ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

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

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

_____ *Владислав МАХИНЯ*

Виконав:
здобувач вищої освіти
група ІСДМ-62
Владислав МАХИНЯ

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

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

Київ – 2026

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 РОБОТУ
Махині Владиславу Сергійовичу
(прізвище, ім'я, по батькові здобувача)

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

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

затверджені наказом вищого навчального закладу від «30» жовтня 2025 року № 467.

2. Строк подання кваліфікаційної роботи: 26 грудня 2025 року.

3. Вихідні дані до кваліфікаційної роботи: датасет Iris;
бібліотеки Python: Sklearn, Matplotlib;
мінімальна відстань в Евклідовому просторі: Eps = 0.4;
алгоритми KMeans, DBSCAN;
науково-технічна література з питань, пов'язаних з наукою про дані.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Дослідження принципів та підходів машинного навчання.

2. Побудова ML-проєкту.
3. Дослідження методів та принципів Unsupervised Learning.
4. Розробка моделі машинного навчання на основі Unsupervised Learning.

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання: 30 жовтня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	30.10 – 05.11.25	
2	Дослідження особливостей машинного навчання	06.11 – 10.11.25	
3	Дослідження особливостей Unsupervised Learning	11.11 – 17.11.25	
4	Аналіз типів даних для побудови моделі	18.11 – 21.11.25	
5	Дослідження методу кластеризації k-means	24.11 – 08.12.25	
6	Дослідження алгоритму DBSCAN	09.12 – 12.12.25	
7	Оформлення роботи: вступ, висновки, реферат	15.12 – 19.12.25	
8	Розробка демонстраційних матеріалів	22.12 – 25.12.25	

Здобувач вищої освіти _____ Владислав МАХИНЯ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

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

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 78 стор., 32 рис., 7 табл., 30 джерел.

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

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

Предмет дослідження – методи, алгоритми та підходи до побудови моделі машинного навчання без вчителя.

Короткий зміст роботи: Досліджені принципи та підходи машинного навчання. Визначені етапи побудови ML-проєкту. Здійснено аналіз типів даних, які використовуються для побудови моделі машинного навчання. Досліджені особливості навчання Unsupervised Learning. Обгрунтовано вибір програмного забезпечення для розробки ML-моделі. Виконано розробку моделі машинного навчання на основі Unsupervised Learning. Виконано навчання та оцінку ML-моделі.

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

ABSTRACT

Text part of the master`s qualification work: 78 pages, 32 pictures, 7 table, 30 sources.

The purpose of the work is to develop a machine learning model based on Unsupervised Learning.

Object of research is the process of developing and training an ML model based on Unsupervised Learning.

Subject of research is methods, algorithms, and approaches to building an unsupervised machine learning model.

Summary of the work: Principles and approaches of machine learning are studied. The stages of building an ML project are determined. The types of data used to build a machine learning model are analyzed. The features of Unsupervised Learning are studied. The choice of software for developing an ML model is justified. A machine learning model based on Unsupervised Learning is developed. The ML model is trained and evaluated.

KEYWORDS: MACHINE LEARNING, UNSUPERVISED LEARNING, PYTHON PROGRAMMING LANGUAGE, VISUALIZATION, TECHNOLOGY, ALGORITHM, SOCKET, PROTOCOL, INTERACTIVE ENVIRONMENT, INFORMATION SYSTEM, MODEL.

ЗМІСТ

ВСТУП……..……9
РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРИНЦИПІВ ТА ПІДХОДІВ МАШИННОГО НАВЧАННЯ……...…11
1.1 Характеристика основних аспектів машинного навчання. Навчання з учителем……..…11
1.2 Навчання без вчителя……..14
1.3 Навчання з підкріпленням……..…19
РОЗДІЛ 2. ПОБУДОВА ML-ПРОЄКТУ……...……..22
2.1 Проєкт з машинним навчанням……..22
2.2 Визначення етапів ML-проєкту……..…26
2.3 Перевірка якості моделі машинного навчання…….29
2.4 Аналіз типів даних……...…31
РОЗДІЛ 3. ДОСЛІДЖЕННЯ МЕТОДІВ ТА ПРИНЦИПІВ UNSUPERVISED LEARNING……...……..37
3.1 Порівняльна характеристика класичних методів навчання……37
3.2 Особливості навчання Unsupervised Learning……..….39
3.3 Вибір та обгрунтування програмного середовища для розробки ML-моделі...42
РОЗДІЛ 4. РОЗРОБКА МОДЕЛІ МАШИННОГО НАВЧАННЯ НА ОСНОВІ UNSUPERVISED LEARNING……..53
4.1 Особливості кластерного аналізу даних……53
4.2 Алгоритм кластеризації k-means……....55
4.3 Процес кластеризації……...58
4.4 Реалізація ML-моделі з використанням методу k-means…….62
4.5 Реалізація алгоритму DBSCAN……..67
4.6 Ієрархічний кластерний аналіз……...……71
ВИСНОВКИ……...77
ПЕРЕЛІК ПОСИЛАНЬ…….79
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)……....82

ВСТУП

Актуальність теми. У сучасних умовах стрімкого розвитку цифрових технологій та постійного зростання обсягів даних машинне навчання стало одним із ключових інструментів для автоматизації аналітики, прогнозування та прийняття рішень. Особливе місце серед підходів машинного навчання посідають методи навчання без вчителя (Unsupervised Learning), які дозволяють працювати з неструктурованими або невідомо маркованими даними. В умовах, коли отримання великих обсягів розмічених даних є дорогим, тривалим або навіть неможливим процесом, алгоритми навчання без вчителя забезпечують можливість виявлення прихованих закономірностей, кластеризації, зниження розмірності та побудови узагальнених моделей даних без попереднього втручання людини. Це робить їх надзвичайно цінними для задач у галузях фінансів, медицини, кібербезпеки, Інтернету речей (IoT), обробки природної мови, комп'ютерного зору та інших перспективних і потрібних галузей.

Розроблення ефективних моделей машинного навчання на основі Unsupervised Learning сприяє підвищенню якості аналізу даних, оптимізації бізнес-процесів, покращенню систем рекомендацій і прогнозування. Крім того, такі моделі мають важливе значення для створення інтелектуальних систем, здатних до самонавчання та адаптації до нових умов без потреби у ручному маркуванні даних.

Отже, дослідження та побудова моделі машинного навчання на основі Unsupervised Learning є актуальним науково-практичним завданням, що відповідає сучасним тенденціям розвитку штучного інтелекту та потребам ринку інформаційних технологій.

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

Для досягнення поставленої мети необхідно виконати наступні *завдання*:

- дослідити особливості машинного навчання (ML);

- дослідити особливості класичного машинного навчання без вчителя (Unsupervised Learning);
- обгрунтувати вибір інструментів для аналізу та обробки даних, роботи з технологіями ML;
- здійснити вибір середовища розробки моделі ML;
- розробити модель ML на основі Unsupervised Learning;
- виконати оцінку якості роботи ML-моделі.

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

Предметом дослідження є методи, алгоритми та підходи до побудови моделі машинного навчання без вчителя.

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

Наукова новизна отриманих результатів: реалізовано модель машинного навчання типу *Unsupervised Learning* із застосуванням сучасних методів оптимізації та проведено її аналіз.

Практична значущість кваліфікаційної магістерської роботи. Результати дослідження можуть бути використані для створення моделей машинного навчання у сферах, що потребують автоматичної обробки даних. Запропонована модель на основі *Unsupervised Learning* підвищує точність прогнозування, оптимізує прийняття рішень та автоматизує аналіз даних. Отримані результати можуть використовуватися як навчальний матеріал для студентів, аспірантів і фахівців у галузі штучного інтелекту.

Апробація результатів та публікації:

Махиня В. С. «Етапи побудови та впровадження ML-проєктів». Тези доповіді на III Всеукраїнській науково-технічній конференції «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу». – Київ, 18 листопада 2025 року.

1 ДОСЛІДЖЕННЯ ПРИНЦИПІВ ТА ПІДХОДІВ МАШИННОГО НАВЧАННЯ

1.1 Характеристика основних аспектів машинного навчання. Навчання з учителем

Машинне навчання – це область штучного інтелекту (рис. 1.1).

Рис. 1.1 Місце машинного навчання в області штучного інтелекту

Штучний інтелект (Artificial intelligence, AI) - це галузь комп'ютерних наук, яка займається створенням систем і програм, здатних виконувати завдання, що зазвичай потребують людського інтелекту. До таких завдань належать: аналіз даних, розпізнавання образів і мовлення, прийняття рішень, планування, навчання на основі досвіду та прогнозування. У науковому контексті штучний інтелект - це сукупність методів, алгоритмів і технологій, що дозволяють комп'ютерам імітувати когнітивні функції людини, такі як розуміння, навчання, адаптація та самовдосконалення.

Машинне навчання (Machine learning, ML) - це напрям штучного інтелекту, який розробляє методи й алгоритми, що дозволяють комп'ютерним системам автоматично навчатися на основі даних без явного програмування під кожне завдання. Інакше кажучи, замість того, щоб задавати точні правила, система виявляє закономірності у даних і використовує їх для прогнозування або прийняття рішень. Слід зазначити, що машинне навчання - це галузь штучного інтелекту, що вивчає методи створення моделей і алгоритмів, здатних аналізувати

дані, виявляти закономірності та робити прогнози або приймати рішення без прямого втручання людини.

Алгоритми машинного навчання формують модель на основі навчальних даних (вибірки), що дозволяє системі здійснювати прогнозування або приймати рішення без необхідності явного програмування під кожне завдання. Такі алгоритми аналізують дані, виявляють у них закономірності та самостійно розв’язують поставлену задачу без втручання розробника.

Є три області машинного навчання: навчання з учителем, навчання без вчителя, навчання з підкріпленням (рис. 1.2).

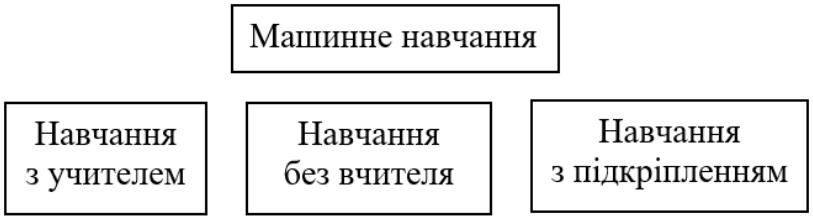


Рис. 1.2 Області машинного навчання

У *навчанні із вчителем* є навчальна пара об’єктів. Є приклади, на яких модель запам’ятовує побудову задачі. Наприклад, ми захотіли навчити модель машинного навчання на класифікацію зображень. На картинці голуб або кіт. У нас є пара об’єктів: картинка кота і цільова мітка – кіт, картинка голуба і цільова мітка – голуб (рис. 1.3).

Кіт Голуб

Рис. 1.3 Навчання із вчителем

Ці цільові мітки були отримані за допомогою спеціально навчених людей: розмітників (асессорів), які дивилися на наші дані і виставляли мітку класу. Тим самим отримуємо, що навчання із вчителем полягає в навчанні моделі на

розмічених прикладах. Ось так модель буде намагатися перевершити свого розмітника, свого вчителя. В області навчання із вчителем вирішуються задачі *класифікації* та *регресії*.

Класифікація – коли модель намагається побудувати розділяючу площину між класами (наприклад, розділити клас плюсів від класу кіл). Цільові значення мають дискретний категоріальний тип даних. Задача регресії – задача, де модель будує лінію, яка повторює закон, якому слідують наші дані. Цільові значення при цьому мають неперервний числовий тип даних (рис. 1.4).

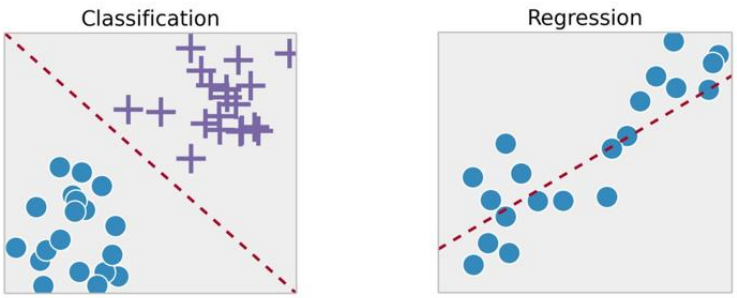


Рис. 1.4 Задачі класифікації та регресії

Оскільки цільове значення у класифікації дискретне, категоріальне, то сюди підходять будь-які задачі, де ми можемо передбачити категорію об’єкту. Перша задача – класифікація зображень. Тут на картинках можуть бути числа від 0 до 9 і тому маємо класифікацію на 10 класів. Це задача багатокласової класифікації (рис. 1.5).

Рис. 1.5 Багатокласова класифікація

Друга задача – передбачення відтоку клієнтів. Клієнт або з нами, або вже не з нами. Тільки 2 стани – задача бінарної класифікації. Тільки на 2 класи. Третя задача – знаходження шахрайських транзакцій: транзакція або шахрайська, або стандартна. Знову маємо бінарну класифікацію. Четверта задача – кредитний скоринг – приймаємо рішення за допомогою моделі - кому видаємо кредит, а кому

не видаємо, так як людина не добросовісна на думку моделі. Це також задача бінарної класифікації.

Задача регресії – передбачення неперевного числового цільового значення. Перша задача – передбачення вартості нерухомості (рис. 1.6). Цікаво дізнатися, яку об’єктивну вартість можна поставити будинку за допомогою моделі ML. Друга задача – передбачення прибутку. Зазвичай це відноситься до торгових точок, в яких ми прогнозуємо потенційний прибуток на період, щоб знати як можна використовувати ці ресурси.

Рис. 1.6 Регресія

Третій приклад – передбачення продажів або попиту на товари. Дані задачі можуть бути корисними, коли ми хочемо спрогнозувати поставку товарів. Особливо актуально для продуктів, які швидко псуються. Не хочеться, щоб дані продукти зіпсувалися із-за того, що покупці їх не купували.

1.2 Навчання без вчителя

Навчання без вчителя основане на знаходженні закономірностей всередині даних (рис. 1.7).

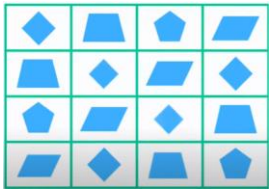


Рис. 1.7 Закономірності всередині даних

Моделі самостійно без підказок вчителя (розробника або асессора) намагаються зрозуміти суть задачі. У даному типі навчання у нас немає розмітки.

Спочатку ми навіть не знаємо, що нам буде видавати модель ML. Але за допомогою моделі ML без вчителя ми можемо глибше зрозуміти влаштування наших даних. Приведемо приклад з котами та голубами, але для задачі без вчителя. Якщо у нас є багато картинок котів і голубів, і ми їх хочемо відсортувати. Можемо скористатися хорошою переробкою зображень, а потім навчити модель кластеризації на сегментацію даних. Після того, як наша модель навчиться, ми зможемо отримати групи схожих зображень. В одній групі будуть фото котів, а в іншій групі будуть зображення голубів. При цьому дані групи будуть називатися не як клас кіт і клас голуб, а як кластер 1 і кластер 2, так як модель нічого не знає про класи наших об'єктів (рис. 1.8). Вона тільки зрозуміла, що картинка кота більше схожа на іншу картинку кота, ніж на картинку голуба. Це перша задача з області навчання без вчителя – *кластеризація*.

Інша задача називається *зниження розмірності*. Наші дані, на яких навчаємо моделі, дуже багатовимірні. Навіть наш набір даних у задачах класифікації по передбаченню відтоку клієнтів для операторів мобільного зв'язку може мати вигляд як представлено у табл. 1.1.

Рис. 1.8 Задача без вчителя

Таблиця 1.1

Набір даних у задачах класифікації по передбаченню відтоку клієнтів для операторів мобільного зв'язку

	Client	Gender	Minutes	SMS	Internet
0	0	0	10	3	3
1	1	1	0	100	600
2	2	0	400	230	20
3	3	0	30	25	400
4	4	1	300	60	50

По рядкам – клієнти. По стовбцям – характеристики, які описують наших клієнтів. Є унікальний ідентифікатор клієнта (стовпчик Client), є його належність до статі, є ознака кількості хвилин, які розмовляла людина, кількість SMS відправлених з номеру, кількість гігабайт, витрачених на користуванням Інтернетом. Можна знайти ще багато характеристик по нашим користувачам, але навіть на цьому прикладі маємо 5 ознак, 5 розмірностей і значить візуалізувати їх не зможемо. Якщо візьмемо тільки 2 ознаки – id-клієнта і його стать, то отримаємо візуалізацію на площині: id-клієнта по осі абсцис і стать клієнта – по осі ординат (рис. 1.9).

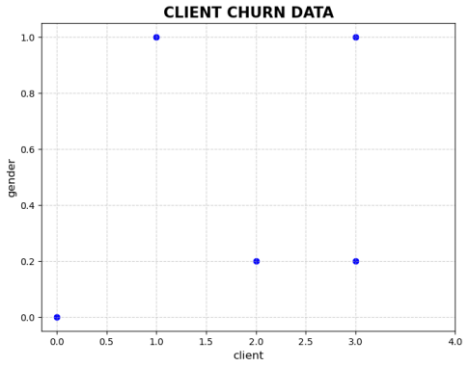


Рис. 1.9 Візуалізація даних у двохвимірному просторі

Якщо візьмемо 3 ознаки: id і стать клієнта, а також хвилини, то отримаємо візуалізацію у тривимірному просторі: id-клієнта по осі абсцис, стать клієнта по осі ординат, кількість хвилин – по третій осі (рис. 1.10).

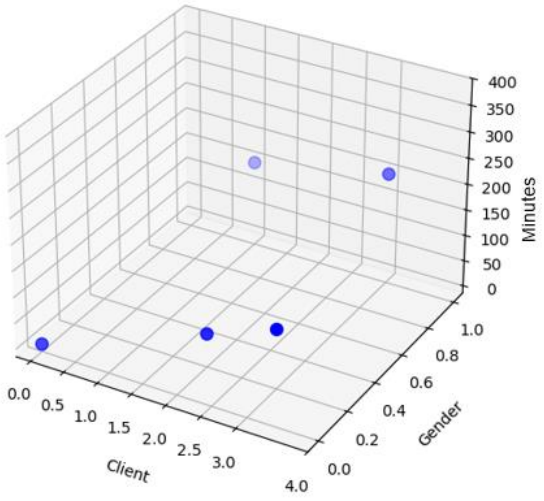


Рис. 1.10 Візуалізація даних у тривимірному просторі

Якщо візьмемо 4 ознаки (3 попередні + кількість SMS), то отримаємо візуалізацію у чотирьохвимірному просторі (рис. 1.11).

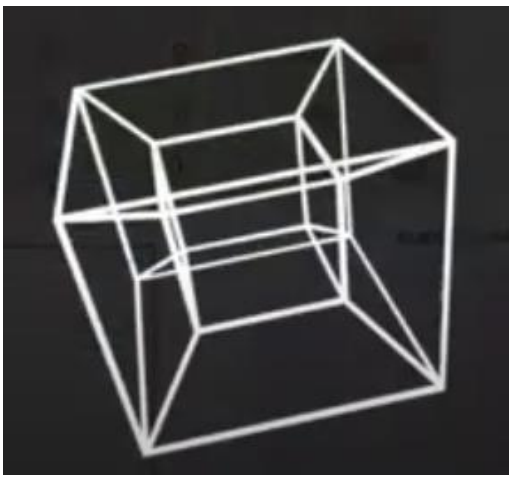


Рис. 1.11 Чотирьохвимірний простір

Але в силу фізіологічних особливостей будови наший очей і мозку ми не зможемо сприймати такий простір, а значить не зможемо проаналізувати наші дані. Але у нас є модель зниження розмірності, яка допоможе нам стиснути дані з більшою розмірністю.

Отримуємо, що моделі зменшення розмірності можуть стискати наші дані і отримувати їх основну інформацію у просторі меншої розмірності. Зробимо те саме з нашими п’ятивимірними даними.

Стиснемо розмірність до двох компонентів за допомогою моделі зменшення розмірності (табл. 1.2).

Таблиця 1.2

Зменшення розмірності

	Component_1	Component_2
0	-0.150934	0.767394
1	-0.792986	-0.415525
2	0.955911	0.087406
3	-0.293932	0.301077
4	0.281941	-0.740352

Тепер можемо візуалізувати наших клієнтів на площині і повивчати (рис. 1.12).

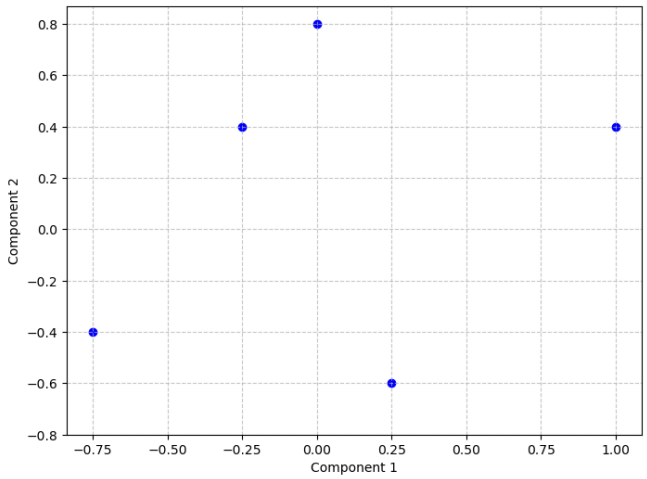


Рис. 1.12 Візуалізація даних зменшеної розмірності

Проаналізуємо декілька задач в області без вчителя. Спочатку кластеризація (рис.1.13). Найпростіший приклад взяти дані по клієнтам магазину і сегментувати, отримавши тим самим групи клієнтів.

Рис. 1.13 Приклад задачі кластеризації

Якщо клієнти потрапляють до однієї групи, це свідчить про наявність певних спільних характеристик між ними. Таку інформацію можна використати у системах рекомендацій, пропонуючи користувачам товари, які цікавлять схожих покупців. Інший приклад застосування кластеризації - групування продуктів харчування. Поділ їх на сегменти зі схожими властивостями дає змогу раціональніше формувати меню та оптимізувати розташування позицій у ньому.

У задачах із зниженням розмірності дані обробляються за загальною схемою. Спочатку набір даних подається на модель зменшення розмірності, після чого отримується стиснуте представлення даних. Далі проводиться візуалізація та

аналіз отриманого представлення. Незалежно від типу даних або конкретної задачі, порядок виконання цих кроків залишається незмінним.

1.3 Навчання з підкріпленням

Наступною областю є навчання з підкріпленням. Цей підхід базується на використанні позитивних та негативних підкріплень. У процесі навчання існує агент, який взаємодіє з оточуючим середовищем, отримуючи зворотний зв’язок для вдосконалення своїх дій (рис. 1.14).

Рис. 1.14 Навчання з підкріпленням

У навчанні з підкріпленням агент отримує винагороду за дії, що сприяють досягненню мети, яка відома лише розробнику. Дії, що не наближають агента до мети, супроводжуються покаранням. Агент не знає конкретної мети, але на основі отриманих сигналів винагороди або покарання навчається вибирати оптимальні дії, максимізуючи суму балів.

Або так само можна навчати агентів ходи з різного роду перешкодами. Перший агент навчився перестрибувати через перешкоди (рис. 1.15). Спочатку агент не мав конкретної мети навчитися перестрибувати через стіни; він орієнтувався лише на отримані винагороди та покарання. Навчання супроводжувалося труднощами: агент зазнавав невдач, не долав перешкоди, падав у ями і процес його адаптації був поступовим та експериментальним. Проте з часом, завдяки системі підкріплення та накопиченому досвіду, агент почав поступово вдосконалювати свої дії, навчався уникати помилок і, зрештою,

опановував необхідні дії для успішного виконання завдань. Такий підхід демонструє принцип навчання з підкріпленням, коли ефективність досягається через проби, помилки та отриманий зворотний зв’язок.

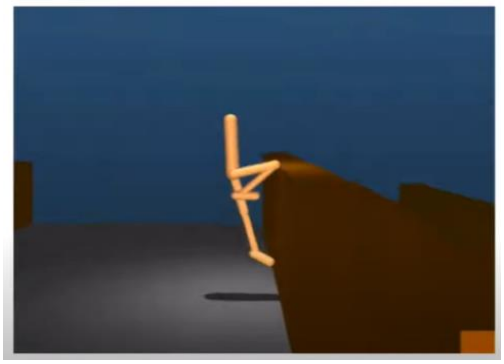


Рис. 1.15 Навчати агентів ходи з різного роду перешкодами

Отже, у навчанні з учителем наявні цільові значення, які заздалегідь розмічені у навчальному наборі даних. Цільова змінна може бути дискретною або неперервною числовою, що визначає тип задачі - класифікацію або регресію. Завдяки цьому модель навчається прогнозувати або класифікувати нові дані на основі відомих результатів. У навчанні без вчителя цільових значень немає і алгоритми намагаються виявити приховані закономірності та структуру в даних. До таких задач відносяться кластеризація та зниження розмірності, що дозволяє краще зрозуміти склад даних і взаємозв’язки між ознаками. У навчанні з підкріпленням існують агенти, які навчаються виконувати поставлене завдання за допомогою позитивного та негативного підкріплення. Агент отримує винагороди за правильні дії та покарання за помилкові, що дозволяє йому поступово підвищувати ефективність і адаптуватися до змін у середовищі.

Ознаки (предиктори) (features, predictors) – це вхідні дані, за якими алгоритм навчається робити передбачення. Вони описують об’єкти у вигляді ***числових*** або ***категоріальних характеристик***.

Наприклад:

- зріст, вага, вік людини;
- кількість кліків на сайті;
- середня температура, вологість, час доби.

Цільова змінна (target, label, dependent variable) – це те, що потрібно передбачити або класифікувати на основі ознак. У задачах класифікації вона категоріальна (приймає обмежену кількість класів).

Наприклад:

- здоровий/хворий (медична діагностика);
- спам/не спам (фільтрація пошти);
- купити/не купити (маркетинг);
- клас 1/клас 2/клас 3 (багатокласова класифікація).

Всі три підходи належать до галузі машинного навчання і мають спільну ціль - навчити модель робити узагальнення на основі даних. У кожному випадку алгоритми працюють з даними (вхідними прикладами, ознаками). Усі підходи передбачають пошук оптимальної моделі або політики, що мінімізує (або максимізує) певну функцію втрат або винагород. Порівняльна характеристика типів машинного навчання представлена у табл. 1.3.

Таблиця 1.3

Порівняльна характеристика типів машинного навчання

Ознака	**Навчання з учителем**	**Навчання без вчителя**	**Навчання з підкріпленням**
Наявність цільових значень	Кожен приклад має правильну відповідь (мітку)	Модель сама шукає закономірності в даних	Є зворотний зв’язок у вигляді винагороди або покарання
Мета навчання	Навчити модель прогнозувати або класифікувати результати	Виявити структуру, схожість або приховані зв’язки між даними	Навчити агента приймати послідовність дій, що максимізує винагороду
Тип задач	Класифікація, регресія	Кластеризація, зниження розмірності, асоціативні правила	Оптимальне керування, ігри, навігація
Тип навчального сигналу	Явний – правильні відповіді відомі	Неявний – правильних відповідей немає	Частковий - оцінка якості дії через винагороду
Приклад	Розпізнавання зображень, прогнозування цін	Групування клієнтів за поведінкою	Робот, який навчається ходити або грає в шахи
Основна складність	Потреба у великій кількості розмічених даних	Складність інтерпретації отриманих структур	Баланс між дослідженням нових дій і використанням отриманого досвіду

2 ПОБУДОВА ML-ПРОЄКТУ

2.1 Проєкт з машинним навчанням

Визначимо черговість етапів ML-проєкту. Розглянемо, яким чином здійснюється побудова етапів ML-проєкту. Розглянемо побудову проєкту на прикладі задачі класифікації об’єктів на дорозі для безпілотного автомобіля. Обмежимося класами машин, людей і світлофорів. Для початку потрібно визначити бізнес-проблему і визначити чи потрібне тут машинне навчання. Ми хочемо без водія бачити пішоходів, які знаходяться поряд, машини, які проїжджають і вчасно зупинятися на червоне світло світлофору. Проблема ясна, а чи потрібне тут машинне навчання? Також так. Картинок з машинами, людьми та світлофорами багато. Тут можемо навчити модель на знаходження потрібних нам класів, якщо візьмемо зображення з відеореєстратора і розмітимо потрібні нам об’єкти (рис. 2.1).

Рис. 2.1 Приклад зображення з відеореєстратора

Для успішного вирішення завдання потрібно його перевести на мову машинного навчання. Потрібно зрозуміти, що будемо передбачати, яке цільове значення і на яких даних будемо це робити. Для безпілотника ми передбачаємо місцеположення потрібних об’єктів на зображенні. Наприклад, для пішоходів або для машин. Задача називається детекцією. Які для цього потрібні дані? Потрібні картинки і розмітка – координати об’єктів, а вони можуть бути представлені різними способами. Використаємо спосіб представлення даних через координату

мінімальних *x* та *y*, і координати максимальних *x* та *y*. Візьмемо найбільший автомобіль та його координати. Мінімальна з координатою x=367 та координатою y=300, і максимальна з x=745 та y=648 (рис. 2.2).

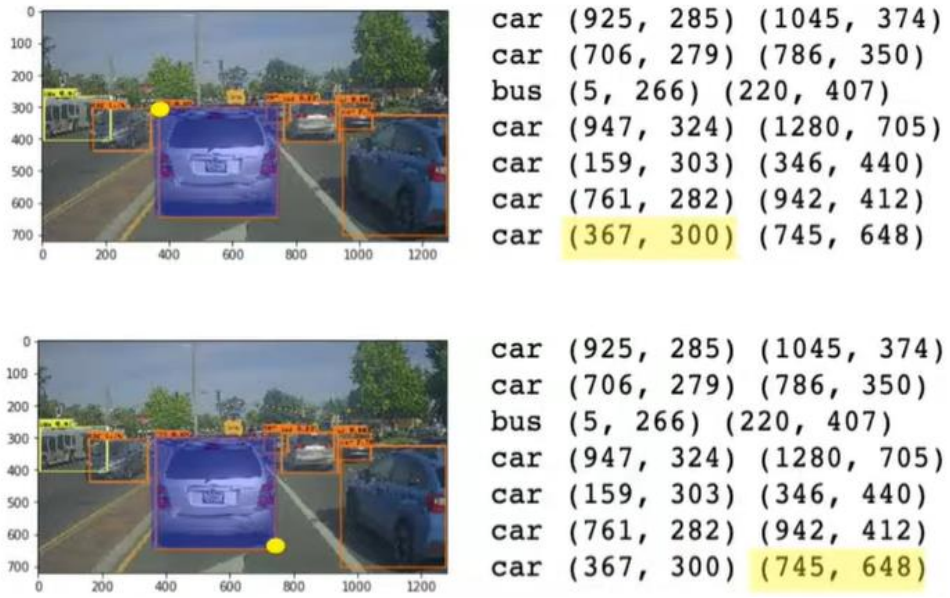


Рис. 2.2 Приклад представлення даних для моделі машинного навчання

Наступний крок – це вибрати метрики, за допомогою яких ми будемо розуміти, на скільки добре вирішуємо задачу. Виділяють два типи метрик: *метрики машинного навчання* – ними користуються датасаєнтисти, коли навчають модель, і *метрики для бізнесу* – вони потрібні, щоб оцінити користь від впровадження ML-рішення. Для нашої задачі метрикою для ML може виступати кількість правильно знайдених об’єктів. Якщо ми знайшли всі машини, всі світлофори і всіх пішоходів, то метрика хороша, висока. А якщо ми знайшли тільки машини й не побачили світлофори, та людей, то метрика гірша.

Для бізнесу підійде метрика, яка оцінює кількість дорожньо-транспортних пригод з безпілотниками. Якщо їх кількість після впровадження ML-рішення ніяк не зміниться або ДТП стане більше, що гірше, то метрика низька. А якщо ДТП стане менше, що краще для людей, то бізнес-метрика буде високою.

Підготовчі етапи для вирішення задачі закінчилися. Тепер наступає серйозніша робота, адже потрібно шукати дані. І тут є 2 шляхи. Якщо ми будемо працювати із

замовником і він буде зацікавлений у якісному рішенні, то він надасть свої дані. У такому випадку замовник може надати таблиці, доступ до баз даних (БД), доступ до сховища даних, якщо це картинки або звуки (рис. 2.3).

	A	B	C	D	E	F	G
1	Donation ID	Donor ID	Type	Method	Status	Amount	Date
2	D153	S067	Donation	Credit card	Completed	$100,00	02.05
3	D154	S123	Shirt	Credit card	Abandoned	$25,00	03.05
4	D155	S345	Shirt	Paypal	Completed	$25,00	07.05
5	D156	S367	Donation	Cash	Completed	$50,00	07.05
6	D157	S121	Shirt	Paypal	Failed	$25,00	07.05
7	D158	S112	Donation	Credit card	Completed	$75,00	10.05
8	D159	S055	Donation	Credit card	Completed	$100,00	12.05
9	D160	S089	Donation	Paypal	Completed	$60,00	13.05
10	D161	S523	Shirt	Credit card	Failed	$25,00	15.05
11	D162	S123	Shirt	Cash	Completed	$25,00	17.05
12	D163	S165	Donation	Paypal	Abandoned	$35,00	17.05
13	D164	S094	Donation	Paypal	Completed	$75,00	21.05
14	D165	S054	Donation	Credit card	Completed	$50,00	22.05
15	D166	S031	Shirt	Credit card	Completed	$25,00	24.05
16	D167	S451	Donation	Cash	Completed	$100,00	24.05
17	D168	S112	Shirt	Credit card	Abandoned	$25,00	27.05
18	D169	S002	Donation	Paypal	Failed	$50,00	27.05
19	D170	S112	Shirt	Credit card	Completed	$25,00	30.05
20	Total May					$895,00	

Рис. 2.3 Дані від замовника

Бувають випадки, коли потрібно самостійно виконувати пошук даних. Тоді потрібно завантажувати дані з просторів Інтернету, виконувати парсинг або скрапінг даних, або навіть самостійно збирати датасет.

Наступний етап – етап вивчення даних. Потрібно зрозуміти, на скільки дані правильні та корисні. Це етап аналізу даних – розвідувальний аналіз даних або ж первинний аналіз даних (EDA, Exploratory Data Analysis). На цьому етапі обчислюються статистики, будують візуалізації та вивчають характеристики даних (рис. 2.4).

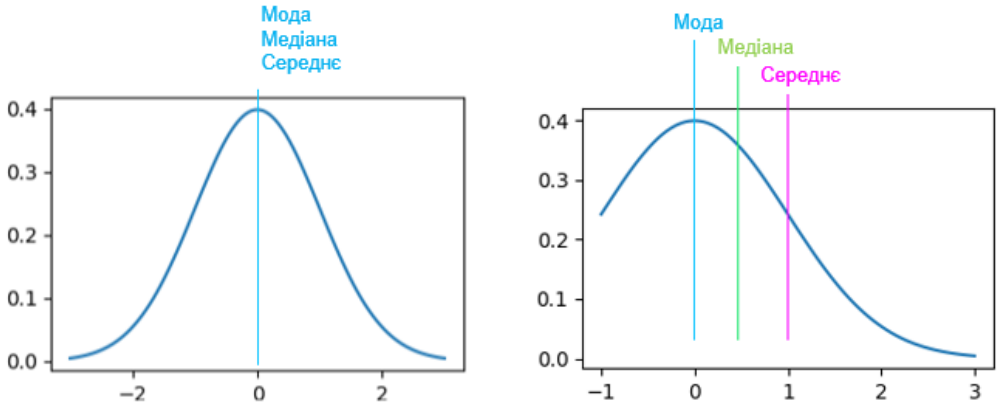


Рис. 2.4 Вивчення даних

Обов’язково варто повивчати цільову ознаку, її розподіл, подивитися відповідність розмітки та даних. А також варто повивчати набір даних взагалі, які вони: чисті чи брудні. З безпілотником можна подивитися розподіл цільової ознаки – це наші об’єкти на дорозі. У наших даних могло так вийти, що найчастіше на зображеннях дороги зустрічаємо машини (синім), потім – людей (червоним), а потім – світлофори (жовтим) (рис. 2.5).

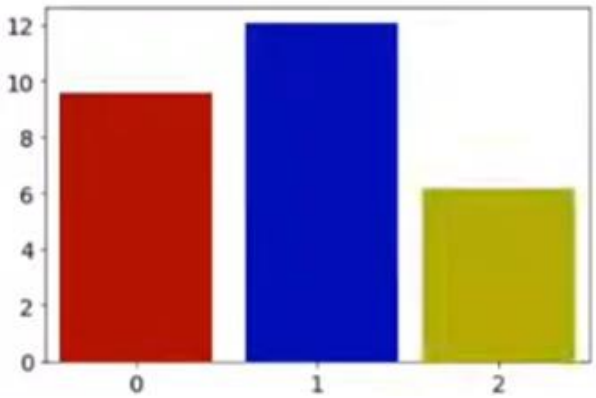


Рис. 2.5 Розподіл цільової ознаки

І також тут можемо подивитися чи відповідає розмітка дійсним об’єктам. Бувають випадки, коли розмітка отримана не вірно. Відповідно такі ситуації потрібно відслідковувати на процесі аналізу даних. Потрібно в цілому повивчати датасет. Неякісні дані потрібно видалити із навчального набору.

Наступний етап – передобробка даних. Даний етап більш обширний для задач з табличними даними. Тут потрібно буде обробити пропуски, обробити викиди, масштабувати дані і придумати нові корисні характеристики. На прикладі нашої задачі список дій звужується до очищення неякісних картинок та масштабування. Видалення поганих картинок потрібне для того, щоб модель ML навчалася якісніше і правильніше, а масштабування – для того, щоб модель навчалася простіше. Також картинки зазвичай ресайзять до однакового розміру для простоти навчання. Після отримання даних гарної якості, можна навчати модель машинного навчання. Складність цього етапу полягає в тому, що потрібно підібрати саме ту модель, яка підходить для нашої задачі і вибирати приходиться з дуже великої кількості моделей. Є класичні одиночні алгоритми, є композиції моделей, а є й множина архітектур нейронних мереж (рис. 2.6 – рис. 2.8).

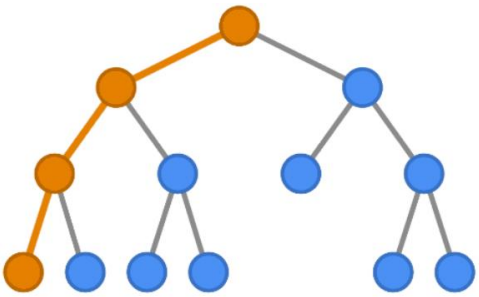


Рис. 2.6 Класичний алгоритм машинного навчання

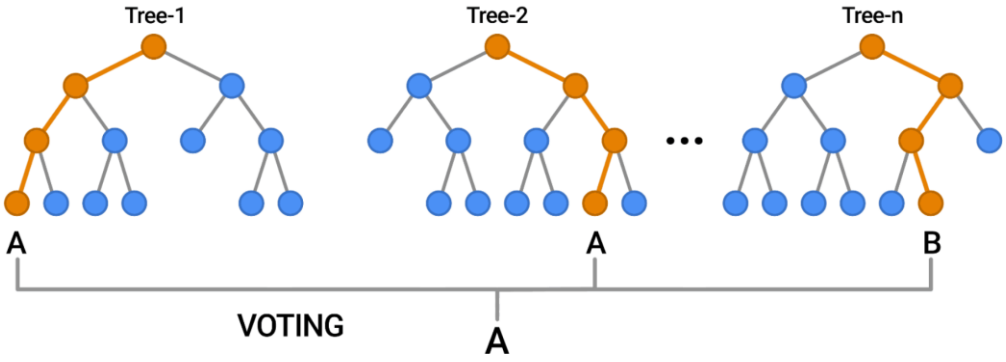


Рис. 2.7 Композиції моделей

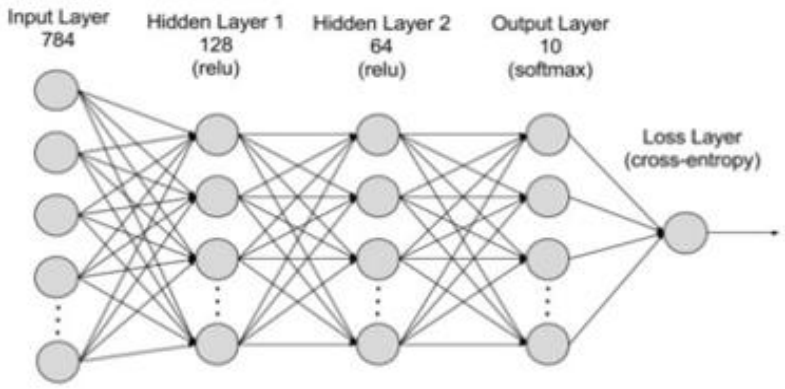


Рис. 2.8 Нейронна мережа

2.2 Визначення етапів ML-проєкту

У рамках завдання необхідно визначити етапи ML-проєкту.

Мета етапу *визначення бізнес-проблеми* - зрозуміти, яку конкретну задачу потрібно вирішити та яку цінність це принесе бізнесу, користувачам або організації. На цьому етапі визначають, чому виникла потреба у використанні машинного навчання і якого результату очікують. Аналітик або дослідник повинен глибоко зануритися у контекст предметної області, щоб правильно сформулювати задачу в термінах даних і моделей. Результат етапу:

чітко сформульована проблема у бізнес-термінах, перетворена на аналітичну або технічну задачу, яку можна розв’язати методами машинного навчання.

Після того як зрозуміло, у чому полягає бізнес-проблема, її потрібно представити у термінах машинного навчання. Мета етапу - *перекласти бізнес-проблему у форму, яку можна вирішити за допомогою методів машинного навчання*, тобто визначити, який тип ML-задачі потрібно розв’язати, які дані потрібні та які метрики ефективності будуть використовуватися. Результат етапу: сформульована ML-задача з чітко визначеними цілями, типом моделі, ознаками та метриками оцінки. Тобто бізнес-проблема перетворюється на технічно вимірюване завдання, яке можна реалізувати програмно.

Етап вибору відповідних метрик дозволяє забезпечити об’єктивне вимірювання якості моделі машинного навчання шляхом визначення показників ефективності (метрик), які найкраще відображають досягнення поставленої цілі. Метрики - це кількісні показники, що дозволяють оцінити точність, стабільність і придатність моделі для вирішення конкретної задачі. Вибір метрики залежить від типу проблеми (класифікація, регресія, кластеризація, прогнозування) і контексту застосування. Неправильний вибір метрики може призвести до помилкових висновків про ефективність моделі або навіть до прийняття хибних рішень у бізнесі.

Мета *етапу отримання даних* - зібрати необхідні вихідні дані, на основі яких буде виконуватись навчання, аналіз і побудова моделі машинного навчання. На цьому етапі здійснюється збір, об’єднання та попередня оцінка якості даних, які можуть надходити з різних джерел. Головне завдання - отримати репрезентативну, повну і достовірну вибірку, що відображає реальний об’єкт або процес, який потрібно моделювати.

На *етапі аналізу даних* формується розуміння того, з чим працює модель. Аналітик досліджує набір даних, щоб з’ясувати його якість, розподіл змінних, наявність пропусків, аномалій або нерівномірності. Результати аналізу дозволяють прийняти рішення про подальшу передобробку, відбір ознак і вибір алгоритму машинного навчання.

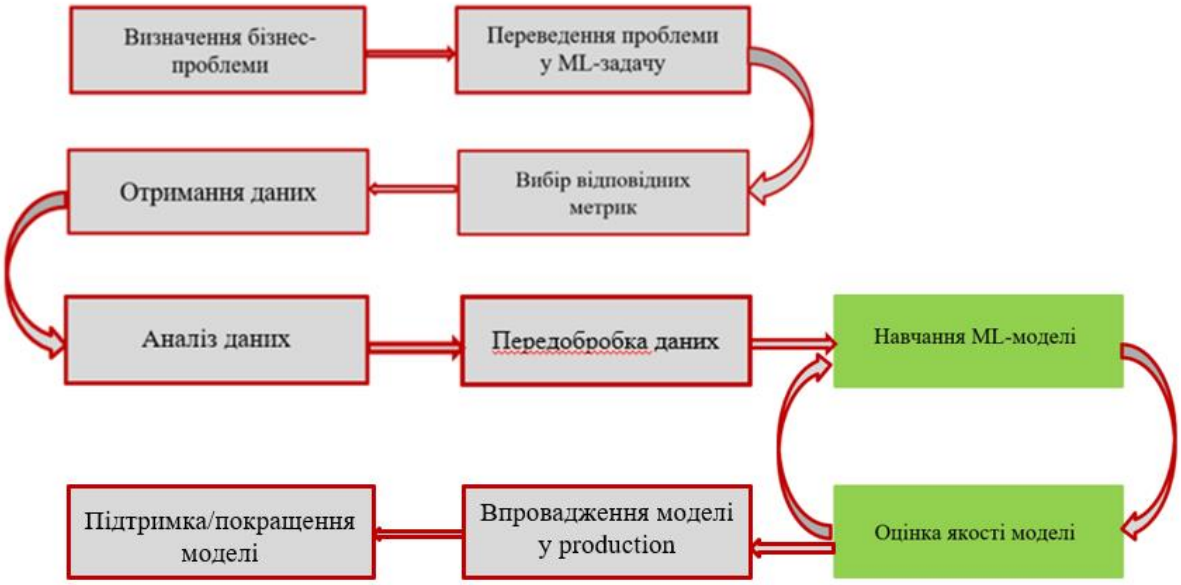


Рис. 2.9 Етапи ML-проєкту

Мета *етапу передобробки даних* - підготувати дані до ефективного використання у моделях машинного навчання шляхом очищення, трансформації та стандартизації. Результат етапу: отримання очищеного, узгодженого і підготовленого набору даних, готового до подальшого використання у моделюванні. Після передобробки дані матимуть оптимальний формат і структуру, що забезпечить стабільну та точну роботу алгоритмів машинного навчання.

Після того як дані очищені та підготовлені, настає етап, коли обирається алгоритм машинного навчання і здійснюється *процес навчання моделі*. Мета - знайти оптимальні параметри моделі, які мінімізують похибку прогнозування або максимізують точність класифікації. Тип алгоритму повинен залежати від поставленої задачі.

Після навчання та перевірки модель має бути перенесена з експериментального середовища (наприклад, Jupyter Notebook, Google Colab) у реальну систему, де вона буде виконувати свої функції: прогнозувати, класифікувати, рекомендувати та інше. *Етап впровадження моделі у production* вимагає не лише технічної інтеграції, а й забезпечення стабільності, масштабованості та безпеки роботи моделі.

Мета *етапу підтримки/покращення моделі* - забезпечити стійку, актуальну та ефективну роботу моделі після її впровадження у production. Модель машинного навчання не є статичною, з часом дані, середовище або поведінка користувачів змінюються, тому її потрібно регулярно контролювати, оновлювати й оптимізувати. Модель постійно залишається актуальною, точною та надійною, адаптуючись до нових даних і змін у середовищі. Система підтримки гарантує довгострокову ефективність і стабільність роботи ML-рішення.

2.3 Перевірка якості моделі машинного навчання

Крім моделі, потрібно вибрати ще й найоптимальніші параметри для неї, яких також може бути дуже багато. Оскільки існує великий різновид моделей, то для того, щоб зрозуміти, що знайдена нами модель є кращою, для цього є етап, який називається оцінка якості моделі (рис. 2.9). При цьому етап навчання моделі і оцінка якості моделі постійно йдуть один за одним.

Потрібно навчити першу модель й оцінити її якість, потім навчити другу модель й оцінити її якість. І так робити до тих пір, поки не отримаємо найкращу модель на нашу думку та на думку метрик, які ми визначили обчислювати. Тут розуміємо, яка з навчених моделей показує найкращу якість. При цьому потрібно зробити розбиття всіх даних на дві вибірки: навчальну та тестову (рис. 2.10).

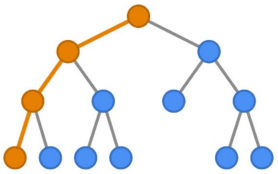
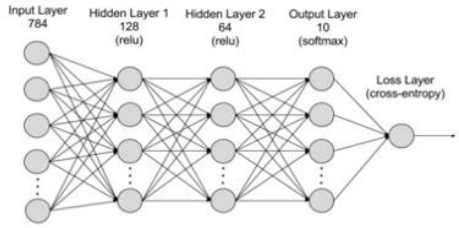
Рис. 2.10 Розбиття даних на навчальну та тестову вибірки

На навчальній вибірці модель навчається, шукає закономірності, залежності та запам’ятовує, як знаходити на даних машини, людей та світлофори. А на тестовій ми перевіряємо як модель вловила зміст задачі з навчальної вибірки. І по успішності моделі на тестовій вибірці - робимо висновок про той алгоритм, який найбільше підходить.

Наприклад, якщо ми хочемо проекспериментувати з двома моделями, беремо першу модель і навчаємо на навчальній вибірці. А щоб перевірити якість моделі, проганяємо її на тестових даних і дивимося на передбачення.

Таблиця 2.1

Оцінка якості моделі машинного навчання

Модель_1			Модель знайшла 3 об’єкти з 11
Модель_2			Модель знайшла 10 об’єктів з 11

На етапі вибору метрик була визначена метрика - кількість правильно знайдених об’єктів. Перша модель знайшла 3 об’єкти з 11 можливих. Кількість правильних класифікацій дорівнює 3 – не найкращий показник. Показуємо навчальну вибірку другій моделі, а потім показуємо тестові дані, на яких вона ще не навчалася. Друга модель знайшла 10 об’єктів з 11 можливих (табл. 2.1). Виходить, що метрика другої моделі краща, значить це і буде сама ідеальна модель по метриці.

Далі цю модель потрібно відправити у реальний світ і дати доступ клієнту до неї - етап впровадження моделі у production (рис. 2.9). Адже модель - це ще не кінцева система, яку бачить користувач, а це тільки невелика її внутрішня частина. Тому на цьому етапі в роботу вступають й інші розробники, які створюють мобільні додатки, веб-додатки, десктопні рішення або займаються налаштуванням інфраструктури. При цьому не завжди модель буде спокійно знаходитися на фізичному або хмарному сервері та працювати через API.

Іноді потрібно буде користуватися моделлю ML прямо з мобільного додатку або потрібно буде покласти її поряд із всією системою розпізнавання як у нашому

прикладі з безпілотником. Тому що автомобіль може бути на трасі, де не завжди ловить мережа, тому зв’язок з моделлю буде втрачатися, запити до неї не будуть доходити. Тому доцільно користуватися компактними вбудованими модулями і класти їх поряд з мозком нашої самостійної машини.

Модель реалізована. Далі потрібно займатися підтримкою та покращенням моделі. Це і є завершальний етап у проєкті. Через зміни у даних іноді приходиться перенавчати моделі на новинках. Особливо це актуально для передбачення вартості або прибутку. Так як передбачення моделі можуть дуже відрізнятися від того, що є насправді. Ми могли свого часу навчити модель на низьких цінах, через декілька років ціни зросли і тепер модель показує неправдиві ціни. Варто її перенавчити на нових актуальних даних, щоб вона відповідала реаліям. Або ж клієнт може захотіти добавити більше класів у модель, щоб розширити задачу. Й тут приходиться займатися донавчанням моделі, щоб покривати всі бажання клієнта.

Проаналізуємо, що, наприклад, може відбутися з нашим безпілотником. У нас спочатку задача заключалася у знаходженні трьох класів на зображенні: клас машин, пішоходів та світлофорів. Хочемо розширити задачу і знаходити на фотографіях ще і знаки дорожнього руху - новий об’єкт, на якому модель не навчалася. Значить потрібно пройти знову деякі етапи: отримати дані, проаналізувати їх, виконати передобробку даних, навчити модель та виконати оцінку її якості. Якщо додавання нового класу у модель нічого нам не зламало (а краще звичайно, щоб ця дія призвела до покращення метрик), то можемо цю модель відправляти у production, а далі знову підтримувати або покращувати модель до нових змін. Знову повторили деякі етапи нашого проєкту і це цілком нормально – це і є здоровий цикл отримання якісної моделі машинного навчання.

2.4 Аналіз типів даних

При побудові моделі машинного навчання важливо розуміти, який тип даних у характеристик, щоб більш коректно їх аналізувати і правильно підготовлювати до

моделі. Дослідження типів даних проведемо на передбаченні кількості продажів у торговій точці в залежності від різних характеристик (табл. 2.2).

Таблиця 2.2

Характеристики даних

	date	shop_id	shop_rating	city	item_id	desciption	quantity	item_price	is_promo	sales
0	2021-12-17	3	5	IEV	9	Відчуйте себе професійним гонщиком...	188	4081.88	0	2
1	2021-12-17	14	4	LWO	8	Пряме та блискуче волосся всього за ...	238	3162.27	1	19
2	2021-12-17	15	5	ODS	4	Відмінно підійде для розвитку дитини...	37	1078.41	0	17
3	2021-12-18	6	3	IEV	7	Королівський подарунок для краси та ...	234	215.30	0	15
4	2021-12-18	16	4	LWO	16	Чайник – розумний приклад з диста...	143	3147.63	1	6

Перша ознака - дата. Далі йдуть ознаки, які пов’язані із самою торговою точкою (їх 3). Її ідентифікаційний номер, її рейтинг і місто, в якому розміщується магазин. Наступна група ознак, які пов’язані з товаром. Маємо ознаку *id* товару, його опис, кількість товару у торговій точці, його актуальна вартість і чи є акція на даний товар. Наша остання ознака (вона ж цільова) - кількість продажів у даній торговій точці визначеного товару.

Досліджені характеристики дуже різні. Є цифри цілі, десяткові, є рядкові ознаки, при цьому з малою кількістю літер та з досить великою, що аж не влазить у всю комірку таблиці. Ще є дата. І на всьому цьому ми можемо навчати моделі з невеликою передобробкою.

Ознаки діляться на якісні, кількісні, текстові і на ознаки з типом даних *дата*. Кількісні у свою чергу поділяються на неперервні та дискретні. А якісні діляться на порядкові та номінативні. При цьому у номінативних є приватний випадок – бінарні ознаки.

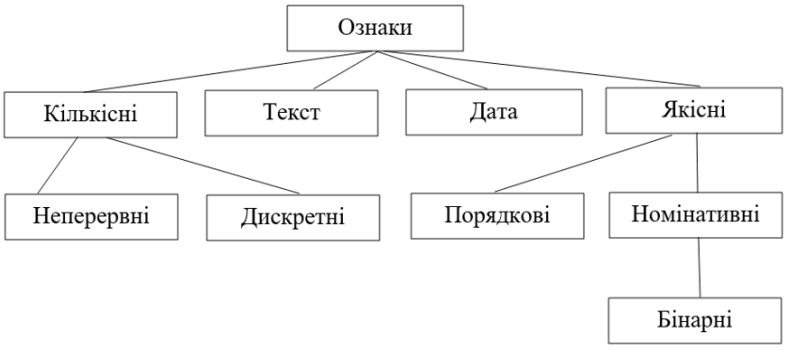


Рис. 2.11 Типи ознак

Кількісні ознаки - це характеристики даних, які можна виміряти і представити у вигляді чисел. У нашому прикладі з передбаченням продажів у магазині такими ознаками є всі стовбці з числовими даними, так як всі вони представляють собою цифри. Але насправді, справжніми кількісними ознаками є тільки *quantity, item_price* і *sales*. Інші характеристики відносяться до інших типів даних.

Кількісні ознаки можна виміряти і порівняти між собою. Виміряти ідентифікатор магазину ми не можемо. Це всього лише унікальний ідентифікатор нашого магазину - глибокого числового сенсу в ньому не має.

Для того, щоб перевірити ознаку на вимірюваність, можна порівняти між собою значення. Чи можемо ми сказати, що магазин з 15-тим id-шником є більшим і кращим, ніж магазин з 6-тим id? Звичайно ж ні, так як такі речі неможливо порівняти між собою. А кількісні ознаки ми можемо вимірювати порівнювати. Так що наша ознака shop_id не є кількісною ознакою.

Shop_rating. Магазин з рейтингом 5 звучить краще, ніж з рейтингом 3 знову ж на перший погляд, але ж ми не знаємо як вимірювався даний рейтинг. Так що об'єктивно оцінити цей рейтинг і оформити його в цифру не можемо. А значить дана характеристика не кількісною, а є якісною.

item_id. По аналогії з shop_id ми не можемо порівняти id товарів між собою. А значить ознака не може бути віднесена до кількісної.

Далі *quantity* - кількість товару в магазині. Цю характеристику ми можемо виміряти, порахувати товар на складі не складе труднощів. І значення ознак можемо порівнювати: кількість товару 188 - це значно більше, ніж 37. Так що врешті решт знайшли першу кількісну ознаку.

Наступна характеристика item_price - вартість товару. Виміряти її можемо, порівняти вартість між собою теж можемо. Це є кількісною ознакою.

Далі is_promo - чи знаходиться зараз товар на акції. 0 - неакційний товар, 1 – акційний товар. Виміряти мітку акційності не можемо, так що маємо справу не з кількісною характеристикою, а з якісною.

І залишається цільова ознака sales - кількість продажів. Порахувати, виміряти і порівняти скільки товарів продано ми можемо. Знову кількісна ознака.

Тепер залишається зрозуміти, а до якого типу кількісних ознак відносяться наші характеристики: неперервних чи дискретних.

Неперервні дані можуть приймати практично будь-які значення, у тому числі і дробові, десяткові значення. Ну і приклади таких ознак - це вік людини, який може бути 20 років, а може бути 30.5 років. Також сюди відносять зріст людини, який може бути 169 см, а може бути 169,3 см. Швидкість машини. По аналогії швидкість машини може бути 60 км/год, а може бути 45.5 км/год. І вартість товару, яка містить копійки.

Визначимо, які ознаки із наших даних відносяться до неперервних. Кількість товару на складі приймає тільки ціле значення, так що вже не підходить. Ціна товару приймає будь-які значення від 0 до нескінченності. Так що це неперервна ознака. Кількість продажів приймає тільки ціле значення, тому що продати половину телевізора ми не можемо, так що це ознака не неперервна.

Дискретні або ж *перервні ознаки* - це такі ознаки, які утворилися через підрахунок у випадках частоти, кількості, результатів. Тобто дані можуть бути тільки цілими (можуть приймати тільки цілі значення). Приклади таких ознак – кількість жителів (у нас не може бути тільки половина людини), кількість кредитів або частота серцевих скорочень.

У нашому датасеті ми підраховуємо кількість товарів на складі, а значить маємо величину дискретну. Ознака quantity - перервна. Також підраховуємо кількість продажів, а значить знову отримуємо величину дискретну.

Тепер переходимо до наступного типу характеристик - *текстовим*. Сюди відносяться ті ознаки, які складаються з великої кількості літер. У наших даних є дві текстові ознаки. Але в одного - city - унікальні значення ознаки обмежені. Торгова точка може розміщуватися тільки у трьох містах: Києві, Львові та Одесі. Так що ознака більше підходить не до текстової, а до категоріальної.

А ось в ознаці опис товару кожне значення є унікальним, немає повторень і значить справжньою текстовою ознакою є колонка description.

Дата. У нашому датасеті ознака date містить дійсно дату: рік, місяць, день.

Якісні (категоріальні) ознаки - це такі характеристики даних, які важко виміряти, а якщо вже й вийшло виміряти, то часто ці вимірювання суб'єктивні, інакше кажучи - якісні ознаки складаються з деяких категорій.

Є некількісна ознака *shop_id* (ідентифікатор магазину). Ми не можемо порівнювати значення між собою. Магазин з 6-тим id не більший, ніж магазин з id=3. А значить це точно категоріальна ознака.

Shop_rating. Магазин з рейтингом 4 кращий, ніж з рейтингом 3, але це доволі суб'єктивна думка. Перед нами категорія.

Item_id по аналогії з shop_id - не можемо порівнювати id товарів між собою. Ознака не кількісна.

Далі ознака is_promo: акція є або акції не має. Категорія.

Залишилася ще одна категоріальна ознака - це city.

Порядкові ознаки - це ті ознаки, в яких є категорії і до того ж між ними можна вибудувати порядок.

Впорядковані категорії. Наприклад, рейтинг ресторану - три зірки (краще, ніж дві). Рівень освіти (бакалаврат зазвичай нижче, ніж магістратура). Або ж стаж роботи (10 років та 1 рік).

У нас в даних із порядкових ознак є тільки рейтинг магазину. В інших – вибудувати порядок ми не можемо, так як одне значення категорії не більше, ніж інше.

Номінативні - це ті ознаки, які не можемо виміряти, тобто маємо категорії. Їх значення не мають порядку. Приклади ознак: сімейний стан (не зрозуміло, що краще – бути одруженим чи бути неодруженим), так як це неможливо порівняти – це просто категорії. Місто, де проживає клієнт – також не можна порівняти, яке місто краще. Колір волосся – теж між собою порівнювати не можемо, а якщо і можемо – це все справа смаку.

У даних номінативними ознаками є shop_id (категорії унікального ідентифікаційного номеру магазину), item_id (категорії унікального ідентифікаційного номеру товару), city (категорія виду міста, де розміщується магазин) та is_promo (категорія у вигляді мітки - є акція або не має).

При цьому на ознаку is_promo ми можемо подивитися уважніше: вона приймає тільки 2 значення, а значить ознака відноситься до більш вузької категорії номінативних ознак - до бінарних. Бінарні характеристики приймають тільки 2 значення. Наприклад, сюди можемо віднести ознаку статі людини; оформив підписку клієнт або ні; ходить людина у спортзал або ні.

Тепер ми знаємо до якого типу даних відноситься кожна наша ознака. Date - дата. Shop_id - категорія номінативна. Shop_rating - категорія порядкова. City - категорія номінативна. Item_id - номінативна ознака. Description - текст. Quantity - кількісна дискретна ознака. Item-price - неперервна кількісна ознака. Is_promo - бінарна ознака. Sales - дискретна ознака. Маючи цю інформацію, ми можемо більш коректно підготовлювати ознаки для моделі машинного навчання.

Отже, кількісні - це ознаки, які складаються із чисел і їх можна об’єктивно вимірювати. При цьому є неперервні кількісні ознаки, які можуть приймати будь-які числа як цілі так і дробові. Є дискретні кількісні ознаки - вони можуть використовувати тільки цілі числа. Далі подивилися на текст - це ознаки, які містять в собі абзаци тексту. Також побачили ознаки, які містять дати. І на останок залишили якісний типи даних - ознаки, які приймають категорії. Дані значення ми можемо виміряти тільки суб’єктивно і вони у свою чергу розділяються на порядкові (категорії між якими можна встановити деяку ієрархію); номінативні (категорії без ієрархії, без впорядкованості). А є ще приватний випадок номінативних ознак - це бінарний тип - дані категорії приймають тільки два значення.

3 ДОСЛІДЖЕННЯ МЕТОДІВ ТА ПРИНЦИПІВ UNSUPERVISED LEARNING

3.1 Порівняльна характеристика класичних методів навчання

У першому та другому розділах магістерської кваліфікаційної роботи були розглянуті основні аспекти машинного навчання, приклади задач класифікації та регресії, особливості навчання з підкріпленням. Визначено, що собою представляє проєкт машинного навчання, визначені етапи проєкту, проаналізовані типи даних для задач машинного навчання. Отже, класичне навчання - це сукупність традиційних методів машинного навчання, які базуються на математичних моделях і статистичних принципах для виявлення закономірностей у даних. У класичному підході основну увагу приділяють структурі ознак (feature engineering), підбору алгоритму та налаштуванню параметрів моделі (рис. 3.1). Класичне навчання зазвичай вимагає менше обчислювальних ресурсів і менше даних, ніж глибоке навчання, але забезпечує високу інтерпретованість результатів. Воно залишається основою для багатьох практичних рішень і використовується тоді, коли важлива швидкість, прозорість та пояснюваність моделі [1-3].

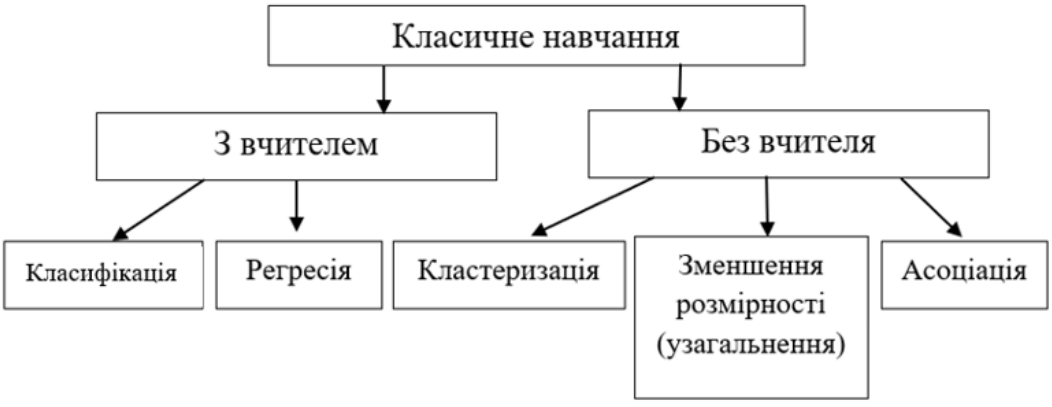


Рис. 3.1 Класичне навчання

У задачах класифікації необхідно передбачати категорію, у задачах регресії – передбачати значення (рис. 3.2). Дослідивши алгоритми для задач класифікації, слід зазначити, що найпопулярнішими з них станом на сьогоднішній день є:

наївний Байєс (Naive Bayes), дерева рішень (Decision Trees), логістична регресія (Logistic Regression), k-найближчих сусідів (k-NN), метод опорних векторів (SVM), а також сюди відносяться нейронні мережі (Neural Networks) та алгоритм Random Forest. Найрозповсюдженішими алгоритмами для задач регресії є: лінійна регресія (Linear Regression), поліноміальна регресія (Polynomial Regression), Ridge/Lasso Regression, метод найближчих сусідів (k-NN), градієнтний бустинг (XG Boost, Light GBM).

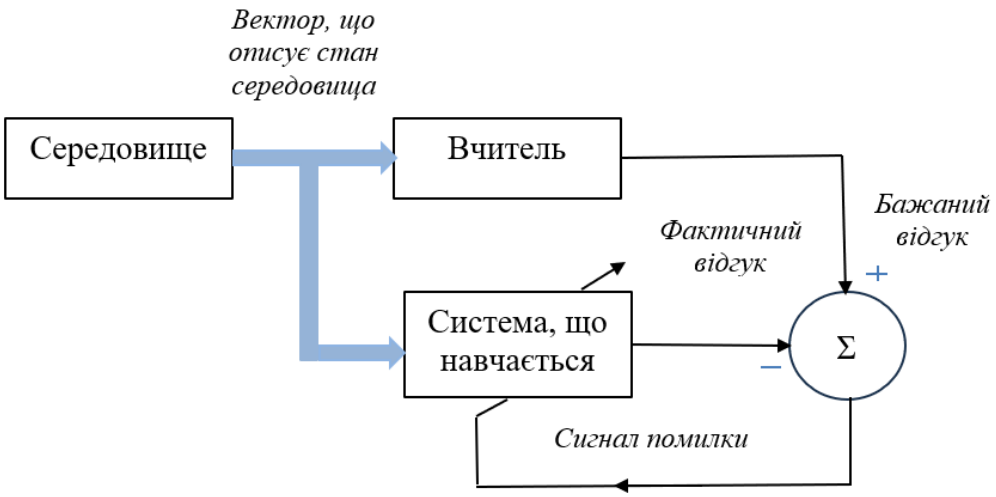


Рис. 3.2 Supervised Learning

Навчання без вчителя (Unsupervised Learning) - це тип машинного навчання, при якому модель працює з нерозміченими даними, тобто без відомих правильних відповідей або цільових змінних. Мета цього підходу - виявити приховані закономірності, структуру або взаємозв'язки у даних без зовнішнього контролю. Модель самостійно групує або структурує дані, шукаючи схожість між об'єктами. Навчання без вчителя дозволяє моделі самостійно досліджувати дані, знаходити внутрішні залежності та структуру, що є основою для подальшого аналізу або прийняття рішень (рис. 3.3).

У навчанні без вчителя модель працює з набором невпорядкованих і нерозмічених даних, що означає відсутність цільових значень або правильних відповідей для прикладів. Машина повинна самостійно виявляти структуру, закономірності та взаємозв'язки між об'єктами, аналізуючи схожість та

відмінності у даних. Наприклад, алгоритми кластеризації можуть групувати клієнтів за схожою поведінкою, а методи зменшення розмірності дозволяють стискати або візуалізувати великі набори даних.

На практиці алгоритми навчання без вчителя використовуються рідше, ніж підходи із вчителем, і здебільшого служать інструментом для дослідження та підготовки даних, а не для прямого виконання конкретного завдання. Проте у випадках, коли розмітка даних неможлива, надто дорога або зайняла б багато часу, методи без вчителя стають критично важливими. Вони дозволяють не лише виявляти приховані закономірності, але й формувати гіпотези для подальшого аналізу, а також підвищувати ефективність наступних етапів ML-проєкту, зокрема підготовки ознак і навчання моделей із вчителем [4].

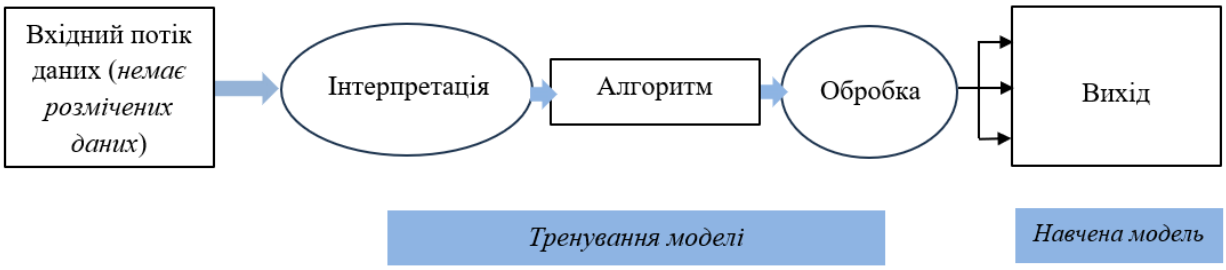


Рис. 3.3 Unsupervised Learning

3.2 Особливості навчання Unsupervised Learning

У рамках магістерської кваліфікаційної роботи визначені основні особливості навчання без вчителя (Unsupervised Learning):

1) Відсутність цільових значень - модель працює з даними, де немає заздалегідь відомих правильних відповідей або міток. Вона сама повинна знаходити структуру та закономірності в даних.

2) Самостійне виявлення закономірностей - алгоритми самостійно групують схожі об’єкти, виділяють аномалії або зменшують розмірність даних без зовнішнього контролю.

3) Гнучкість застосування - підходи Unsupervised Learning можна використовувати для кластеризації, виявлення аномалій, асоціацій та підготовки даних для подальшого навчання моделей із вчителем.

4) Відсутність прямого оцінювання точності - оскільки немає правильних відповідей, оцінити «правильність» результатів складніше; зазвичай застосовуються внутрішні метрики (наприклад, силует кластера, відстань між групами) або візуалізацію результатів.

5) Висока роль попередньої обробки даних - ефективність алгоритмів сильно залежить від якості, масштабу та підготовки даних, оскільки модель повинна сама виділити структуру.

6) Потенціал для відкриття нових закономірностей - моделі без вчителя дозволяють знаходити приховані взаємозв'язки та закономірності, які людина могла не помітити.

На основі підходу Unsupervised Learning вирішуються задачі, пов'язані з виявленням прихованої структури в даних та їх аналізом без наявності цільових значень. До основних типів задач Unsupervised Learning відносяться:

- *задачі кластеризації* (Clustering) - об'єднання об'єктів у групи (кластери) за схожими ознаками. Варто виділити наступні приклади задач кластеризації: сегментація клієнтів за поведінкою у покупках; групування товарів за характеристиками; об'єднання близьких точок на карті; аналіз соціальних мереж, де відбувається виявлення груп користувачів із схожими інтересами або зв'язками; біоінформатика: кластеризація генів або білків за схожістю у структурах або функціях; обробка зображень (розбиття зображення на області за схожими характеристиками), тобто сегментація зображення;

- *зменшення розмірності* (Dimensionality Reduction) - спрощення структури даних при збереженні основної інформації. Наприклад, стискання зображень, візуалізація великих наборів даних у 2D або 3D. Іншими прикладами задач зменшення розмірності можуть бути: візуалізація даних (перетворення багатовимірних даних) у дво- або тривимірний простір для більш зручної візуалізації та аналізу; стиснення даних (зменшення розмірності даних без значної

втрати інформації); підготовка даних перед моделюванням: видалення зайвих або корельованих ознак для зменшення обчислювальної складності та підвищення ефективності моделі; визначення тематики та пошуку схожих документів;

- *вивчення асоціацій* (Association Analysis) - пошук закономірних залежностей між ознаками. Основними типами задач цього напрямку є: прогноз акцій і розпродажів; аналіз товарів, які купують разом (маркетинг); розташування товарів на полицях; рекомендаційні системи: виявлення асоціацій між продуктами або послугами для побудови рекомендацій; оптимізація розташування товарів у супермаркеті: визначення товарів, які слід розміщувати поруч для збільшення продажів.

У табл. 3.1 приведені досліджені особливості машинного навчання без вчителя (Unsupervised Learning).

Таблиця 3.1

Особливості Unsupervised Learning

№ п/п	*Елементи та етапи навчання*	*Особливість*
1	Збір і підготовка даних	Модель працює з нерозміченими даними, тому важлива якість та чистота даних.
2	Попередній аналіз даних (EDA)	Дозволяє виявити структуру, закономірності та аномалії у даних перед навчанням.
3	Вибір алгоритму без вчителя	Залежить від типу задачі: кластеризація, зменшення розмірності, виявлення аномалій.
4	Навчання моделі	Модель самостійно шукає закономірності, групує об’єкти або виділяє ознаки.
5	Оцінка результатів	Виконується за допомогою внутрішніх метрик або візуалізації, бо відсутні цільові значення.
6	Інтерпретація та використання результатів	Результати застосовуються для аналізу, рекомендацій або підготовки даних для подальшого навчання із вчителем.

Таким чином, навчання без вчителя дозволяє досліджувати дані, виявляти структури та взаємозв’язки, підвищуючи ефективність подальшого аналізу або прогнозування.

3.3 Вибір та обгрунтування програмного середовища для розробки ML-моделі

Google Colab - безкоштовне інтерактивне хмарне середовище розробки від Google. Завдяки блокнотам Colab можна використовувати в одному документі код виконання, форматований текст, зображення, розмітку HTML, набір LaTeX. Блокноти Colab зберігаються на Google Disk. Можна відкривати до них доступ колегам, друзям, дозволяючи переглядати або редагувати документ, а також залишати коментарі.

Фізично середовище знаходиться на серверах Google. Починаючи роботу з Google Colab, ми отримуємо доступ до віддаленої машини, де розгортається віртуальна машина з піднятим на ній Jupyter Notebook, а це і є інтерактивне середовище розробки (рис. 3.4).

Рис. 3.4 Середовище розробки ML-моделей

Тут є комірки з кодом, які можна виконувати і є комірки з текстом, що дуже зручно для фіксації ідей та результатів експериментів. Для того, щоб розпочати роботу з Google Colab, можна у будь-якій пошуковій системі набрати Google Colab і перейти буквально за першим посиланням. Щоб створити ноутбук, потрібно в меню вибрати «Файл» і потім створити блокнот [5].

Colaboratory або просто Colab дозволяє писати та виконувати код Python у браузері. При цьому не потрібно ніякого налаштування; отримуємо безкоштовний доступ до графічних процесорів; надавати доступ до документів іншим людям дуже просто. Це відмінне рішення для студентів, спеціалістів з обробки даних та дослідників у галузі штучного інтелекту. Щоб дізнатися розміщення створеного блокноту, достатньо у вкладці «Файл» перейти на «Показати місцеположення на диску».

Створений блокнот буде розміщуватися у папці із всіма Google Colab ноутбуками. Його можна переміщати куди завгодно в межах нашого Google Disk. Ноутбуком у Google Colab дуже легко ділитися з іншими людьми. Достатньо натиснути на кнопку «Поділитися» і вибрати, з ким ми хочемо поділитися ноутбуком. Можна зробити доступ за посиланням для всіх, якщо натиснути на «Дозволити доступ всім, у кого є посилання». Тоді з'явиться посилання на наш ноутбук і права доступу. Можна зробити людей не читачами, а коментаторами або редакторами. Але ніколи не потрібно надавати права доступу на редагування неперевіреним людям. Якщо нам надають доступ як читачу, то редагувати наданий ноутбук не вийде. Тому можна створити свою копію на Google Disk. Робиться це через «Файл» → «Створити копію на Диску».

Щоб запустити код в інтерактивному середовищі, можна натиснути на кнопку «Підключитися» або ж запустити будь-яку комірку. Щоб виконати комірку, потрібно натиснути на кнопку *Play* поряд з коміркою, або натиснути поєднання клавіш, яке приведене у таблиці 3.2.

Таблиця 3.2

Основні поєднання клавіш в середовищі Google Colab

Поєднання клавіш	Функція
Ctrl + Enter	виконується комірка і виділення переходить на комірку нижче
Shift + Enter	виконується комірка і виділення залишається в цій комірці
Alt + Enter	виконується комірка і створюється нова під нею, на неї і переходить виділення

Щоб створити нову комірку з кодом, можна натиснути на спливаючу кнопку + код. Або ж можна натиснути на клавішу *A* для створення комірки над поточною. Головне - знаходитися у потрібному режимі: *режимі введення команд*, коли можна спокійно переміщатися по коміркам, а не в *режимі редагування*, коли можна переміщатися по вмісту однієї комірки.

A – створення нової комірки; B – створення нової комірки знизу. Якщо ми хочемо із кодової комірки зробити текстову - можна натиснути на поєднання

клавіш *ctrl+m+m*, тоді кодова комірка стане текстовою. Ctrl + m +m – переведення комірки у текст. *Ctrl + m + y* - переведення комірки в код й тоді текстова комірка стане кодовою. Щоб ще створити нову комірку з текстом, потрібно натиснути на спливаючу кнопку + текст. Для видалення комірок можна використовувати команду: *Ctrl + m +d* - видалення комірки.

Google Colab - віддалена віртуальна машина, з якою ми фізично не контактуємо. А значить ми не можемо взяти флешку, переслати на неї набір даних і вставити флешку у комп'ютер. Ми не знаємо, де знаходиться комп'ютер. Ця проблема вирішується трьома варіантами. Перший - передати локальні файли з папки в ліву область у Google Colab. Другий варіант – це скористатися модулем *Files* із бібліотеки Google Colab, потрібно написати: *from google colab import Files* і викликати метод *upload*. Виведеться вікно, де потрібно буде вибрати необхідний файл. І після завантаження даних файл буде у лівій області робочого середовища. Але ці файли з лівої області зникнуть, як тільки ми відключимося від виділеної для нас віртуальної машини. Якщо ми виберемо «Управління сеансами» і натиснемо на «Завершити», сеанс роботи з середовищем для нас закінчиться. І наш файл зліва теж зникне і більше ми його ніколи не побачимо. Щоб не втратити наші файли, можемо користуватися третім і самим надійним варіантом - приєднання Google диска до віртуальної машини. Можна в лівій області натиснути на «Підключити диск», потім потрібно буде авторизуватися через Google-пошту. При цьому у лівій області у з'явилася папка *drive* - це і є весь вміст нашого Google диску (рис. 3.5).

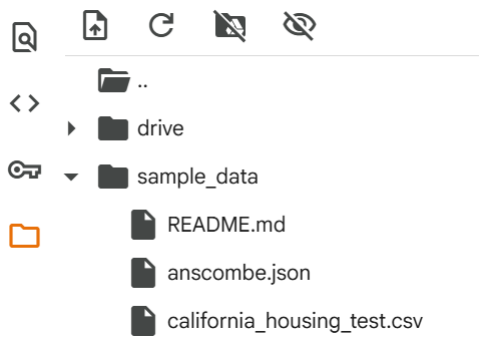


Рис. 3.5 Приєднання Google диска до віртуальної машини

Рекомендовано користуватися саме цим підходом для роботи з файлами, так як він є самим надійним та перевіреним [10].

Для побудови та експериментів із моделями машинного навчання зручно використовувати інтерактивні середовища програмування Google Colab і Jupyter Notebook. Обидва середовища дозволяють виконувати код поетапно, аналізуючи результати після кожного блоку. Це значно спрощує процес відлагодження, тестування гіпотез і візуалізації проміжних результатів. Google Colab і Jupyter Notebook підтримують інтеграцію з багатьма бібліотеками, що дозволяє відображати графіки, діаграми, матриці кореляцій та інші візуальні аналітичні інструменти безпосередньо у робочому середовищі.

Обидві платформи повністю сумісні з найпоширенішими бібліотеками Python, такими як Scikit-learn, TensorFlow, Keras, PyTorch, NumPy, Pandas. Це забезпечує гнучкість і масштабованість при створенні моделей різної складності. На відміну від локального Jupyter Notebook, Google Colab не потребує встановлення програмного забезпечення на комп'ютері. Користувач отримує доступ до безкоштовних хмарних обчислювальних ресурсів (у тому числі GPU і TPU), що дозволяє навчати складні моделі навіть на малопотужних пристроях.

У Google Colab легко ділитися проєктами через посилання, що сприяє командній роботі. У Jupyter Notebook зручно зберігати та відтворювати експерименти, що є важливим для наукових досліджень і звітності.

Можливість комбінувати код, текст (у форматі Markdown), формули та візуалізації в одному документі робить ці середовища ідеальними для оформлення дослідницьких робіт і звітів. Використання Google Colab і Jupyter Notebook забезпечує ефективність, гнучкість і наочність процесу розробки моделей машинного навчання, дозволяє поєднувати код, аналітику та пояснення в єдиному середовищі, а також спрощує спільну роботу над проєктами.

Jupyter Notebook як і Google Colab є одним із кращих інструментів для машинного навчання. Основна мова програмування для ML – це Python. Для програмування на ньому потрібне середовище розробки. Це окрема програма, де ми можемо писати і виконувати код. Для цієї мети і використовується популярну

оболонку Jupyter Notebook. Щоб використовувати Jupyter Notebook, потрібно завантажити і встановити набір бібліотек для машинного навчання - Anaconda, куди і входить Jupyter Notebook.

Вводимо в Google - Anaconda. Перше посилання веде на потрібний сайт *Anaconda.org*. Потрібно перейти на нього і натиснути вгорі у правому куті вікна Download Anaconda (рис. 3.6).

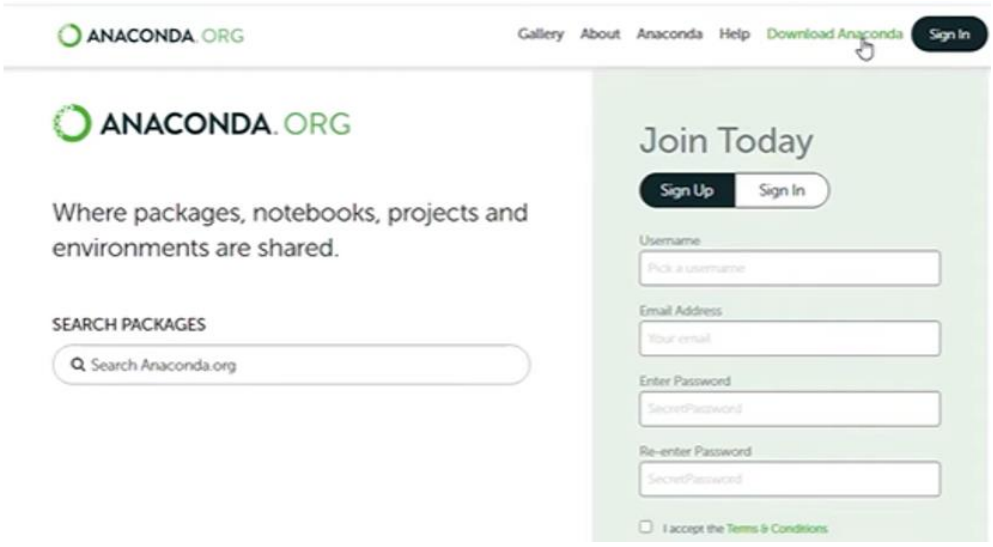


Рис. 3.6 Anaconda – дистрибутив Python

Рис. 3.7 Інсталятори для Windows, MacOS, Linux

В залежності від нашої операційної системи потрібно завантажити необхідний дистрибутив [14].

За замовчуванням ноутбук називається *Untitled* (рис. 3.8). Назву можна змінити. Для цього необхідно клацнути по назві і ввести нову назву, наприклад, *1_Notebook*:

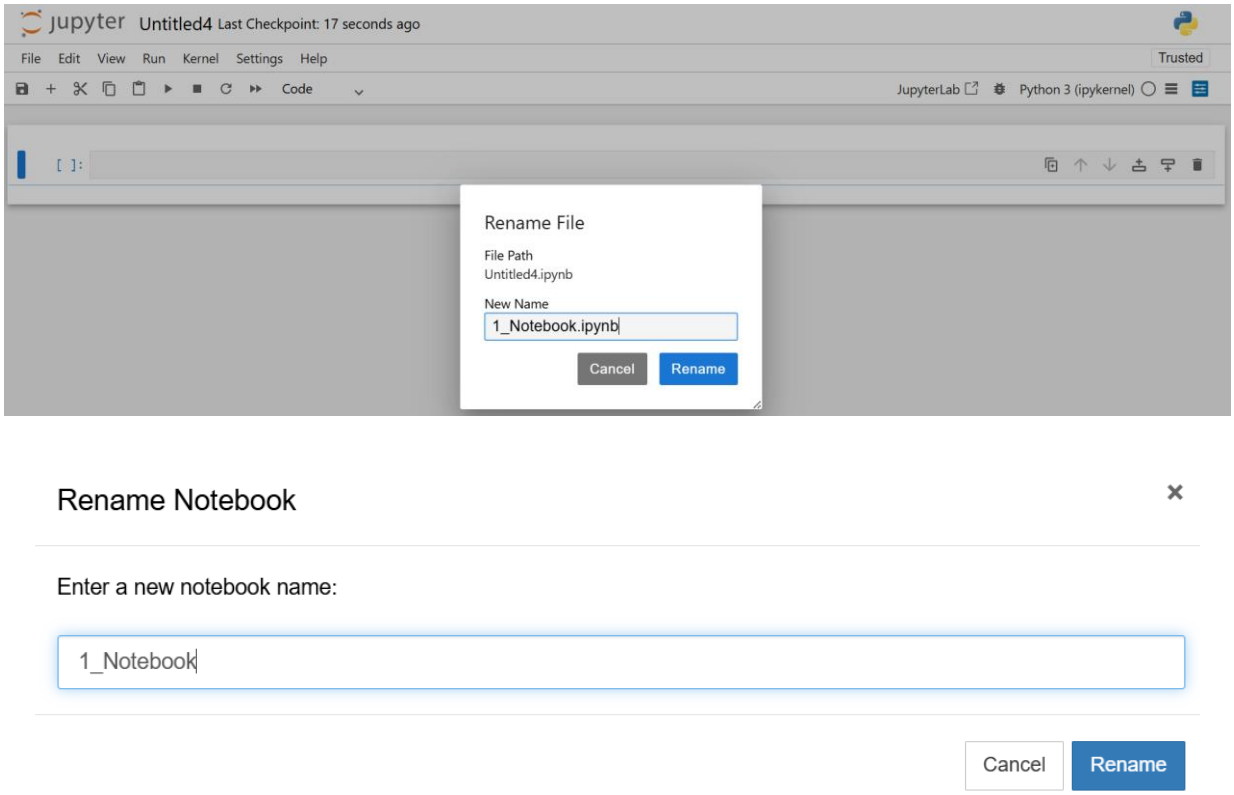


Рис. 3.8 Rename Notebook

Під назвою розміщується *рядок меню* для дій з ноутбуком, комірками та ядром, а також меню допомоги (Help). Нижче розміщується *панель інструментів* з кнопками для найчастіших дій. Якщо затримати курсор на кнопці, то з'явиться функція цієї кнопки. Робочу область ноутбука складають комірки (рис. 3.9). Їх можна скільки завгодно вставляти, видаляти (+, ножиці), копіювати. Є 2 режими роботи ноутбука: перший - це *режим редагування*, коли ми всередині комірки вводимо код. У цей час комірки виділяються зеленим кольором, а у правому верхньому куті з'являється олівець.

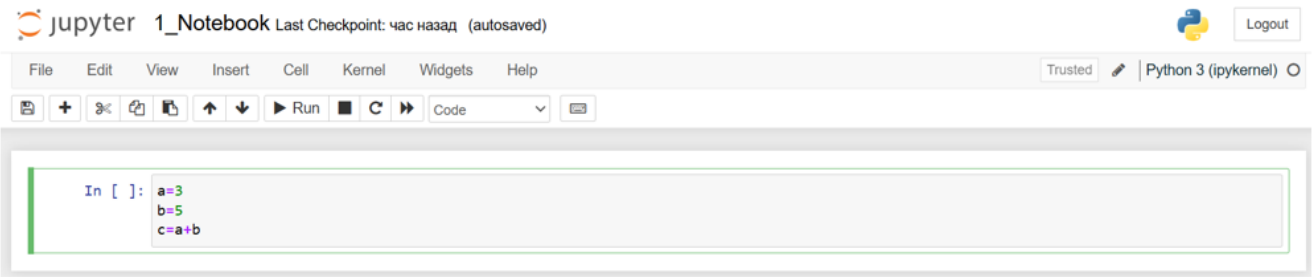


Рис. 3.9 Робоча область ноутбука

Для того, щоб код виконався, потрібно запустити комірку (рис. 3.10). Робиться це або клавішею «Запуск» (Run), або комбінацією клавіш *Shift+Enter*. Коли комірка виконалася, їй присвоюється порядковий номер і виділення переходить на наступну комірку:

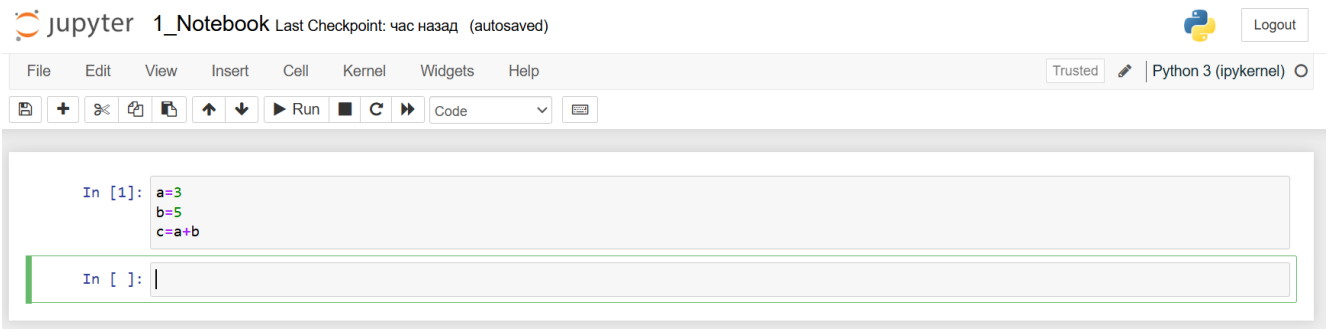


Рис. 3.10 Запуск комірки в Jupyter Notebook

Також у ноутбуці можна дивитися виведення, якщо вони є. Виконаємо другу комірку і під нею у нас з'являться за допомогою функції *print* результати операції додавання (рис. 3.11).

Рис. 3.11 Результат виконання операції в середовищі

Частина, де з'являється виведення, називається Output. Другий режим роботи ноутбуку - це *командний режим*. Щоб у нього перейти з режиму редагування, потрібно натиснути *Esc*. Тоді у нас комірка виділяється синім кольором і у правому верхньому куті зникає значок олівця (рис. 3.12).

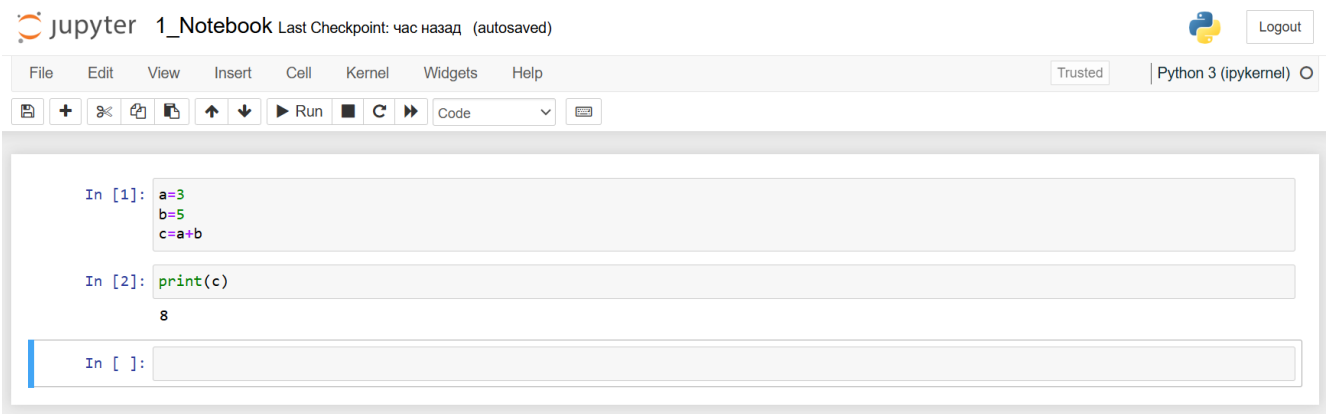


Рис. 3.12 Командний режим

У цьому режимі можна вставляти нові комірки за допомогою + на панелі інструментів або комбінації клавіш *Ctrl+B*, якщо ми хочемо вставити комірки нижче даної, або *Ctrl+A*, якщо ми хочемо вставити комірки вище. Для того, щоб видалити комірки, потрібно натиснути комбінацію клавіш *Ctrl+DD*. Для того, щоб скопіювати комірку, потрібно її вибрати, натиснути *Ctrl+C*, перейти на потрібну позицію і натиснути *Ctrl+V*.

Також у командному режимі можна виділяти декілька комірок зразу, затиснувши при цьому Shift і натискаючи на стрілочки вгору або вниз: Shift+↑↓ (рис. 3.13).

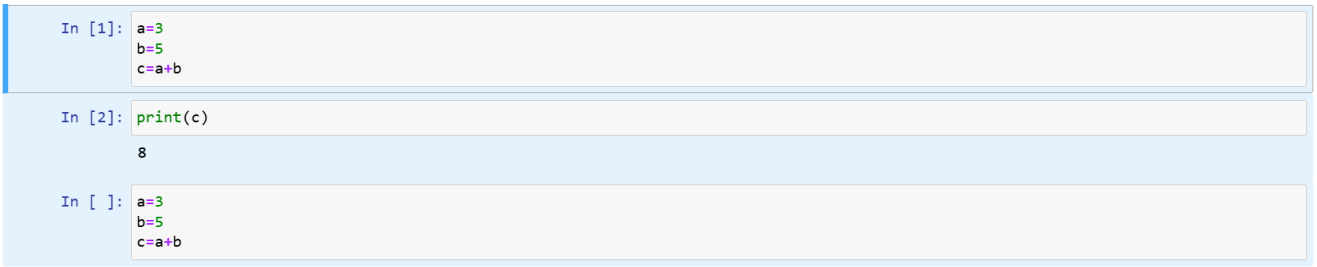


Рис. 3.13 Виділення комірок

Також можна переміщати комірки, натискаючи на панелі інструментів: ↑↓.

Ноутбук періодично сам зберігається, вгорі середовища можна побачити, коли він сам зберігався. І завжди можна примусово зберегти ноутбук, натиснувши кнопку «*Зберегти на панелі інструментів*» або комбінацію клавіш *Ctrl+S*.

Можна писати коментарі до коду всередині комірок за допомогою знаку # (рис. 3.14). Коментар виділяється зеленим кольором та курсивом:

```
In [4]: # коментар до коду
        a*=10
        a
Out[4]: 30
```

Рис. 3.14 Коментарі до коду

У ноутбуці можна виводити різні графіки. Для виведення графіків потрібно імпортувати бібліотеку Matplotlib, задати координати і намалювати графік (рис. 3.15).

```
In [5]: import matplotlib.pyplot as plt

In [6]: x=[-2,-1,0,1,2]
        y=[4,1,0,1,4]

In [7]: plt.plot(x,y)
Out[7]: [<matplotlib.lines.Line2D at 0x19fe28f8460>]
```

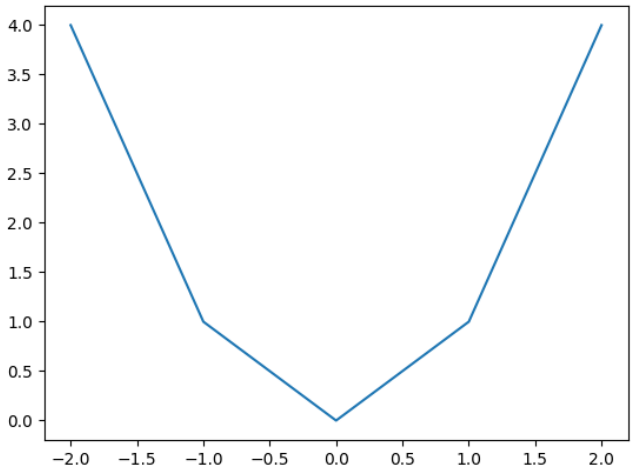


Рис. 3.15 Графік з використанням бібліотеки Matplotlib

З’явився графік, все виходить дуже інтерактивним та наглядним. В одному місці у нас код і графіки, і текст. Графіки можна тонко налаштовувати. Обирати розмір, колір, тип ліній, робити надписи і за один раз виводити декілька графіків. Це тема візуалізації в Jupyter Notebook.

Ще у ноутбуці можна вводити консольні команди. Особливо це зручно в операційній системі Linux. Щоб ввести команду, потрібно надрукувати знак оклику, а потім набирати команду (рис. 3.16).

```
In [9]: ! chcp 65001
        Active code page: 65001

In [10]: ! ping www.google.com

         Pinging www.google.com [142.250.203.196] with 32 bytes of data:
         Reply from 142.250.203.196: bytes=32 time=19ms TTL=114
         Reply from 142.250.203.196: bytes=32 time=19ms TTL=114
         Reply from 142.250.203.196: bytes=32 time=20ms TTL=114
         Reply from 142.250.203.196: bytes=32 time=17ms TTL=114

         Ping statistics for 142.250.203.196:
             Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
         Approximate round trip times in milli-seconds:
             Minimum = 17ms, Maximum = 20ms, Average = 18ms
```

Рис. 3.16 Консольні команди

У комірок ноутбука є 2 основних типи: *Code* і *Markdown*. Змінити тип можна у панелі інструментів (рис. 3.17).

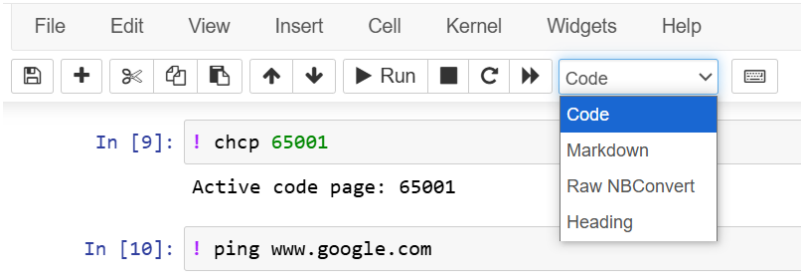


Рис. 3.17 Зміна типу комірок

Або клавішами *Ctrl+M* перейти в режим *Markdown* і *Ctrl+Y* - повернутися у режим коду.

У режимі Markdown можна писати текст, заголовки, формули. Markdown - мова розмітки для форматування тексту. У цьому режимі можна писати заголовки різних рівнів. Можна створювати списки, приводити частини коду, таблиці. Також можна використовувати html-теги, за допомогою них можна виводити заголовки. У ноутбуці ми пишемо код, де код з'являється, а сам код виконується в ядрі, яке називається *Kernel*. При закритті ноутбуку Kernel не закривається. Щоб повністю закрити ноутбук, потрібно закрити головне вікно і закрити консоль.

Використання Jupyter Notebook під час розробки моделей машинного навчання є доцільним завдяки його інтерактивності, зручності та гнучкості. Це середовище дозволяє виконувати програмний код поетапно, що спрощує процес тестування, налагодження та аналізу результатів роботи моделі.

Однією з головних переваг Jupyter Notebook є можливість поєднання коду, тексту, формул, графіків та візуалізацій у межах одного документа. Такий підхід сприяє кращому розумінню логіки побудови моделі, полегшує документування досліджень і підготовку звітів. Jupyter Notebook підтримує інтеграцію з основними бібліотеками Python, що забезпечує можливість реалізації повного циклу машинного навчання - від завантаження даних і їх обробки до побудови, навчання й оцінювання моделей [21-23].

Середовище Jupyter Notebook дозволяє швидко візуалізувати дані та результати кластеризації або прогнозування, що підвищує наочність і зручність аналітичної роботи.

Jupyter Notebook є ефективним інструментом для створення, тестування та аналізу моделей машинного навчання завдяки своїй інтерактивності, підтримці популярних бібліотек Python, можливості документування процесу та візуалізації результатів у зручному форматі.

4 РОЗРОБКА МОДЕЛІ МАШИННОГО НАВЧАННЯ НА ОСНОВІ UNSUPERVISED LEARNING

4.1 Особливості кластерного аналізу даних

Дослідивши принцип навчання із вчителем, можна узагальнити, що це коли у нас є *X* і ми хочемо отримати *y*. І ці *y* нам вже дані (*X*→*y*). Якщо це регресія, у нас є вхідні дані *X* і ми хочемо отримати якесь значення з якогось діапазону (якесь число) – це задачі регресії. У нас були *y* і наші моделі навчалися по тим даним, які ми вже підготували. Коли ми виконували класифікацію, у нас були вхідні дані *X* і у нас були мітки класів *y* - до якого класу належить конкретно цей зразок. Але так часто буває, що мітки ніхто не збирав. Тобто у нас є дані по *X*-сам, а як взяти *y*-ки взагалі не зрозуміло. Тобто ми хочемо зробити класифікацію. Ми припускаємо, що наші об’єкти у нашій базі мають якийсь кластер. Ми говорили, що у нас є дані, ми намалювали один клас та інші клас (рис. 4.1). Але суть в чому. Тут і без кольорів зрозуміло, що ось ці три області щось об’єднує.

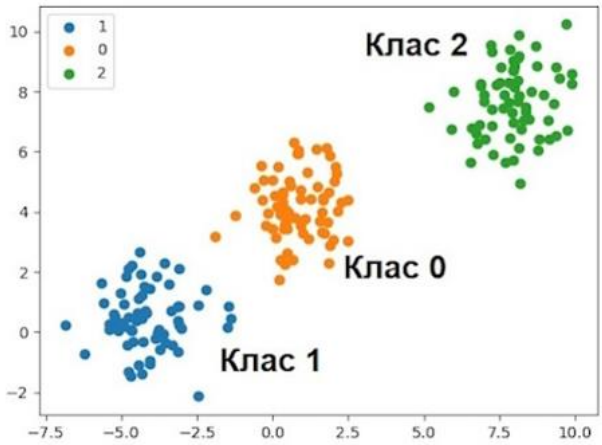


Рис. 4.1 Задача класифікації

У даному випадку нам не треба *y*-ків (хоча вони спрощують нам життя). У даному прикладі нам не треба знати *y*-ки цих точок, щоб сказати, що новий зразок, який з’явиться, належить до об’єктів якогось із класів.

Кластерами називається сукупність схожих об’єктів. *Кластеризація* - це метод, який допомагає нам наші дані, у яких не має *y*-ків (ми не знаємо який це

клас) розбити на певні кластери, тобто сказати, до якої із сукупності елементів відноситься кожен з них. Вони просто нумеруються: нульовий, перший, другий.

Досліджуючи кластеризацію, слід зазначити, що вона, по-перше, використовується для сегментації. У нас не має *y*-ків, але ми дані наші візуалізуємо і бачимо, що у нас є сукупність елементів, що вони скупчені у певних областях, а не нормально або рівномірно розподілені. Тому ми можемо просегментувати це все, намагаючись пояснити на словах, що це за кластери і що їх об'єднує. Це може використовуватися у методиках зменшення розмірності для знаходження аномалій. Якщо взяти рис. 4.1, аномалію можна позначити точкою між двома сусідніми кластерами.

Ми знаємо, що у нас всі кластери десь зосереджені. У нас з'являється тут точка, вона – аномалія. Ми можемо її відкинути. Кластеризація не підлягає ніяким розподілам і допомагає нам знайти наші аномальні спостереження. Це у нас може бути для часткового навчання. Наприклад, є дані, частина яких вже розмічена, але й є дані, де не має *y*-ків.

Можна натренувати модель, яка буде предиктити і вже допредиктити те, що нам потрібно. Це один із підходів. Другий підхід - зробити кластеризацію, а потім у зразках, які належать до кластера, вибрати їхній результат як більшість розмічених елементів, які попали у цей кластер.

Кластеризацію можна робити для сегментації зображень. Тобто у нас на картинці щось намальовано, ми можемо картинку розглядати як масив пікселів. У пікселів є координата по *x*-су, координата по *y* і RGB-значення. Якщо на зображенні буде зелений фон і біла квітка, то зображення, де у нас біла квітка, у нього буде певна відстань від зеленої трави, тому що колір - це так само число, ці числа будуть відрізнятися, тому що в зеленій траві - значення R та B близькі до 0, тому вона й зелена: x, y, RGB, де R=B=0. У білій квітці будуть значення і в R, і в G, і в B, тому що білий колір - це сукупність всіх трьох кольорів. Відповідно у нас є значення, ми можемо зробити кластеризацію, кожен піксель віднести до певного кластеру і зможемо зрозуміти, які саме пікселі у відповідають за квітку, а які ні.

Приблизно так працює те, що ми на телефоні виділяємо об’єкт і для цього використовувалася кластеризація, зараз це опрацьовують нейронні мережі.

4.2 Алгоритм кластеризації k-means

Для дослідження алгоритму k-means визначимо приклад із даними (рис. 4.2).

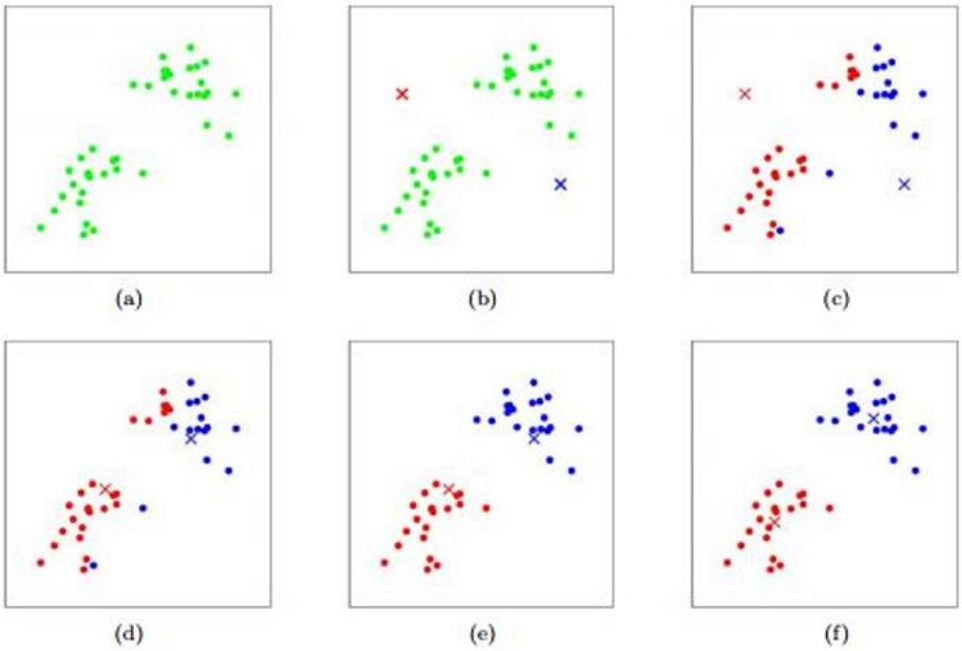


Рис. 4.2 Кластеризація методом k-середніх (k-means clustering)

Маємо двовимірний простір, але в реальності у нас може бути вимірів більше. Маємо 5 кластерів. Яким чином працює цей алгоритм? Єдине, що нам потрібно знайти – це центроїди. *Центроїд* - це точка, яку можна розуміти як центр кластера. У навченій моделі у нас знайдені координати центроїдів. Центроїди відповідають за кожен кластер і потім буде будуватися діаграма Воронова по цим точкам. Що таке діаграма Воронова? Беремо 2 кластери, які ми зобразили. У нас між ними проведеться лінія. І буде існувати точка, де вони перетнуться. Коли нова точка попадає в певну область, вона віднесеться до того кластеру, який центроїд до неї найближчий. Якщо ми візьмемо будь-яку точку, то центроїд, якому належить область, буде найближчим до цієї точки.

Дослідимо, як визначити центроїди, їх координати. Перше, що у нас відбувається - це випадкова ініціалізація центроїдів. Тобто для цього алгоритму

ми маємо вказати, на яку кількість кластерів ми розраховуємо. Є методи, які дозволяють визначити потрібну кількість кластерів. Якщо дані у нас мають тільки 2 вхідні ознаки (*X, y*), по яким ми хочемо робити кластеризацію, то ми можемо просто зробити візуалізацію, подивитися скільки кластерів і вказати.

На наступному кроці будується діаграма Воронова і кожному об'єкту присвоюється певний кластер. Тобто, до кожного *X*, до кожної чорної точки ми знаходимо найближчий випадковий ініціалізований центроїд і відносимо його до кластеру цього центроїду. Тобто ми кожному X_i присвоюємо кластер. Кожний центроїд переноситься в таку точку, щоб мінімізувати середню відстань до кожного з *X*.

До кожної точки від центроїда ми можемо порахувати відстань. Взагалі у нас є координати цих точок, які належать до кожного з цих кластерів. Ми можемо порахувати середнє значення координат цих точок і перемістити туди наші кластери. Після того, як наші центроїди переїхали, ми можемо перемалювати діаграму Воронова і зробити точно теж саме. І так за декілька кроків скоріше за все (не 100% гарантії, а скоріше за все) центроїди зійдуться в оптимальне рішення.

Іноді таке буває, що цей алгоритм не сходиться правильно, тобто знаходить субоптимальне рішення там, де кластери поділилися погано. Рідко, але таке буває. Тобто цей алгоритм не дає правильного 100%-го сходження у максимально оптимальні значення. Але є метрика, яка означає, наприклад, суму квадратів всіх точок до центроїду. Якщо ми це зробимо це в Python, тобто зробимо його імпорт із Sklearn, навчимо, то він буде робити випадкову ініціалізацію 10 разів (вона там теж не випадкова) і вона вибере той варіант, де рішення було найбільш оптимальним. Це число, що 10 разів робиться ініціалізація, 10 разів будується алгоритм, його теж можна налаштувати, зробити більшим або змінити random_state, якщо використовуємо у Python, а не пишемо самі, то з дуже великою ймовірністю кластери зійдуться так як потрібно. Центроїд переміщається у центр об'єктів його кластеру. І ці кроки виконуються до тих пір, поки кластери не стабілізуються. Тобто на якомусь етапі вони просто перестануть рухатися.

Недоліком алгоритму є те, що центроїди на першому етапі рандомно ініціалізуються. Якщо вони всі будуть десь поряд або не в правильній послідовності, то алгоритм може не оптимально зійтися або довго працювати. Для того, щоб уникнути цієї проблеми, по-перше, ми можемо самі ініціалізувати центроїди, тобто виконати ініціалізацію вручну. Можна виконати візуалізацію, подивившись приблизно, які мають бути значення по нашим *X*-сам, викликавши метод *init,* ініціалізувати все і потім запустили навчання. Тобто метод 1 – це метод ручної ініціалізації.

Другий метод – це метод k-means++. Алгоритм даного методу наступний. Перший центроїд $C^{(1)}$ вибирається випадково із набору даних. Наступний центроїд $C^{(i)}$ вибирається із ймовірністю:

$$\frac{D(x^{(i)})^2}{\sum_{j=1}^{m} D(x^{(j)})^2}\ , \tag{4.1}$$

де $D(x^{(i)})^2$ – відстань між x^i та найближчим центроїдом. Тобто ми вибираємо перший кластер рандомно із тих точок, із тих зразків *X*, які у нас є. Далі для кожного зразка буде рахуватися число, яке буде означати його квадрат відстані до центроїда поділити, тобто усереднити це все. Для кожного зразка рахуємо його відстань у квадраті центроїда, потім ділимо на суму всіх для всіх точок. І наступний кластер вибереться із ймовірністю, яка буде порахована як відношення відстаней. Ми скоріше за все виберемо наступний кластер той, який буде далі від всього. Тобто ми вибрали один кластер випадково, а далі чим далі точка *X* від цього кластеру, тим більша ймовірність, що ця точка вибереться наступним кластером. І так ми будемо робити до тих пір, поки не ініціалізуємо всі кластери, а далі алгоритм піде по старій схемі вчитися за цими пунктами.

Наступним питанням є кількість кластерів. Дуже добре мати приклад, коли є 2 виміри і візуально видно, що тут 5 кластерів. Але часто буває, що *X=5, X=6* і ми можемо знайти найбільш впливові *X*. Якщо ми будемо, наприклад, використовувати Random Forest, у Random Forest є дерева і коли ми вибирали якесь значення і порогове значення, то ми старалися зменшити ентропію або

забрудненість Джині. І той з *X*, які ми вибрали і він сильно збільшує ентропію - його можна вважати більш вагомим. Можемо написати у полі *model* і подивитися, на скільки наші зразки значимі. Ми можемо вибрати 2 або 3 впливових зразки, модель подивиться, що саме найбільш впливає. Можна зробити візуалізацію і подивитися. По-перше, для підстраховки, по-друге, для автоматизації певних процесів нам потрібно без участі людини навчитися це робити.

Є спеціальна метрика, яка називається інерція. *Інерцією* називається середня відстань від кожного зразка (кожної нашої точки) до центроїда. Якщо ми візьмемо 5 кластерів і порахуємо ці відстані, вони будуть достатньо малими. Якщо ми візьмемо менше кластерів, тоді відстані до найбільшого центроїду будуть дуже великими, вони підносяться у квадрат і наша інерція сильно зростає. Тому, очевидно, що потрібно взяти таку модель, де інерція буде найменшою. Якщо добавити ще один кластер, зміниться діаграма і з'явиться ще 1 лінія, яка розділяє кластери. Інерція не буде зростати. Чим більше у нас центроїдів, чим більше ми вибрали кластерів, тим менша у нас інерція.

4.3 Процес кластеризації

Кластеризація - розбиття певної множини об'єктів на певні групи. Ці групи називаються кластерами. Коли ми займаємося кластеризацією, ми можемо робити багато кластерів, можемо задати довільну кількість кластерів - стільки, скільки нам потрібно. Якщо у нас є певний набір даних для класифікації і в цьому наборі даних ми точно знаємо кількість класів, то без помилки в обробці даних ми зробити не зможемо дану задачу. На відміну від класифікації, кластеризація не передбачає чітко заданих параметрів. Здебільшого той процес, який розбиває певні елементи на кластери, відбувається під капотом. Ми просто кажемо, що виділи нам у цій множині 5, 6, 10, 20 - скільки нам потрібно підмножин. А як уже у нас цей процес відбувається – залежить від математики. При якісній кластеризації у кожному кластері мають опинитися «схожі» об'єкти, а об'єкти різних кластерів мають бути якомога більш відмінними (рис. 4.3). Головна

різниця між кластеризацією та класифікацією полягає в тому, що перелік параметрів груп чітко не заданий і визначається в процесі роботи алгоритму.

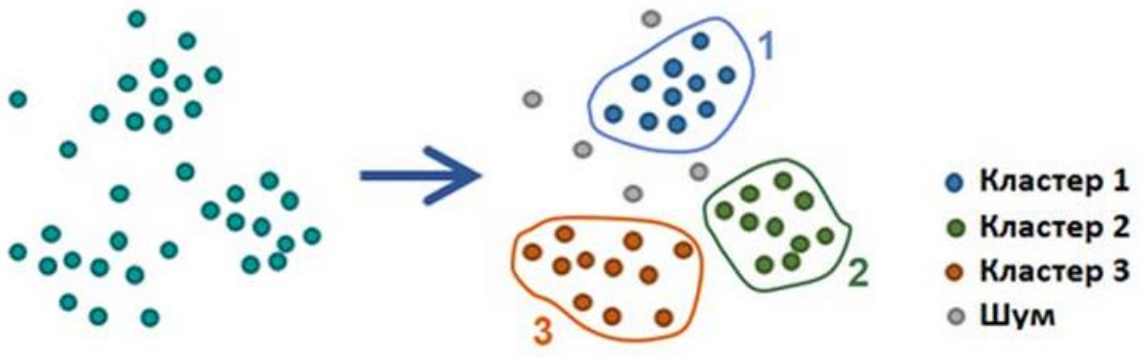


Рис. 4.3 Кластерний аналіз даних

Кластеризація є алгоритмом проміжної обробки даних, коли потрібно певні дані розбити на певні групи. У цьому конкретному випадку ми використовуємо той чи інший алгоритм, який нам дозволяє ці групи отримати. Але який би ми алгоритм не використовували, яку б методику ми не застосовували, весь процес кластеризації у нас зводиться до певних етапів. Спочатку нам потрібно означити нашу вибірку для кластеризації.

Після цього нам потрібно визначити певну множину змінних, за якими у нас буде оцінюватися набір наших об’єктів. Зазвичай цей набір змінних вибирається алгоритмами автоматично, але іноді нам в це доводиться втручатися. Після цього дуже важливим етапом виділення кластерів є обчислення значень міри подібності одного об’єкту по відношенню до певних груп об’єктів. Коли у нас обчислена група подібності - це і є той показник, на основі якого якісь елементи певної множини у нас відносяться до того чи іншого кластеру. У нас відбувається процес кластеризації. Далі ми оглядаємо наші дані, дивимося, що у нас вийшло, за потреби ми вносимо певні корективи або перезапускаємо наш алгоритм. Наприклад, ми перезапускаємо алгоритм з метою збільшення або зменшення кластерних ядер. Коли ми детально вирішили все розбити, а нам потрібно більш генералізувати наші кластерні підмножини, тоді ми задаємо просто меншу кількість ядер. Ключовим поняттям при здійсненні кластеризації є міри відстані кластерів. По показникам мір відстаней найчастіше і відбувається віднесення того

чи іншого елемента до певного кластера, визначається його приналежність до певного кластера. По мірам відстаней і розділяються всі існуючі алгоритми.

Кластерний аналіз є одним із основних методів багатовимірної статистики та машинного навчання, що спрямований на виявлення внутрішньої структури даних шляхом об’єднання об’єктів у групи (кластери) за певними критеріями подібності. Процес кластеризації включає кілька послідовних етапів.

1) Формування вибірки об’єктів для кластеризації. На цьому етапі визначається сукупність об’єктів дослідження, що становлять основу для подальшого аналізу. Вибірка має бути репрезентативною, щоб забезпечити достовірність отриманих результатів.

2) Визначення множини змінних (ознак). Обираються змінні, які найкраще характеризують об’єкти та впливають на процес їх групування. За потреби виконується нормалізація або стандартизація значень змінних для усунення впливу різних масштабів вимірювання, що дозволяє зробити внесок кожної ознаки у процес кластеризації порівняним.

3) Обчислення міри подібності або відстані між об’єктами. Для цього використовують певні метрики відстані (наприклад, евклідова, мангеттенська, косинусна), які кількісно відображають ступінь схожості або відмінності між об’єктами у багатовимірному просторі ознак.

4) Застосування методу кластерного аналізу. На цьому етапі обирається конкретний алгоритм кластеризації (ієрархічний, k-середніх, DBSCAN, агломеративний), який на основі розрахованих відстаней формує групи подібних об’єктів. Кожен алгоритм має свої переваги, обмеження та параметри, що впливають на кінцевий результат.

5) Інтерпретація та оцінювання результатів. Отримані кластери аналізуються з точки зору їх змістової узгодженості, однорідності всередині груп і відмінностей між ними. За необхідності виконується коригування вхідних параметрів - наприклад, зміна метрики відстані, кількості кластерів або самого алгоритму.

Кластерний аналіз є ітеративним процесом, у якому вибір міри подібності та методу кластеризації має вирішальне значення для якості групування. Різні

метрики відстані можуть призводити до різних результатів, тому їх підбір здійснюється з урахуванням характеру даних і цілей дослідження.

Проведемо дослідження основних мір відстані, які використовуються у кластеризації. Міра відстані визначається як деякий критерій, на основі якого проходить порівняння декількох об'єктів. Найчастіше при кластеризації застосовуються: Евклідова відстань, квадратична Евклідова відстань, Манхеттенська відстань і відстань Чебишова.

Евклідова відстань - геометрична відстань між об'єктами у багатомірному просторі:

$$\rho(x, x') = \sqrt{\sum_{i}^{n} (x_i - x_i^{'})^2} \tag{4.2}$$

Квадратична Евклідова відстань надає більші ваги більш віддаленим один від одного об'єктам:

$$\rho(x, x') = \sum_{i}^{n} (x_i - x_i^{'})^2 \tag{4.3}$$

Манхеттенська відстань (відстань міських кварталів) – середня різниця по координатах:

$$\rho(x, x') = \sum_{i}^{n} | x_i - x_i^{'} | \tag{4.4}$$

Відстань Чебишова дозволяє визначити як відрізняються об'єкти один від одного тільки по одній координаті:

$$\rho(x, x') = \max(| x_i - x_i^{'} |) \tag{4.5}$$

Як і у всіх інших моделей потрібно оцінювати якість роботи моделей кластеризації. Для цього є спеціальні метрики. Метрики об'єднання кластерів:

1. *Одинарний зв'язок (відстань найближчого сусіда)* полягає у визначенні відстані між двома кластерами як мінімальної відстані між будь-якими двома об'єктами, що належать різним кластерам.

2. *Повний зв'язок (відстань найбільш віддалених сусідів)* визначає відстань між двома кластерами як максимальну відстань між будь-якими двома об'єктами, що належать різним кластерам.

3. *Незважене попарне середнє* визначає відстань між двома кластерами як середнє арифметичне всіх попарних відстаней між об'єктами, що належать цим кластерам.

4. *Зважене попарне середнє* є аналогічним до незваженого, однак під час обчислення враховується розмір кластерів - кількість об'єктів у кожному з них використовується як ваговий коефіцієнт.

5. *Незважений центроїдний метод* визначає відстань між кластерами як відстань між їх центроїдами, тобто центрами вагів об'єктів у кожному кластері.

6. *Зважений центроїдний метод* аналогічний незваженому, але при обчисленні відстані враховується розмір кластерів, який використовується як ваговий коефіцієнт.

Найчастіше в якості метрик використовуються 1,2, тому що вони найпростіші і тому що результати роботи кластеризації настільки очевидні, що іноді ми на око оцінюємо як у нас відпрацювала наша модель. У різних фреймворках існує багато моделей, які проводять той чи інший тип кластеризації. Розглянемо три основні методики.

4.4 Реалізація методу k-means

Цей алгоритм ще називають алгоритмом плаваючих центроїдів. Ми спочатку обираємо бажану кількість кластерів - задаємо їх у параметрах моделі (2, 3, 4, скільки потрібно). За такого підходу побудова цих кластерів до певної міри штучна, тому що ми самі визначаємо, а не за якимись об'єктивними показниками, скільки в цій множині може бути кластерів. Коли починає працювати алгоритм, він випадковим чином призначає центри вибраних кластерів. Потім він починає по відношенню до кожного центроїду шукати найближчого сусіда із сукупності елементів, які знаходяться у нашій множині. У процесі пошуку сусіда центроїди поступово починають зміщуватися, де цих точок найбільше. За таким принципом

за декілька циклів, яких може бути багато (залежить від того, скільки кластерів ми задаємо, яка у нас множина для дослідження), починає відбуватися поступове зміщення даного центру і із зміщенням центру у нас поступово відбувається перегляд належності точок до того чи іншого центру, тобто до ядра майбутнього кластеру. В решті решт центри майбутніх кластерів зміщуються до центрів згущення певних точок. У результаті за методом ближчого сусіда у нас визначається, які точки більше відповідають даним конкретним центрам і на основі цього у нас виділяються певні кластери. Ця методика хоч і штучна, але доволі проста. Ми тут ніяким чином не регулюємо за якими параметрами відбувається вимір. Цей вимір між центром і найближчими сусідами відбувається в деякому багатовимірному Евклідовому просторі, до якого зводиться положення точок на двовимірній діаграмі. Точок (характеристик, предикторів) може бути набагато більше, ніж 2 (рис. 4.2). Але, коли вони зводяться до Евклідового простору, то відповідним чином всі обчислення відбуваються в Евклідовому просторі.

Отже, метод k-середніх - ітеративний алгоритм кластеризації, що грунтується на мінімізації сумарних квадратичних відхилень координат точок кластерів від центроїдів цих кластерів. Етапи алгоритму:

1. Обирається бажана кількість кластерів k.

2. Випадковим чином відбираються k об’єктів вибірки, що призначаються центроїдами класів.

3. Для кожного центроїда шукається найближчий сусід, що додається до кластера.

4. Перераховується положення центроїда. Алгоритм зупиняється, коли координати центроїдів перестають змінюватися.

Для практичної реалізації методу k-середніх необхідно імпортувати датасет, метрики, модель кластеризації (рис. 4.4). У даному випадку для кластеризації методом k-середніх будемо використовувати модель KMeans.

Для кластеризації використаємо відомий датасет Wine (рис. 4.5).

```
[1]: from sklearn import datasets
     from sklearn import metrics
     from sklearn.cluster import KMeans
     import matplotlib.pyplot as plt
```

Рис. 4.4 Імпорт моделі кластеризації KMeans

```
[2]: wine_df=datasets.load_wine()
     y=wine_df.target
     y
```

```
[2]: 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, 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, 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])
```

Рис. 4.5 Датасет

Бачимо поділ на три класи. Їх не будемо класифікувати, а застосуємо до них модель кластеризації. Імпортуємо модель KMeans. Для того, щоб нам вказати, скільки ми хочемо отримати кластерів, потрібно використати головний параметр n_clusters - кількість кластерів і саме головне, якщо ми задамо без цього параметру, то він нам спробує кластеризувати на свій розсуд (рис. 4.6).

```
[3]: model=KMeans(n_clusters=3)
```

Рис. 4.6 Визначення кількості кластерів

Далі необхідно виконати тренування моделі машинного навчання (рис. 4.7).

```
[4]: model.fit(wine_df.data)
```

```
E:\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are les
s chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

```
[4]: KMeans
     KMeans(n_clusters=3)
```

Рис. 4.7 Тренування моделі машинного навчання

Передамо на ML-модель номери класів (0, 1, 2). Отримаємо з моделі номери класів, які вона створила (рис. 4.8).

```
[5]: labels=model.labels_
     y=wine_df.target
```

Рис. 4.8 Підготовка до оцінки якості кластеризації

Для того, щоб порахувати точність кластеризації нашої моделі, потрібно визначити метрики, які ми будемо використовувати для того, щоб оцінити наскільки точно у нас відбулася кластеризація.

```
[6]: print(f"Homogeneity: {metrics.homogeneity_score(y, labels):.3f}")
     print(f"Completeness: {metrics.completeness_score(y, labels):.3f}")
     print(f"V-measure: {metrics.v_measure_score(y, labels):.3f}")
     print(f"Adjusted Rand Index: {metrics.adjusted_rand_score(y, labels):.3f}")
     print(
         "Adjusted Mutual Information:"
         f"{metrics.adjusted_mutual_info_score(y, labels):.3f}"
     )
     print(f"Silhouette Coefficient: {metrics.silhouette_score(wine_df.data, labels):.3f}")

     Homogeneity: 0.429
     Completeness: 0.429
     V-measure: 0.429
     Adjusted Rand Index: 0.371
     Adjusted Mutual Information:0.423
     Silhouette Coefficient: 0.571
```

Рис. 4.9 Попередні метрики ML-моделі

На рис. 4.9 бачимо попередні метрики нашої моделі. Як бачимо, показник тримається в межах 50%. Результати кластеризації (рис. 4.10):

```
[7]: predictions=model.predict(wine_df.data)
     predictions

[7]: array([1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
            1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0,
            0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 0, 2, 0, 2, 2, 0,
            2, 2, 0, 0, 0, 2, 2, 1, 0, 2, 2, 2, 0, 2, 2, 0, 0, 2, 2, 2, 2, 2,
            0, 0, 2, 2, 2, 2, 2, 0, 0, 2, 0, 2, 0, 2, 2, 2, 0, 2, 2, 2, 2, 0,
            2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2,
            2, 0, 0, 0, 0, 2, 2, 2, 0, 0, 2, 2, 0, 0, 2, 0, 0, 2, 2, 2, 2, 0,
            0, 0, 2, 0, 0, 0, 0, 2, 0, 2, 0, 0, 2, 0, 0, 0, 0, 2, 2, 0, 0, 0,
            0, 2], dtype=int32)

[8]: y=wine_df.target
     y

[8]: 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, 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, 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])
```

Рис. 4.10 Результати кластеризації

Якщо вивести значення поряд, то побачимо, що є деякі відмінності. Саме тому метрики всі, крім останньої, не виходять за межі 0.42 по результатам кластеризації. Виведемо результати на діаграму (рис. 4.11). Результати кластеризації значно інформативніше аналізувати графічно, оскільки візуалізація дозволяє краще зрозуміти структуру даних та якість сформованих кластерів. На діаграмі можна побачити, як саме розташовані об’єкти у просторі ознак, наскільки вони щільно згруповані та чи справді утворюють окремі скупчення (кластери).

Візуальне представлення дає змогу швидко визначити, чи добре алгоритм розділив об’єкти, а саме чи чітко відмежовані кластери; чи існують накладання або перетини між ними; чи є “шумові” точки (аномалії). За допомогою діаграм легко порівнювати результати кластеризації, отримані різними алгоритмами або при зміні параметрів (наприклад, кількості кластерів). Візуалізація результатів кластеризації у вигляді діаграм (розсіювання, теплових карт, дендрограм) забезпечує більш повне, інтуїтивне й змістовне уявлення про структуру даних, що робить аналіз значно ефективнішим.

```
[9]: claster_0=wine_df.data[predictions==0]
     claster_1=wine_df.data[predictions==1]
     claster_2=wine_df.data[predictions==2]
     plt.scatter(claster_0[:,0], claster_0[:,6], color='red')
     plt.scatter(claster_1[:,0], claster_1[:,6], color='black')
     plt.scatter(claster_2[:,0], claster_2[:,6], color='yellow')

[9]: <matplotlib.collections.PathCollection at 0x27f29d68410>
```

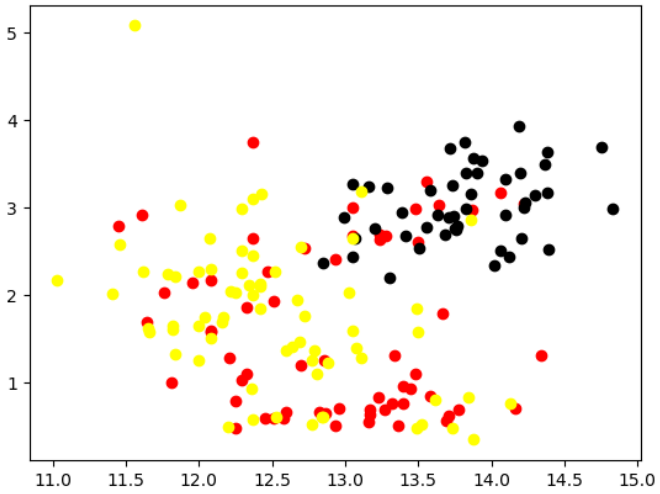


Рис. 4.11 Результати кластеризації

На діаграмі бачимо три кластери, створені за допомогою алгоритму KMeans (рис. 4.11). З діаграми видно, що у нас більш менш обосіблений тільки кластер 1. Що стосується 0 і 2 кластерів, у нас вони практично зливаються і ми з цим поки що нічого не можемо зробити. У даного методу є перевага, що тут ми просто задаємо кількість кластерів, на які нам потрібно розбити множину наших даних і все. Але для того, щоб отримати пристойні результати, пристойні метрики, нам потрібно спробувати виставити показник n_clusters у різних значеннях, тому що з точки зору класифікації у нас 3 класи, але з точки зору кластеризації методом k-

середніх у нас може бути оптимально 2 або 10 кластерів, бо множина, яку ми бачимо, може бути оптимально розбита на 10 кластерів.

Значення метрик Homogeneity, Completeness та V-measure близько 0.43 вказують на середній рівень якості кластеризації, що є типовим для даних, які не мають чітко розділених груп. Метрики Adjusted Rand Index (0.371) і Adjusted Mutual Information (0.423) підтверджують, що знайдені кластери частково відповідають реальним класам, але містять певні помилки класифікації. Разом із тим, Silhouette Coefficient (0.571) демонструє достатньо добру внутрішню структуру кластерів, що свідчить про відносно правильний вибір кількості кластерів (k) і прийнятну якість моделі.

Дослідивши метод кластеризації k-means, було визначено наступне. Коли ми виставили фіксовану кількість кластерів n_clusters=3, а дані почали надходити до моделі в подальшому, коли почали її експлуатувати, які виходять далеко за діапазони ядр кластерів, звичайно модель буде їх відносити до того чи іншого кластеру. Наприклад, якщо точки у нас будуть локалізуватися у правому верхньому куті, модель їх скоріше за все віднесе до кластеру 1. Якщо це у нас опосередкований набір даних, які створюють свій власний кластер, то що робити в цьому випадку? У цьому випадку модель KMeans нам не дуже добре підходить. Потрібно пробувати інші моделі для створення цих самих кластерів. Однією з таких моделей, яка більш універсальна, є модель DBSCAN.

4.5 Реалізація алгоритму DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) - це алгоритм кластеризації на основі щільності, який об’єднує точки у кластери, якщо вони розташовані досить близько одна до одної, а точки, що не належать до жодного кластера, вважає шумом (noise). Тобто DBSCAN - це щільнісний алгоритм просторової кластеризації з урахуванням шуму. Особливістю даної моделі є те, що тут в даному конкретному випадку нам не обов’язково задавати повний перелік ядер. Головне - задати мінімальну кількість точок, менше від яких у нас не буде вважатися, що дана конкретна сукупність є кластером. Її будуть

розглядати просто як шум. А от що стосується ядер і відповідної кількості кластерів навколо них, дана модель буде шукати сама. Це більш кращий варіант, який нам дозволяє використовувати дану модель у більш автоматизованому режимі і якщо в процесі експлуатації з'являється більше кластерів, ми можемо розраховувати на те, що вона нам виділить ці кластери і ми отримаємо те, що насправді хочемо отримати, тобто точний результат.

Модель DBSCAN досить популярна, використовується разом з іншими моделями на етапі передобробки даних, зокрема з моделями глибокого навчання перед тим, як дані передаються на нейронні мережі. Дана модель реалізована у багатьох фреймворках і бібліотеках. Вона реалізована в тому числі як окрема модель у таких бібліотеках як TensorFlow - бібліотеці для створення нейронних мереж. Під модель DBSCAN є клас у Sklearn. Скористаємося даною моделлю. DBSCAN лежить в пекетжі *sklearn.cluster*. Іноді для роботи з DBSCAN доводиться використовувати модель для декомпозиції даних. Для цього використаємо просту модель PCA [27]. Завантажимо набір даних з бібліотеки Scikit-learn (рис. 4.12).

```
[10]: iris_df=datasets.load_iris()
```

Рис. 4.12 Завантаження датасету з бібліотеки Scikit-learn

Створимо модель DBSCAN (рис. 4.13). Eps = 0.4 - мінімальна відстань в Евклідовому просторі. Параметр min_samples == 4 буде визначати мінімальну кількість точок, які будуть належати до кластера. Якщо буде виділятися ядро в межах кількості точок 3, 2, 1, то ці точки будуть вважатися шумом. Ядром буде вважатися тільки певне поле, яке має не менше чотирьох точок. Цей показник можна змінювати в різні сторони.

```
[11]: from sklearn.cluster import DBSCAN
      from sklearn.decomposition import PCA

[12]: dbscan=DBSCAN(eps=0.4, min_samples=4)

[13]: dbscan.fit(iris_df.data)

[13]: DBSCAN
      DBSCAN(eps=0.4, min_samples=4)
```

Рис. 4.13 Модель DBSCAN

Отримали модель. Натренували її на існуючих даних. Тепер нам потрібно з цими даними виконати декомпозицію. Ми наші кластери виділяємо по декільком предикторам, але оскільки ці предиктори перераховуються в Евклідову площину, то ми не можемо отримати більш менш об'єктивної картини з приводу розміщення наших кластерних ядер. Для того, щоб ми це отримали, нам потрібно провести декомпозицію наших даних, для цього потрібно підвантажити клас моделі PCA. PCA спрощує набір даних у плані використання предикторів для проведення кластеризації. Зокрема цю кількість предикторів ми можемо вказати у параметрах нашої моделі (там є спеціальний параметр *n_components*). Створимо модель PCA, задамо для неї *n_components=2*, зразу виконаємо *fit* на *wine_df.data*. Відбудеться навчання моделі PCA по даним *wine_df.data* тільки по першим двом стовпчикам, оскільки ми тут задали тільки 2 компоненти. Вже після цього ми можемо візуалізувати кластери. Для цього по отриманій моделі потрібно виконати трансформацію даних. Тобто ми їх із Евклідового простору знову повертаємось на 2d-площину, яку нам буде зручно створити за допомогою Matplotlib.

[14]:
```
pca=PCA(n_components=2).fit(iris_df.data)
pca_2d=pca.transform(iris_df.data)
```

Рис. 4.14 Зменшення розмірності даних

У межах нашої моделі кластеризації буде 4 групи елементів, тому що у нас буде 3 групи кластерів (0, 1, 2) і той самий шум, який є у будь-якій моделі кластеризації і який ми локалізували, вказавши для нього мінімальну кількість кластерів. У нас будуть осередки по 2, по 3, по одному, нам їх теж треба якимось чином виділити.

Результат обробки кластеру представлений на рис. 4.15.

[15]:
```
for i in range(0, pca_2d.shape[0]):
    if dbscan.labels_[i]==0:
        c1=plt.scatter(pca_2d[i,0], pca_2d[i,1], c='r', marker='+')
    elif dbscan.labels_[i]==1:
        c2=plt.scatter(pca_2d[i,0], pca_2d[i,1], c='g', marker='o')
    elif dbscan.labels_[i]==2:
        c3=plt.scatter(pca_2d[i,0], pca_2d[i,1], c='b', marker='*')
    elif dbscan.labels_[i]==-1:
        c4=plt.scatter(pca_2d[i,0], pca_2d[i,1], c='y', marker='d')
```

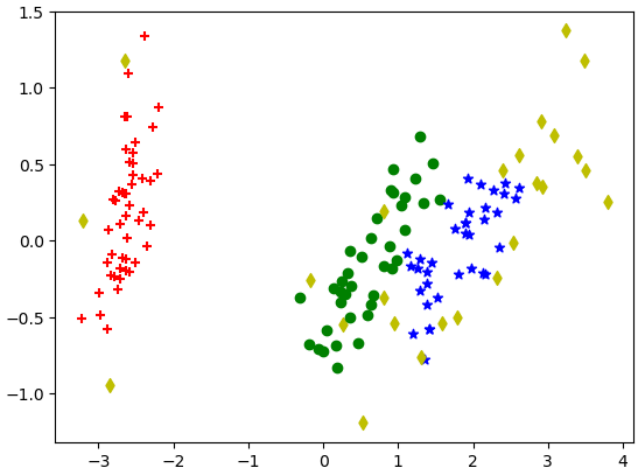


Рис. 4.15 Результат обробки кластеру

На графіку 4.15 вище бачимо результат обробки кластеру. Бачимо більш якісніший результат. Чітко виділено три кластери. Бачимо 2 кластери у безпосередній близькості один до одного. Ці кластери мають чіткі окреслення, мінімальну ступінь змішування і є ще так званий шум (жовтим). Ці кластери з точки зору моделі не відповідають критерію *min_samples=4*. Якщо ми розширимо датасет, добавивши туди додатковий клас, який, наприклад, може локалізуватися між червоним та зеленим кластером, відповідно DBSCAN нам цей клас окреслить у цих межах і нам не потрібно буде задавати та визначати, скільки всього може бути кластерів.

Визначимо метрики моделі (рис. 4.16).

```
[16]: labels=dbscan.labels_
      y=iris_df.target

[17]: print(f"Homogeneity: {metrics.homogeneity_score(y, labels):.3f}")
      print(f"Completeness: {metrics.completeness_score(y, labels):.3f}")
      print(f"V-measure: {metrics.v_measure_score(y, labels):.3f}")
      print(f"Adjusted Rand Index: {metrics.adjusted_rand_score(y, labels):.3f}")
      print(
          "Adjusted Mutual Information:"
          f"{metrics.adjusted_mutual_info_score(y, labels):.3f}"
      )
      print(f"Silhouette Coefficient: {metrics.silhouette_score(wine_df.data, labels):.3f}")

      Homogeneity: 0.810
      Completeness: 0.614
      V-measure: 0.699
      Adjusted Rand Index: 0.684
      Adjusted Mutual Information:0.692

      Silhouette Coefficient: 0.325
```

Рис. 4.16 Метрики моделі DBSCAN

Як бачамо, результат набагато кращий. Гомогенність моделі 0.810 - це вже достатньо високо і достатньо добре для того, щоб сказати, що модель дійсно

кластеризує дані. На основі метрики одинарного зв’язку ми можемо сказати, що наша модель дійсно виконує кластеризацію.

Метод DBSCAN виявив меншу узгодженість із еталонними класами, проте є ефективним у виявленні шумових точок та аномалій, які k-means не розпізнав. K-means доцільно використовувати для чітко структурованих та збалансованих даних, тоді як DBSCAN - для виявлення нетипових об’єктів, кластерів довільної форми та аналізу даних з шумами.

4.6 Ієрархічний кластерний аналіз

Крім DBSCAN, ще є інші підходи до кластеризації. Є підхід ієрархічної кластеризації. Суть ієрархічної кластеризації полягає в тому, що розбиття об’єктів на окремі кластери відбувається шляхом встановлення певних ієрархічних одиниць, які знаходяться на певних рівнях. Всю множину об’єктів ми розбиваємо за певними ознаками на 2 кластери. У свою чергу наступні 2 кластери з кожного боку, що у нас утворилися, теж на черговому рівні розбиваємо ще на 2 кластери і т.д. до самого найнижчого рівня. Таким чином, даний підхід дозволяє кластеризувати всі елементи досліджуваної множини елементів до найнижчого рівня, тобто тоді, коли кожен елемент буде представляти собою окреме кластерне ядро. Даний підхід, на відміну від попередніх, дозволяє нам отримати саму детальну кластеризацію з усіх можливих, тому що тут у нас йде розбиття до останнього елементу.

Якщо нам не потрібно, щоб в якості кластеру був кожен елемент, то ми по результатам даної кластеризації можемо просто вибрати певний рівень, який нам потрібний, і розбивати елементи по кластерам до певного рівня. Наприклад, вибрали рівень, який знаходиться в області 1.5 одиниці - тут у нас буде виділятися 13 кластерів. І нехай вони розділяються по цим 13 кластерам. Іншою перевагою даного підходу є те, що тут кількість кластерів динамічна, тобто ми не вказуємо, скільки у нас всього кластерних ядер буде потрібно. У нас на основі певного об’єктивного розбиття відбувається виділення кластерів в автоматичному режимі,

що іноді на користь, а іноді на шкоду. Це специфіка даного підходу і її потрібно враховувати.

Для реалізації ієрархічної кластеризації нам знадобляться моделі *linkage* (реалізує алгоритм висхідної кластеризації) та *dendrogram* (формує дендрограму). Їх можна імпортувати із *scipy.cluster.hierarchy*.

Для дослідження ієрархічного кластерного аналізу в коді, доцільно використовувати бібліотеку SciPy, тому що дана бібліотека дає великий доступ до різних математичних і статистичних функцій, в тому числі до функцій перевірки множин даних на їх певних характеристиках, наприклад, характеристиках розподілу, який може бути нормальним, біноміальним, Пуассонівським та іншими. Бібліотека дуже велика. Крім Sklearn обов'язково треба знати NumPy, Pandas, Matplotlib, SciPy. SciPy хоч і є надбудовою над NumPy, але ця надбудова дуже важлива.

Імпортуємо необхідні бібліотеки. Із пекетжа *cluster.hierarchy* імпортуємо *linkage* і *dendrogram* (рис. 4.17).

```
[2]: from scipy.cluster.hierarchy import linkage, dendrogram
```

Рис. 4.17 Імпорт бібліотек linkage, dendrogram

Linkage - бібліотека, яка виділяє кластери, dendrogram - бібліотека, яка формує кластери у певну ієрархію і потім ми її можемо відобразити в якості дендрограми, тобто linkage виконує ієрархічну кластеризацію (обчислює матрицю зв'язків між об'єктами); dendrogram будує дендрограму (дерево кластерів) на основі результатів linkage.

Створюємо модель. У linkage досить багато параметрів, два основних - це параметри методу і параметри датасету. Вкажемо попередній датасет. Якщо ми перейдемо у документацію SciPy, то побачимо, що кожний з методів розписується, є формули, по яким відбувається розрахунок для даного параметру. Наприклад, метод вираховує дистанцію між двома кластерами. Тобто кожного разу, коли у нас будується нове розгалуження, він за допомогою однієї з цих формул (методів) вираховує дистанцію між двома кластерами.

Завантажуємо набір даних Iris, виконуємо ієрархічну кластеризація (рис. 4.18).

```
[4]: from sklearn import datasets
     from scipy.cluster.hierarchy import linkage, dendrogram
     import matplotlib.pyplot as plt

[5]: iris_df = datasets.load_iris()

[6]: mergings = linkage(iris_df.data, method='complete')

[7]: plt.figure(figsize=(8, 4))
     dendrogram(mergings)
     plt.show()
```

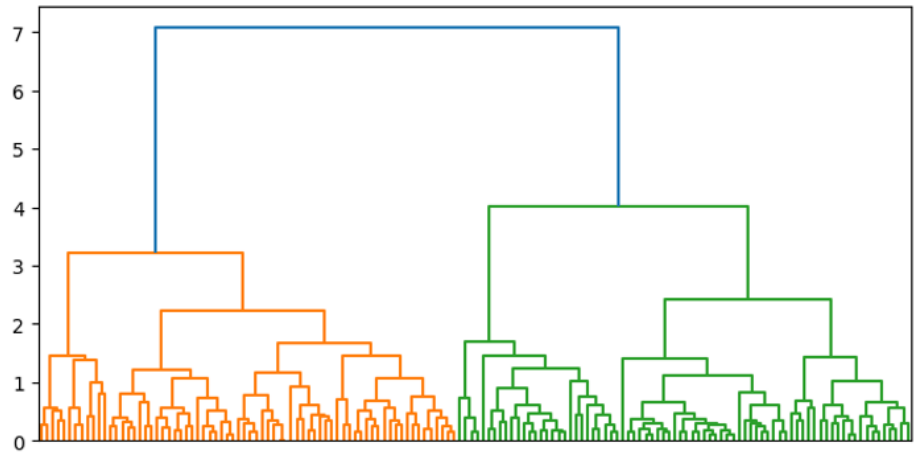


Рис. 4.18 Ієрархічна кластеризація

Отримали кластеризацію датасету Iris за допомогою методу dendrogram (рис. 4.18).

```
[9]: wine_df=datasets.load_iris()
     y=wine_df.target
     y
```

```
[11]: claster_0=wine_df.data[predictions==0]
      claster_1=wine_df.data[predictions==1]
      claster_2=wine_df.data[predictions==2]
      plt.scatter(claster_0[:,0], claster_0[:,3], color='red')
      plt.scatter(claster_1[:,0], claster_1[:,3], color='black')
      plt.scatter(claster_2[:,0], claster_2[:,3], color='yellow')
```

Рис. 4.19 Розподіл об’єктів за кластерами

Отримані результати візуалізовано на площині двох вибраних ознак (0-ї та 3-ї), де кожен кластер позначено власним кольором (рис. 4.20). Візуальний аналіз показав, що об’єкти всередині кожного кластера мають схожі значення ознак, тоді як між кластерами спостерігається певна відмінність, що свідчить про ефективність роботи алгоритму кластеризації для даного набору даних.

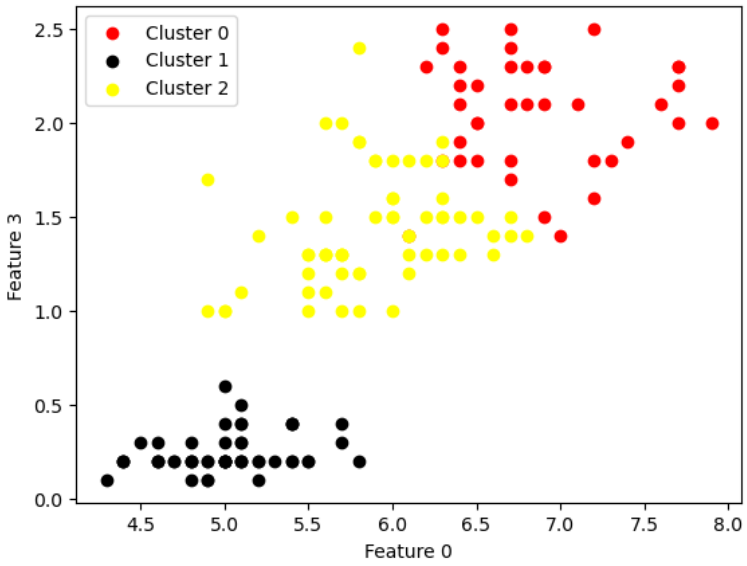


Рис. 4.20 Розподіл даних між кластерами

Щоб оцінити якість кластеризації, після побудови дендрограми потрібно: отримати мітки кластерів, порівняти їх із справжніми мітками, обчислити стандартні метрики - Homogeneity, Completeness, V-measure, Adjusted Rand Index, Adjusted Mutual Information і Silhouette Coefficient (рис. 4.21).

```
•[3]: from sklearn import datasets, metrics
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
import matplotlib.pyplot as plt


iris_df = datasets.load_iris()


mergings = linkage(iris_df.data, method='complete')


plt.figure(figsize=(8, 4))
dendrogram(mergings)
plt.title("Dendrogram (Complete Linkage)")
plt.xlabel("Samples")
plt.ylabel("Distance")
plt.show()
```

```
labels = fcluster(mergings, 3, criterion='maxclust')


y_true = iris_df.target

print("Оцінка якості ієрархічної кластеризації")
print(f"Homogeneity: {metrics.homogeneity_score(y_true, labels):.3f}")
print(f"Completeness: {metrics.completeness_score(y_true, labels):.3f}")
print(f"V-measure: {metrics.v_measure_score(y_true, labels):.3f}")
print(f"Adjusted Rand Index: {metrics.adjusted_rand_score(y_true, labels):.3f}")
print(f"Adjusted Mutual Information: {metrics.adjusted_mutual_info_score(y_true, labels):.3f}")
print(f"Silhouette Coefficient: {metrics.silhouette_score(iris_df.data, labels):.3f}")
```

Рис. 4.21 Обчислення метрик якості для ієрархічної кластеризації

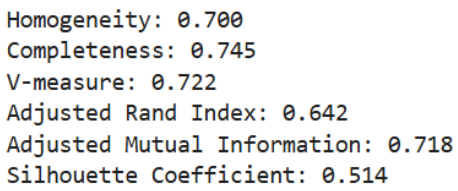


Рис. 4.22 Оцінка якості ієрархічної кластеризації

Результати ієрархічної кластеризації свідчать про досить якісне групування об’єктів. Homogeneity = 0.700 означає, що більшість елементів у кожному кластері належать до одного реального класу (достатньо добра чистота кластерів). Completeness = 0.745 означає, що класи не надто розпорошені між різними кластерами, але є невелике перекриття. V-measure = 0.722 - стійка якість кластеризації, що свідчить про адекватне співпадіння знайдених кластерів із реальними класами даних. Adjusted Rand Index (ARI) = 0.642 - вимірює схожість між кластеризацією та реальними мітками з урахуванням випадковості. Значення 0.64 - це досить високий результат, означає, що близько 64% рішень про спільність/відмінність об’єктів збігаються з істинними. Adjusted Mutual Information (AMI) = 0.718 показує, наскільки інформація, що міститься в отриманих кластерах, узгоджується з реальною розміткою. Значення 0.718 підтверджує високу інформативність та адекватність структури кластерів. Silhouette Coefficient оцінює, наскільки чітко кластери відокремлені один від одного (0 - погано, 1 - дуже добре). Значення 0.514 вказує, що кластерні межі досить чіткі, об’єкти добре належать до своїх груп, хоча можливе часткове перекриття між межами деяких кластерів.

Кластеризація як метод може використовуватися як самостійно, так і як важливий компонент каскаду наших моделей. Вона застосовується досить часто. Існує дуже багато алгоритмів для вирішення задач по кластеризації. Алгоритм методу k-means дозволяє виділяти фіксовану кількість кластерів. Його доцільно застосовувати тоді, коли ми чітко знаємо скільки у нас кластерів може бути у даній множині. Більш гнучкий і більш об’єктивний алгоритм - це алгоритм DBSCAN, який автоматично визначає кількість кластерів, а також вміє визначати

такі компоненти як шум, тобто елементи, які не входять ні до якого іншого кластеру. Більш приватний метод - кластеризація методом дендограм.

Метод DBSCAN показав найвищі показники узгодженості кластерів із класами даних, що свідчить про його ефективність при наявності шумових або нерівномірно розподілених даних. Алгоритм k-means формує добре розділені, компактні кластери, але його точність нижча через припущення про сферичну форму кластерів. Ієрархічна кластеризація продемонструвала найбільш збалансовані результати - хорошу якість поділу та інтерпретованість структури, що робить її оптимальним підходом для даного набору даних.

Якщо оцінювати узгодженість кластерів із реальними класами (Homogeneity, ARI, AMI), то найкращим виявився DBSCAN. Якщо враховувати структурну чіткість кластерів (Silhouette Coefficient), то перевага за k-means. Якщо ж брати збалансованість усіх метрик, то найбільш оптимальним є ієрархічний метод кластеризації.

ВИСНОВКИ

У результаті виконання магістерської кваліфікаційної роботи досліджені особливості навчання Unsupervised Learning. Обгрунтовано вибір програмного забезпечення Jupyter Notebook для розробки ML-моделі. Виконано розробку моделі машинного навчання на основі Unsupervised Learning. Виконано навчання та оцінку ML-моделі.

На відміну від навчання з учителем (Supervised Learning), алгоритми навчання без вчителя (Unsupervised Learning) працюють тільки з вхідними даними, без відомих правильних відповідей. Модель самостійно виявляє структуру та закономірності в даних. Алгоритми здатні знаходити схожі групи об’єктів (кластери), шаблони, асоціації або аномалії.

Навчання без вчителя не потребує розмітки даних; дозволяє знаходити сегменти, аномалії, закономірності, які складно побачити вручну; допомагає аналітикам і бізнесу отримати структуровану інформацію з великих даних та на її основі формувати стратегії або прогнози; є основою для подальшого навчання із вчителем. Результати Unsupervised Learning можна використовувати для генерації міток, класифікації або підготовки даних для алгоритмів Supervised Learning.

Виконавши реалізацію моделі машинного навчання з використанням алгоритму k-means, можна зробити висновки, що даний алгоритм легко реалізувати за допомогою сучасних бібліотек (наприклад, Scikit-learn), а отримані результати - зрозумілі та інтерпретовані. Метод k-means є балансом між простотою, швидкістю та ефективністю, його доцільно застосовувати як у наукових дослідженнях, так і у промислових аналітичних системах. Результати, отримані у магістерській кваліфікаційній роботі, свідчать, що алгоритм k-means зміг виявити у наборі даних певну кластерну структуру, однак розподіл об’єктів між кластерами не є повністю однорідним.

Подальше підвищення точності кластеризації можливе за рахунок попередньої нормалізації ознак, відбору інформативних параметрів або застосування більш складних методів, таких як DBSCAN, або ієрархічна кластеризація. Після

реалізації алгоритму DBSCAN, визначено, що метрики Homogeneity, Completeness, V-measure і ARI для DBSCAN збігаються або близькі до значень k-means. Це означає, що DBSCAN змоделював подібну структуру кластерів, проте з певними відмінностями у внутрішній щільності.

Отримані у магістерській кваліфікаційні роботі результати демонструють, що ієрархічний метод кластеризації достатньо ефективно виявив природну структуру даних. Кластери мають помірно високу чистоту і повноту. Межі кластерів досить добре виражені, а результати узгоджуються з реальними класами даних (особливо порівняно з методом k-means).

Unsupervised Learning підходить для обробки великих обсягів інформації, де немає попередньо відомих результатів. Дані організовуються у кластери або структури без людського втручання, що дозволяє спрощувати аналіз і виявляти закономірності.

Кластеризація дозволяє знайти природні групи або патерни в даних, які не є очевидними на перший погляд. Це допомагає виявити закономірності, аномалії або сегменти користувачів, що має практичне значення у бізнесі, медицині, маркетингу, кібербезпеці. Кластеризація допомагає спростити структуру даних, що підвищує ефективність подальших алгоритмів машинного навчання. Результати кластеризації можуть бути використані для оптимізації управлінських рішень, наприклад, для визначення типів поведінки користувачів, ризикових груп або сегментів продукції. Це дає змогу створювати точніші моделі прогнозування й автоматизувати процес аналізу великих обсягів даних.

Кластеризація є невід’ємною складовою машинного навчання, оскільки забезпечує розуміння структури даних, виявлення прихованих закономірностей та оптимізацію процесів обробки інформації. Завдяки своїй гнучкості, незалежності від навчальних міток та здатності працювати з великими масивами даних, методи кластеризації залишаються одними з найбільш затребуваних інструментів у сучасних системах штучного інтелекту та аналітики даних.

ПЕРЕЛІК ПОСИЛАНЬ

1. https://www.python.org/
2. https://matplotlib.org/
3. https://pytorch.org/
4. https://numpy.org/
5. https://matplotlib.org/
6. https://www.tensorflow.org/
7. 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.
8. 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.
9. 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].
10. Wang, Haofan et al. (2020). Score-CAM: Score-Weighted Visual Explanations for Convo-lutional Neural Networks. arXiv: 1910.01279 [cs.CV].
11. 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.
12. 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.
13. Jake VanderPlas Python Data Science Handbook: Essential Tools for Working with Data, 2022, 551p.
14. Грас Д. Data Science. Наука про дані з нуля: Пер. з англ. – 2-е видання., перероб. та доп. – 2021. – 416 с.

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

16. 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.

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

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

19. 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.

20. 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.

21. C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. arXiv:1611.03530. https://arxiv.org/abs/1611.03530

22. L. Zhang, C. Aggarwal, and G.-J. Qi. Stock Price Prediction via Discovering Multi-Frequency Trading Patterns. ACM KDD Conference, 2017.

23. 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.

24. 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.

25. Гайдур Г.І., Бондаренко З.З. Теорія інформації та кодування: Навчальний посібник для підготовки до практичних занять. – К.: ДУІКТ, 2024 – 43 с.

26. Холод О.М. Комунікаційні технології [текст] підручник / О.М. Холод – К.: «Центр учбової літератури», 2021. – 213 с.

27. Ткаленко О.М., Сторчак К.П. Розподіл інформації в системах інтегрованого доступу. – Навчальний посібник. – Київ, ДУТ, 2019. – 56 с.

28. Величко О. М. Інтелектуальні інформаційні системи: структура і застосування: підручник / О. М. Величко, Т. Б. Гордієнко. – Херсон: Олді+, 2022. – 728 с.

29. Сєрих С. О. Вибір та налаштування кінцевого обладнання інформаційних систем. Керівництво до проведення і виконання практичних занять. – К.: ДУТ, 2020. – 93 с.

30. Кутковецький В. Я. Розпізнавання образів: навчальний посібник / В. Я. Кутковецький. – Миколаїв: Вид-во ЧНУ ім. Петра Могили, 2017. – 420 с.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)