, алгоритми та підходи до побудови моделі машинного навчання з учителем.

Методи дослідження: аналіз літературних джерел та огляд існуючих рішень; метод моделювання; методи математичної статистики; метод комп'ютерного моделювання.

Наукова новизна отриманих результатів: розроблено та проаналізовано модель машинного навчання на основі Supervised Learning з урахуванням сучасних алгоритмів та методів оптимізації.

Практична значущість кваліфікаційної бакалаврської роботи. Результати дослідження можуть бути використані для розробки та впровадження моделей машинного навчання у різних сферах, де важлива автоматична обробка даних. Запропонована модель на основі Supervised Learning дозволяє підвищити точність та ефективність прогнозування, оптимізувати процес прийняття рішень та автоматизувати аналіз великих обсягів інформації. Також, результати можуть бути використані як навчальний матеріал для студентів, дослідників і розробників, які працюють у сфері штучного інтелекту та аналізу даних.

Апробація результатів та публікації:

1. Коркушко О. О. «Оцінка якості моделей машинного навчання». Тези доповіді на VI Міжнародній науково-технічній конференції «Сучасний стан та перспективи розвитку IoT». – Київ, 15 квітня 2025 року.

1 ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ МАШИННОГО НАВЧАННЯ

1.1 Штучний інтелект та його похідні

Штучний інтелект (Artificial Intelligence, AI), машинне навчання (Machine Learning, ML) і глибоке навчання (Deep Learning, DL) - це взаємопов'язані, але різні поняття в галузі комп’ютерних наук.

Штучний інтелект - це широка галузь, яка охоплює всі технології, що дозволяють комп’ютерам виконувати завдання, які зазвичай потребують людського інтелекту. AI включає алгоритми, правила, логіку, машинне навчання, обробку природної мови, робототехніку тощо, наприклад, голосові помічники, чат-боти, рекомендаційні системи (Netflix, YouTube).

Машинне навчання - це підгалузь AI, яка використовує алгоритми для аналізу даних і самостійного навчання на основі досвіду. Машина не просто програмується, а вчиться знаходити закономірності в даних. ML поділяється на типи, основними з яких є: класичне навчання, навчання з підкріпленням, ансамблі, нейромережі і глибоке навчання. У свою чергу класичне навчання поділяється на два підтипи: навчання із вчителем, яке ще називають наглядовим навчанням (Supervised Learning) та навчання без вчителя (без нагляду - Unsupervised Learning) [1-4].

Supervised Learning – це навчання, при якому модель навчається на мічених даних (наприклад, розпізнавання облич у фото).

Unsupervised Learning шукає структуру в немічених даних (наприклад, кластеризація клієнтів у маркетингу).

Навчання з підкріпленням (Reinforcement Learning) – тип навчання, де агент навчається взаємодії з середовищем (наприклад, навчання гри в шахи).

Глибоке навчання - це підгалузь машинного навчання, яка використовує штучні нейронні мережі, на кшталт роботи людського мозку. Глибокі нейронні мережі можуть мати сотні шарів та обробляти великі обсяги даних. Глибокі нейромережі використовуються у комп’ютерному зорі (розпізнавання

облич, автономні автомобілі); для обробки природної мови (Google Translate, ChatGPT); у генеративних моделях (створення тексту, зображень, музики).

Якщо досліджувати, як пов’язані штучний інтелект, машинне навчання та глибоке навчання, можна визначити, що штучний інтелект містить машинне навчання, а машинне навчання включає глибоке навчання (рис. 1.1).

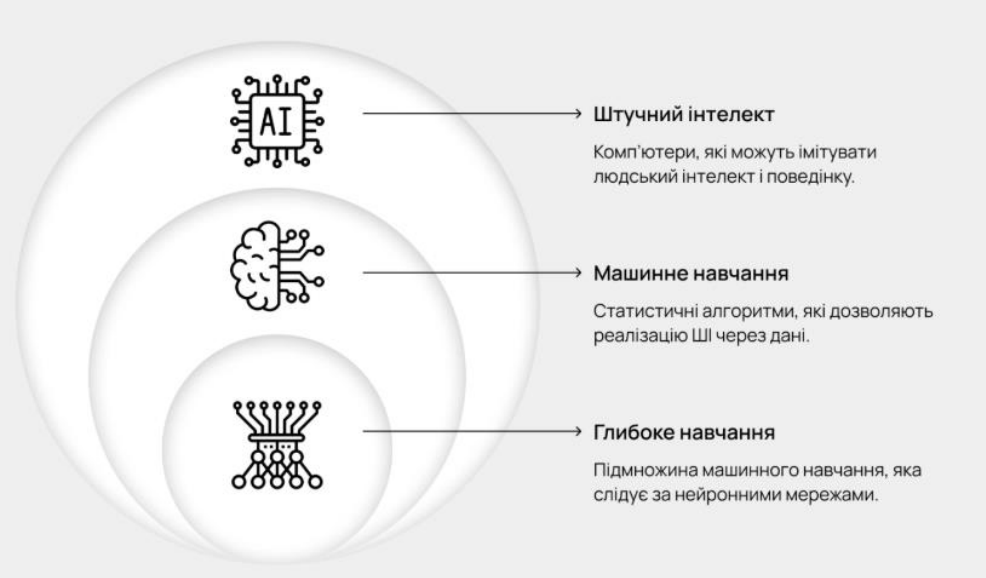


Рис. 1.1 Взаємозв’язок штучного інтелекту, машинного та глибокого навчання

Глибоке навчання – це найпотужніший, але й найвимогливіший підхід у AI. Приводячи, наприклад, аналогію з грою у шахи, можна сказати що, AI - якби комп’ютер міг грати в шахи; ML - коли комп’ютер вчиться грати в шахи, аналізуючи партії; DL - коли комп’ютер використовує нейронні мережі, як AlphaGo і перемагає чемпіонів [5, 6].

Отже, під штучним інтелектом слід розуміти програмно-технічні засоби, здатні адаптуватися і донавчатися в процесі своєї експлуатації. Донавчатися мається на увазі змінюватися. Машинне навчання – технологія, призначена для створення моделей машинного навчання. Моделі машинного навчання – це моделі, які навчаються по навчальним датасетам і які не змінюють результатів своєї роботи в процесі експлуатації. Глибоке навчання - технологія роботи з нейронними мережами в основному.

Основна мета машинного навчання - навчити комп'ютерні системи автоматично знаходити закономірності в даних і робити прогнози або приймати рішення без явного програмування. Головне завдання машинного навчання - зробити так, щоб комп'ютерна система самостійно навчалася з даних і покращувала свої результати без постійного втручання програмістів. У табл. 1.1 приведені більш конкретні цілі ML.

Таблиця 1.1

Конкретні цілі ML

№ п/п	Цілі ML	Приклад застосування
1	Автоматизація складних завдань	Автоматичне сортування електронної пошти (розпізнавання спаму)
2	Аналіз та прогнозування	Прогнозування цін на акції або погоди на основі історичних даних
3	Розпізнавання та класифікація	Розпізнавання облич у фотографіях або діагностика хвороб за рентгенівськими знімками
4	Кластеризація та пошук схожих елементів	Рекомендація фільмів на Netflix, групування клієнтів у маркетингу
5	Оптимізація рішень	Оптимізація логістичних маршрутів або управління ресурсами на виробництві
6	Взаємодія з людьми (Natural Language Processing, NLP)	Голосові помічники або чат-боти, які розуміють мову
7	Навчання агентів через підкріплення	Створення систем штучного інтелекту для гри в шахи або керування самокерованими автомобілями

1.2 Визначення складових машинного навчання

Машинне навчання за вхідними даними має передбачати результат. Для того, щоб результат передбачення був точнішим, дані, на базі яких відбуваються передбачення, мають бути різноманітними і їх має бути багато. Для машинного навчання важливими є наступні складові: дані, ознаки, алгоритми [7-10].

Машинне навчання - це як навчання людини, але замість книг і досвіду комп'ютер використовує дані. Без даних алгоритм нічому не навчиться – так

само, як людина не зможе навчитися читати без літер або малювати без засобів для малювання (фарб, олівців, фломастерів). Чим більше якісних даних - тим кращі результати. Якщо навчати систему на поганих або неправильних даних, вона зробить помилки (наприклад, якщо навчити її розпізнавати котів за картинками собак, вона буде плутатися). Дані відіграють важливу роль для навчання, перевірки та прийняття рішень.

У процесі навчання система повинна аналізувати дані, знаходити в них закономірності та запам’ятовувати їх. Після навчання систему перевіряють на нових даних, щоб оцінити її точність. І прийняття рішень – це коли на основі побачених даних система робить прогнози або розпізнає об’єкти (наприклад, визначає чи є хвороба на знімку).

Якщо, наприклад, ми хочемо навчити комп’ютер розпізнавати яблука і банани, то даними будуть тисячі фото яблук і бананів; навчання – це коли алгоритм аналізуватиме кольори, форми, текстури; перевіркою буде те, що система отримуватиме нове фото та визначатиме, що на ньому – яблуко або банан. Чим більше правильних та різноманітних фото ми дамо, тим кращим буде результат. *Дані* - це основа всього машинного навчання. Без них система просто не знатиме, що робити [11].

Дані, які використовуються у машинному навчанні можна розділити на структуровані, неструктуровані, напівструктуровані та часові ряди. Структуровані дані - це організовані дані, які можна легко зберігати у таблицях або базах даних (табл. 1.2).

Таблиця 1.2

Структуровані дані

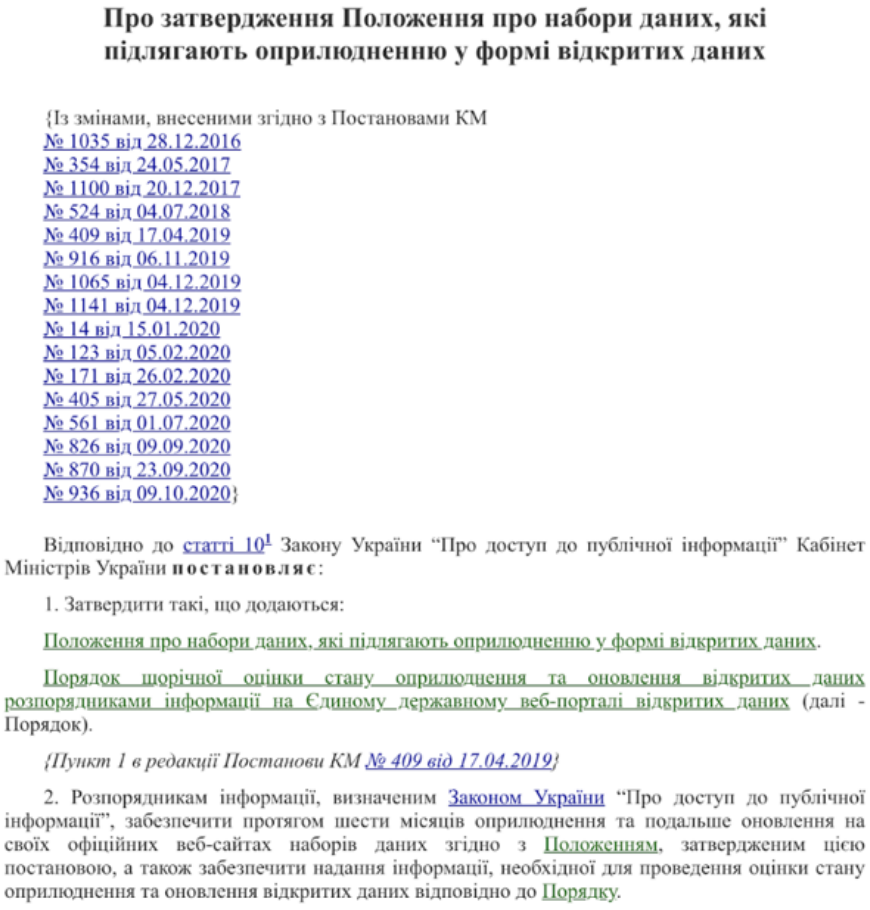
Місто	**Підприємство**	**Дата**	**Прибуток**
Київ	ДП «Комунсервіс»	2024-05-01	15503148
Київ	ДП «Комунсервіс»	2024-03-30	1834126
Київ	ДП «Комунсервіс»	2024-02-15	1950456
Львів	ДП «Львівкартон»	2024-10-18	1675362
Львів	ДП «Львівкартон»	2024-07-08	946194
Харків	ДП «Льодова арена»	2024-02-23	2453173
Харків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

Структуровані дані складаються з рядків і стовпців, де кожна колонка – це певна ознака. Ці дані можуть бути числовими або категорійними [12-15].

Неструктуровані дані – це неорганізовані або сирі дані, які не можна легко представити у вигляді таблиці. Для їх аналізу потрібні спеціальні алгоритми обробки, наприклад, нейронні мережі, обробка зображень. До неструктурованих даних належать (рис. 1.2):

- текстові дані (повідомлення, відгуки, статті);
- зображення (фотографії, рентгенівські знімки);
- аудіо (голосові команди, музика);
- відео (записи з камер, фільми).

Методами обробки неструктурованих даних є технологія NLP (Natural Language Processing), яка використовується для аналізу тексту і комп'ютерний зір (Computer Vision) - для розпізнавання зображень.



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 **постановляє**:

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. 1.2 Неструктуровані дані

Дані ще можуть бути напівструктурованими – це дані, які мають часткову структуру – вони не такі чітко організовані, як таблиці, але все ж мають певні правила. Прикладами частково структурованих даних є формати для збереження даних у мережі Інтернет (JSON, XML) та логи серверів, які містять події у вигляді записів із часовими мітками (рис. 1.3).

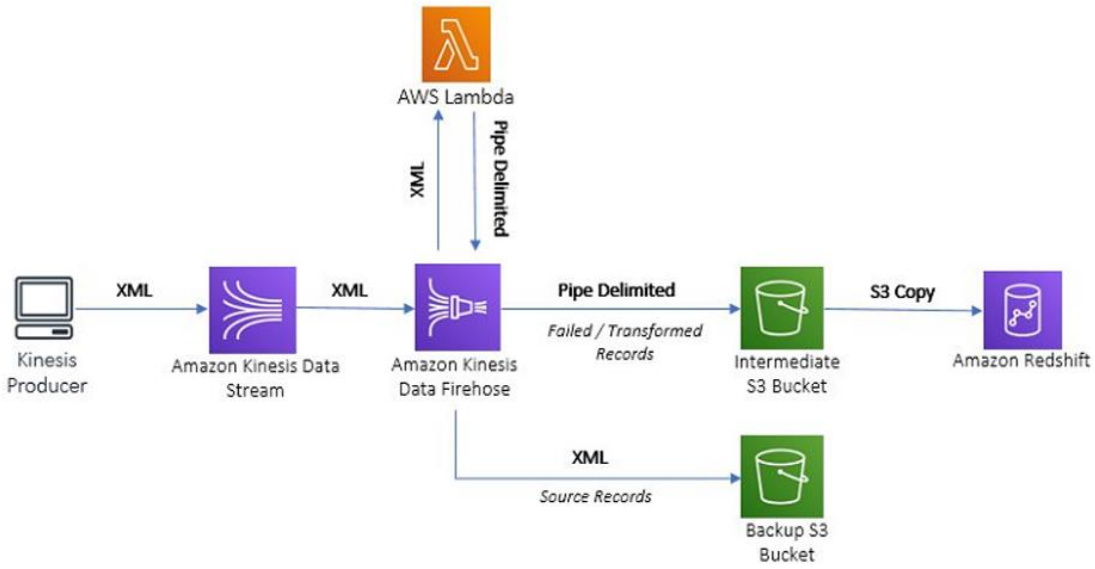


Рис. 1.3 Представлення даних JSON та XML

Часові ряди - це дані, зібрані в певні моменти часу, де порядок спостережень має значення, тобто це дані з часовою залежністю. Відмінна риса таких даних – залежність від часу, так як значення у майбутньому часто залежать від попередніх спостережень [16].

Прикладами часових рядів є: біржові котирування, погодні дані, запити в Google за днями, навантаження на сервер, серцевий ритм у медицині.

Часовий ряд – це набір даних, в якому є одна важлива деталь – часовий лаг. Часовий лаг – це певний період або в секундах, або у хвилинах, або у днях, місяцях, кварталах, але він там присутній у вигляді якогось певного поля, де вказується, що там йде перелік по рокам.

Дані такого типу можна використовувати для прогнозування. Прогнозувати можна все, що завгодно: врожайність, захворюваність, динаміку курсу валют, економічні показники, тобто все, що завгодно, що у нас змінюється в часі. При

цьому в якості навчального датасету вибирається залежність між часовим лагом і певними шукомими показниками за відомі періоди. Іншими словами, ми беремо такий часовий ряд, навчаємо нашу модель на відомих нам даних часового ряду, а потім просимо її зробити предикцію вже на час невідомий. При чому ця предикція може бути зроблена як прогноз наперед так і ретраспективно назад, тобто у нас є певний проміжок і ми хочемо за допомогою цього проміжку з'ясувати, а що у нас було до цього проміжку, бо за минулі періоди у нас даних немає. Для цього і використовуються часові ряди [17].

Є наступні види часових рядів:

- одновимірні – тільки одна змінна змінюється з часом (наприклад, температура повітря);
- багатовимірні – враховують кілька змінних (наприклад, температуру, вологість, тиск);
- стаціонарні – статистичні характеристики (середнє, дисперсія) не змінюються з часом;
- нестаціонарні – характеристики змінюються (наприклад, тренд або сезонність).

Проаналізувавши різні типи даних, представимо їх у табл. 1.3.

Таблиця 1.3

Основні 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

У машинному навчанні *ознаки* (features) - це важливі характеристики або властивості, які допомагають алгоритму робити правильні висновки. Якщо представити, що ми навчаємо комп'ютер розрізняти яблука і банани, то дані - це

багато фотографій фруктів; ознаки - це характеристики, за якими їх можна відрізнити. Прикладами таких ознак можуть бути: колір (зелений, червоний, жовтий); форма (кругла або довга); розмір (великий або малий). Алгоритм використовуватиме ці ознаки, щоб навчитися визначати, який це фрукт.

Ознаки є важливою складовою машинного навчання, так як без правильних ознак алгоритм не зможе добре навчитися. Якщо ознаки погані або недостатні, система буде помилятися. А якщо вибрати занадто багато ознак, алгоритм може стати надто складним.

Наприклад, для передбачення цін на нерухомість ознаками можуть бути площа будинку, кількість кімнат, розташування. Для визначення фальшивих транзакцій ознаками можуть бути сума платежу, місце, час покупки, а для розпізнавання мови – тон голосу, швидкість вимови, акцент.

Ознаки – це те, що допомагає алгоритму зрозуміти дані і робити правильні прогнози. У табл. 1.4 приведені приклади ознак, які найчастіше використовуються в задачах машинного навчання.

Таблиця 1.4

Ознаки, які використовуються в ML-задачах

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

У машинному навчанні алгоритм – це набір правил або інструкцій, які допомагають комп’ютеру аналізувати дані, знаходити закономірності та приймати рішення. Алгоритм встановлює правила та навчається. Наприклад, якщо повернутися до простої задачі, яка використовувалася для пояснення розуміння даних та ознак, то можна зазначити, що якщо кругле та червоне – це, швидше за все, яблуко, а якщо жовтий та довгий – це банан.

Алгоритм сам знаходить зв’язки між ознаками та правильними відповідями. Алгоритми визначають, як саме модель навчається та приймає рішення. Алгоритми дозволяють машині узагальнювати знання – навіть якщо вона бачить новий об’єкт, вона все одно може зробити правильний прогноз. Вони впливають на швидкість і точність роботи моделі [18].

В якості прикладів алгоритмів машинного навчання, які досить часто використовуються у задачах, можна привести лінійну регресію, дерева рішень, нейронні мережі (рис. 1.4). Лінійна регресія передбачає, наприклад, ціну квартири на основі її площі. Дерева рішень допомагають у виборі, наприклад, кредитного ризику. Нейронні мережі використовуються у складних завданнях, наприклад, розпізнавання облич або самокеровані автомобілі.

Алгоритми – це "мозок" машинного навчання. Вони визначають, як саме система буде аналізувати дані та робити прогнози.

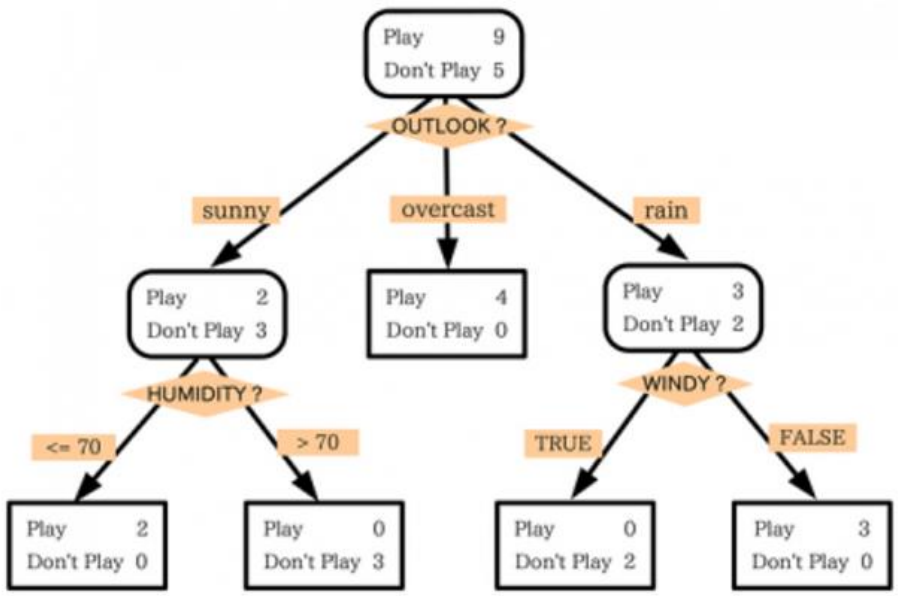


Рис. 1.4 Дерева рішень

1.3 Типи машинного навчання

Машинне навчання можна поділити на чотири основних типи в залежності від того, як модель отримує знання з даних: класичне навчання, навчання з

підкріпленням, ансамблеві методи, нейронні мережі та глибоке навчання (рис. 1.5).

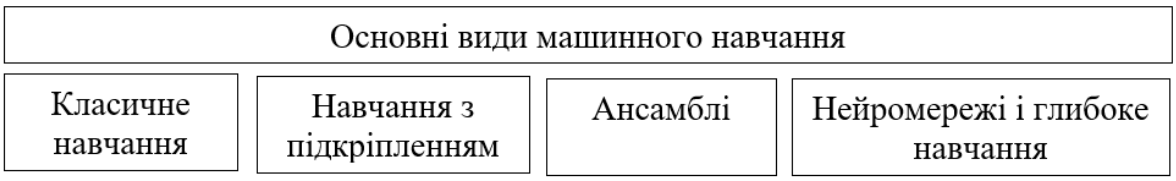


Рис. 1.5 Типи машинного навчання

Дослідження підходу Supervised Learning детально буде розглянуте у другому розділі бакалаврської кваліфікаційної роботи, на основі якого буде реалізована модель машинного навчання.

Навчання без вчителя – це тип машинного навчання, в якому модель працює з немічеными даними (тобто без правильних відповідей). Вона самостійно шукає закономірності та групує дані за схожістю.

Основні особливості навчання без вчителя:

- немає готових відповідей – модель сама вирішує, які групи або структури є в даних;
- відкриває приховані закономірності – допомагає знайти групи або зв’язки, які неочевидні для людини;
- корисне для великих обсягів даних – особливо коли їх важко розмічати вручну;
- застосовується у рекомендаційних системах, кластеризації, виявленні аномалій.

Принцип роботи навчання без вчителя полягає у тому, що модель отримує дані, аналізує ці дані, шукає схожі об’єкти, групує їх або знаходить закономірності. Основними методами навчання без вчителя є: кластеризація, зниження розмірності, асоціація [19, 20].

Ансамблі – це метод, при якому кілька моделей об’єднуються для отримання більш точного та надійного прогнозу, ніж одна окрема модель, так як декілька слабших моделей можуть працювати краще, ніж одна потужна. В ансамблевих методах кожна модель робить свій прогноз. Їхні результати комбінуються (середнє значення, голосування тощо). У підсумку результат має бути більш точним та стабільним.

Особливістю ансамблів є те, що вони зменшують похибку – знижують ризик, що одна модель зробить грубу помилку. Ансамблеві методи забезпечують менше переобчислення (overfitting), так як поєднання різних моделей дає більш узагальнене рішення. Навіть якщо одна модель трохи помиляється, загальний ансамбль цього не допустить, тому результати є стабільнішими.

Навчання з підкріпленням – це тип машинного навчання, у якому агент навчається через взаємодію із середовищем і отримує нагороду за правильні дії та штраф за неправильні. Модель не отримує готових відповідей, як у звичайному навчанні, а сама пробує різні дії, щоб досягти найкращого результату. Агент не має готових міток, а експериментує і навчається на власних помилках. У навччанні з підкріпленням потрібно пробувати нові дії (дослідження) і використовувати вже знайдені хороші стратегії (експлуатація). Агент може отримати меншу нагороду зараз, але більшу в майбутньому. Навчання з підкріпленням застосовується у складних завданнях, таких як ігри, робототехніка та фінанси (рис. 1.6).

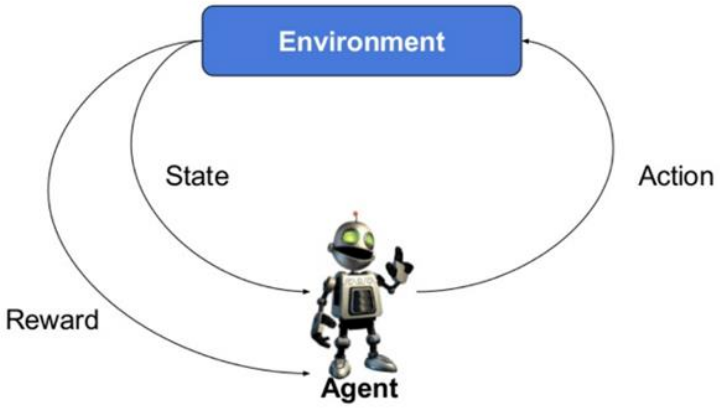


Рис. 1.6 Навчання з підкріпленням

Навчання з підкріпленням добре працює там, де потрібно вибирати найкращі дії у довгостроковій перспективі. RL може працювати у динамічному середовищі та адаптуватися до змін, допомагає автоматизувати прийняття рішень у складних системах і є потужним інструментом для задач, де потрібне автономне навчання та оптимізація дій.

2 ДОСЛІДЖЕННЯ ПІДХОДУ SUPERVISED LEARNING У МАШИННОМУ НАВЧАННІ

2.1 Визначення особливостей класичного навчання

У першому розділі бакалаврської кваліфікаційної роботи була розглянута класифікація основних типів машинного навчання, одним з яких є класичне навчання, яке у свою чергу поділяється на навчання із вчителем (Supervised Learning) та навчання без вчителя (Unsupervised Learning) (рис. 2.1).

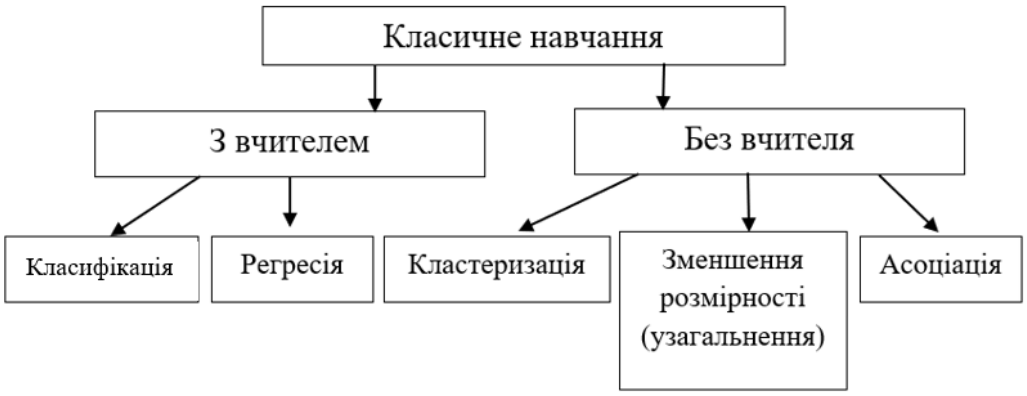


Рис. 2.1 Supervised Learning

Навчання із вчителем (Supervised Learning) – це підхід у машинному навчанні, при якому модель навчається на основі мічених даних, тобто кожен вхідний приклад має відповідну мітку (правильну відповідь).

Наприклад, якщо потрібно навчити дитину розпізнавати фрукти: яблука та апельсини. Для цього потрібно показати їй багато зображень фруктів і кожного разу сказати, який це фрукт. Через деякий час дитина навчиться самостійно розрізняти фрукти. Прикладом задачі Supervised Learning у машинному навчанні може бути приклад з медицини, де на сьогоднішній день активно впроваджуються методи ML. Якщо у нас є дані про пацієнтів (вік, стать, симптоми) та їхні діагнози, можна навчити алгоритм передбачати, чи є у нового пацієнта хвороба, використовуючи ці історичні дані.

Основні властивості навчання із вчителем:

- використання мічених даних – кожен приклад має відповідний вихід (наприклад, клас або числове значення);

- алгоритм навчається на основі закономірностей у даних;

- необхідна якісна вибірка – якщо вхідні дані містять помилки, модель також буде робити неправильні передбачення;

- підходить для двох основних типів задач: *класифікація* (Classification) – передбачення категорій (наприклад, спам/не спам); *регресія* (Regression) – передбачення числових значень (наприклад, прогнозування ціни нерухомості).

Навчання із вчителем - один із найпопулярніших підходів у машинному навчанні (рис. 2.2). Воно дозволяє створювати точні та ефективні алгоритми для передбачення майбутніх подій та автоматизації складних завдань. Однак воно потребує великих обсягів якісних даних, що є його головним обмеженням.

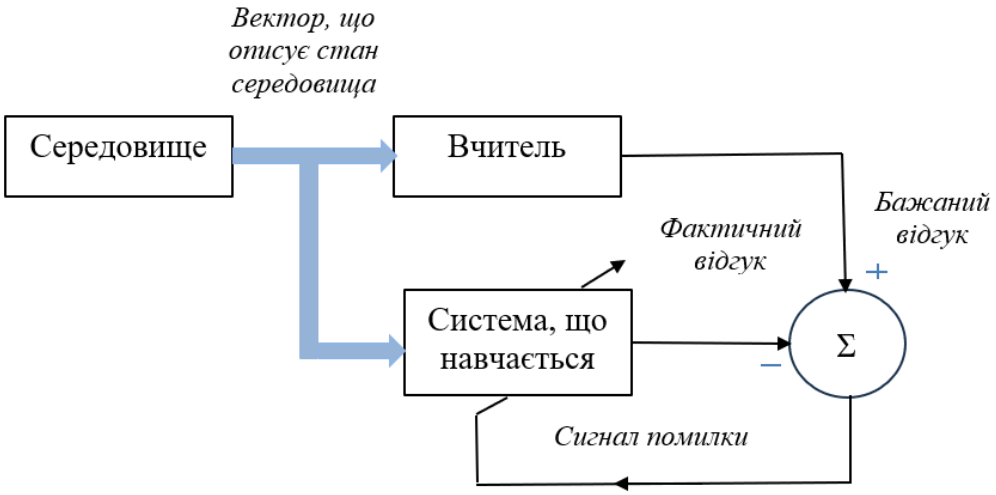


Рис. 2.2 Навчання із вчителем

Вчитель у машинному навчанні - це мітка або правильний результат, який алгоритм повинен передбачити. Це фактично правильна відповідь для кожного вхідного прикладу, яку модель має навчитися прогнозувати. У контексті цього підходу вчитель виступає як джерело навчання для моделі, допомагаючи їй коригувати свої помилки та покращувати точність.

Якщо необхідно навчити модель передбачати чи буде e-mail спамом, то мітка або правильна відповідь (вчитель) для кожного e-mail - це "спам" або "не спам".

Якщо необхідно передбачити ціну нерухомості, то мітка для кожного прикладу - це реальна ціна об'єкту.

Можна представити вчителя як керівника або наставника, який постійно надає правильні відповіді (мітки), щоб модель могла порівнювати свої передбачення з цими реальними результатами та коригувати свої стратегії для досягнення більш точної роботи у майбутньому. У цьому контексті вчитель не є окремою сутністю, а більше концептом, що позначає правильні мітки/відповіді, які використовуються для навчання моделі.

У табл. 2.1 приведені досліджені особливості машинного навчання із вчителем, які визначають, чому навчання із вчителем є популярним і ефективним методом у багатьох галузях ML.

Таблиця 2.1

Особливості машинного навчання із вчителем

№ п/п	*Елементи та етапи навчання*	*Особливість*
1	Мічені дані	Для навчання моделі використовуються дані, де кожен вхід має відповідну мітку або цільове значення.
2	Типи задач	- Класифікація - передбачення категорії (наприклад, чи є e-mail спамом). - Регресія - передбачення числового значення (наприклад, прогнозування ціни на нерухомість).
3	Залежність від якості даних	Якість моделі напряму залежить від якості та обсягу мічених даних. Помилки у даних можуть призвести до неправильних результатів.
4	Процес навчання	- Алгоритм аналізує вхідні дані та навчається знаходити закономірності між ознаками та мітками. - Використовуються методи оптимізації, такі як градієнтний спуск для мінімізації помилки.
5	Оцінка якості моделі	Після навчання модель перевіряється на тестових даних і використовуються різноманітні метрики, такі як Accuracy, Precision, Recall для класифікації, або MSE (середньоквадратична помилка) для регресії.
6	Прогнозування на нових даних	Після навчання модель може передбачати мітки для нових, невідомих даних, які не були використані під час навчання.
7	Швидкість навчання	Швидше навчання порівняно з іншими методами (наприклад, навчання з підкріпленням), через наявність чітких вхідних

Продовження таблиці 2.1

№ п/п	*Елементи та етапи навчання*	*Особливість*
		та вихідних значень.
8	Узагальнення	Моделі можуть бути добре узагальнені на нові дані, якщо вони були навчальні на достатньо різноманітних прикладах.
9	Необхідність у мічених даних	Для навчання потрібен великий набір мічених даних, що може бути трудомістким і дорогим процесом.
10	Обмеження щодо складних ситуацій	Погано справляється з новими ситуаціями, які сильно відрізняються від даних, на яких модель навчалася.
11	Алгоритми, які використовуються	- Для класифікації: логістична регресія, дерева рішень, SVM. - Для регресії: лінійна регресія, поліноміальна регресія, XGBoost.
12	Застосування	Фінансова сфера, медицина, рекомендаційні системи, розпізнавання образів, аналітика.

2.2 Задачі, які вирішуються на основі підходу Supervised Learning

До задач, які вирішуються на основі підходу Supervised Learning, відносяться:

- класифікація - у цій задачі модель має визначити, до якої категорії належить вхідний об’єкт;
- регресія - тип задач, який передбачає прогнозування числового значення на основі вхідних даних.

Приклади класифікації: визначення чи є e-mail спамом, розпізнавання рукописного тексту, передбачення діагнозу на основі медичних аналізів, класифікація зображень.

Важливими алгоритмами класифікації є:

- наївний Байєс (Naïve Bayes);
- логістична регресія (Logistic Regression);
- дерева рішень (Decision Trees);
- метод опорних векторів (SVM);
- нейронні мережі (Neural Networks);
- Random Forest;

- k-найближчих сусідів (k-NN).

Приклади регресії: прогнозування ціни нерухомості, передбачення погоди, оцінка прибутку компанії.

Алгоритмами регресії є:

- лінійна регресія (Linear Regression);
- поліноміальна регресія (Polynomial Regression);
- Ridge/Lasso Regression;
- метод найближчих сусідів (k-NN);
- градієнтний бустинг (XGBoost, LightGBM).

Процес навчання у Supervised Learning включає 4 кроки:

1) Збір та підготовку даних.
2) Вибір моделі.
3) Навчання моделі.
4) Оцінку якості моделі машинного навчання.
5) Впровадження та вдосконалення моделі ML.

У процесі збору та підготовки даних формується навчальна вибірка (містить вхідні ознаки та мітки), видаляються пропущені значення та аномальні дані, дані можуть нормалізуватися або стандартизуватися для кращої роботи моделі.

У процесі вибору моделі вибирається алгоритм на основі типу задачі (класифікація або регресія), наприклад, для передбачення ціни нерухомості підходить лінійна регресія, а для розпізнавання зображень – нейронні мережі.

Далі відбувається навчання моделі. Алгоритм аналізує вхідні ознаки та знаходить залежності, використовує градієнтний спуск або інші методи оптимізації для мінімізації помилки.

Після навчання модель перевіряється на тестових даних, які не використовувалися під час тренування. Для оцінки якості моделі ML використовуються метрики оцінки.

Для класифікації доцільно використовувати наступні метрики якості: Accuracy, Precision, Recall, F1-score. Для регресії - MSE (Mean Squared Error), R^2 (коефіцієнт детермінації). Якщо якість моделі недостатня, вона може бути доопрацьована

шляхом зміни алгоритму, додаванням більшої кількості даних. Якщо модель працює добре, її можна впровадити у реальне використання.

2.3 Методи оптимізації в задачах Supervised Learning

Математичний аналіз, зокрема інтегрування, похідні та частинні похідні відіграє важливу роль у машинному навчанні. У машинному навчанні основна мета - мінімізувати функцію втрат (наприклад, середньоквадратичну похибку або крос-ентропію). Для цього використовується градієнтний спуск, який базується на частинних похідних. Похідні показують напрямок найшвидшого зменшення функції втрат. Градієнт - це вектор частинних похідних, який використовується для оновлення вагів у нейронних мережах.

Інтегрування використовується у байєсівських методах та ймовірнісних моделях. Лапласіан та градієнтні методи допомагають у зменшенні перенавчання. Багато моделей машинного навчання базуються на ймовірнісних розподілах (наприклад, гауссові процеси, регресія). Інтегрування використовується для обчислення ймовірностей та очікувань. Частинні похідні застосовуються у згорткових нейронних мережах (CNN) для виявлення контурів у зображеннях. Градієнтні методи використовуються для згладжування сигналів та аналізу змін. Похідні допомагають аналізувати зміну параметрів у часі (наприклад, у навчанні з підкріпленням).

Математичний аналіз є основою машинного навчання, оскільки дозволяє будувати, оптимізувати та аналізувати моделі. Похідні використовуються для навчання моделей, інтеграли - для ймовірнісних методів, а частинні похідні - для складних багатовимірних проблем.

Методи оптимізації - це алгоритми, які використовуються для знаходження найкращих параметрів моделі шляхом мінімізації (або максимізації) функції втрат. Основна мета - налаштувати модель так, щоб вона узагальнювала дані якнайкраще. Розповсюдженим методом оптимізації, який часто застосовується для побудови моделей ML, є *градієнтний метод*. Градієнтні методи базуються на

обчисленні похідних функції втрат для коригування параметрів моделі і поділяються на:

- градієнтний спуск (Gradient Descent, GD);
- адаптивні градієнтні методи.

Градієнтний спуск – це метод оновлення вагів:

$$\theta = \theta - \alpha \nabla J(\theta) \ , \tag{2.1}$$

де Θ - параметри моделі;

α - швидкість навчання (learning rate);

▼$J(\Theta)$ - градієнт функції втрат.

Градієнтний спуск є методом оптимізації, який дозволяє знайти мінімум або максимум функції. Максимум нам не потрібний, так як нам не потрібно, щоб наша помилка MSE була червоною. Так що це наш мінімум. І ми можемо знайти мінімум або максимум за допомогою градієнта. Градієнт – це *n*-мірний вектор із частинних похідних.

Наприклад, наша функція *f* залежить від трьох змінних: *x, y, z.* І градієнт цієї функції *f* буде дорівнювати *n*-мірному вектору, тобто вектору розмірності 3 (тому що є три залежні змінні) і це частинні похідні. Було три змінних, стало три частинних похідних. Це і є градієнт. У градієнта є одна цікава властивість - він показує, куди потрібно рухатися, щоб функція зростала.

Типи градієнтного спуску:

- Batch Gradient Descent (BGD) – використовує всі дані, але є повільним;
- Stochastic Gradient Descent (SGD) - оновлює параметри після кожного прикладу;
- Mini-Batch Gradient Descent - компроміс між BGD і SGD, використовує невеликі пакети даних.

Адаптивні градієнтні методи поліпшують класичний градієнтний спуск за рахунок динамічної зміни швидкості навчання. Основними з них є:

- Momentum – враховує попередні градієнти, щоб уникати коливань;
- Adagrad – змінює швидкість навчання для кожного параметра.
- RMSprop – модифікація Adagrad для розв’язання проблеми згасання градієнтів.

- Adam (Adaptive Moment Estimation) - поєднує Momentum і RMSprop, найпоширеніший у глибокому навчанні.

Графічна репрезентація стохастичного градієнтного спуску (SGD) приведена на рис. 2.3.

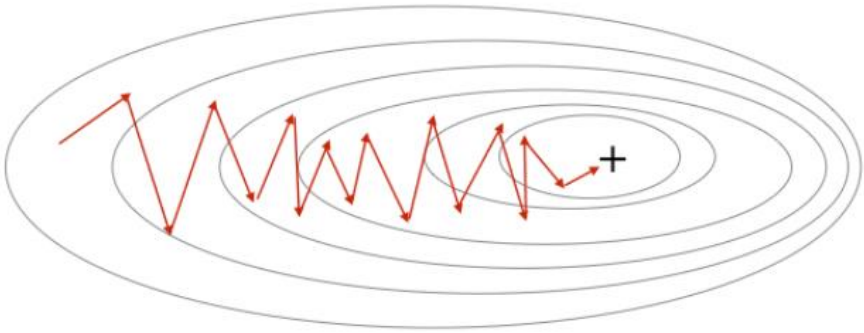


Рис. 2.3 Графічна репрезентація стохастичного градієнтного спуску (SGD)

Стохастичний градієнтний спуск (SGD) - це варіант градієнтного спуску, який оновлює параметри моделі після кожного окремого прикладу з навчальної вибірки. SGD використовує наступне оновлення параметрів:

$$\theta = \theta - \alpha \nabla J(\theta; x_i, y_i) \quad , \tag{2.2}$$

де Θ - параметри моделі (наприклад, ваги нейронної мережі);

α - швидкість навчання (learning rate);

$J(\Theta, x_i, y_i)$ - функція втрат, обчислена для одного випадкового зразка (x_i, y_i);

$\nabla J(\Theta, x_i, y_i)$ - градієнт функції втрат для цього зразка.

Недоліками стохастичного градієнтного спуску є: велика варіативність оновлень, можливість не сходження параметрів через шум, необхідність налаштування швидкості навчання для виключення коливання моделі. Перевагами SGD є те, що він підходить для великих даних, оскільки не потребує обробки всієї вибірки; завдяки випадковості може уникати локальних мінімумів; ефективний для онлайн-навчання (забезпечує адаптацію моделі у реальному часі).

Методи оптимізації є основою машинного навчання, яка визначає ефективність навчання моделей. У більшості випадків використовують Adam або SGD з варіаціями, але вибір методу залежить від конкретної задачі. SGD - основний

алгоритм оптимізації у машинному навчанні, особливо для глибоких нейронних мереж. Він швидкий, добре масштабується та допомагає уникати локальних мінімумів. Однак через шум він менш стабільний, тому часто використовується з методами покращення, такими як Momentum або Adam.

Теорія ймовірностей - розділ математики, що вивчає закономірності випадкових явищ: випадкові події, випадкові величини, їхні функції, властивості й операції над ними.

Теорія ймовірностей виникла з практичних потреб – необхідності передбачати випадкові події. Подією або випадком ми назвемо явище, яке може відбутися, а може і не відбутися. І хоча свій початок ця теорія бере з необхідності передбачати результати азартних ігор, подальший її розвиток привів до значних здобутків у теорії вимірювань, масового обслуговування, надійності. Операції над випадковими подіями приведені на рис. 2.4.

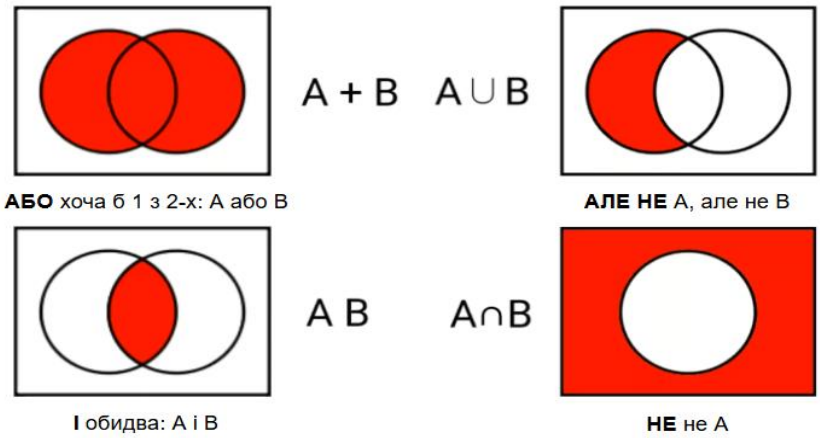


Рис. 2.4 Операції над випадковими подіями

Теорія ймовірності є основою багатьох алгоритмів машинного навчання, оскільки вона дозволяє працювати з невизначеністю, стохастичними процесами та ймовірнісними моделями. Багато алгоритмів базуються на ймовірностях, оскільки дані можуть містити невизначеність або шум. У реальних даних є шум, пропущені значення, помилки вимірювань. Ймовірнісні методи допомагають будувати моделі, які можуть адаптуватися до таких умов. Ймовірність використовується у генеративних моделях для створення нових даних.

Теорія ймовірності дозволяє машинному навчанню працювати з невизначеністю, будувати ймовірнісні моделі та приймати рішення на основі ймовірностей. Вона є основою багатьох алгоритмів, включаючи байєсівські класифікатори, нейронні мережі, HMM, GANs і VAE.

Математична статистика - інструмент для аналізу та інтерпретації даних у машинному навчанні. Вона допомагає будувати моделі, оцінювати їхню якість і приймати рішення в умовах невизначеності. Математична статистика допомагає аналізувати дані перед навчанням моделі, оцінювати параметри та визначати значущість змінних, використовувати ймовірнісні розподіли у моделях, оцінювати точність та запобігати перенавчанню, а також перевіряти гіпотези та оптимізувати моделі.

В цілому, без статистики машинне навчання неможливе, адже воно ґрунтується на обробці й аналізі даних у невизначених умовах.

3 РОЗРОБКА МОДЕЛІ МАШИННОГО НАВЧАННЯ НА ОСНОВІ SUPERVISED LEARNING

3.1 Постановка задачі

У рамках бакалаврської кваліфікаційної роботи розглянуто задачу побудови моделі машинного навчання для класифікації видів ірису на основі датасету Iris. Цей датасет містить інформацію про три види ірису (Iris Setosa, Iris Versicolor, Iris Virginica) та їхні морфологічні характеристики, такі як довжина та ширина чашолистка і пелюстки.

Мета дослідження - розробити, навчити та оцінити ефективність моделі класифікації, яка з високою точністю зможе визначати вид ірису за його морфологічними ознаками.

Основні завдання дослідження:

1) Ознайомитися з датасетом Iris, проаналізувати його структуру та характеристики.

2) Виконати предобробку даних для виявлення закономірностей.

3) Вибрати та навчити класифікаційний алгоритм для вирішення поставленого завдання.

4) Провести оцінку моделі за допомогою метрик точності та обрати найкращу.

Даний вид задачі відноситься до класичного типу навчання, а саме до навчання із вчителем (Supervised Learning). Supervised Learning включає такі типи задач, як класифікація – передбачення категорій та регресія – передбачення числового значення. Поставлена у бакалаврській кваліфікаційній роботі задача відноситься до задачі класифікації (рис. 3.1). Задачі класифікації є одним із ключових напрямків машинного навчання, оскільки вони широко використовуються у багатьох сферах для прийняття рішень та автоматизації процесів. Їхня важливість пояснюється наступними факторами:

- автоматизація прийняття рішень;
- робота з великими обсягами даних;
- поліпшення точності прогнозів;

- адаптивність до різних сфер застосування;
- використання у розпізнаванні образів і мовленні;
- виявлення аномалій і безпека.

Класифікаційні моделі можуть швидко та ефективно приймати рішення без втручання людини, наприклад, визначення чи є транзакція шахрайською у банківській системі. Сучасні компанії мають великі масиви даних, які складно аналізувати вручну. Класифікаційні алгоритми допомагають обробляти ці дані та знаходити закономірності (автоматична категоризація електронних листів як спам або не спам). Класифікація дозволяє моделювати складні залежності між ознаками, що значно підвищує точність передбачень - медична діагностика (визначення, чи є у пацієнта певне захворювання). Класифікація є основою багатьох завдань комп'ютерного зору та обробки природної мови. Класифікація допомагає знаходити аномальні ситуації та можливі загрози.

Класифікація - це основа багатьох сучасних технологій, які дозволяють автоматизувати та оптимізувати процеси у різних сферах. Завдяки їй компанії та організації можуть приймати швидкі, точні та ефективні рішення на основі аналізу даних.

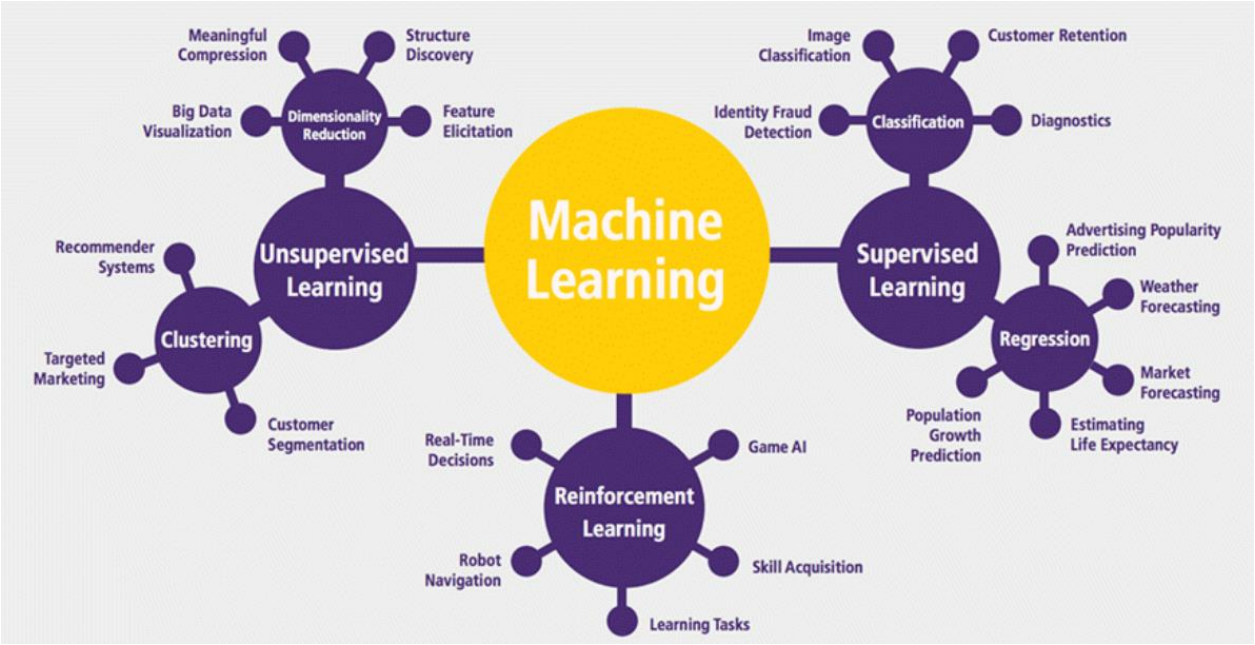


Рис. 3.1 Supervised Learning у машинному навчанні

Класифікація буває декількох типів: бінарна, багатокласова і класифікація з кількома мітками.

Бінарна класифікація - це тип задачі машинного навчання, у якій алгоритм має розподілити вхідні дані в одну з двох можливих категорій (класів) (рис. 3.2). Алгоритмами для бінарної класифікації є:

- логістична регресія;
- дерева рішень (Decision Trees);
- метод опорних векторів (SVM);
- нейронні мережі;
- Random Forest, XGBoost.

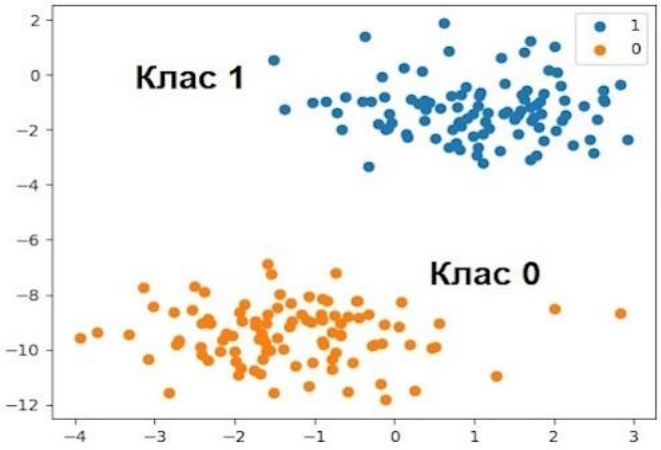


Рис. 3.2 Бінарна класифікація

Багатокласова класифікація (Multiclass Classification) - це тип задачі машинного навчання, в якій алгоритм повинен розподілити об'єкти в одну з більш, ніж дві категорії (класи) (рис. 3.3).

Для багатокласової класифікації доцільно використовувати алгоритми:

- логістична регресія;
- дерева рішень (Decision Trees);
- Random Forest;
- метод опорних векторів (SVM) з "one-vs-one" або "one-vs-all" підходами;
- нейронні мережі (особливо глибокі CNN для розпізнавання зображень).

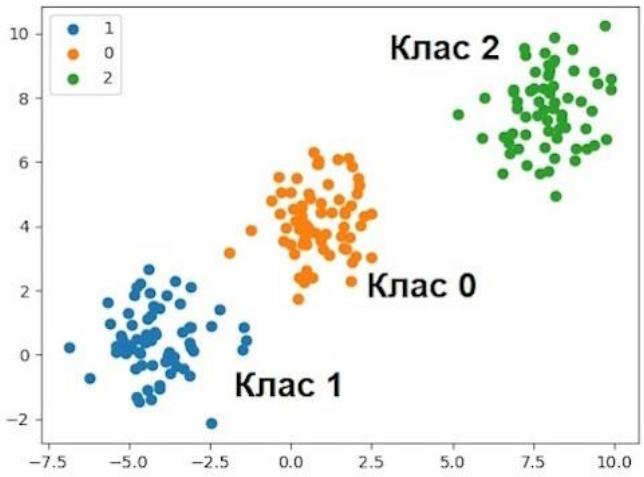


Рис. 3.3 Багатокласова класифікація

Класифікація з кількома мітками - це тип задачі машинного навчання, в якій один об'єкт може належати одразу до кількох класів (міток). Приклади класифікації з кількома мітками:

- тегування тексту, коли стаття може мати одночасно мітки "Політика", "Економіка", "Технології";
- рекомендаційні системи - фільм може бути позначений як "Комедія", "Фантастика" і "Пригоди";
- медична діагностика, наприклад, коли один пацієнт може мати одразу кілька хвороб;
- аналіз зображень, коли на фото можуть бути одночасно люди, машини та тварини.

Для реалізації моделі машинного навчання на основі Supervissed Learning були досліджені дві мови програмування R та Python.

Мова програмування R широко використовується у машинному навчанні, особливо в аналізі даних і статистичному моделюванні. Вона має потужні бібліотеки, гнучкість у візуалізації даних та зручний синтаксис для роботи з числовими та категоріальними даними.

Мова програмування R була створена для статистики, тому має багато вбудованих функцій для аналізу даних, основними з яких є:

- робота з регресією, класифікацією, кластерами;

- використання статистичних тестів;
- обробка тимчасових рядів та аналіз кореляцій.

Мова R проста у вивченні для статистиків, економістів, біологів, оскільки її синтаксис схожий на мову математики. Водночас, ця мова дає велику гнучкість для дослідників, які можуть створювати власні статистичні моделі та аналітичні методи.

Дослідивши, коли варто використовувати R для ML, визначена доцільність, якщо потрібно швидко аналізувати дані та візуалізувати результати; якщо проєкт містить статистичні методи або економетричний аналіз. Також доцільно використовувати мову програмування R в області ML для розробки прототипів та досліджень у сфері науки та фінансів, а також для створення інтерактивних ML-додатків.

Популярною мовою програмування для побудови моделей ML є Python завдяки своїй гнучкості, масштабованості та потужним бібліотекам. Для розробки моделі ML у дипломній роботі була вибрана мова програмування Python, так як вона має кращу підтримку машинного та глибокого навчання, ніж R. Результати порівняння вищевказаних мов зведені у табл. 3.1.

Таблиця 3.1

Порівняння мов програмування Python та R у сферах ML та DL

№ п/п	Особливість	Python	R
1	Простота вивчення	Легкий синтаксис	*Специфічний синтаксис*
2	Машинне навчання	Scikit-learn, XGBoost	Caret, RandomForest
3	Глибоке навчання	TensorFlow, PyTorch	*Слабка підтримка*
4	Продуктивність	Швидкий (C/C++ під капотом)	*Повільніший*
5	Масштабованість	Використовується в production	*Більше для досліджень*
6	Big Data	Spark, Dask	*Обмежена підтримка*
7	Візуалізація	Matplotlib, Seaborn	ggplot2, Lattice
8	Інтеграція з хмарою	AWS, GCP, Azure	*Мінімальна підтримка*
9	Спільнота	Велика	*Менша, більше академічна*

На основі проведеного аналізу, доцільно визначити, в яких випадках краще використовувати мову програмування Python, а в яких – R (табл. 3.2).

Таблиця 3.2

Задачі, в яких доцільно використовувати мови програмування Python та R

№ п/п	***Python***	***R***
1	Побудова складних ML-моделей	Швидкий аналіз даних
2	Продакшен-рішення та масштабування	Статистичний аналіз
3	Глибоке навчання та нейромережі	Економетричне моделювання
4	Робота з великими даними	Сфера науки, медицини або соціології
5	Інтеграція з іншими мовами та хмарними сервісами	Візуалізація даних (ggplot2 - потужний пакет)

У результаті досліджень було визначено, що Python - найкраща мова для машинного навчання, оскільки він простий, продуктивний, гнучкий та має потужні бібліотеки. Мова R більше підходить для статистичних досліджень, але програє у масштабованості та підтримці глибокого навчання. Тому, якщо потрібне машинне навчання для реальних продуктів, доцільніше використовувати мову програмування Python.

3.2 Використання математичного апарату для розробки моделі ML

Машинне навчання (ML) базується на математичних концепціях та алгоритмах, які дозволяють моделям аналізувати дані, знаходити закономірності та робити прогнози. Основними напрямами математики, що використовуються в ML, є лінійна алгебра, математичний аналіз, теорія ймовірностей, статистика та оптимізація.

Лінійна алгебра є основою для роботи з багатовимірними даними, які представлені у вигляді векторів і матриць. Вектори представляють окремі

спостереження або ознаки, матриці використовуються для зберігання наборів даних. Множення матриць необхідне для обчислення вихідних значень у моделях.

У лінійній регресії використовується формула:

$$y = Xw + b \ , \tag{3.1}$$

де X – матриця вхідних даних;

w - вектор коефіцієнтів;

b – зміщення;

y – прогноз.

У нейронних мережах кожен шар представлений як множення матриці вагів на вхідний вектор.

Всі моделі машинного навчання працюють із статистичними закономірностями у даних. Для навчання моделей потрібна оптимізація функцій, що базується на диференціальному численні. Алгоритм оновлює параметри моделі для мінімізації функції втрат:

$$w := w - \eta \cdot \nabla L(w) \ , \tag{3.2}$$

де w - параметри моделі;

η - швидкість навчання;

▼L(w) - градієнт функції втрат.

Фундаментальним поняттям, будівельною цеглою, яка лежить в основі всієї лінійної алгебри, є вектор.

У представленні для фізики вектори – це стрілки, які направлені у простір. При цьому вектор визначає довжина і напрямок, в який він вказує. Поки ці два фактори не змінюються, ми можемо переміщувати його як завгодно і це буде той самий вектор. Вектори, які знаходяться на площині, дворозмірні (two-dimensional), а ті, які знаходяться у розширеному просторі, в якому ми живемо, - трирозмірні (three-dimensional). У представленні студента computer science вектори – це впорядковані списки чисел. Наприклад, нехай ми робимо аналіз цін на будинки. Єдині властивості, які нас цікавлять, це площа та ціна. Ми можемо

змоделювати кожний дім парою чисел – перше означає площу, друге – ціну. Порядок тут важливий. Будинки можна моделювати у вигляді дворозмірних векторів, де у даному контексті вектор – слово замість слова «список». Дворозмірним вектором робить його те, що його довжина дорівнює 2.

У математиці задалися узагальненням двох підходів і визначили, що по суті вектор може бути чим завгодно, де дотримується ідея додавання двох векторів і множення вектора на скаляр. Деталі такого підходу досить абстрактні. Ідеї векторного додавання та множення на числа відіграють важливу роль у всьому курсі лінійної алгебри.

Коли ми будемо представляти нову серію векторів, потрібно спочатку думати про стрілку, а саме про стрілку в системі координат, такою як площина *x*, *y* з «хвостом» на початку координат (рис. 3.4). Це трохи відрізняється від представлення студента-фізика, де вектори можуть вільно знаходитися на площині, де вони хочуть. У лінійній алгебрі майже завжди наш вектор буде закріплений на початку координат.

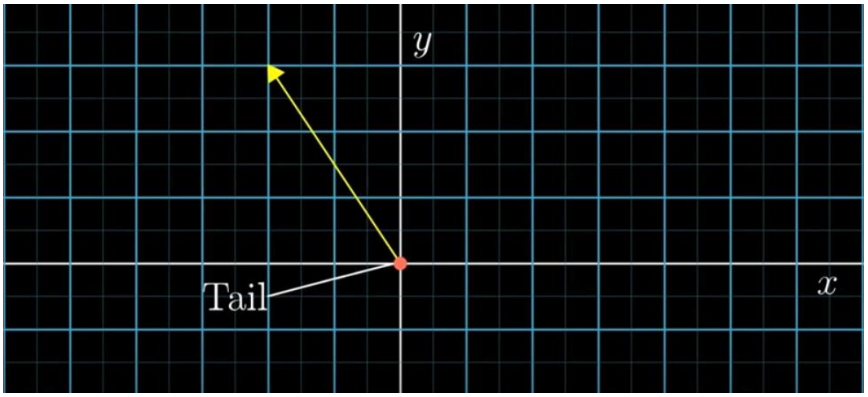


Рис. 3.4 Вектори у машинному навчанні

Потім, коли ми зрозуміли нову концепцію у контексті стрілок у просторі, ми переводимо це у список числового представлення, що ми можемо зробити, розглядаючи координати вектора.

Координатна система - це місце, де всі переходи у прямому і зворотньому напрямках між двома перспективами в лінійній алгебрі і відбуваються. Маємо два виміри.

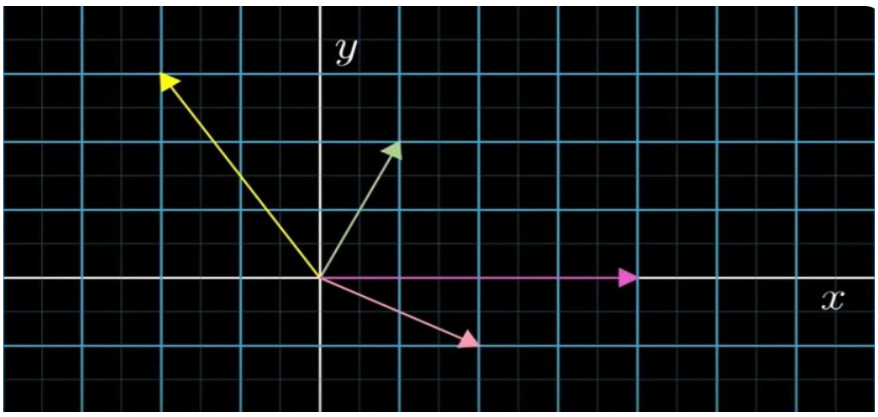


Рис. 3.5 Координати вектору

Вектори та матриці є основою для представлення, обробки та аналізу даних у машинному навчанні. Вони дозволяють ефективно працювати з великими наборами даних, виконувати обчислення швидко та ефективно і будувати складні моделі.

Вектори використовуються для представлення ознак (features), дозволяють легко порівнювати об'єкти і є основою для всіх алгоритмів машинного навчання.

Матриця – це таблиця чисел (або набір векторів), яка використовується для представлення кількох об'єктів одночасно (рис. 3.6).

		Прогнозований клас	
	Загальна кількість = P + N	Predicted Positive (PP)	Predicted Negative (PN)
Справжній клас	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Рис. 3.6 Матриця невідповідностей

Матриці використовуються для зберігання великих масивів даних, дозволяють виконувати паралельні обчислення (наприклад, у GPU) і використовуються у всіх алгоритмах ML. Більшість операцій у машинному навчанні - це робота з векторами та матрицями.

3.3 Вибір середовища розробки моделі ML

Для розробки моделі ML були розглянуті наступні середовища: PyCharm, Google Colab, Jupyter Notebook.

PyCharm є потужним інтегрованим середовищем розробки (IDE) для мови Python, яке широко використовується для машинного навчання (ML) та штучного інтелекту (AI) (рис. 3.7). Воно надає розширений функціонал для написання, тестування та відлагодження коду, що робить процес розробки ML-моделей зручним та ефективним. Крім цього, PyCharm забезпечує інтеграція з середовищем Jupyter Notebook, забезпечуючи можливість запускати ML-експерименти безпосередньо у PyCharm.

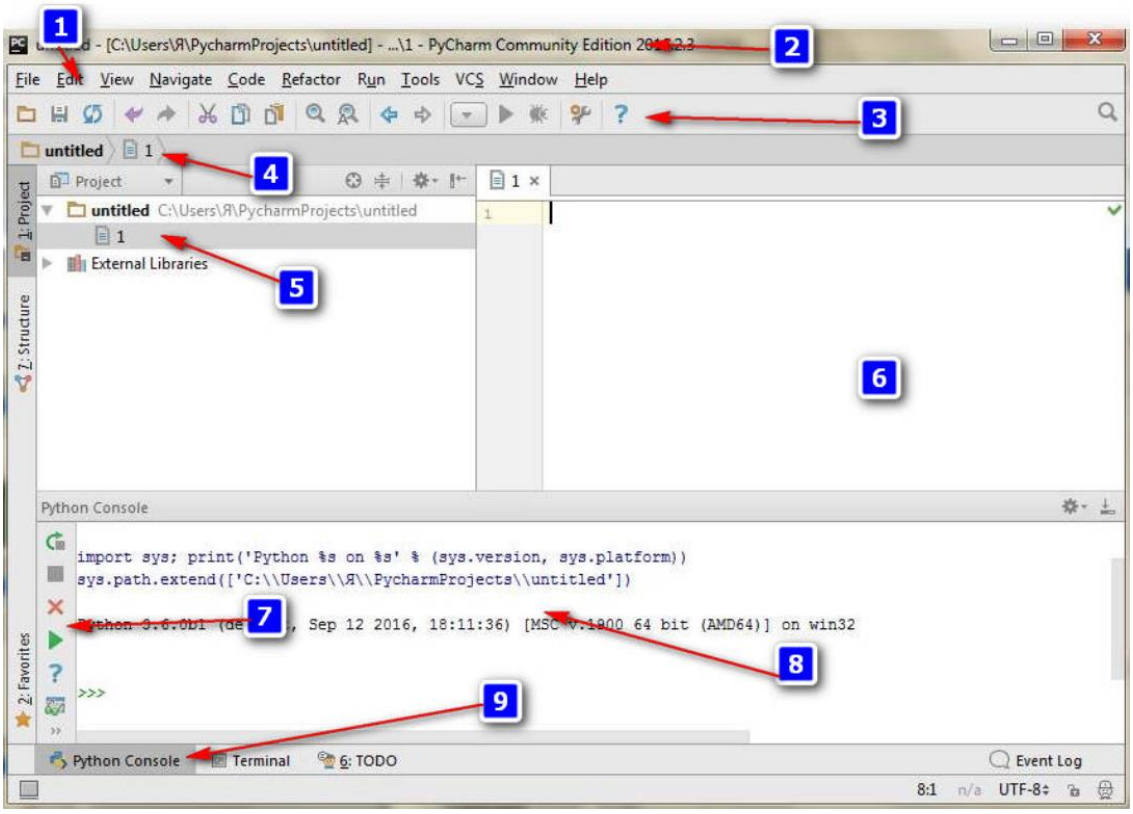


Рис. 3.7 PyCharm - інтегроване середовище розробки для мови програмування Python

1 - Головне меню PyCharm знаходиться у верхній частині вікна IDE і містить основні функції для роботи з проєктом та інструментами.

2 - Заголовок діалогового вікна у PyCharm - це частина інтерфейсу, що знаходиться у верхній частині вікна діалогу або вікна налаштувань. Заголовок

містить назву діалогового вікна та додаткову інформацію, яка допомагає користувачу зрозуміти, в якому контексті знаходиться програма та яку задачу потрібно вирішити.

3 - Основна панель інструментів - це панель, яка розташована під головним меню і надає швидкий доступ до найпоширеніших функцій IDE. Вона дозволяє користувачеві здійснювати ключові операції, такі як запуск програм, налаштування середовища, перемикання між вікнами та інші важливі дії, не виходячи з основного робочого інтерфейсу.

4 - Панель навігації - важлива частина інтерфейсу IDE, яка забезпечує зручний доступ до різних частин проєкту, файлів і структур коду. Вона допомагає швидко переміщатися між різними елементами програми, знаходити файли, класи, методи та інші об'єкти, що значно підвищує продуктивність розробника.

5 - Інспектор об'єктів у PyCharm - це потужний інструмент для дослідження об'єктів та даних під час виконання програми, який дозволяє переглядати та аналізувати значення змінних і об'єктів у реальному часі. Зазвичай інспектор об'єктів використовується під час налагодження (debugging) коду, що дає змогу розробникам детально розуміти поведінку програми і знаходити помилки.

6 – Редактор - основний компонент IDE, який дозволяє писати, редагувати та аналізувати код. Редактор у PyCharm забезпечує зручне середовище для програмування, з численними інструментами, які допомагають прискорити розробку та зробити її ефективною.

7 - Вікно інструментів у PyCharm - це панель, яка містить різні вкладки та функціональні інструменти для ефективної роботи з кодом, відлагодження, тестування та управління проєктом. Вікна інструментів допомагають організувати робочий процес, забезпечуючи доступ до важливих можливостей IDE без необхідності відкривати додаткові вікна або меню.

8 - Стрічка стану у PyCharm - це нижня панель у вікні IDE, яка відображає важливу інформацію про поточний стан проєкту, середовища розробки та процесів, що виконуються. Основними елементами стрічки стану є: поточний статус IDE, інформація про файл, Git та VCS (Version Control System), режими

редактора. Стрічку стану можна налаштовувати: приховувати, відображати потрібні модулі або додавати нові компоненти для зручності роботи.

9 - Режим роботи середовища. Додаткові опції. Режим роботи середовища у PyCharm визначає, як IDE працює з проєктами, кодом, відлагодженням та іншими функціями. PyCharm підтримує різні конфігурації та налаштування, що дозволяють розробникам адаптувати середовище під свої потреби. До додаткових опцій відносяться: інтерпретатор Python, конфігурації запуску, встановлення плагінів, профілювання коду, підтримка віддаленого запуску, автоматичне збереження та відновлення.

PyCharm надає гнучкі режими роботи для розробки, тестування, налагодження та контролю версій. Додаткові опції, такі як підтримка плагінів, робота з інтерпретаторами Python, профілювання коду, роблять його потужним інструментом для машинного навчання, веб-розробки та інших сфер програмування.

Google Colab (Colaboratory) - це безкоштовне хмарне середовище для роботи з Python, яке надає можливість писати та виконувати код у Jupyter Notebook без необхідності встановлення додаткового програмного забезпечення. Воно ідеально підходить для розробки та навчання моделей машинного навчання, зокрема з використанням TensorFlow, PyTorch, Keras та Scikit-learn (рис. 3.8).

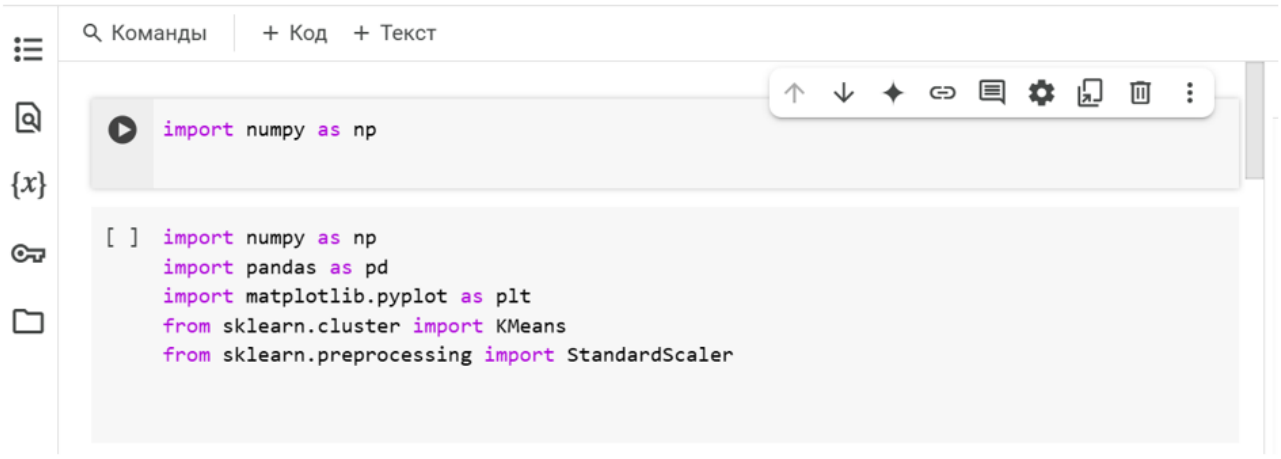


Рис. 3.8 Середовище Google Colab

Основні особливості Google Colab:

- робота у хмарі;
- безкоштовний доступ до GPU та TPU;
- підтримка Jupyter Notebook;
- інтеграція з Google Drive та GitHub;
- легка установка бібліотек;
- автоматичне збереження та контроль версій;
- візуалізація результатів.

Код у Google Colab виконується на сервері Google, тому немає потреби у локальних обчислювальних ресурсах. Немає необхідності встановлювати Python або бібліотеки, так як все налаштовано автоматично. Colab надає можливість використовувати графічні процесори (GPU) та тензорні процесори (TPU) безкоштовно, що прискорює навчання моделей машинного навчання, особливо при роботі з великими наборами даних.

Google Colab використовує формат Jupyter Notebook, який дозволяє комбінувати код, текст (Markdown), візуалізації. Інтеграція з Google Drive та GitHub забезпечує збереження та відкривання файлів безпосередньо з Google Drive, доступ до репозиторіїв на GitHub для роботи з кодом у спільних проєктах.

Colab підтримує більшість бібліотек Python, наприклад, TensorFlow, PyTorch, Pandas, Scikit-learn (рис. 3.9).

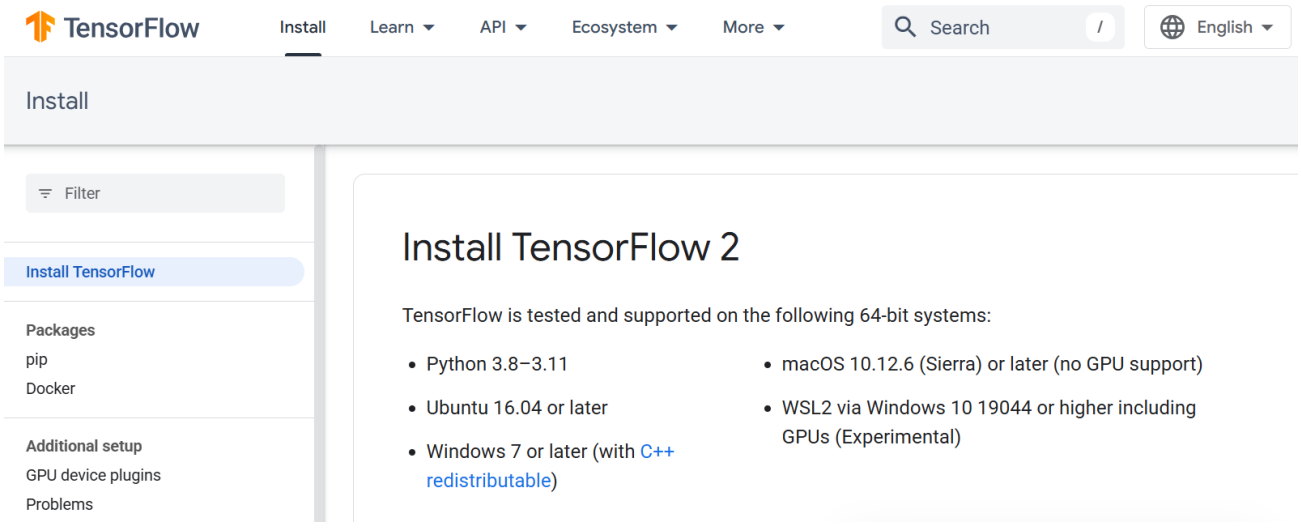


Рис. 3.9 Фреймворк TensorFlow

Автоматичне збереження та контроль версій забезпечує автоматичне збереження усіх змін у Google Drive. Colab підтримує вбудовані графіки та діаграми, використовуючи такі бібліотеки як Matplotlib, Seaborn, Plotly.

Google Colab є ідеальним середовищем для швидкого прототипування моделей машинного навчання, особливо якщо у нас немає потужного комп'ютера. Воно дозволяє безкоштовно використовувати GPU/TPU, працювати у Jupyter Notebook, інтегруватися з Google Drive і GitHub, а також легко спільно працювати над проєктами.

Jupyter Notebook - інтерактивне середовище для написання та виконання Python-коду, яке широко використовується у сфері машинного навчання, аналізу даних та наукових обчислень. Воно дозволяє комбінувати код, текст, графіки та візуалізації в одному документі. Його інтеграція з популярними бібліотеками та можливість поетапного виконання коду роблять його одним із найкращих інструментів для роботи з машинним навчанням.

У Jupyter Notebook код виконується покроково, що дозволяє тестувати частини коду без необхідності запуску всього скрипта, є можливість перезапускати окремі комірки та експериментувати з параметрами моделей. У ноутбуці можна додавати форматований текст, заголовки, списки та формули, що дозволяє документувати процес розробки моделі машинного навчання (рис. 3.10).

Рис. 3.10 Середовище Jupyter Notebook

Jupyter Notebook повністю інтегрований з бібліотеками Matplotlib, Seaborn, Plotly для побудови графіків. Хоча основною мовою є Python, Jupyter підтримує також R, Julia, Scala, Java та інші. В Jupyter Notebook можна легко імпортувати дані з CSV, JSON, SQL, Google Drive. Jupyter Notebook працює через веб-інтерфейс, що робить його зручним для віддаленої роботи.

Дане середовище підтримує моделювання нейронних мереж та інші алгоритми машинного навчання, надає можливість запуску експериментів та візуалізації результатів у реальному часі. Середовище підтримує багато фреймворків та бібліотек для роботи з даними, у тому числі такі відомі як TensorFlow, Keras, PyTorch (рис. 3.11, 3.12).

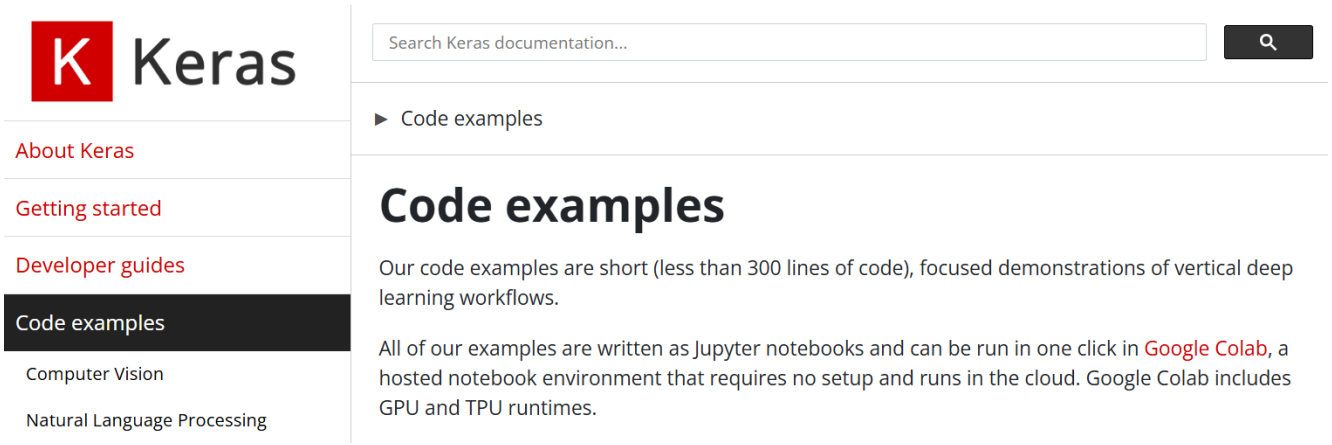


Рис. 3.11 Фреймворк Keras

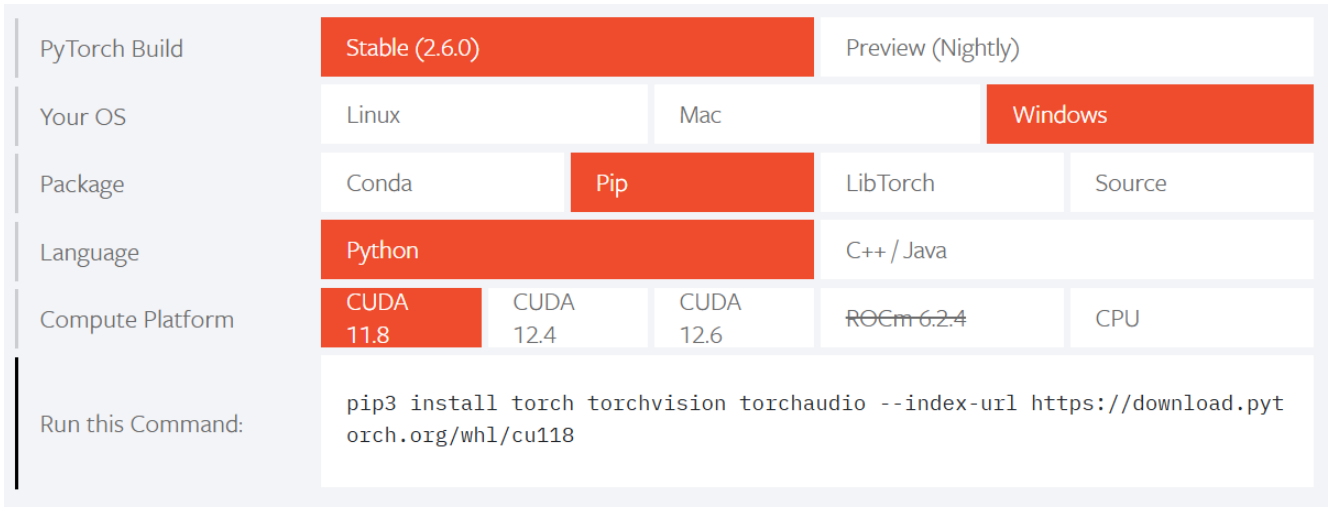


Рис. 3.12 Сітка PyTorch

3.4 Вибір алгоритму навчання для поставленої задачі

Для розробки моделі машинного навчання на основі Supervised Learning було вибране середовище розробки Jupyter Notebook, мова програмування Python. Актуальними та поширеними бібліотеками та фреймворками для розробки моделей ML є наступні (табл. 3.3).

Таблиця 3.3

Бібліотеки Python для машинного навчання

№ п/п	Бібліотека/фреймворк	Призначення
1	NumPy	Бібліотека для числових обчислень, забезпечує швидкі операції над масивами та матрицями, що є основою багатьох ML-алгоритмів.
2	pandas	Використовується для роботи з табличними даними, забезпечує ефективну обробку, аналіз і підготовку даних перед передаванням у модель.
3	matplotlib	Бібліотека для візуалізації даних, допомагає аналізувати та інтерпретувати дані перед тренуванням моделі, а також оцінювати результати навчання.
4	seaborn	Бібліотека для візуалізації даних, побудована на основі Matplotlib, дозволяє створювати більш інформативні та стильні графіки, що особливо корисно в аналізі даних перед навчанням моделі машинного навчання.
5	scikit learn	Забезпечує зручні інструменти для побудови, тренування та оцінки моделей машинного навчання. Бібліотека включає алгоритми для класіфікації, регресії, кластеризації та оцінки моделей.
6	TensorFlow	Забезпечує інструменти для розробки та тренування моделей машинного навчання та глибокого навчання, а також для розгортання цих моделей на різних платформах, включаючи мобільні пристрої та сервери.
7	Keras	Високорівнева бібліотека для розробки та тренування нейронних мереж.
8	PyTorch	Забезпечує високопродуктивну підтримку для нейронних мереж, глибокого навчання, а також для наукових обчислень із використанням тензорів (багатовимірних масивів).

Датасет Iris - один із найвідоміших наборів даних у машинному навчанні. Його часто використовують для навчання та тестування алгоритмів класифікації, візуалізації даних і демонстрації методів аналізу. Даний датасет містить 150 зразків (квіток) ірису і три види (класи) ірису, зазначених у розділі 3.1 бакалаврської кваліфікаційної роботи.

На першому етапі потрібно завантажити необхідні бібіліотеки: NumPy, Scikit Learn, завантажити датасет Iris. Датасет має 4 числові характеристики (ознаки квітки):

- довжину чашолистка (*sepal length*);
- ширину чашолистка (*sepal width*);
- довжину пелюстки (*petal length*);
- ширину пелюстки (*petal width*).

Основними задачами, в яких доцільно використовувати датасет Iris, є наступні: класифікація; візуалізація та аналіз даних; навчання алгоритмів зниження розмірності; кластеризація; виявлення аномалій. Датасет Iris невеликий, добре структурований і підходить для багатьох алгоритмів.

```
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
```

На наступному етапі необхідно завантажити дані:

```
iris = load_iris()
```

```
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],
```

```
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 є атрибутом, що містить мітки класів (цільові значення) для кожного зразку у датасеті Iris.

```
iris.target_names
```

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

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

Команда *iris.target_names* - це атрибут, який містить назви класів (категорій) видів ірису у датасеті Iris. Кожен елемент відповідає класу *iris.target* (табл. 3.4).

Таблиця 3.4

Відповідність значення *iris.target* класу *iris.target_names*

Числове значення (*iris.target*)	Назва класу (*iris.target_names*)
0	‘setosa’
1	‘versicolor’
2	‘virginica’

Нижче у коді бачимо, що параметри (6.5, 3. , 5.2, 2.) має клас рослин 2, параметри (6.2, 3.4, 5.4, 2.3) має також клас рослин 2.

Параметри, які розташовуються по середині, відносяться до класу рослин ‘versicolor’, тобто до класу 1.

```
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, 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]),
 'frame': None,
```

Потрібно написати модель ML, яка б по чотирьом значенням розпізнавала, до якого класу віднести ту чи іншу рослину – 0, 1 або 2, тобто виконати задачу класифікації. Модель має навчитися визначати закономірності для того, щоб надати відповідь класу.

Визначити значень *X* та *y*:

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

Далі необхідно виконати розділення датасету на тестову та тренувальну вибірки:

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

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],
```

```
len(X_train)
```

```
120
```

```
len(X_test)
```

```
30
```

Для задачі класифікації використовуються такі алгоритми: наївний Байєс (Naive Bayes), дерева рішень (Decision Trees), логістична регресія (Logistic Regression), k-найближчих сусідів (k-NN), метод опорних векторів (SVM), нейронні мережі, Random Forest.

Порівнюючи дані алгоритми, визначимо, який з них найкраще підходить для датасету Iris. Результати порівняння приведено у табл. 3.5.

Таблиця 3.5

Основні характеристики алгоритмів класифікації

№ п/п	Алгоритм	Простота	Час навчання	Пояснювальність	Чутливість до шуму	Підходить до лінійних даних	Гнучкість для складних даних
1	*Наївний Байєс (NB)*	Простий	Швидкий	Висока	Чутливий	Так	Обмежена
2	*Дерева рішень (DT)*	Простий	Середній	Висока	Середня	Ні	Гнучкий
3	*Логістична регресія (LR)*	Простий	Швидкий	Висока	Середня	Так	Погано працює з нелінійністю
4	*k-NN (k-найближчих сусідів)*	Простий	Повільний	Низька	Дуже чутливий	Так	Гнучкий
5	*Метод опорних векторів (SVM)*	Середній	Середній	Середня	Чутливий	Так	Гнучкий
6	*Нейронні мережі (NN)*	Складний	Дуже повільний	Низька	Чутливий	Так	Дуже гнучкий
7	*Random Forest (RF)*	Середній	Середній	Середня	Стійкий	Ні	Гнучкий

Логістична регресія - найкращий вибір для датасету Iris, тому що у даному датасеті дані добре розділяються лінійно, алгоритм швидкий та простий, забезпечує високу точність. Для складніших даних (з більшою кількістю

особливостей або нелінійних залежностей) варто використовувати алгоритми Random Forest або SVM.

У рамках дипломної роботи був вибраний алгоритм логістичної регресії. Логістична регресія - алгоритм бінарної або багатокласової класифікації, який прогнозує ймовірність належності зразка до певного класу. Незважаючи на назву, логістична регресія використовується не для регресії, а для класифікації. Вона базується на логістичній (сигмоїдній) функції, яка перетворює вхідні значення у ймовірності від 0 до 1.

У математичній моделі логістичної регресії для заданого набору ознак $X = (x_1, x_2, \ldots x_n)$ обчислюється лінійна комбінація:

$$z = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b \ , \tag{3.3}$$

де w_i - коефіцієнти (ваги моделі);

b - зсув (bias);

z - лінійна сума.

Потім потрібно застосувати функцію активації, у даному випадку сигмоїдну функцію:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.4}$$

Це перетворює значення z у ймовірність p належності до класу 1:

$$p = P(y = 1|X) = \frac{1}{1 + e^{-(w_1x_1 + w_2x_2 + \ldots + b)}} \tag{3.5}$$

Якщо $p \geq 0.5$, модель відноситиме приклад до класу 1, інакше – до класу 0.

Наступним етапом є створення та навчання ML-моделі:

```
model = LogisticRegression()
```

Команда *model = LogisticRegression()* створює екземпляр моделі логістичної регресії з бібліотеки Scikit-learn. *LogisticRegression()* - це конструктор класу, який створює модель. *Model* - об’єкт моделі, який можна навчати, прогнозувати та оцінювати.

Для навчання моделі ML необхідно виконувати метод *fit()*, куди потрібно передати *X_train*, *y_train*.

Команда *model.fit(X_train, y_train)* навчає модель логістичної регресії на тренувальних даних, де *X_train* - матриця ознак (вхідні дані для тренування); *y_train* - мітки класів (правильні відповіді); *fit()* - метод, який налаштовує вагові коефіцієнти моделі на основі тренувальних даних.

Створюємо модель ML.

```
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()
```

```
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])
```

Тепер модель ML може виконувати класифікацію наших даних.

Подамо деякі значення на ML-модель:

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

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

У результаті матимемо відповідь: клас 2. Тобто рослина, у якої показники значень були б [1, 2, 3, 4], має належати до класу 2. Якщо на ML-модель передати інші значення, наприклад, чотири, сім, три, чотири - відповіддю буде клас 0 клас.

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

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

3.5 Оцінка якості моделі ML

Оцінка якості моделі ML необхідна, щоб переконатися, що модель правильно узагальнює закономірності у даних і не є випадковою або перенавченою. Потрібно знати, наскільки добре модель передбачає правильні відповіді. Якщо модель має низьку точність, її потрібно покращити або вибрати інший алгоритм. Часто для розв'язання задачі потрібно випробувати різні алгоритми, а оцінка якості допомагає вибрати найкращий варіант.

В цілому, оцінка моделі необхідна, щоб перевірити її точність та ефективність. Це допомагає уникнути перенавчання або недонавчання моделі, дозволяє вибрати найкращий алгоритм для конкретної задачі, допомагає покращити модель шляхом налаштування параметрів.

Команда *classification_report()* виконується після тренування моделі та прогнозування. *y_test* - реальні мітки класів, *y_predict* - передбачені мітки, які отримані від моделі.

```
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(y_test, y_predict)* виводить розширену статистику оцінки якості моделі у задачах класифікації. Вона показує основні метрики:

- precision (точність);
- recall (повнота);
- F1-score (збалансована оцінка);
- support (кількість прикладів у кожному класі).

У результаті характеристики вищевказаних метрик, визначено, що:

- метрика *precision* – це частка правильних передбачень для кожного класу:

$$Precision = \frac{TP}{TP + FP} \quad , \tag{3.6}$$

де *TP* (True Positives) - кількість правильних передбачень для позитивного класу;
FP (False Positives) - кількість неправильних передбачень позитивного класу.
- метрика *recall* означає, скільки прикладів класу модель правильно розпізнала:

$$Recall = \frac{TP}{TP + FN} \quad , \tag{3.7}$$

де *FN* (False Negatives) – кількість випадків, коли модель пропустила позитивний клас (помилково віднесла позитивний об'єкт до негативного).
- метрика *F1-score* - гармонічне середнє між precision та recall:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad , \tag{3.8}$$

де *precision (точність)* - яка частка передбачених позитивних класів є правильними;
recall (повнота) - яка частка реальних позитивних випадків була знайдена.
- метрика *support* - кількість зразків у кожному класі.

Отримані результати класифікації моделі машинного навчання:

1 - для кожного класу precision, recall і F1-score рівні 1.00 (100%), що означає, що модель абсолютно правильно класифікує всі зразки.

2 - Support показує, скільки тестових прикладів було у кожному класі (наприклад, 10 для класу 0; 9 для класу 1; 11 для класу 2).

Загальну оцінку моделі представимо у табл. 3.6.

Accuracy - метрика оцінки ML-моделі, яка показує, яку частку передбачень модель зробила правильно.

Macro avg (макро-середнє) - це середнє значення метрик, таких як precision, recall або F1-score для кожного класу без врахування їхніх часток або вагів у загальній кількості зразків. Тобто, кожен клас має однакову вагу незалежно від того, скільки прикладів є у кожному класі.

Weighted avg (вагове середнє) - середнє значення метрик (наприклад, precision, recall або F1-score), де кожен клас має різну вагу залежно від кількості зразків цього класу у тестовому наборі. Це означає, що класи з більшою кількістю прикладів мають більший вплив на загальну оцінку.

Таблиця 3.6

Загальна оцінка розробленої моделі ML для задачі класифікації

№ п/п	Метрика	Значення
1	Accuracy (загальна точність)	1.00 (100%)
2	Macro avg	1.00
3	Weighted avg	1.00

Accuracy = 1.00 (100%) означає, що модель абсолютно правильно класифікувала всі 30 тестових зразків.

Macro avg = 1.00 - це середнє арифметичне значення всіх F1-score для кожного класу.

Weighted avg = 1.00 - усереднене значення, враховуючи кількість зразків у кожному класі.

Модель правильно передбачає всі тестові приклади для всіх класів.

ВИСНОВКИ

У результаті виконання бакалаврської роботи було розроблено модель машинного навчання для задачі класифікації на основі датасету Iris, який містить інформацію про види рослин (Iris Setosa, Iris Versicolor, Iris Virginica) та їхні характеристики. Було виконане навчання моделі ML, здійснено оцінку її ефективності, яка з високою точністю зможе визначати вид рослини за її морфологічними ознаками.

Датасет Iris містить 150 записів, що дозволяє швидко тестувати алгоритми без потреби в потужному обладнанні. Він є добре збалансованим, містить по 50 зразків для кожного з трьох класів, що запобігає проблемам із дисбалансом класів, не містить пропущених значень, інтегрований у бібліотеку Scikit-learn, добре підходить для ознайомлення з алгоритмами класифікації та методами обробки даних. Iris є одним із найпопулярніших датасетів у сфері машинного навчання та аналізу даних.

Машинне навчання із вчителем (Supervised Learning) відбувається на основі розмічених даних, де кожен вхідний приклад має відповідний цільовий вихід (мітку). Двома основними завданнями Supervised Learning є класифікація та регресія. Модель навчається на основі навчального набору даних (training set), модель перевіряється на тестовому наборі (test set), щоб оцінити її продуктивність. Supervised Learning дозволяє вирішувати широкий спектр завдань із високою точністю.

Процес навчання у Supervised Learning включав етапи збору та підготовки даних; вибір моделі; вибір алгоритму навчання; навчання моделі; оцінку якості моделі ML.

У процесі збору та підготовки даних було сформовано навчальну вибірку, пропущені значення та аномальні дані в датасеті Iris були відсутніми, тому їх заповнювати та нормалізовувати було не потрібно. При використанні датасетів інших видів можлива нормалізація або стандартизація даних для кращої роботи моделі.

Для розробки ML-моделі було вибране інтерактивне середовище Jupyter Notebook, яке знайшло широке використання у сфері машинного навчання, аналізу даних та наукових обчислень. Інтеграція Jupyter Notebook з популярними бібліотеками та можливість поетапного виконання коду роблять його одним із найкращих інструментів для роботи з машинним навчанням.

У процесі розробки моделі ML для вирішення задачі класифікації були використані бібліотеки NumPy та Scikit-learn. NumPy забезпечує швидкі операції над масивами та матрицями, що є основою багатьох ML-алгоритмів. Бібліотека Scikit-learn має зручні інструменти для побудови, тренування та оцінки моделей машинного навчання. Бібліотека включає алгоритми для класіфікації, регресії, кластеризації та оцінки моделей.

У процесі вибору моделі був застосований алгоритм для задачі класифікації - логістична регресія (Logistic Regression). Логістична регресія - алгоритм бінарної або багатокласової класифікації, який прогнозує ймовірність належності зразка до певного класу. Даний алгоритм є швидким, інтерпретованим, добре підходить для задач реального часу, задач Supervised Learning, де важливі швидкість і точність. Підходить для задач, особливо коли дані мають лінійну залежність між ознаками та цільовим класом.

Для оцінки якості моделі ML використовувалися метрики оцінки. Для класифікації доцільно використовувати наступні метрики якості: Accuracy, Precision, Recall, F1-score. Якщо якість моделі недостатня, вона може бути доопрацьована шляхом зміни алгоритму, додаванням більшої кількості даних. Якщо модель працює добре, її можна впровадити у реальне використання.

Для класів рослин setosa, versicolor та virginica показники точності, повноти, та F1-оцінки становили 1.00, що означає, що всі передбачення були вірними.

Навчання із вчителем (Supervised Learning) є одним із найпопулярніших підходів у машинному навчанні. Воно дозволяє створювати точні та ефективні алгоритми для передбачення майбутніх подій та автоматизації складних завдань. Але воно потребує великих обсягів якісних даних, що є його основним обмеженням.

Перевагами Supervised Learning є: висока точність за умови якісних даних; чіткий процес навчання, так як алгоритм працює на основі конкретних міток; швидке навчання у порівнянні з іншими типами навчання (наприклад, Reinforcement Learning). Даний метод навчання має надзвичайно широку область застосування, наприклад, у фінансовій сфері використовується для виявлення шахрайських транзакцій; у медицині – для діагностики хвороб на основі аналізів; автономні системи – системи керування автомобілями; розпізнавання мови та тексту – голосові помічники, на кшталт Google Assistant; у рекомендаційних системи – персоналізовані рекомендації на Netflix, YouTube.

Разом з цим, для навчання Supervised Learning потрібно багато розмічених даних; даний вид навчання не підходить для складних невизначених середовищ, де важко зібрати правильні мітки. Є проблеми з узагальненням - модель може погано працювати на нових або незнайомих даних. Також присутня чутливість до шуму у даних - якщо у вибірці багато помилкових міток, модель навчиться неправильно.

Для успішної розробки моделей машинного навчання та роботи з ними потрібно розумітися у наступних знаннях з математики: лінійна алгебра; математичний аналіз (похідні, частинні похідні); теорія оптимізації; теорія ймовірності та математична статистика.

Розроблена та проаналізована модель машинного навчання на основі Supervised Learning з урахуванням сучасних алгоритмів може бути використана для ефективної обробки великих обсягів даних, для сучасних задач Big Data та автоматизації процесів у бізнесі.

ПЕРЕЛІК ПОСИЛАНЬ

1. https://pytorch.org/
2. https://numpy.org/
3. https://matplotlib.org/
4. https://www.tensorflow.org/
5. 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.
6. 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.
7. 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].
8. Wang, Haofan et al. (2020). Score-CAM: Score-Weighted Visual Explanations for Convo-lutional Neural Networks. arXiv: 1910.01279 [cs.CV].
9. Лі, Кай-Фу Штучний інтелект: 10 передбачень для майбутнього / Кай-Фу Лі, Чень Цюфань; пер. з англ. А. Райчук. – Київ: Форс Україна, 2022. – 464 с.
10. 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.
11. Jake VanderPlas Python Data Science Handbook: Essential Tools for Working with Data, 2022, 551p.
12. Грас Д. Data Science. Наука про дані з нуля: Пер. з англ. – 2-е видання., перероб. та доп. – 2021. – 416 с.

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

14. 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.

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

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

17. Yao, H., Mai, T., Jiang, C., Kuang, L., Guo, S. AI Routers & Network Mind: A Hybrid Machine Learning Paradigm for Packet Routing. IEEE Computational Intelligence Magazine, 14(4), 2019. — 21-30 с.

18. Li, L., Wen, X., Lu, Z., Pan, Q., Hu, W. J. A. Z. Energy-efficient UAV-enabled MEC system: Bits allocation optimization and trajectory design. Sensors, 19(20), 2019. — 4521 с.

19. Wu, G., Miao, Y., Zhang, Y., Barnawi, A. Energy efficient for UAV-enabled mobile edge computing networks: Intelligent task prediction and offloading. Comput. Commun., 150, Jan. 2020. — 556-562 с.

20. 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.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)