НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра математичного моделювання та аналізу даних

«На правах рукопису»
УДК 004.85

До захисту допущено:
Завідувач кафедри
_____ Наталія КУССУЛЬ
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Математичні методи моделювання, розпізнавання образів та комп'ютерного зору**»**

зі спеціальності 113 «Прикладна математика»

на тему: «Дослідження методів зменшення розмірності даних в галузі інтелектуального аналізу»

Виконав:
студент ІІ курсу, групи ФІ-01мп
Хомініч Дмитро Сергійович _____

Науковий керівник:
Доцент, к.ф.-м.н.
Орєхов Олександр Арсенійович _____

Рецензент:
Доцент кафедри ІБ «ФТІ», к.ф.-м.н., с.н.с.
Смирнов Сергій Анатолійович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2021 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра математичного моделювання та аналізу даних

Рівень вищої освіти – другий (магістерський)
Спеціальність (освітня програма) – 113 Прикладна математика («Математичні методи моделювання, розпізнавання образів та комп'ютерного зору»)

До захисту допущено:

Завідувач кафедри

_____ Наталія КУССУЛЬ

«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту

Хомініч Дмитро Сергійович
(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження методів зменшення розмірності даних в галузі інтелектуального аналізу.

_____ ,
керівник роботи _____
доцент кафедри ММАД, к.ф.-м.н., Орєхов Олександр Арсенійович ___,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «____ » _____ 2021 р. №

2. Термін подання студентом роботи 30 листопада 2021 р.

3. Вихідні дані до роботи

1. Попередні дослідження, а також навчальні посібники.
2. Алгоритми машинного навчання.
3. Датасет з ознаками.

4. Зміст роботи

1)Вивчити різні підходи та методи для зменшення розмірності даних.
2)Ознайомитись з проблемами розмірності даних.
3)Побудувати модель класифікатора з методом зниження розмірності та порівняти із стандартним методом.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)
Методи зменшення розмірності даних — презентація.

6. Дата видачі завдання 12.02.2021

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
	Отримання завдання	12.02.2021	
	Збір інформації	20.02.2021	
	Дослідження предметної області та існуючих рішень	06.03.2021	
	Розробка плану роботи	10.03.2021	
	Побудова моделі	01.10.2021	
	Проведення тестування моделі	05.10.2021	
	Оцінка результатів	18.10.2021	
	Оформлення дипломної роботи	01.11.2021	
	Отримання допуску	30.11.2021	

Студент _____ _____
(підпис) (ініціали, прізвище)

Керівник роботи _____ _____
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Обсяг роботи 79 сторінки, 28 ілюстрації, 2 таблиць, 1 додатки, 7 джерело літератури.

Дослідження методів та алгоритмів зниження розмірності в даних, та побудова наявної моделі класифікації з використанням технік машинного навчання. Суть в тому, щоб побудувати модель з використанням технік зменшення розмірності даних, щоб уникнути деяких проблем пов'язаних з переосназченими даними, та щоб збільшити точність передбачень класифікатора.

Методом дослідження було опрацювання робот по методам машинного навчання, присвячені теоретичним і практичним аспектам їх застосування, а також документація до методів обробки даних. У роботі використана теорія алгоритмів, математичний аналіз, теорія ймовірності, та програмування, а також математична статистика та аналіз даних.

Об'єктом дослідження є використання визначеної термінології у публікаціях, та навчальних посібниках різних напрямів та нормативно-правових документах. Були пророблено різні підходи визначено кращі напрями, та враховані усі рекомендації.

Предметом дослідження є методи машинного навчання, алгоритми та математичні методи.

Ключові слова: МАШИННЕ НАВЧАННЯ, КЛАСИФІКАТОР, ВІДСТАНЬ, ЯКІСТЬ, КРИТЕРІЙ, МОДЕЛЬ, ЕКСПЕРИМЕНТ, КОМПОНЕНТИ, ЗНИЖЕННЯ РОЗМІРНОСТІ.

ABSTRACT

Volume of work 79 pages, 28 illustrations, 2 tables, 1 appendices, 7 source of literature.

Research of methods and algorithms for reducing the dimensionality in the data, and construction of the existing model of classification using machine learning techniques. The point is to build a model using data dimension reduction techniques to avoid some data issues and to increase the accuracy of classifier predictions.

The method of research was the development of robots on the methods of machine learning, devoted to theoretical and practical aspects of their application. The paper uses algorithm theory, mathematical analysis, probability theory, and programming, as well as mathematical statistics and data analysis.

The object of research is the use of certain terminology in publications and textbooks in various fields and legal documents. Different approaches were identified, the best directions were identified, and all recommendations were made.

The subject of research is machine learning methods, algorithms and mathematical methods.

Keywords: MACHINE LEARNING, CLASSIFIER, DISTANCE, QUALITY, CRITERION, MODEL, EXPERIMENT, COMPONENTS, DIMENSIONALITY REDUCTION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

d(x,y) – скалярна метрика;

cluster – мітка призначення об’єкту деякому класу;

ОС – обчислювальна система;

ПК – персональний комп’ютер;

dataset – набір даних;

Confusion matrix – матриця похибок, або матриця неточності;

Completeness score – критерій повноти;

Accuracy — точність методу;

ВСТУП

Актуальність роботи: Робота присвячена дослідженню методів і алгоритмів зменшення розмірності даних. Зменшення розмірності даних використовується в системах інтелектуального аналізу даних для зменшення кількості функцій в даних для простішого пошуку рішення.

В цифровому всесвіті дуже багато інформації накопичено на серверах, в базах даних, та будь-яка людина може скористуватись цією інформацією скориставшись всесвітньою мережею – інтернет. Але як зрозуміло, інформації настільки багато, що дуже важко знайти щось цікаве, тому що інформація зберігається в необробленому вигляді. Щоб знайти щось важливе, та корисне людині, інформацію треба якось оброблювати. Цим займаються статистичні методи та методи штучного інтелекту.

Статистичні методи обробки інформації займають малу кількість потреб в аналізі тому з’явилися, так звані, методи штучного інтелекту, які можуть оброблювати інформацію на власному досвіді. Цим займається інтелектуальний аналіз даних, що займає велику роль в аналізі та дослідженні великих обсягів даних[5].

На даний момент широку популярність отримали методи штучного інтелекту, що займаються пошуком та аналізом даних. Вони шукають нетипові закономірності в даних, з подальшою обробкою та прогнозуванням в нових задачах. На сьогодні методи штучного інтелекту дуже часто зустрічаються в нашому житті.

Основні методи штучного інтелекту:

1. Класифікація;
2. Регресія;
3. Зниження розмірності даних;
4. Кластеризація;

Алгоритми зниження розмірності можуть сильно спростити вирішувану задачу, бо якщо набір даних з великою кількістю функцій, це може привести до погіршення якості роботи. Ці алгоритми переводять дані з вхідного простору ознак до меншого оптимального, що дозволяє прискорити роботу класифікатора, та іноді навіть

покращити кінцевий результат. Ще вони дозволяють уникнути прокляття розмірності даних. Це така проблема розріджених даних, коли відстані між точками становляться дуже великими. Прокляття розмірності уповільнює роботу алгоритмів, потребує багато часу та пам'яті комп'ютера для обробки. Алгоритми зменшення розмірності даних видаляють сильно залежні змінні та шум, який дуже погано впливають на класифікатори.

Мета і завдання дослідження

Метою даної роботи є дослідження методів та алгоритмів зниження розмірності даних, які не потребують попередньої інформації про структуру даних, та побудова моделі класифікатора на основі алгоритмів пониження розмірності даних.

Об'єкт дослідження: використання визначеної термінології у публікаціях та навчальних посібниках різних напрямів та нормативно-правових документах.

Предмет дослідження: методи машиного навчання без вчителя, методи машинного навчання з вчителем.

Методи дослідження: роботи по методам класифікації, зниження розмірності даних, присвячені теоретичним і практичним аспектам їх застосування. У роботі використана дискретна і обчислювальна математика, а також математична статистика та програмування.

Наукова новизна одержаних результатів:

1. Дослідження проблеми великої розмірності даних**;**
2. Методика розробки вкладеної моделі класифікації з використанням методів зниження розмірності.
3. Застосування налаштованих алгоритмів на інших даних.

Практичне значення одержаних результатів: Вкладене рішення для задачі класифікації з великими датасетами, використовуючи алгоритми зниження розмірності даних, дають хороші результати в обробці інформації.

Апробація результатів роботи: Основні положення та результати доповідалися і обговорювалися на VIII Всеукраїнській науково-практичну конференцію студентів, магістрів, аспірантів «Current Trends in Young Scientists' Research», 22 квітня 2021 року в Державному університеті «Житомирська політехніка».

1 ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

1.1 Задачі інтелектуального аналізу даних

Розвиток обчислювальної техніки та комунікацій призвела до створення суспільства, яке живиться інформацією. Це призвело до росту обсягів інформації і потребі якось її обробляти. Більша частина інформації знаходиться в необробленому вигляді. Якщо дані характеризуються як зафіксовані факти, то інформація - це набір закономірностей або спостережень, що лежать в основі даних. У базах даних зберігається величезна кількість інформації, яка потенційно важлива, але ще не виявлена або не інтерпретована. Наша задача полягає в тому, щоб знайти неявні закономірності в даних та якось ними скористатись у різних областях людської діяльності.

Інтелектуальний аналіз даних - область знань, що відноситься до обробки даних, що вивчає пошук і опис прихованих, нетривіальних і практично корисних закономірностей в досліджуваних даних[1]. Ідея полягає в тому, щоб створювати комп'ютерні програми, які збирають та описують інформацію з різних баз даних в пошуках залежностей або закономірностей. Крім етапу необробленого аналізу, він також включає аспекти бази даних і управління даними, попередню обробку даних, розгляд моделі і логічного висновку, метрики інтересу, міркування складності, постобробку виявлених структур, візуалізацію і онлайн-оновлення. Сильні патерни, якщо вони будуть знайдені, узагальнюються, щоб зробити точні прогнози щодо майбутніх даних. Різниця між аналізом даних та інтелектуальним аналізом даних полягає в тому, що аналіз даних використовується для перевірки моделей і гіпотез щодо набору даних, незалежно від обсягу даних, а інтелектуальний аналіз даних навпаки використовує машинне навчання і статистичні моделі для виявлення таємничих або прихованих закономірностей у великому обсязі даних.

Звичайно часто зустрічаються проблеми. Багато знайдених шаблонів будуть тривіальними та нецікавими в рамках деякої задачі. Інші будуть помилковими, залежними від випадкових збігів в конкретному використовуваному наборі даних. Взагалі реальні дані недосконалі:

- Неповні: відсутність певних атрибутів або їх значень, що представляють інтерес, або містять тільки сукупні дані.
- Містять шум: в значеннях присутні помилки або викиди. Викиди — це екстремальні значення у вхідних даних, які знаходяться далеко за межами інших спостережень.
- Неузгоджені: містять невідповідності в кодах або іменах.

Точність аналізу цих даних буде зменшуватись, результати будуть неточними, а закономірності викривленими. Перш ніж можна буде використовувати алгоритми інтелектуального аналізу даних, необхідно зібрати цільовий набір даних. Оскільки інтелектуальний аналіз даних може виявити тільки шаблони, фактично присутні в даних, цільовий набір даних повинен бути достатньо великим, щоб містити ці шаблони, але при цьому залишатися досить коротким, щоб можна було добути результати протягом прийнятного періоду часу. Попередня обробка необхідна для аналізу багатовимірних наборів даних перед інтелектуальним аналізом. Потім цільовий набір очищається. Очищення даних видаляє спостереження, що містять шум, і спостереження з відсутніми даними.

Задачі попередньої обробки даних, підготовки даних:

- Очищення даних: заповнення пропущених значень, виявлення і видалення викривлених даних і викидів.
- Інтеграція даних: використання декількох баз даних, кубів даних або файлів.
- Перетворення даних: нормалізація і агрегація.
- Скорочення даних: зменшення обсягу даних (незалежних параметрів).
- Дискретизація даних: перетворення типів атрибутів на потрібні для обробки алгоритму.
- Очищення тексту: видалення впроваджених символів, які можуть порушувати вирівнювання даних, наприклад впроваджених символів табуляції у файлі з

роздільником-табуляцією, впроваджених нових ліній, які можуть розбивати записи. Видалення сполучників, та інших неінформативних даних.

Стандартний процес обробки даних методами інтелектуального аналізу виглядає так:

1. Розуміння поставленої задачі.
2. Розбір і розуміння даних.
3. Підготовка даних до аналізу (обробка даних).
4. Моделювання, робота з методами машинного навчання.
5. Оцінювання моделі та результатів.
6. Оформлення робочої моделі, та її результатів.

Алгоритми повинні бути досить надійними, щоб справлятися з недосконалими даними і витягувати корисні закономірності.

1.2 Класифікація задач інтелектуального аналізу даних

Інтелектуальний аналіз даних включає шість загальних класів задач:

1. Виявлення аномалій — виявлення нестандартних екземплярів даних, які відрізняються від основних, але можуть бути цікаві, або помилок в даних, що вимагають подальшого аналізу та вивчення.
2. Вивчення правил асоціації (моделювання залежності в даних) — пошук взаємозв'язків між змінними. Наприклад, будь-який магазин може збирати дані про вподобання клієнтів, аналізуючи їх за допомогою правил асоціації. Цю інформацію можна використовувати в різноманітних маркетингових цілях.
3. Кластеризація — це задача виявлення груп і структур в даних, які "схожі" за деяким критерієм, а саме розбиття множини об’єктів на групи, які називаються кластерами, в яких знаходяться схожі елементи.

4. Класифікація — це завдання узагальнення відомої структури для застосування до нових даних. Наприклад, спам-фільтр має класифікувати електронний лист як “спам” або як "повідомлення".
5. Регресія — намагається знайти функцію, яка моделює дані з найменшою помилкою, тобто для оцінки відносин між даними або наборами даних.
6. Узагальнення — забезпечення більш компактного представлення набору даних, включаючи візуалізацію і створення звітів.

Ці задачі поділяються на 2 основні типи моделі технології інтелектуального аналізу даних. Мета технології Data Mining — знаходження даних таких моделей, які не можуть бути знайдені звичайними методами. Існують два види моделей: прогнозні та описові.

Прогнозні (predictive) моделі будуються на підставі набору даних з відомими результатами. Вони використовуються для прогнозування результатів на основі інших наборів даних. При цьому потрібно, щоб модель працювала максимально точно, була статистично значущою й оправданою для вирішення задачі. На основі вже налаштованої моделі прогнозуються результати роботи з іншим набором даних. **Прогнозні задачі** працюють в два етапи:

1. На основі набору даних з відомими результатами будується деяка модель. Обирати, яку модель використовувати, залежить від поставленої задачі, та типу набору даних. Налаштування моделі відбувається вручну експертом у даної галузі, до якої належить задача, або автоматизовано шляхом налаштування гіперпараметрів відповідними методами машинного навчання (Наприклад, Налаштування параметрів пошуку в сітці).
2. Модель тестується, та використовується для передбачення результатів на нових схожих наборах даних.

До даного виду задач відносяться класифікація та регресія:

- Моделі **класифікації** — описують правила або набір правил, у відповідності з якими можна віднести опис будь-якого нового об'єкта до одного з класів. Такі правила будуються на підставі інформації про існуючі об'єкти шляхом розбиття їх на класи.

- Моделі **послідовностей** — описують функції, що дозволяють прогнозувати зміну неперервних числових параметрів. Вони будуються на підставі даних про зміну деякого параметра за минулий період часу.

Описові (descriptive) моделі покращують розуміння аналізованих даних, приділяють увагу суті закономірностей в наборі даних, взаємному впливу різних факторів, та покращення опису інших задач побудови емпіричних моделей різних систем. Ключовий момент в таких моделях — легкість і прозорість для сприйняття людиною. Саме описові задачі шукають нетривіальні зв'язки, та показують їх. Інструмент для вирішення задач інтелектуального аналізу даних, це область знань, яка називається машинне навчання (machine learning). За способами реалізації, задачі класифікують на supervised learning та unsupervised learning.

В **supervised learning** також відоме як машинне навчання з учителем, є підкатегорією машинного навчання і штучного інтелекту. Він визначається використанням позначених наборів даних для навчання алгоритмів, які дозволяють точно класифікувати дані або передбачати результати. По мірі того, як вхідні дані вводяться в модель, вона коригує свої ваги до тих пір, поки модель не буде належним чином підігнана, що відбувається як частина процесу перехресної перевірки. Контрольоване навчання допомагає організаціям вирішувати різні реальні проблеми в будь-якому масштабі, наприклад класифікувати спам в окремій папці від вашої поштової скриньки.

В **unsupervised learning**, також відоме як машинне навчання без вчителя, використовує алгоритми машинного навчання для аналізу немаркованих наборів даних. Ці алгоритми виявляють приховані закономірності або групи даних без необхідності втручання людини. Ці алгоритми дозволяють добре описувати та виявляти подібності та відмінності в інформації. Машинне навчання без вчителя дуже хороший інструмент для дослідницької діяльності, кластеризації текстів, розпізнавання зображень та інше.

Висновки до розділу 1

В цьому розділі було визначено поняття інтелектуального аналізу даних, розглянуті основні задачі та методи інтелектуального аналізу. Визначено основні етапи підготовки даних та аналізу. Класифіковано різновиди моделей інтелектуального аналізу. Було визначено чітку класифікацію задач. Було описано основні інструменти, та класифікацію алгоритмів за способом реалізації.

2 МАШИННЕ НАВЧАННЯ

2.1 Новий підхід аналізу даних

Алгоритм машинного навчання — це алгоритм, здатний навчатися на даних. "Навчання" - це здатність моделі навчатися, виявляти приховані закономірності, на основі побудованих правил та коригувань. Алгоритм після навчання, починає видавати результати, щось прогнозувати або описувати. Центральне місце в алгоритмах займають не дані, а цільова функція. Коли починається рішення практичної задачі, дуже важливо визначити цільову функцію, обов'язково визначитись, як оцінювати результати. Вибір цільової функції повністю визначає всю подальшу роботу, і навіть в схожих завданнях різні цільові функції можуть привести до зовсім різних моделей.

Машинне навчання (ML) — це галузь штучного інтелекту (AI) та інформатики, яка фокусується на використанні даних та алгоритмів для імітації способу навчання людей, поступово підвищуючи його точність. За допомогою статистичних методів алгоритми навчаються класифікувати дані, будувати прогнози і виділяти важливу інформацію в ході проектів зі збору та аналізу даних. За допомогою самонавчальних алгоритмів ми можемо перетворити дані в знання. Завдяки тому, що за останні кілька років були розроблені чисельні та потужні бібліотеки з відкритим кодом, область машинного навчання стала дуже популярна. Це дозволяє спеціалістам швидко навчатись та виявляти в даних приховані зв'язки, описувати дані, робити дослідження і прогнози про майбутні події. Згодом такі рішення покращують зручність життя людства, стимулюють прийняття рішень в наукових, освітніх й практичних сферах життя. Машинне навчання набуває все більшої значущості в наукових дослідженнях в області інформатики, воно починає грати все більш активну роль в нашому повсякденному житті.

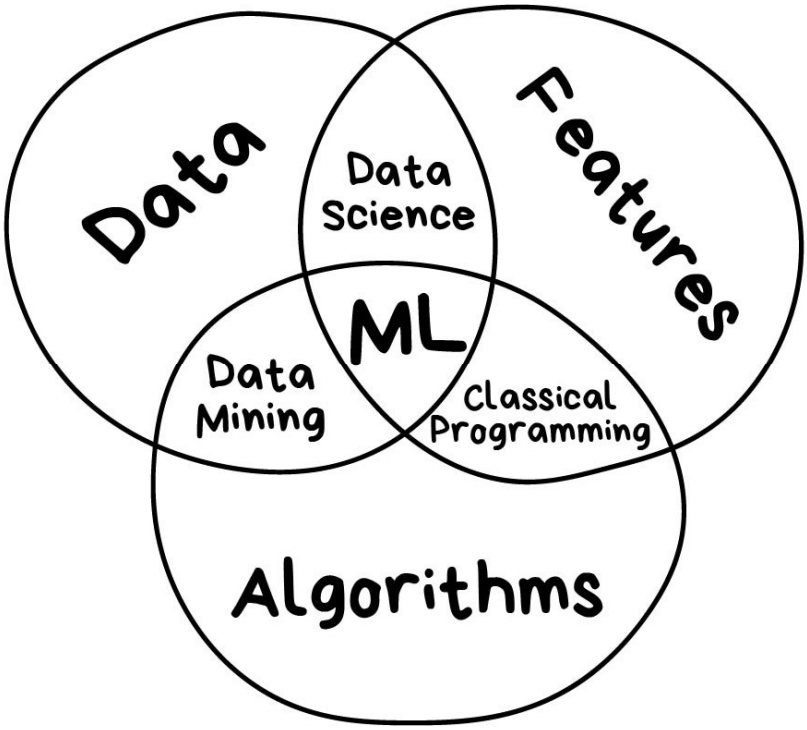


Рисунок 2.1 – Межі машинного навчання

Методологія машинного навчання пропонує для відокремлення знань з даних поступове покращення якості прогнозних моделей і прийняття рішень, коригованих даними. Область, в якій машинне навчання досягає успіху — це проблеми, які занадто складні для традиційних підходів, або не мають відомого алгоритму. Наприклад, розглянемо розпізнавання мови. Очевидно, що стандартна алгоритмічна техніка не буде масштабуватися до тисяч слів, вимовлених мільйонами дуже різних людей на десятках мов. Краще рішення — написати алгоритм, який навчається сам по собі, враховуючи безліч прикладів записів для кожного слова. Машинне навчання може допомогти людям вчитися: алгоритми ML можна перевірити, щоб побачити, як вони навчилися. Наприклад, як тільки спам-фільтр був навчений достатній кількості спаму, його можна легко перевірити, щоб виявити список слів і комбінацій слів, які є кращими предикторами спаму. Це дозволяє виявити несподівані взаємозв'язки або нові тенденції і тим самим привести задачу до кращого розуміння проблеми. Застосування методів ML для обробки великих обсягів даних може допомогти

виявити закономірності, які не були відразу очевидні. Програми машинного навчання можуть виконувати завдання, навіть не будучи запрограмованими явним чином. Він включає в себе навчання комп'ютерів на наданих даних для виконання певних завдань. Для простих завдань, можна запрограмувати алгоритми, що повідомляють машині, як виконувати всі кроки, необхідні для вирішення даної проблеми. Для більш складних завдань людині може бути складно вручну створити необхідні алгоритми. На практиці може виявитися більш ефективним допомогти машині розробити свій власний алгоритм, ніж змушувати програмістів вказувати кожен необхідний крок.

Підводячи підсумок, машинне навчання дуже добре підходить для:

- Проблеми, для яких існуючі рішення вимагають великої ручної настройки або довгих списків правил: один алгоритм машинного навчання часто може спростити код і поліпшити продуктивність.
- Складні проблеми, для яких взагалі немає хорошого рішення з використанням традиційного підходу: кращі методи машинного навчання можуть знайти рішення.
- Мінливі умови: система машинного навчання може адаптуватися до нових даних.
- Отримання інформації про складні проблеми і великих обсягах даних.

2.2 Класи та задачі машинного навчання

У роботі Mitchell (1997) дано визначення: "Комп'ютерна програма навчається на досвіді E відносно деякого класу задач T і міри якості P, якщо якість на задачах з T, виміряне за допомогою P, зростає з ростом досвіду E"[2].

Задача T. Машинне навчання дозволяє вирішувати задачі, які дуже важкі для вирішування за допомогою програм, написаних і сконфігурованих людиною. Завдання машинного навчання описуються в термінах того, як система машинного навчання повинна обробляти приклад. Прикладом є набір ознак, отриманих в

результаті вимірювання деякого об'єкта або події, які система повинна навчитися обробляти. Як правило, приклад представляється у вигляді вектора $x \in \mathbb{R}_n$, де кожен елемент це ознака. За допомогою машинного навчання можна вирішувати багато цікавих задач:

- **Класифікація**. В цих задачах програма повинна відповісти, до якої категорії k належить деякий екземпляр з набору даних. Для рішення задачі потрібно породити функцію $f: \mathbb{R}^n \to \{1..k\}$. Якщо $f(x) = y$, то модель відносить вхідний екземпляр, який описується вектором x до класу y. Прикладом задачі класифікації є розпізнавання об'єктів, коли на вхід подається зображення, а на виході виходить числовий код, який ідентифікує присутній в зображенні об'єкт.
- **Регресія.** В цій задачі програма повинна спрогнозувати числове значення по вхідним даним. Це в основному класифікація, при якій ми прогнозуємо число, а не категорію. Прикладами є ціна автомобіля по пробігу, трафік за часом доби, обсяг попиту по зростанню компанії. Регресія дуже добре працює коли щось залежить від часу.
- **Транскрипція**. В таких задачах програма повинна проаналізувати неструктуроване представлення даних, щоб перетворити її у текстову форму. Наприклад якщо на вхід подано фотографію тексту, то на виході повинен бути сам текст у виді послідовності символів**.**
- **Машинний переклад.** В таких задачах на вхід подається послідовність символів на одній мові, а на виході отримаємо послідовність на іншій мові. Такі системи використовуються для обробки природніх мов, наприклад, для перекладу текстів.
- **Структурний висновок.** Під структурним висновком розуміється будь-яке завдання, в якій на виході породжується вектор (або інша структура, що містить кілька значень), між елементами якого існують важливі зв'язки[2].
- **Виявлення аномалій.** В цих задачах програма слідкує та аналізує безліч подій, або об’єктів, та маркує деякі з них як нетипові. Приклад, це шахрайство з банківськими картами, кібератаки, та фільтри спаму для електронної пошти.

- **Синтез та вибірка.** У задачах цього типу алгоритм машинного навчання повинен генерувати нові приклади, схожі на навчальні дані[2]. Синтез і вибірка методами машинного навчання корисні в мультимедійних додатках, коли генерування великих обсягів даних вручну, не варто затрачених ресурсів. Наприклад, відеоігри вміють автоматично генерувати текстури для об'єктів, не змушуючи робітників працювати над кожним пікселем.
- **Знешумлення.** До алгоритму машинного навчання подається зашумлений приклад. Алгоритм повинен відновити чистий приклад або видалити лишні шуми.

Міра якості P. Алгоритми машинного навчання повинні бути досить точні для правильного рішення задачі, й можливості покращити цю точність. Для того щоб оцінювати алгоритми машинного навчання вводиться поняття кількісної міри якості. Для кожної задачі машинного навчання, існують свої міри якості. Зазвичай вони специфічні для вирішуваної задачі.

Для задач типу класифікації вводиться поняття вірності (accuracy). Вона видає кількісну долю правильно класифікованих прикладів. Також є міра, яка називається частота помилки, вона вимірює долі прикладів, які було класифіковано невірно. Іноді частота помилок описується бінарною функцією втрат, яка приймає для прикладу значення 0, якщо він класифікований правильно, і 1, якщо неправильно[2].

Взагалі нас цікавить як алгоритм машинного навчання працює на даних. Нам потрібно знати як модель працює на нових даних, яких раніше не бачив. Це потрібно враховувати, щоб знати як усе буде працювати в реальних додатках. Тому міри якості вираховуються окремо для навчальної вибірки, та тестової. Взагалі важко знайти універсальну міру, яка буде добре працювати на всіх задачах, та яка буде відповідати очікуваної поведінки системи. Треба добре розуміння задачі, та побудови системи, щоб обрати якісну міру оцінки якості алгоритму. Для налаштування алгоритмів часто використовують підхід, який використовує міру якості для корегування гіперпараметрів моделі, для підвищення точності роботи алгоритмів, та підвищення якості результатів.

Досвід E. Алгоритми машинного навчання можна поділити на два великих класи. Алгоритми навчання з вчителем та без вчителя. Вони відрізняються тим, на якому досвіді навчається алгоритм. За досвід мається на увазі набір даних. Сукупність великої кількості прикладів, ще їх називають вимірами або точками. Кожен приклад має свої ознаки. Є два основні способи отримання даних – ручний та автоматичний. Дані, зібрані вручну, містять набагато менше помилок, але для їх збору потрібно більше часу, що в цілому робить його дорожчим. Автоматичний підхід дешевше - ви збираєте все, що можете знайти, і сподіваєтеся на хороший результат.

Для алгоритмів **з вчителем**, дані є марковані, тобто приклади помічені деякими мітками, або цільовими класами об’єктів. Для алгоритмів **без вчителя** дані нічим не марковані. Подається присутні в наборі даних ознаки, а алгоритми повинні виявити структурні властивості набору даних.

2.3 Класи алгоритмів машинного навчання

Існує великий напрямок машинного навчання, який включає в себе стандартні алгоритми, нейроні мережі, та новітні методи deep learning. **Штучний інтелект** — це галузь знань з обробки інформації, яка дозволяє комп'ютерам навчатися на власному досвіді, адаптуватися до параметрів і виконувати ті завдання, які раніше були під силу тільки людині. **Машинне навчання** — це великий підрозділ штучного інтелекту, який вивчає методи побудови та налаштування алгоритмів, здатних навчатися. **Глибоке навчання** — це сучасний метод побудови, навчання та використання нейронних мереж. По суті це нова архітектура яка використовує нейроні мережі. У рисунку 2.2 позначено загальну ієрархію області штучного інтелекту.

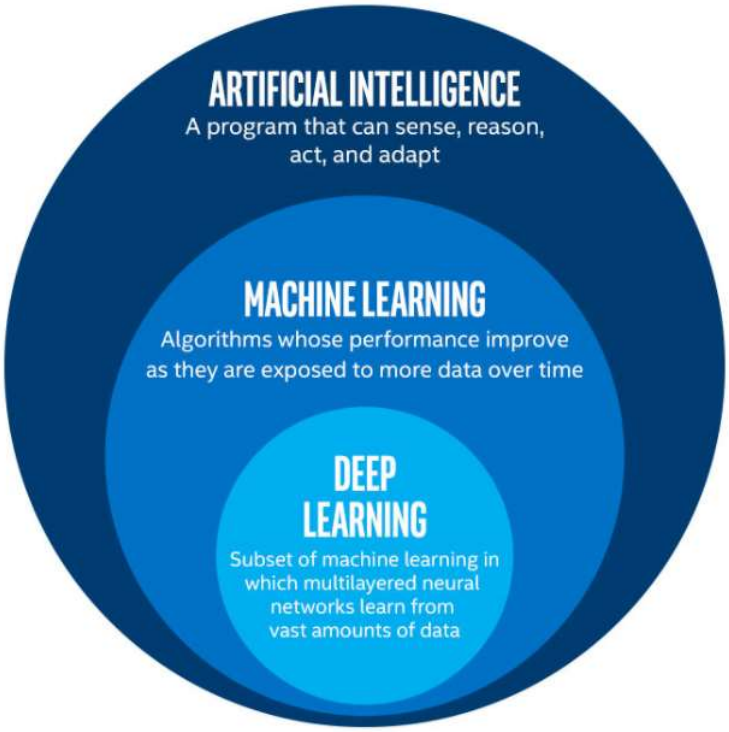


Рисунок 2.2 – Ієрархія області штучного інтелекту

Існує безліч різноманітних алгоритмів машинного навчання. Рішення вибору алгоритму завжди залишається на увазі спеціалістів. Ті чи інші алгоритми завжди дають різну точність. Деякі алгоритми краще працюють за однією задачею, а деякі з іншою. Завжди треба пам'ятати – у світі машинного навчання ніколи не буває єдиного способу вирішення проблеми. Завжди є кілька відповідних алгоритмів, і вам потрібно вибрати, який із них підходить краще. Дорожна карта алгоритмів приведено у рисунку 2.3.

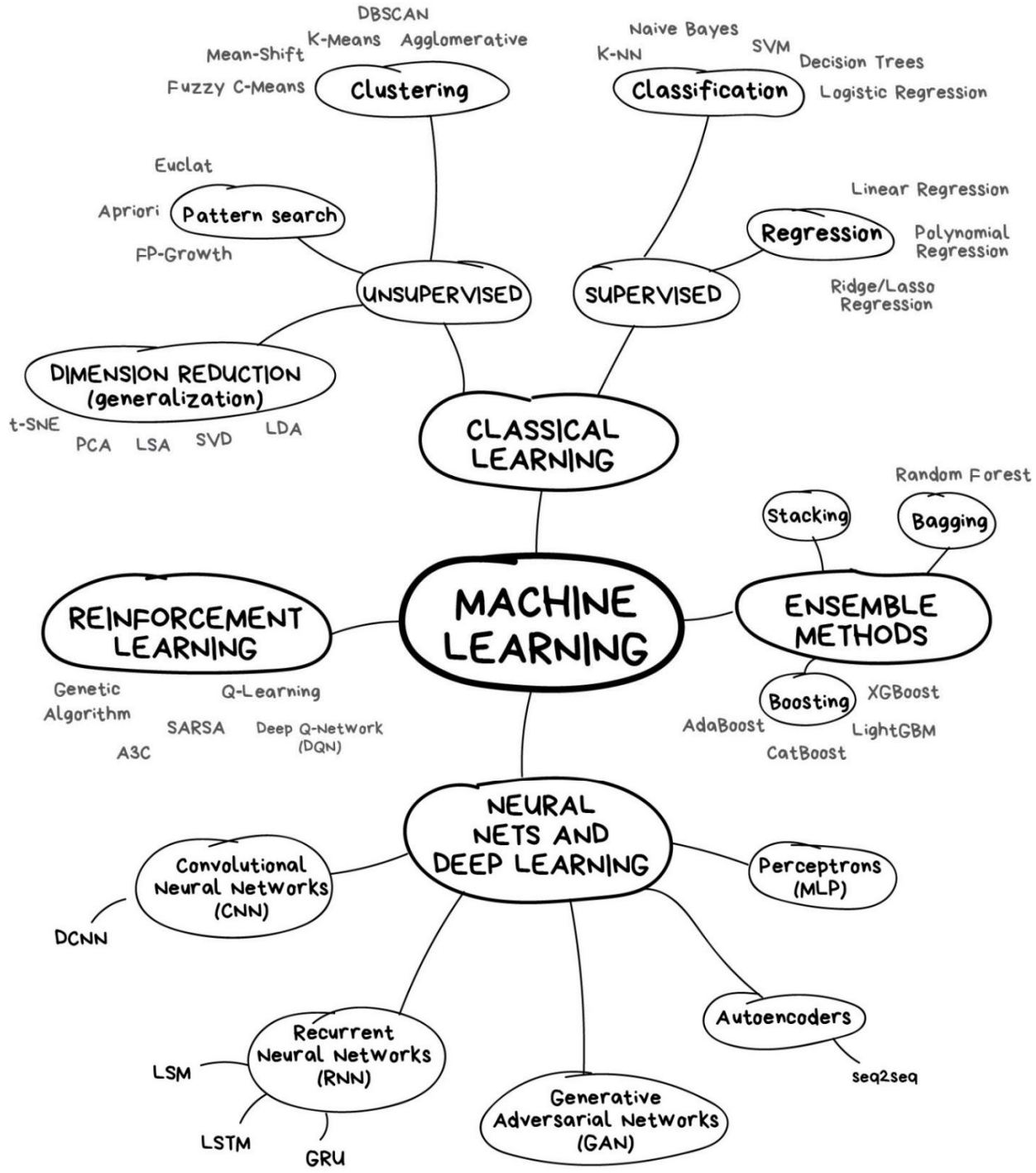


Рисунок 2.3 – Карта алгоритмів машинного навчання

Виділяють два основні класи задач машинного навчання — алгоритми **з вчителем (supervised learning)** та **без вчителя (unsupervised learning)**. Загальна структура класичного машинного навчання позначено Рисунком 2.4.

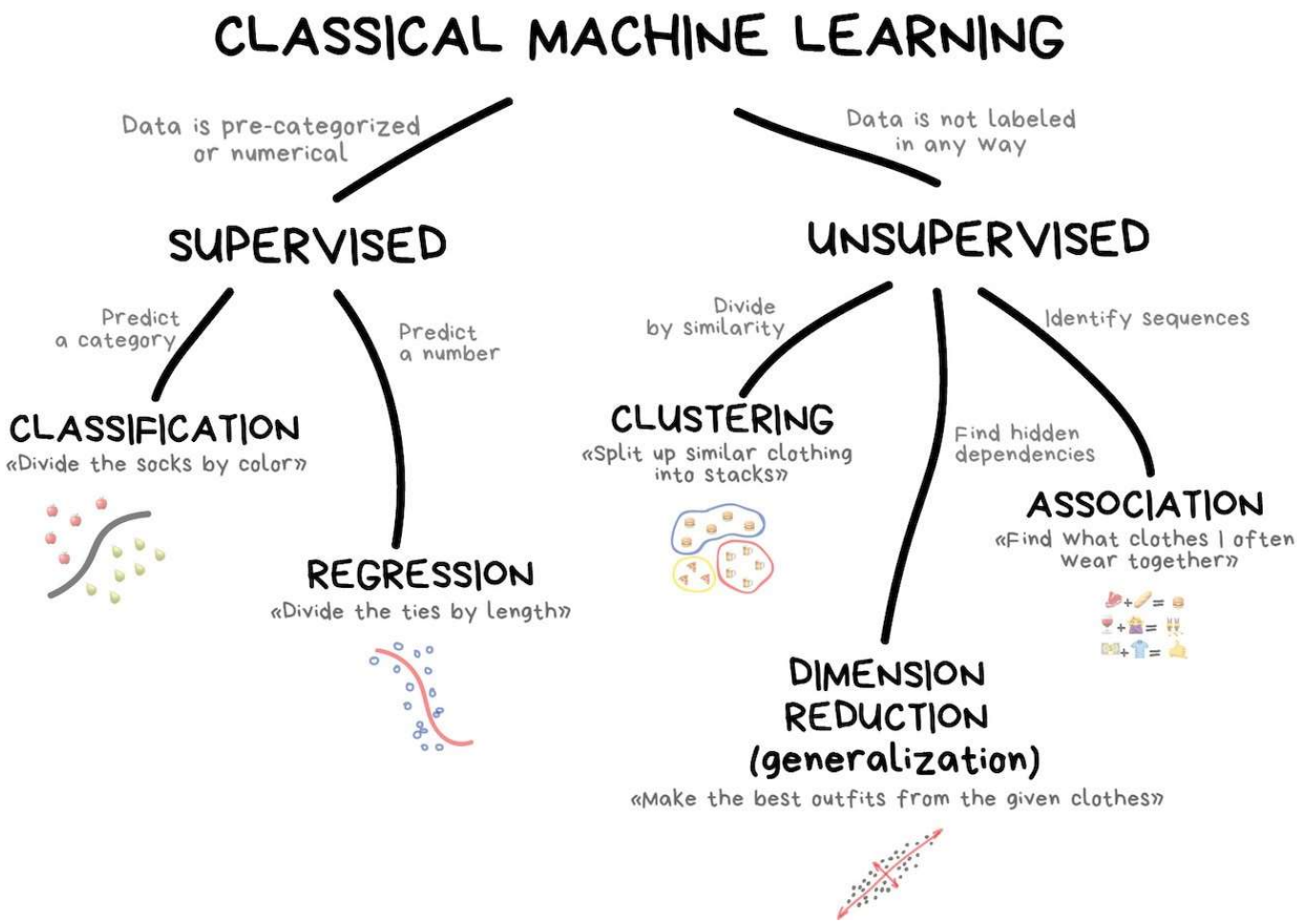


Рисунок 2.4 – Supervised learning

Алгоритми **навчання з вчителем** засновані на тому, що дані які подаються на вхід, містять в собі правильні рішення, які називаються мітками. Типовий приклад, це задача класифікації, та регресії. Загальна модель навчання з вчителем позначено Рисунком 2.5.

Алгоритм класифікації на основі вхідних даних з вірними мітками класів, повинна навчитися вірно класифікувати нові вхідні приклади.

Алгоритм регресії повинен передбачати цільове числове значення. На основі предикторів (заданий набір цільових ознак) алгоритм повинен передбачити якесь числове значення. Щоб навчити алгоритм потрібно на вхід подати багато прикладів, враховуючи набір цільових ознк. Регресія вирішує такі завдання: передбачення цін на товари за наявними даними, розрахунок суми кредитів для видачі банком людині, прогнозування вартості ціних бумаг та інше. Регресія дуже добре працює коли дані залежать від часу. Ще є таке зауваження, що алгоритм регресії, може вирішувати

задачу класифікації. Цей алгоритм називається логістичною регресією. Логістична регресія використовується для класифікації, бо вона може спрогнозувати значення, яке відповідає ймовірності належності до певного класу.

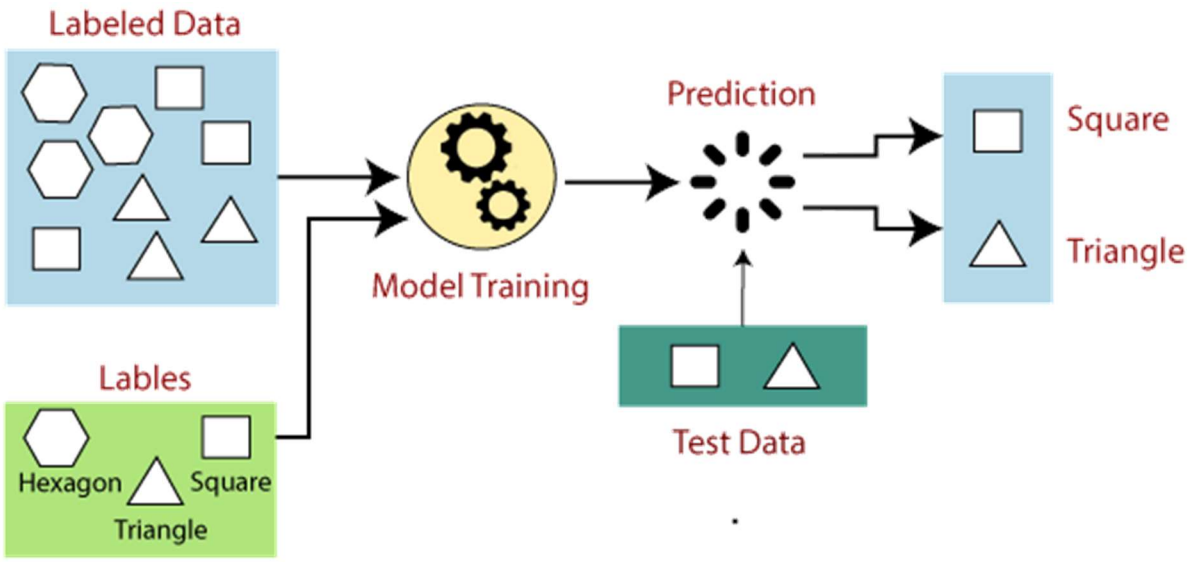


Рисунок 2.5 – Supervised learning

Найважливіші алгоритми машинного навчання з вчителем:

- k-Nearest Neighbors
- Support Vector Machines (SVMs)
- Linear Regression
- Logistic Regression
- Decision Trees and Random Forests
- Neural networks (але деякі нейронні мережі можуть бути unsupervised learning)

В навчанні **без вчителя** дані не позначені правильними відповідями. Задача стоїть в тому, що система сама вчиться без правильних відповідей, знаходити якісь приховані, цікаві закономірності. Наприклад, алгоритми кластеризації можуть виявити непересічні групи об’єктів. Цей алгоритм буде розділяти точки до кожного з кластерів, поки вони не будуть сформовані. В одному кластері будуть дуже схожі об’єкти, а в різних кластерах об’єкти будуть дуже сильно відрізнятись один від

одного. Це, наприклад, дозволяє будувати системи рекомендацій для кожної групи клієнтів магазину.

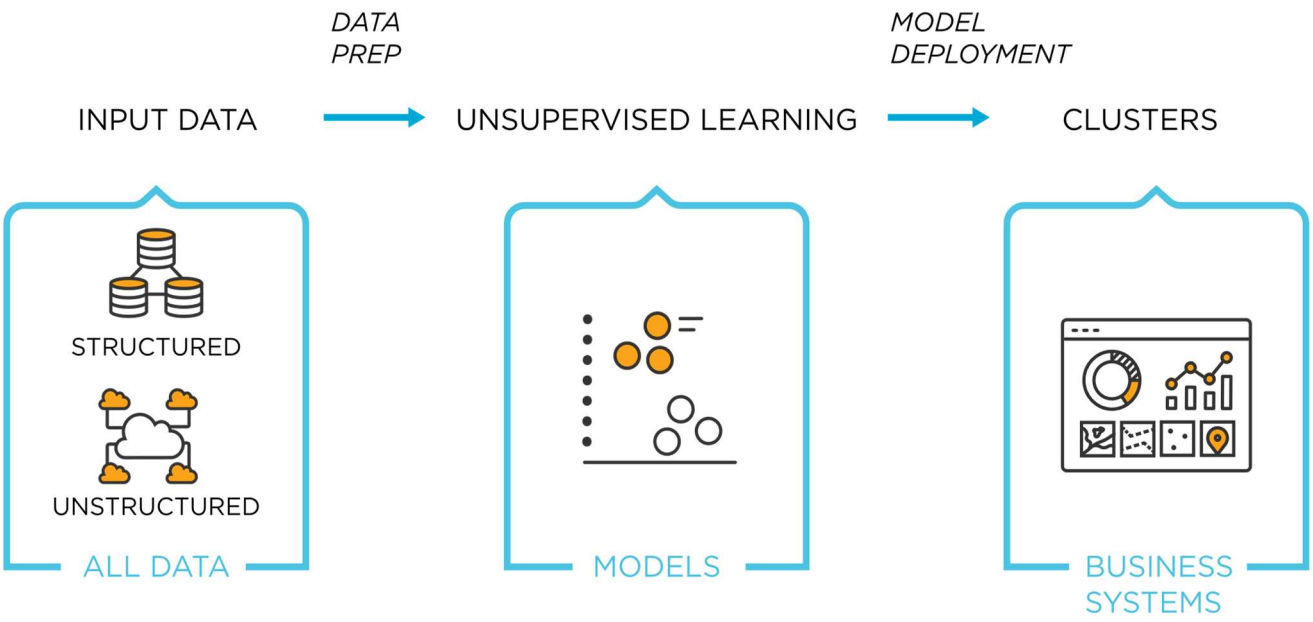


Рисунок 2.6 – Unsupervised learning

Дуже хороший приклад навчання без вчителя, це алгоритми **візуалізації**. В ці алгоритми надається багато складних, та немаркованих даних, з багатьма параметрами. Вони будують двовимірне, або тривимірне представлення цих даних зі збереженням внутрішньої структури, які можна легко побудувати на графіку. Ці алгоритми намагаються зберегти більше структури, щоб ви могли зрозуміти, як організовані дані, і, можливо, визначити непередбачувані закономірності[3]. Супутнім завданням є **зменшення розмірності**. Мета зменшення розмірності, це спростити дані так, щоб велика кількість інформації була незмінна, а розмір даних суттєво зменшився, щоб обчислювальні затрати теж зменшилися. Алгоритм об’єднує декілька схожих функцій в одну. Наприклад, пробіг автомобіля може дуже корелювати з його віком, тому алгоритм зменшення розмірності об’єднає їх в одну ознаку, яка представляє старіння автомобіля[3]. Майже завжди зменшення розмірності даних за допомогою алгоритму є хорошою ідеєю, перш ніж передати дані іншому алгоритму навчання (наприклад, алгоритму навчання з учителем). Алгоритм буде працювати набагато швидше, дані займатимуть менше місця на диску та

оперативній пам’яті, та в деяких випадках він буде працювати краще. Деякі сучасні книги по машинному навчанню рекомендують використовувати методи зменшення розмірності, як передобробку до кожної задачі. Алгоритми зменшення розмірності допомагають уникнути перенавчання, шуму, та проблеми “прокляття розмірності”.

Одна з важливих задач, це **виявлення аномалій.** Видалення викидів з набору даних, перед подачею його в інший алгоритм навчання, дасть покращити точність алгоритму. Модель навчається за допомогою звичайних екземплярів, коли вона бачить новий екземпляр, вона може визначити, чи виглядає він як нормальний, чи це аномалія.

Задача виявлення **правил асоціації** теж відноситься до задач машинного навчання без вчителя. Суть задачі, знайти відносини між атрибутами в даних. Це дозволяє використовувати ці правила у різних сферах життя, але в основному використовуються в магазинах, щоб розуміти що хоче покупець, та як йому допомогти обрати щось інше.

Найважливіші алгоритми машинного навчання без вчителя:

- Clustering
  - K-Means
  - Hierarchical Cluster Analysis (HCA)
  - Expectation Maximization
- Visualization and dimensionality reduction
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Locally-Linear Embedding (LLE)
  - t-distributed Stochastic Neighbor Embedding (t-SNE)
  - Uniform Approximation and Projection (UMAP)
- Association rule learning
  - Apriori
  - Eclat

Ще існує такий вид алгоритмів, як **навчання з підкріпленням (Reinforcement Learning).** Частина цієї системи, називається агентом, в контексті моделі. Він може

спостерігати за довкіллям, та обирати якісь дії. Якщо дії вірні, агент отримує позитивну нагороду, якщо же дія була невірною, він отримує штраф у виді негативної нагороди. За цією схемою агент повинен обрати найкращу стратегію, яка називається політикою, щоб отримати максимальну нагороду за всі часи. Політика визначає яку дію повинен зробити агент, коли він знаходиться в даній ситуації. Наприклад, сучасні роботи використовують саме навчання з підкріпленням, щоб навчитися ходити, або виконувати якісь дії. Ще дуже відома сфера навчання з підкріпленням це ігри, бо саме машина, яка навчалася з підкріпленням, отримала перемогу над діючим чемпіоном світу. Як писали видання, алгоритм проаналізував мільйони игор, а після багато грав сам проти себе параллельно навчаючись. Після цього він отримав свою найкращу політику, та за допомогою цього обиграв чемпіона світу. Навчання під час ігри було вимкнено, тому машина використовувала тільки свою, отриману під час навчання, політику. На рисунку 2.7 зображено навчання агента.

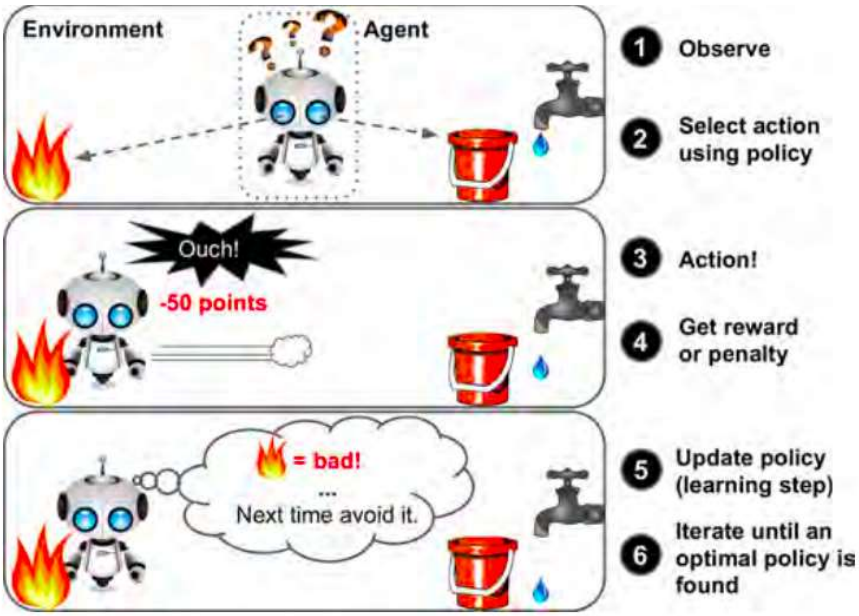


Рисунок 2.7 – Reinforcement Learning

2.4 Підготовка даних

Для того, щоб наші алгоритми добре працювали, треба зробити підготовку даних. По перше, після того, як ми вибрали, або зібрали датасет для аналізу, ми повинні видалити порожні функції, бо деякі алгоритми не вміють працювати з ними. Якщо в даних відсутні деякі значення, то ми повинні видалити атрибут, або заповнити ці місця середнім, медіанним або нульовим значенням, або видалити даний екземпляр. Наприклад, в бібліотеці Scikit-Learn для цього є визначені методи dropna(), drop(), та fillna().

По друге нам потрібно перетворити усі **текстові та категоріальні атрибути на числові**, бо більша кількість алгоритмів працюють саме з чисельними атрибутами. Scikit-Learn передбачив цю можливість, зробивши метод-кодувальник, який називається LabelEncoder. Після виконання цього методу можна використовувати чисельні результати в алгоритмі. Також можна подивитися результат зіставлення цього кодувальника, й подивитись що ми отримали. Однакові значення атрибутів в одній функції мають один клас. Одна проблема в цьому методі, що алгоритми припускають, що два сусідніх значення більше схожі, ніж два окремих. Щоб виправити цю проблему, звичайним рішенням є створення одного двійкового атрибуту для кожної категорії: один атрибут дорівнює 1, і 0 інакше. Це називається гарячим кодуванням. Scikit-Learn надає кодувальник OneHotEncoder для перетворення цілих категоріальних значень в гарячі вектори.

Обов’язково треба зробити, один з найважливіших методів — **масштабування функцій**. В машинному навчанні масштабування називають передобробкою числових ознак з метою приведення їх до загальної шкали без втрати інформативності. Алгоритми машинного навчання можуть не працювати, або мати погану точність, коли масштаби функцій різні. Наприклад, коли значення атрибуту лежить в інтервалі від 1 до 5000, а більшість значень лежить в інтервалі від 5 до 7, це може мати дуже негативний вплив на якість моделі. Існує багато різних підходів до масштабування, але найпопулярніші це **нормалізація** та **стандартизація**.

Нормалізація — метод обробки даних, при якому усі значення ознак зміщуються та приводять до одного масштабу в діапазонах [0;1], або [-1;1]. Ми вираховуємо нові значення за формулою:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

В Scikit-Learn для цього існує функція під назвою MinMaxScaler. Він має гіперпараметр feature_range, який дозволяє вам змінити діапазон. Якщо треба зробити нормалізацію для довільного інтервалу [a,b], то для цього використовується формула:

$$X' = a + \frac{X - X_{min}}{X_{max} - X_{min}}(b - a)$$

Стандартизація — метод обробки даних, при якому всі вихідні значення набору даних, незалежно від їх початкових розподілів та одиниць вимірювання, до набору значень з розподілу з нульовим середнім та стандартним відхиленням, рівним 1. В результаті формується стандартизована шкала, яка визначає місце кожного значення набору даних, вимірюючи його відхилення від середнього в одиницях стандартного відхилення:

$$z_i = \frac{x_i - \overline{X}}{\sigma_x}$$

Де x_i — значення ознаки, $\overline{X}$ — середнє значення, σ_x — стандартне відхилення ознаки, за набором даних.

2.5 Задача класифікації

Класифікація — це підрозділ машинного навчання, який вирішує задачу віднесення об’єктів, розділених деякими зв’язками на групи, які називаються класами. Важно розуміти те, що є набір даних в якому об’єкти вже розділені на класи, задача в тому, щоб на основі поділу вже відомих об’єктів, класифікувати нові дані.

Маркування об’єктів за класами називається мітками. Потрібно спрогнозувати мітки для нових вхідних даних. Класифікація відноситься до задач навчання з вчителем.

Існує декілька типів задач класифікації:

1. Бінарна класифікація
2. Мультикласова, або багатокласова класифікація
3. Класифікація з багатьма мітками

2.5.1 Бінарна класифікація

Бінарна класифікація — класифікація з бінарною міткою класу. Тобто існує лише 2 класи які позначаються як 0 або 1. Це найпростіша задача класифікації. В таких задачах вирішується належність об’єктів до одного з двох класів. Будь-яка задача класифікації може бути зведена до бінарної класифікації. За допомогою цього, можна спростити, та краще розуміти та інтерпретувати задачу. Для того щоб розробити правильно працюючу систему, використовують міри оцінки ефективності алгоритмів класифікації.

Перша міра ефективності, це оцінка за допомогою перехресної перевірки. Перехресна перевірка — це метод оцінки моделей машинного навчання шляхом навчання, декількох з них на підмножинах вхідних даних, та їх оцінок на іншій додатковій підмножині. Така перевірка використовується для виявлення перенавчання, тобто перевіряється нездатність розпізнати патерни в даних. Необхідно перевірити, що модель має відображення більшості патернів в даних, що її ефективність не падає від шумів присутніх в даних. Якщо модель нестабільна, то кажуть, що вона має низький рівень зміщення і дисперсії.

Валідація - це процес прийняття рішення про прийнятні числові результати, що визначають передбачувані взаємозв'язки між змінними як характеристики даних. Зазвичай оцінка помилки для моделі вираховується після навчання, більш відомого як оцінка точності прогнозів (accuracy). Вираховується чисельна оцінка, як помилка

в навчанні алгоритму — різниці у результатах навчання та справжніх відповідях. Це дає нам розуміння як модель працює на тренувальних даних. Однак ймовірно, що модель схильна до перенавчання або недонавчання. Проблема цього методу оцінки полягає в тому, що він не гарантує прийнятного рівня узагальнення нових даних.

Для оцінки, нам потрібен метод, який надає багато інформативних даних для навчання, та для перевірки. Крос-валідація за К-блоками (K-Fold Cross Validation) дозволяє нам перевірити якість на достатньої кількості даних. При перевірці K-Fold дані поділяються на k підмножин. Тепер утримання повторюється k разів, так що кожен з k підмножин використовується як перевірочний набор, а інші підмножини k-1 об'єднуються, щоб сформувати навчальний набір. Помилка усереднюється по всіх випробуваннях, щоб отримати узагальнену ефективність нашої моделі. Кожна точка даних потрапляє в набір для перевірки рівно один раз і потрапляє до навчального набору k-1 разів. Це значно знижує зміщення, оскільки ми використовуємо більшу частину даних для підгонки моделі, а також значно скорочуємо дисперсію, оскільки більшість даних також використовується в наборі для перевірки. Перестановка тренувального та тестового наборів також підвищує ефективність цього методу. За замовчуванням K дорівнює 5 або 10, але може приймати будь-яке інше значення.

Матриця неточностей. Матриця неточностей — це зведення результатів прогнозу завдання класифікації. Ідея полягає в тому, щоб підрахувати кількість випадків, коли екземпляри першого класу класифікуються як другий клас. Щоб обчислити матрицю неточностей, спочатку потрібно мати набір прогнозів, щоб їх можна було порівняти з вірними мітками класів.

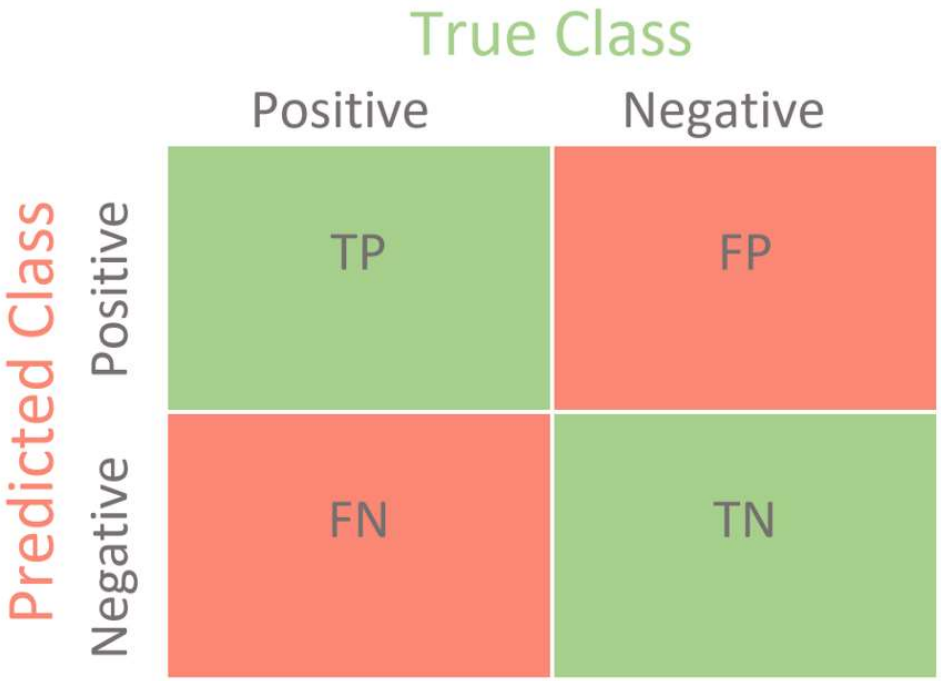


Рисунок 2.8 – Confusion matrix

По головній діагоналі матриці показуються усі правильно класифіковані екземпляри, а по побічній діагоналі усі неправильно класифіковані екземпляри. В даному випадку можна розрізнювати Negative = 0, а Positive = 1, як два бінарних класи.

True positive (TP) — результат тесту, який вказує на правильно класифікований перший клас наявної мітки, у порівнянні з вірними класами об’єктів.

True negative (TN) — результат тесту, який вказує на правильно класифікований другий клас наявної мітки, у порівнянні з вірними класами об’єктів.

False positive (FP) — результат тесту, який вказує на невірно класифікований перший клас наявної мітки, у порівнянні з вірними класами об’єктів.

False negative (FN) — результат тесту, який вказує на невірно класифікований другий клас наявної мітки, у порівнянні з вірними класами об’єктів.

Матриця неточності дає нам багато інформації, але можна використовувати більш стислий показник точності. Завжди можна подивитись на точність позитивних прогнозів класифікатора. Така міра називається точністю класифікатора, або precision measure:

$$precision = \frac{TP}{TP + FP}$$

де TP – кількість істинно позитивних результатів, а FP – кількість помилкових результатів.

Точність зазвичай використовується разом з іншим показником, який називається recall, який називають чутливістю або справжньою позитивною швидкістю. Це співвідношення позитивних випадків, які правильно виявлені класифікатором:

$$recall = \frac{TP}{TP + FN}$$

FN — кількість помилкових негативних результатів.

Ще існує комбінована міра яка поєднує в собі precision та recall в єдиний показник для порівняння класифікаторів. Оцінка F1 — це середнє гармонійне значення precision та recall. У той час як звичайне середнє розглядає всі значення однаково, середнє гармонічне надає набагато більшу вагу низьким значенням. В результаті класифікатор отримає високу оцінку F1, лише якщо precision і recall високі:

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = 2 \times \frac{precision \times recall}{precision + recall} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

Оцінка F1 надає перевагу класифікаторам, які мають подібну точність і чутливість. В деяких задачах ви здебільшого підвищуєте точність, а в інших справді спираєтесь на чутливість. Збільшення точності зменшує чутливість, і навпаки. Це називається торгівлею точності/чутливості. Але можна зробити компроміс між точністю та чутливістю алгоритму. Багато алгоритмів вираховують для кожного екземпляра оцінку на основі цільової функції, якщо оцінка перевищує деякий поріг, вона відносить екземпляр до позитивного класу, або до негативного класу. Зниження або збільшення порогу для класифікації несе такий сенс, наприклад, зниження порогу збільшує чутливість і знижує точність, або навпаки. Один зі способів обрати компроміс між precision та recall, це побудувати графік безпосередньо між ними обираючи декілька порогів для класифікації.

Ще одна з мір якості це **ROC Curve**. ROC є ще одним поширеним інструментом, який використовується з бінарними класифікаторами. Вона дуже схожа на криву точність/чутливість, але замість того, щоб відображати точність і

чутливість, крива ROC відображає відношення TPR (true positive rate) проти FPR (false positive rate). FPR – це відношення негативних випадків, які неправильно класифікуються як позитивні. TPR — відношення позитивних випадків, які правильно класифікуються.

$$TPR = \frac{TP}{TP + FN} = 1 - FNR$$

$$FPR = \frac{FP}{FP + TN} = 1 - TNR$$

$$FNR = \frac{FN}{FN + TP} = 1 - TPR$$

$$TNR = \frac{TN}{TN + FP} = 1 - FPR$$

Щоб побудувати криву ROC, спочатку потрібно обчислити TPR і FPR для різних порогових значень, використовуючи функцію roc_curve() з sklearn.

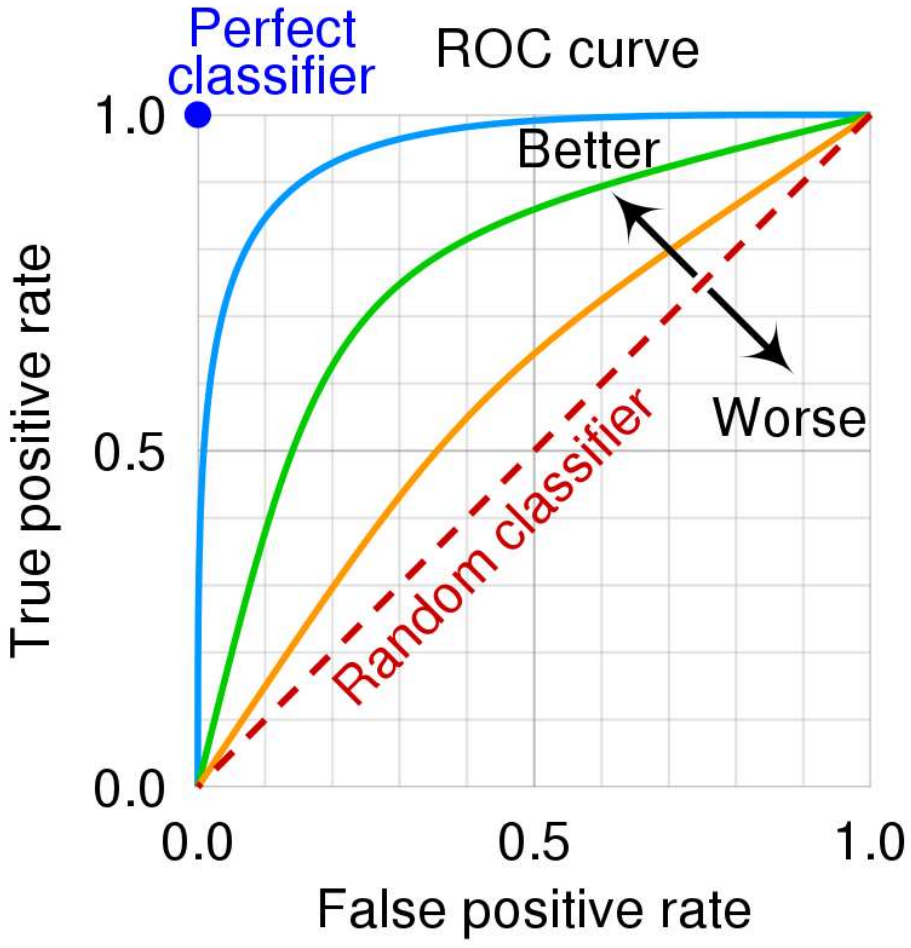


Рисунок 2.9 – ROC curve

Знову можна зробити компроміс, чим вище відкликання (TPR), тим більше помилкових спрацювань (FPR) видає класифікатор. Пунктирна лінія представляє криву ROC випадкового класифікатора. Якісний класифікатор залишається як можна далі від цієї лінії до верхнього лівого кута. Одним із способів порівняння класифікаторів є вимірювання площі під кривою (AUC). Ідеальний класифікатор матиме ROC AUC, рівний 1, тоді як випадковий класифікатор матиме ROC AUC, рівний 0,5. Крива ROC curve дуже схожа на precision/recall (крива PR), але яку краще використовувати не зовсім зрозуміло. Як правило, потрібно використовувати саме криву precision/recall, коли позитивний клас рідкісний або коли ви більше дбаєте про хибнопозитивні результати, ніж про помилкові негативи, і кривій ROC в іншому випадку[3].

2.5.2 Мультикласова класифікація

В підрозділі 2.5.1 ми розібрали бінарну класифікацію, яка працює тільки с двома класами об’єктів, мультикласова класифікація, може працює з більш ніж двома класами. Деякі алгоритми вміють працювати з багатьма класами, такі як random forest, k-nearest neighbors, логістична регресія, дерева рішень та інші. Однак існують стратегії, які дозволяють використовувати бінарні класифікатори для виконання мультикласової задачі.

Стратегія **один проти всіх (one-versus-all)**, її ідея в тому, що якщо в нас є 6 класів, треба навчити 6 бінарних класифікаторів, по одному до кожного класу, а далі щоб зрозуміти на нових даних де який клас, ми отримуємо оцінку рішення від кожного класифікатора для цього набору даних і вибираємо клас, чий класифікатор виводить найвищий бал.

Стратегія **один проти одного (one-versus-one)**, ідея в тому, щоб навчити бінарний класифікатор для кожної пари класів, наприклад 0 та 1, 0 та 2, 0 та 3 й так

далі для всієї множини міток. Якщо є N класів, нам потрібно навчити $N \times (N-1)/2$ класифікаторів[3]. Якщо в нас є 6 класів, то треба навчити 15 класифікаторів. Перевага цієї стратегії в тому, що кожен класифікатор повинен вчитися на даних, для яких визначена пара класів, які він повинен розрізняти.

Для алгоритмів, які погано масштабуються, наприклад, support vector machine рекомендують використовувати стратегію один проти одного, бо набагато швидше й краще навчити багато класифікаторів на маленьких даних, ніж декілька класифікаторів на великих. Для більшості бінарних класифікаторів надається перевага в стратегії один проти всіх.

Аналіз та підвищення точності на мультикласової класифікації. Як було сказано для бінарної класифікації, для мультикласової теж працює підвищення точності за аналізом помилок. Для початку треба проаналізувати матрицю неточності (confusion matrix), а далі налаштовувати алгоритм, щоб спробувати уникнути помилок роботи алгоритму.

2.5.3 Класифікація з багатьма мітками

До цього ми розглядали бінарну та мультиноміальну класифікацію, коли кожному екземпляру у відповідність становиться один клас. Для класифікації з багатьма мітками у кожного екземпляру може бути декілька міток класів. Система класифікації, яка виводить декілька бінарних міток називають системою з багатьма мітками. Наприклад, якщо на фото декілька людей, то класифікатор може для тих, кого він навчився розпізнавати, ставити мітки присутніх на фото людей. Якщо він навчився і впізнав на фото людей, він ставить одиниці, якщо впізнав, та 0 якщо людини нема на фото. Це виглядає для розпізнавання 3 людей, як [1,0,0], якщо на фото присутній лише одна людина, [1,1,0] якщо 2 людини, [1,1,1] якщо на фото присутні всі 3 особи. При чому, цей кортеж фіксований для кожної людини.

Існує багато способів оцінити класифікатор з багатьма мітками, і вибір правильної метрики залежить від вашого проекту. Наприклад, один підхід полягає в

тому, щоб виміряти оцінку F1 для кожної окремої мітки (або будь-якої іншої метрики бінарного класифікатора, про яку йшлося раніше), а потім просто обчислити середній бал.

2.6 Задача регресії

Задача регресії — задача машинного навчання, яка моделює вимірюванні дані, й досліджує їх властивості. Задача прогнозує неперервні дані, на основі існуючих значень. Велику роль грають саме змінні та їх закономірності присутні в даних. Дані складаються зі значень залежних змінних, та незалежних. Регресійна модель це функція незалежної змінної, й параметрів з додаванням випадкової величини. Модель налаштовується таким чином, щоб найкращим образом наближати дані. Критерієм якості наближення (цільовою функцією) зазвичай є середньоквадратична помилка: сума квадратів різниці значень моделі та залежної змінної для всіх значень незалежної змінної як аргумент:

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

Регресійний аналіз - розділ математичної статистики та машинного навчання. Передбачається, що залежна змінна є сума значень деякої моделі та випадкової величини. Щодо характеру розподілу цієї величини робляться припущення, які називають гіпотезою породження даних. Для підтвердження чи спростування цієї гіпотези виконуються статистичні тести, які називають аналізом залишків. При цьому передбачається, що незалежна змінна не містить помилок. Регресійний аналіз використовується для прогнозу, аналізу часових рядів, тестування гіпотез та виявлення прихованих взаємозв'язків у даних.

Регресія - залежність математичного очікування (наприклад, середнього значення) випадкової величини від однієї або кількох інших випадкових величин (вільних змінних), тобто $E(y|x) = f(x)$. Регресійним аналізом називається пошук

такої функції, яка описує цю залежність. Регресія може бути представлена у вигляді суми невипадкової та випадкової складових:

$$y = f(x) + \nu,$$

де f — функція регресійної залежності, а ν — адитивна випадкова величина з нульовим математичним очікуванням. Припущення характер розподілу цієї величини називається гіпотезою породження даних. Зазвичай передбачається, що величина має гауссовий розподіл з нульовим середнім і дисперсією σ_ν^2.

Задача знаходження функції регресійної моделі кількох змінних визначається в такий спосіб:

Задано вибірку $\{x_1,\dots,x_N | x \in \mathbb{R}^M\}$ — множину значень вільних змінних і множину відповідних їм значень залежних змінних $\{y_1,\dots,y_N | y \in \mathbb{R}\}$. Ці множини позначаються як, множина вхідних даних $\{(x,y)_i\}$. Задана регресійна модель — параметричне сімейство функцій $f(w,x)$, які залежить від параметрів $w \in R$ і вільних змінних x . Потрібно знайти параметри $\overline{w}$ для цієї регресійної моделі.

$$\overline{w} = \arg\max_{w \in \mathbb{R}^W} p(y|x,w,f) = p(D|w,f).$$

Функція ймовірності p залежить від гіпотези породження даних і визначається Байєсовським висновком або методом найбільшої правдоподібності.

Лінійна регресія. Алгоритм лінійної регресії:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

,де ŷ – прогнозоване значення, n – кількість ознак, x_i – i-те значення ознаки, θj є j-тим параметром моделі (включаючи член зміщення θ_0, та вагові коефіцієнти $\theta_1, \theta_2, \dots, \theta_n$). Векторизована модель лінійної регресії:

$$\tilde{y} = h_0(x) = \theta^T \cdot x$$

Навчання моделі означає встановлення її параметрів так, щоб модель найкраще відповідала навчальному набору.

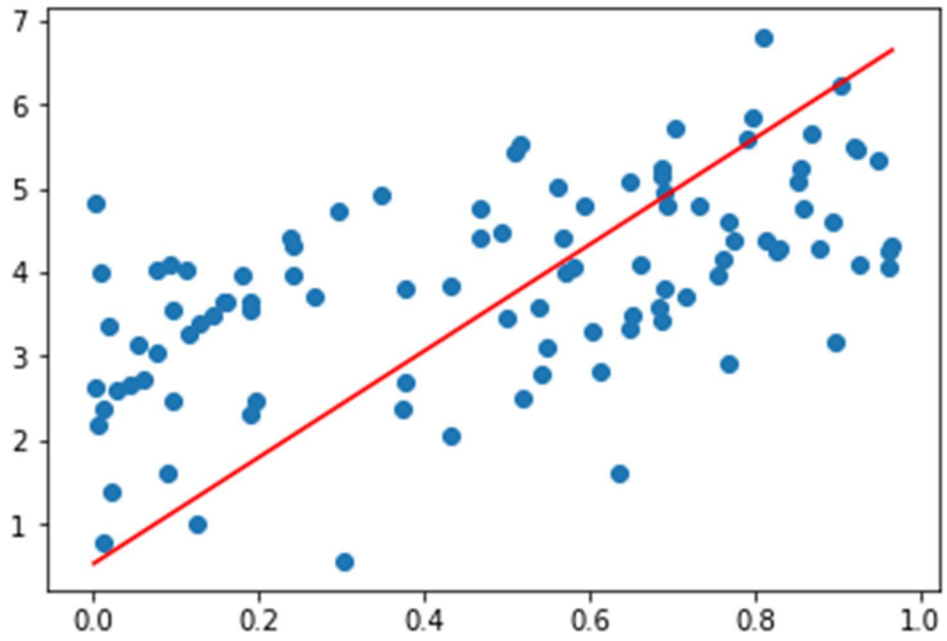


Рисунок 2.10 – Приклад лінійної регресії на штучних даних

Для цього нам спочатку потрібна міра того, наскільки добре модель відповідає навчальним даним. Найпоширенішим показником ефективності регресійної моделі є середньоквадратична помилка (RMSE). Щоб навчити модель лінійної регресії, вам потрібно знайти значення θ, яке мінімізує RMSE. На практиці простіше мінімізувати середню квадратичну помилку (MSE), ніж RMSE, і це призводить до того ж результату (оскільки значення, яке мінімізує функцію, також мінімізує її квадратний корінь). MSE гіпотези лінійної регресії h_θ на навчальній множині X обчислюється за допомогою:

$$MSE(X, h_\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(\theta^T \cdot x^{(i)} - y^{(i)}\right)^2$$

Щоб спростити позначення, часто позначаємо MSE(θ) замість MSE(X, h_θ).

Лінійна регресія потижний механізм аналізу, який використовується у різних сферах. Тому треба розуміти для реальних задач, як дані повинні бути структуровані, щоб краще використовувати модель. Є багато рекомендацій та правил, які допоможуть покращити модель. На практиці, можна використовувати ці правила як, практичні правила при використанні звичайної регресії методом найменших квадратів, найбільш поширеної реалізації лінійної регресії.

- **Лінійне припущення**. Лінійна регресія передбачає, що відносини між введенням та висновком є лінійними. Можливо, потрібно буде перетворити дані, щоб зробити лінійний зв'язок.
- **Видалити шум**. Лінійна регресія передбачає, що вхідні та вихідні змінні не зашумлені. Розгляньте можливість використання операцій очищення даних, які дозволять вам краще виявити та прояснити сигнал у ваших даних. Це важливо для вхідних змінних, та видалити викиди у вихідних змінних.
- **Видалити коллініарні змінні**. Лінійна регресія схильна до перенавчання, якщо в даних присутні сильно корельовані змінні. Навчання буде проходити на сильно корелюваних змінних, що зробить результати хорошими на локальному завдані, але для передбачень на інших задачах, будуть робитися помилки і постраждає точність. Щоб уникнути цього, треба розрахувати матрицю кореляцій та видалити ті зміні, які сильно корелюють, або зробити зменшення розмірності даних відповідними методами машинного навчання без вчителя.
- **Гауссівські розподіли**. Лінійна регресія зробить більш надійні прогнози, якщо вхідні та вихідні змінні мають гауссівський розподіл.
- **Зміна масштабу вхідних даних**: Лінійна регресія так само, як і інші способи потребує масштабування даних. Цей метод дасть точніші результати, якщо провести нормалізацію або стандартизацію даних.

2.7 Задача кластеризації

Задача кластеризації полягає в поділі множини об'єктів на групи "схожих" об'єктів, які називаються кластерами (cluster)[1]. Кластеризація спрямована на виявлення скритих залежностей в даних, та їх подальший опис. Головна перевага кластеризації в тим, що вона розбиває множину об’єктів на класи які не перетинаються.

Методи розбиття данних на кластери майже ніколи використовуються самі по собі, а тільки для отримання груп схожих об'єктів. Після визначення кластерів використовуються інші методи інтелектуального аналізу даних, щоб спробувати встановити, що означає таке розбиття, чим воно викликане[1]. Це робить цей метод універсальним, щоб зрозуміти з якими даними ми маємо справу.

Ще одна з переваг, те що можна зменшити кількість даних, якщо взяти типових представників об’єктів з кожного кластеру, та виділити їх у нову вибірку. Також дуже легко знайти викиди в даних, бо вони будуть дуже відрізнятись від еталонних об’єктів, які знаходяться ближче до центрів кластерів, які називають центроїдами.

Для розбиття на кластери, алгоритми використовують поняття подібності, або схожість до об'єктів. Це ніщо інше, як скалярна метрика. Для того, щоб будь, яка математична метрика працювала для кластеризації, вона повинна бути симетричною, невід’ємною, та відповідати рівнянню трикутника:

$$d(x,\ y) \geqslant 0$$

$$d(x,\ y) = 0 \Leftrightarrow x = y$$

$$d(x,\ y) = d(y,\ x)$$

$$d(x,\ z) \geqslant d(x,\ y) + d(y,\ z)$$

Для того, щоб обрати, яку метрику використовувати, ми повинні знати яка задача стоїть перед нами, яка природа даних, та які типи алгоритмів використовуються. Визначення для розуміння задачі[4]:

- $X \subseteq R^m$ – множина даних, підмножина m-мірного простору
- $x_i = (x_{i1}, \dots, x_{im})$ – елементи множини даних i=1…|X|
- $\bar{x} = \frac{1}{|X|}\sum_{i=1}^{|X|} x_i$ – середнє значення об’єктів множини
- $C = \frac{1}{|X|-1}\sum_{i=1}^{|X|}(x_i - \bar{x})\,(x_i - \bar{x})^t$ – коваріаційна матриця

Метрики які часто використовуються для аналізуЮ та використання в задачі кластеризації[4]:

1. **Евклідова відстань.** Найбільш поширена відстань. Це геометрична відстаннь в багатовимірному просторі:

$$d(x_i, x_j) = \sqrt{\sum_{t=1}^{m} (x_{it} - x_{jt})^2}$$

2. **Відстань по Хемінгу**. Служить метрикою об'єктів однакової розмірності. Є середньою різницею координат:

$$d(x_i, x_j) = \sum_{t=1}^{m} |x_{it} - x_{jt}|$$

3. **Відстань Чебишова**. Визначається як найбільша різниця координат між двома векторами. Цю відстань використовують, коли два об'єкти розрізняються по одній будь-якій координаті:

$$d_{\infty}(x_i, x_j) = \max_{l<t<m} |x_{it} - x_{jt}|$$

4. **Відстань Махаланобіса**. Міра відстані між векторами випадкових величин, що узагальнює поняття евклідової відстані. Погано працює, якщо матриця коваріацій рахується для всієї множини об’єктів, а не для якогось одного класу (групи об’єктів).

$$\mathrm{d}(x_i, x_j) = (x_i - x_j) C^{-1} (x_i - x_j)^T, \text{де C} - \text{коваріаційна матриця}$$

Використовуючи ці міри можна налаштувати алгоритми кластеризації для якісної роботи з даними. Виходячи зі значень цих формул, ми можемо віднести екземпляр до деякого кластеру.

Існує чіткий поділ алгоритмів кластеризації на різні типи, ієрархічні та неієрархічні. Перший тип, це **ієрархічні методи**, які працюють розбиваючи чи навпаки збираючи до кластерів вхідні дані. Цей тип кластеризації ділиться на **агломеративні** та **дивизимні** методи:

- **Агломеративні** методи кластеризації характеризуються тим, що на вхід подається множина екземплярів, кожен кластер це один екземпляр. Тобто на першому кроці 4600 екземплярів, це 4600 кластерів. Далі найближчи кластери зливаються в один більший і так проходить до того моменту, поки усі екземпляри не зійдуться в один великий кластер.

- **Дивізимні** методи кластеризації характеризуються тим, що на вхід подається один великий кластер, який містить усі екземпляри. Далі великий кластер розбивається на декілька кластерів меншого розміру, до тих пір, поки кожен екземпляр буде в 1 своєму кластері. На виході , наприклад, 4600 екземплярів, це буде 4600 кластерів.

Головні переваги цих методів, що не треба заздалегідь знати кількість кластерів. Ці методи опрацьовують усі екземпляри перебором. З цієї переваги витікає важливий мінус цих алгоритмів: не завжди нам потрібно шукати усі розбиття, а також вибір метрики в цих алгоритмах дуже сильно впливає на результат роботи алгоритмів. З плюсів можна ще зазначити, те що ієрархічні алгоритми будують покрокове дерево розбиття, яке називається дендрограмма.

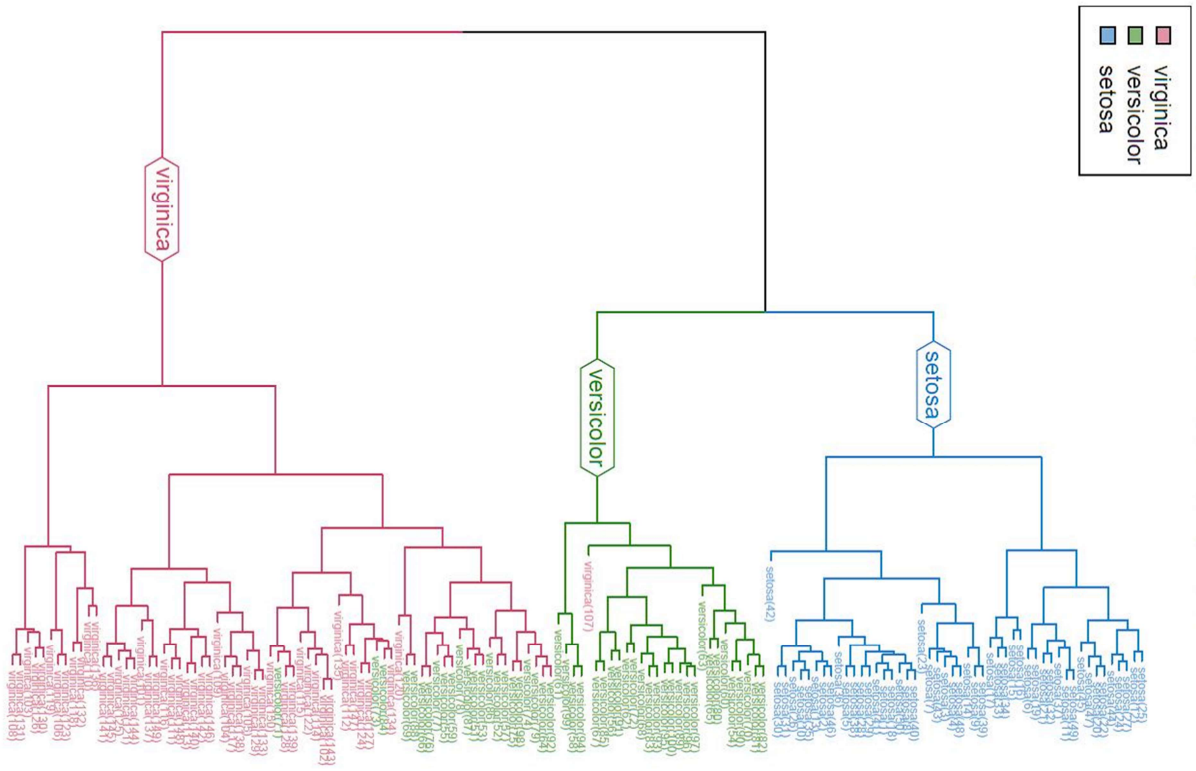


Рисунок 2.11 – Дендрограма на основі набору даних Iris

В **неієрархічних** алгоритмах знаходження рішення задачі кластеризації, полягає в групуванні даних методом знаходження екстремуму цільової функції — її мінімізації[4]. Головна відмінність цього методу від ієрархічного в тому, що він містить багато гіперпараметрів, які потрібно налаштовувати для якісної роботи

алгоритму. Гіперпараметрів існує безліч: задання положення початкових центроїдів, визначення кількості кластерів, задання порогових значень та значень зупинки, задання міри схожості та інші. Дуже важливо в цих алгоритмах задати правильну кількість кластерів, щоб алгоритм міг їх побудувати та якісно обробити. В таких алгоритмах досягається велика гнучкість кластеризації та великі можливості налаштування[4]. Використовуючи критерії якості можна автоматизувати пошук гіперпараметрів, й завдяки цьому можна зробити алгоритм добре масштабованим та якісним.

Висновки до розділу 2

У цьому розділі розглянуті методи машинного навчання. Визначено ієрархію методів штучного інтелекту, а також вкладені задачі. Визначено порядок підготовки даних перед використання алгоритмами, а також рекомендації щодо налаштування. Описані задачі класифікації, регресії та кластеризації. Визначено особливості цих задач та алгоритмів, методи оцінки, та методи налаштування та тестування. Описано переваги та недоліки різних задач. Додатково представлено деякі приклади реалізації.

3 ЗМЕНШЕННЯ РОЗМІРНОСТІ

3.1 Задача зменшення розмірності

Пониження розмірності — задача зменшення розмірності даних, а саме простору ознак. В багатьох задачах реального світу, задачі машинного навчання включають тисячі, а іноді мільйони функцій для кожного вхідного екземпляру. Це все не тільки тягне багато проблем, а також ускладнює пошук оптимального рішення. Цю проблему, ще називають прокляттям розмірності. Але є можливість звести складну задачу до оптимальної, якщо зменшити кількість функцій за допомогою методів пониження розмірності.

Зменшення розмірності призведе до втрати деякої кількості інформації, хоча ці методи прискорять навчання, але воно може також погіршити результати роботи системи. Методи пониження розмірності треба використовувати разом з алгоритмами класифікації, регресії, або кластеризації. Це призводить до того, що система стає дещо складніше, тому її складніше підтримувати. Однак задачі пониження розмірності даних видаляють викиди, шуми, та непотрібні змінні.

Крім видалення шумів, непотрібної інформації, прискорення навчання, завдяки зведенню простору ознак до 2-х та 3-х вимірювань методи зменшення розмірності дуже корисні для візуалізації даних. Це дозволяє зобразити багатовимірний навчальний набір даних на графіці і часто отримувати деякі важливі ідеї візуально виявляючи шаблони, такі як кластери[3].

Людині дуже важко представити розмірність простору більше трьох. Навіть базовий чотиривимірний гіперкуб неймовірно складно уявити в умі, не кажучи вже про 200-мірний еліпсоїд, вигнутий у 1000-мірному просторі[3]. На рисунку 3.1 вказано різні простори. Прокляття розмірності — проблема, пов'язана з експоненціальним зростанням кількості даних завдяки збільшенню розмірів просторів. При збільшенні розмірності даних виникає проблема інформативності в

даних (перенавчання, мультиколленіарність, відстань між парами об’єктів збільшується).

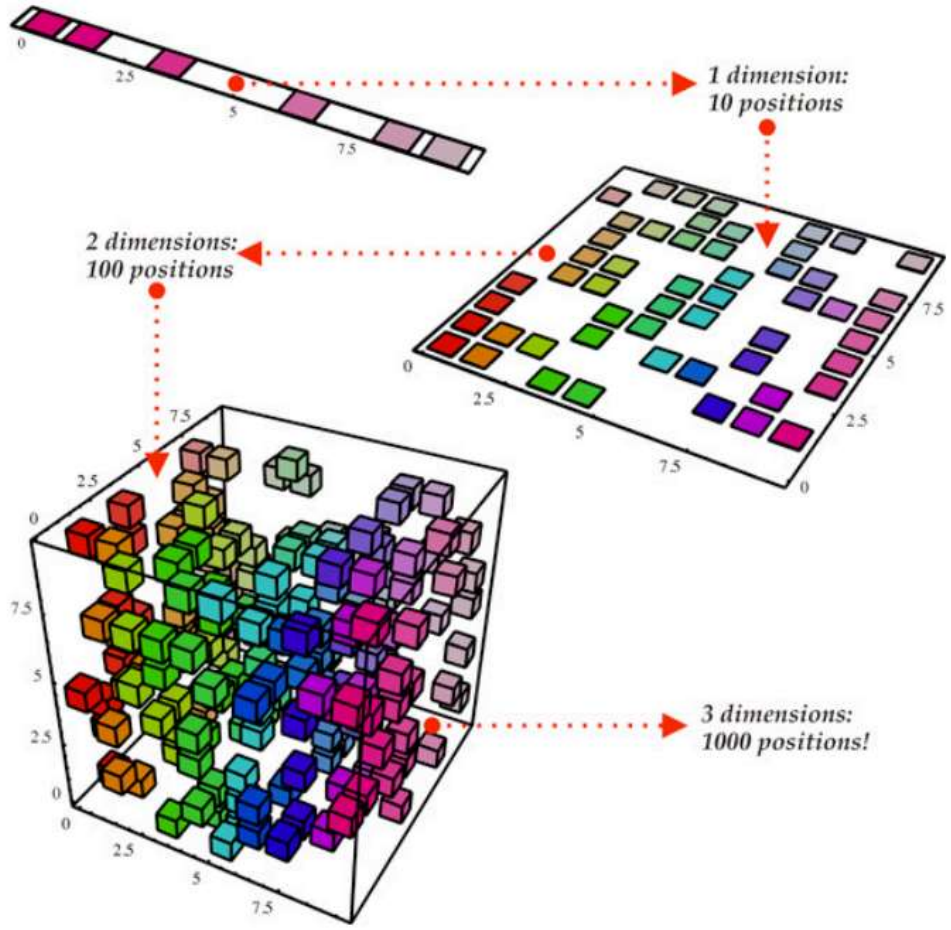


Рисунок 3.1 – Розмірність простору

Масиви даних з великими розмірами можуть бути дуже розрідженими: більшість навчальних екземплярів, будуть далеко один від одного. Звичайно, це також означає, що новий екземпляр, буде далеко від будь-якого навчального екземпляра, що робить прогнози менш надійними, ніж у нижчих вимірах, оскільки вони будуть базуватися на набагато більших екстраполяціях. Чим більший розмір тренувального набору, тим більший ризик його переобладнання. Теоретично одним із рішень прокляття розмірності може бути збільшення розміру навчального набору для досягнення достатньої щільності навчальних екземплярів. На жаль, на практиці кількість навчальних екземплярів, необхідних для досягнення заданої щільності, зростає в геометричній прогресії із збільшенням кількості вимірів. Маючи всього 100

функцій, знадобиться більше навчальних екземплярів, ніж атомів у спостережуваному всесвіті, щоб навчальні екземпляри були в середньому в межах 0,1 один від одного, припускаючи, що вони рівномірно розподілені по всіх вимірах.[3]

У більшості реальних проблем екземпляри навчання не розподіляються рівномірно по всіх вимірах. Багато функцій майже константи, тоді як інші сильно корелюють. В результаті всі навчальні екземпляри фактично лежать в межах (або близько) підпростору значно нижчого виміру простору високої вимірності[2].

Якщо розглядати, що усі точки навчальної вибірки лежать близько до площини, як на рисунку 3.2, це площина меншої вимірності (2D), у просторі більшої вимірності (3D).

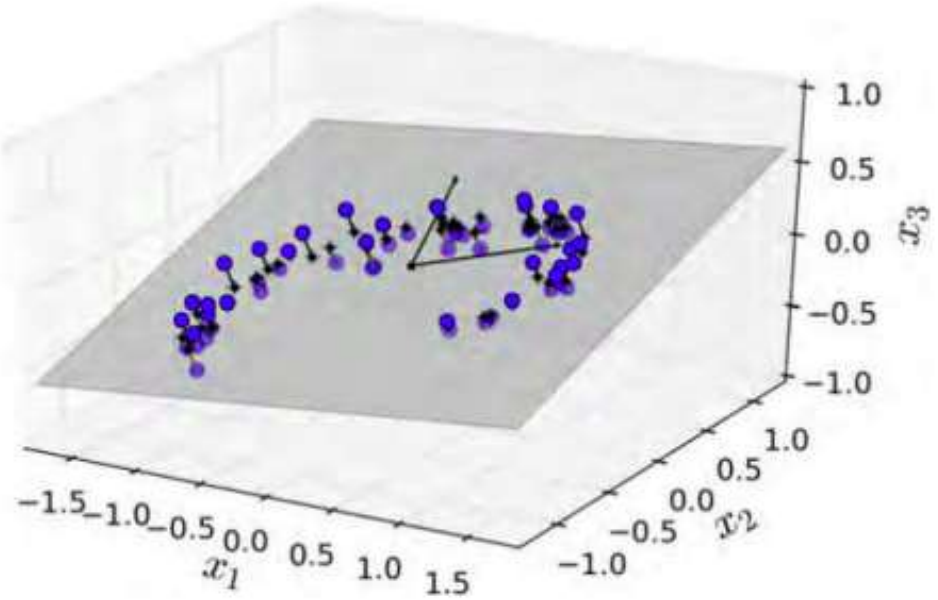


Рисунок 3.2 – Проеціювання простору 3d на 2d

Тоді ми можемо спроеціювати кожен екземпляр ортогонально на цю площину. Після проеціювання отримаємо новий набір даних у нижчій розмірності. Тоді як координати z1 та z2 мають назву – головні компоненти. Зображені ортогональні стрілочки показують напрямок векторів — головних компонентів. Новий набір даних в іншій розмірності зображено на рисунку 3.3.

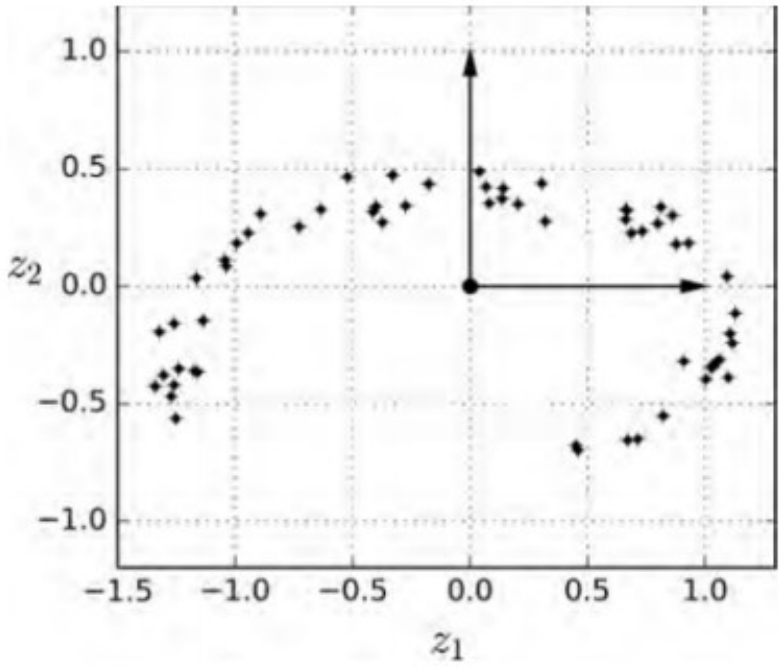


Рисунок 3.3 – Новий набір даних

Але буває таке, що простір даних є нелінійним, більш складним тому проекція може в цих задачах не працювати. В деяких випадках простір може бути сферичним, закрученим, або повернутим, мати форму циліндра тощо. В таких задачах проекції будуть працювати дуже погано, кількість інформації втраченою після обробки буде недопустимою в рамках опису задачі. Тому є другий підхід пониження розмірності даних, шляхом виділення багатовидів.

Багато алгоритмів зниження розмірності даних працюють саме моделюючи багатовид у якому лежить навчальна вибірка. Існує припущення, що більшість реальних даних великої розмірності знаходиться близько до багатовидів набагато меншої розмірності. Це називається гіпотезою багатовидів. Ця гіпотеза дуже часто підтверджується експериментальним шляхом на практиці.

Ще гіпотеза про багатовид стверджується іншим припущенням, про те, що задана задача буде простіша, якщо вона виражена в просторі нижчої вимірності багатовиду. Тобто в деяких випадках можна спростити задачі класифікації та регресії так, що межа рішення, тобто пряма яка повинна відділяти класи, може бути просто деякою простою прямою в 2d, тоді як в 3d складним гіперпростором. Але це супутнє

припущення не завжди працює, бо буває і таке, що куди простіше знайти межу рішення саме в багатовимірному просторі.

Можна сказати, що методи зменшення розмірності дуже добре працюють з прокляттям розмірності, прискорюють роботу алгоритмів, усувають деякі проблеми з даними, але не завжди ці методи покращують та поліпшують пошук рішення, але найсучасніші рукописи, та інструкції рекомендують використовувати ці методи при аналізі багатовимірних даних, в яких двадцять чи більше ознак. Ще є такі типи алгоритмів, які знаходять класи метричним підходом вираховуючи відстані на площині. Ось для таких алгоритмів машинного навчання треба завжди використовувати ці методи, бо як казали раніше, з ростом кількості вимірів відстань між екземплярами збільшується.

3.2 PCA (Principal component analysis)

Аналіз основних компонентів, або PCA, — це метод зменшення розмірності, який часто використовується для зменшення розмірності великих наборів даних шляхом перетворення великого набору змінних у менший, який, як і раніше, містить більшу частину інформації у великому наборі[5]. Аналіз головних компонент (PCA) один з найпопулярніший на сьогодні алгоритм пониження розмірності даних. Зменшення кількості змінних у наборі даних, природно, відбувається за рахунок точності, але хитрість у зменшенні розмірності полягає в тому, щоб торгувати невеликою точністю задля простоти. Оскільки менші набори даних легше досліджувати та візуалізувати, а аналіз даних стає набагато простіше та швидше для алгоритмів машинного навчання без обробки сторонніх змінних. Цей алгоритм користується підходом який використовує проекції, а саме визначає найближчу гіперплощину до даних, а потім робить проекцію усіх точок навчальної множини на цю гіперплощину.

Перш ніж спроектувати навчальну вибірку, треба визначити правильну гіперплощину. Тобто треба обрати таку вісь, для якої проекції точок на цю вісь будуть давати найбільшу дисперсію, це значить що точки повинні бути сильно віддалені один від одного на цієї умовній прямій. Це дає нам розуміння про те, що дисперсія буде зберігатись набагато краще, якщо максимізувати дисперсію проекцій. Треба обрати вісь, яка зберігає максимальну кількість дисперсії, оскільки вона, збереже більше інформації, ніж інші проекції. Інший спосіб обґрунтувати цей вибір полягає в тому, що саме вісь мінімізує середню квадратичну відстань між вихідним набором даних та його проекцією на цю вісь[5].

PCA визначає вісь, на яку припадає найбільша кількість дисперсії в навчальному наборі. Також PCA визначає другу вісь, ортогональну до першої, на яку припадає найбільша кількість дисперсії, що залишилася.

Алгоритм PCA виглядає таким чином:

1. **Стандартизація**. Треба стандартизувати діапазон неперервних змінних, щоб кожна з них вносила рівний вклад в аналіз даних. Головна причина виконання стандартизації перед PCA полягає в тому, що він дуже чутливий до варіацій вихідних змінних. Тобто, якщо є великі відмінності між діапазонами функцій, то ознаки з великими діапазонами домінують над ознаками з невеликими діапазонами, що призведе до неправильних результатів. Таким чином, перетворення даних у порівнянні масштаби може запобігти цій проблемі.
2. **Вирахування матриці коваріацій**. Мета цього кроку в тому, щоб зрозуміти як ознаки корелюють між собою, тобто зрозуміти який між ними зв’язок. Іноді деякі змінні сильно корелюють й містять надмірну інформацію, тому щоб це зрозуміти потрібно обрахувати матрицю коваріацій. В матриці коваріацій велике значення має знак коваріації:
    - якщо позитивний, то: дві змінні збільшуються або зменшуються разом (корельовано)
    - якщо негативно, то: одне збільшується, коли інше зменшується (зворотня кореляція)

3. **Власні вектори та власні значення.** Власні вектори та власні значення – це поняття лінійної алгебри, які нам потрібно вирахувати з коваріаційної матриці, щоб визначити головні компоненти даних. Основні компоненти – це нові змінні, які побудовані як лінійні комбінації чи суміші вихідних змінних. Ці комбінації виконуються в такий спосіб, щоб основні компоненти не корелювали, і більшість інформації у вихідних змінних описана у перших компонентах. Важно розуміти, що головні компоненти не інтерпретуються, бо вони побудовані як лінійні комбінації вхідних змінних. З геометричної точки зору, головні компоненти є напрямками даних, які пояснюють максимальну кількість відхилень , тобто лінії, які захоплюють більшу частину інформації про дані. Зв'язок між дисперсією та інформацією тут полягає в тому, що чим більше дисперсія, що переноситься лінією, тим більше дисперсія точок даних вздовж неї, і чим більше дисперсія вздовж лінії, тим більше інформації вона містить. Це все роблять власні вектори та власні значення, тому що власні вектори матриці коваріації насправді є напрямками осей, де спостерігається найбільша дисперсія, які ми називаємо головними компонентами. А власні значення - це просто коефіцієнти, прикріплені до власних векторів, які дають величину дисперсії, що переноситься у кожному основному компоненті. Впорядковуючи власні вектори в порядку їх власних значень, від максимального до найменшого, ви отримуєте основні компоненти як значущість.
4. **Вектори ознак.** На цьому етапі вибираємо, чи залишити всі ці компоненти або відкинути ті, які мають менше власне значення, і сформувати з матриці векторів — вектор ознак. Вектор ознак – це просто матриця, у стовпцях якої є власні вектори компонентів, які вирішили залишити. Це робить його першим кроком до зменшення розмірності, тому що, якщо ми вирішимо залишити лише p власних векторів (компонентів) з n, остаточний набір даних матиме тільки p вимірів.
5. **Перераховуємо дані.** На цьому етапі, мета полягає в тому, щоб використовувати вектор ознак, сформований з використанням власних векторів коваріаційної матриці, для переорієнтування даних з вхідних осей на головні

компоненти. Це можна зробити, помноживши транспонування вхідного набору даних на транспонування вектору ознак:

$$PCAData = FeaturesVec^T * StandartizedOriginData^T$$

3.3 T-Distributed Stochastic Neighbor Embedding (t-SNE)

T-sne– це метод неконтрольованого зменшення розмірності даних. Цей метод машинного навчання використовується для зменшення розмірності та візуалізації даних великої розмірності у просторі меншої розмірності. T-sne працює шляхом присвоєння кожній точці даних відповідне місце розташування на двох чи трьох вимірній сітці. Метою зменшення розмірності є збереження якомога більшої частини значної структури багатовимірних даних на карті низької розмірності.

Традиційні методи зменшення розмірності, такі як аналіз головних компонентів (PCA) і класичне багатовимірне масштабування (MDS), є лінійними методами, які зосереджуються на тому, щоб утримувати низьковимірні представлення різнорідних точок даних далеко один від одного. Для багатовимірних даних, які лежать поблизу низьковимірного, нелінійного багатовиду, зазвичай важливіше зберігати низьковимірні представлення дуже схожих точок даних близько один до одного, що зазвичай неможливо за допомогою лінійного відображення. t-SNE здатний дуже добре фіксувати більшу частину локальної структури багатовимірних даних, а також розкривати глобальну структуру, таку як наявність кластерів у кількох масштабах.

Стохастичне вбудовування сусідів (SNE) починається з перетворення багатовимірних евклідових відстаней між точками даних в умовні ймовірності, які представляють подібність. Подібність точки даних x_j до точки даних x_i є умовною ймовірністю $p_{j|i}$, це значить, що x_i обере x_j як свого сусіда, якщо сусіди були обрані пропорційно їх щільності за гауссом із центром в x_i. Для найближчих точок даних $p_{j|i}$ є відносно високим, тоді як для відокремлених точок даних $p_{j|i}$ буде майже

нескінченно малим (для дисперсії гаусса σ_i). Математична умовна ймовірність $p_{j|i}$ визначається як:

$$p_{j|i} = \frac{exp\left(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$$

,де σ_i – дисперсія коефіцієнта Гаусса з центром у точці даних x_i.

Для t-sne треба визначити ще p_{ij}:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

, де p_{ij} симетричне, $p_{ii} = 0, \sum p_{ij} = 1$.

Встановлюємо p(i|i) = 0, бо нам потрібно лише попарне моделювання подібності двох точок. Для низьковимірних точок y_i та y_j високовимірних точок даних x_i та x_j можна обчислити подібну умовну ймовірність, яку ми позначимо як $q_{j|i}$.

$$q_{j|i} = \frac{exp\left(-\left\|y_i - y_j\right\|^2\right)}{\sum_{k \neq i} exp(-\|y_i - y_k\|^2)}$$

Якщо точки карти y_i та y_j правильно моделюють подібність між високовимірними точками даних x_i та x_j, умовні ймовірності $p_{j|i}$ та $q_{j|i}$ будуть рівні. Завдяки цьому SNE прагне знайти представлення даних низької розмірності, яке мінімізує невідповідність між $p_{j|i}$ та $q_{j|i}$.

У t-SNE ми використовуємо t-розподіл Стьюдента з одним ступенем свободи, як розподіл з важким хвостом у низьковимірній карті[5]. Використовуючи цей розподіл, спільні ймовірності qi j визначаються як:

$$q_{ij} = \frac{\left(1 + \left\|y_i - y_j\right\|^2\right)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|y_k - y_l\|^2)^{-1}}$$

Мірою порівняння вірності є міра Кульбака-Лейблера. T-SNE мінімізує суму розбіжностей Кульбака-Лейблера для всіх точок даних за допомогою методу градієнтного спуску. Так визначається положення нових точок на карті:

$$KL(P \,||\, Q) = \sum_{i \neq j} p_{ij} log\left(\frac{p_{ij}}{q_{ij}}\right)$$

Коли ми мінімізуємо розбіжність KL, це робить q_{ij} фізично ідентичним P_{ij}, тому структура даних у багатовимірному просторі буде аналогічна структурі даних низьковимірному просторі. Грунтуючись на рівнянні дивергенції KL, якщо P_{ij} велике, то нам потрібне велике значення q_{ij} для представлення локальних точок з вищою подібністю. Якщо P_{ij} мало, нам потрібно менше значення для q_{ij}, щоб представляти локальні точки, які знаходяться далеко один від одного.

На останньому кроці треба використовувати розподіл Стьюдента, щоб визначити подібність між двома точками в низьковимірному просторі. T-SNE використовує розподіл Стьюдента-t з важким хвостом з одним ступенем свободи для обчислення подібності між двома точками в низькорозмірному просторі. T-розподіл створює ймовірнісний розподіл точок у просторі нижчих вимірів, і це допомагає зменшити проблему скупченості точок.

3.4 UMAP

Апроксимація та проекція однорідного різноманіття (UMAP) – це метод зменшення розмірності, який можна використовувати для візуалізації аналогічно t-SNE, але також і для загального нелінійного зменшення розмірності. Алгоритм ґрунтується на трьох припущеннях про дані:

1. Дані поступово розподілені на рімановому багатовиді;
2. Риманова метрика локально стала, або може бути зведена;
3. Багатовид локально складний.

Виходячи з цих припущень, можна змоделювати багатовид з нечіткою топологічною структурою. Вкладення знаходиться шляхом пошуку низьковимірної проекції даних, яка має найближчу можливу еквівалентну нечітку топологічну структуру.

При зниженні вимірності даних, будується зважений граф, з’єднуючи ребрами тільки ті об’єкти, які знаходяться поруч, і є найближчими сусідами. Множина ребер графу — це нечітка множина з функцією приналежності, вона визначається як

ймовірність існування ребра між двома вершинами[5]. Далі алгоритм створює новий граф в низьковимірному просторі, й топологічно наближає його, до початкового.

На початку алгоритму подаються вхідні дані, а заданою метрикою він будує множину найближчих сусідів. Множина найближчих сусідів $T = \{t_1, t_2, \dots, t_k\}$, а множина вхідних даних з n об'єктів $X = \{x_1, x_2, \dots, x_n\}$. Після цього для кожного об'єкту з множини, вираховується відстань до найближчого сусіда: $\rho_i = \min\limits_{t \in T} d(x_i, t)$.

Далі вираховується σ_i, ця величина нормує суму ваг для кожного об'єкта до заданого числа $\log_2 k$ з рівняння:

$$\sum_{t \in T} exp\left(-\frac{d(x_i, t) - \rho_i}{\sigma_i}\right) = \log_2 k$$

Далі алгоритм виконує побудову зваженого орієнтованого графу, в якому ребра з'єднують кожен об'єкт з його найближчим сусідом. Вага ребра x_i об'єкта до його сусіда t_j вираховується з рівняння:

$$w(x_i \to t_j) = exp\left(-\frac{d(x_i, t_j) - \rho_i}{\sigma_i}\right)$$

Може бути таке, що між вершинами є декілька ребер з різною вагою, тому вони об'єднуються в одне з рівняння, та в результаті отримаємо зважений неорієнтований граф:

$$w(x_i, x_j) = w(x_i \to x_j) + w(x_j \to x_i) - w(x_i \to x_j) \cdot w(x_j \to x_i)$$

Множина ребер E такого графа є нечіткою множиною з випадкових величин Бернуллі. Алгоритм створює новий граф у низьковимірному просторі та наближає множину його ребер до вхідного, щоб зберегти топологію. Для цього він мінімізує суму дивергенцій Кульбака-Лейблера для кожного ребра, з вихідної та нової нечітких множини[5]:

$$\sum_{e \in E} w_h(e) log \frac{w_h(e)}{w_l(e)} + \big(1 - w_h(e)\big) log \left(\frac{1 - w_h(e)}{1 - w_l(e)}\right) \to \min_{w_l} \dots$$

,де $w_h(e)$ функція належності з багатовимірного простору, а $w_l(e)$ в низьковимірному просторі відповідно.

Задача вирішується за допомогою стохастичного градієнтного спуску. Отримана множина ребер визначає нове розташування об'єктів і, відповідно, низькорвимірне відображення вхідного простору[5].

Цей метод дуже добре працює з різними даними. В інструкції до бібліотеки цього методу, присутні багато різних задач, від розпізнавання зображень, до кластеризації та класифікації зірок.

Результати UMAP дуже якісні, та схожі на алгоритм t-sne, який вважається одним з найкращих, але працює UMAP набагато швидше.

3.5 Багатовимірне масштабування (MDS)

Багатовимірне масштабування — це метод нелінійного зниження розмірності даних, та візуалізації. Алгоритм будує відображення на основі відстаней на абстрактну декартову площину.

MDS застосовується при візуалізації інформації, для відображення інформації, що міститься в матриці відстані. Це форма нелінійного зменшення розмірності. Виходячи з матриці відстаней між кожною парою об’єктів в наборі даних, та кількості вимірів, MDS розміщує кожний об’єкт, в простір меншої вимірності таким чином, щоб відстані між об’єктами якомога краще зберігалися. MDS прагне апроксимувати уявлення нижчої розмірності, мінімізуючи функцію втрат, а загальні форми функцій втрат називаються напругою в метричному MDS і деформацією в класичній MDS[6]. Діаграма розсіювання дозволяє нам візуально оцінити відстань між кожною парою точок. Фактичну відстань між двома точками, можна обчислити чисельно за допомогою формули евклідової відстані:

$$d_{ij} = \sqrt{\sum_{k=1}^{p} \left(x_{ik} - x_{jk}\right)^2}$$

Ця формула передбачає необмежену кількість вимірів, формула обчислює відстань у p-вимірному просторі, де p може бути більше двох.

Методи MDS виявилися корисними, оскільки часто виникають обставини, коли фактичні координати об'єктів невідомі, але доступна матриця відстаней певного типу. Особливо це стосується психології, де люди не можуть намалювати загальну картину групи об'єктів, але вони можуть виразити, наскільки різні окремі пари об'єктів.

MDS моделює відстані, тому вибір повинен бути, який ґрунтується на відмінностях між фактичними відстанями та їх прогнозними значеннями. Така міра називається напругою і розраховується як значення:

$$stress_D(x_1, x_2, \dots, x_n) = \left(\frac{\sum_{ij} \left(b_{ij} - x_i^T x_j \right)^2}{\sum_{ij} \left(b_{ij} \right)^2} \right)^{\frac{1}{2}}$$

, x_i вектори в N вимірному просторі.

Алгоритм **класичного масштабування** такий:

1. Побудувати та налаштувати матрицю близькості;
2. Зробити подвійне центрування $B = \frac{1}{2} C D^{(2)}$, де $C = I - \frac{1}{n} J_n$, C – матриця центрування;
3. Знайти найбільші власні значення m та власні вектори матриці B;
4. $X = E_m \mu_m^{1/2}$, де E_m матриця власних векторів, μ_m діагональна матриця з m власних значень матриці B;

Метричне масштабування — це підрозділ класичної MDS, який узагальнює процедуру оптимізації на множині функцій втрат і вхідних матриць відомих відстаней з вагами[5]. Функція втрат у цьому контексті називається стресом, який часто мінімізується за допомогою процедури, яка називається мажоризацією стресу. Метричний MDS мінімізує функцію вартості, яка є залишковою сумою квадратів[5]:

$$\boldsymbol{Strain_d} = \left(\sum_{\boldsymbol{i \neq j = 1,\dots,N}} \left(\boldsymbol{d_{ij}} - \left\| \boldsymbol{x_i} - \boldsymbol{x_j} \right\| \right)^2 \right)^{\frac{1}{2}}$$

Неметричне масштабування. Неметрична MDS знаходить як непараметричний монотонний зв'язок між відмінностями в матриці елемент-елемент та евклідовими відстанями між елементами, так і положення кожного елемента в низькорвимірному просторі[3]. Взаємозв'язок зазвичай визначається за допомогою ізотонічної регресії: нехай x позначимо вектор близькості, $f(x)$ монотонне перетворення x, а d точкові відстані,потім необхідно знайти координати, які мінімізують так звану напругу[6]:

$$Strain = \sqrt{\frac{\sum(f(x) - d)^2}{\sum d^2}}$$

Існує кілька варіантів цієї функції втрат. Програми MDS автоматично мінімізують стрес, щоб отримати рішення MDS, ядро неметричного алгоритму MDS – це подвійний процес оптимізації, тому спочатку потрібно знайти оптимальне монотонне перетворення близькості[6]. По-друге, точки конфігурації повинні бути розташовані оптимальним чином, щоб їх відстані як найточніше відповідали масштабованим наближенням. Основні кроки неметричного алгоритму MDS[6]:

1. Обчисліть відстані d між точками;
2. Знайдіть оптимальне монотонне перетворення близькості, щоб отримати оптимально масштабовані дані $f(x)$;
3. Мінімізуйте напругу між оптимально масштабованими даними та відстанями, вирахувавши нову конфігурацію точок;
4. Порівняйте напругу з деяким критерієм. Якщо напруга досить мала, вийдіть із алгоритму, інакше поверніться до 2;

MDS намагається змоделювати дані подібності чи відмінності як відстані у геометричних просторах, а дані можуть бути рейтингами подібності між об'єктами, частотами взаємодії молекул або індексами торгівлі між країнами[7].

Висновки до розділу 3

В третьому розділі було розглянуто задачу зниження розмірності даних, а також продемонстровано основні популярні алгоритми. Було описано відмінності, та покроково розкрито усі подробиці алгоритмів. Було описано переваги та недоліки технології зменшення розмірності даних. Визначено такі проблеми з даними, за яких використання алгоритмів буде найкращим рішенням.

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Метою дослідження було дослідження та практичне тестування алгоритмів зниження розмірності даних на реальних даних з використанням самих алгоритмів, та класифікатору KNN. Дослідження результатів роботи алгоритмів зменшення розмірності дуже важко оцінити, бо вони перетворюють вхідні дані, в дані які належать менш вимірному простору, тому було прийнято рішення досліджувати алгоритми на класифікаторі KNN. Класифікатор був обраний через критерій універсальності, масштабованості, та тим, щоб саме KNN є метричним алгоритмом, тому сильно схильний до проблем пов’язаних з великими вимірами даних.

В дипломній роботі розроблювався класифікатор спаму на основі набору даних Spambase UCI, з використанням методів зменшення розмірності даних. Концепція «спаму» різноманітна: реклама продуктів / веб-сайтів, схеми швидкого заробітку, листи с погрозами та інше. Набір даних містить 56 атрибутів частоти слів в електроному листі, 57 атрибут, це відображення типу листа на спам, чи не спам. Взагалі в датасеті 4601 екземплярів.

Для аналізу використовувалась середа програмування Jupyter notebook, мова програмування Python, з бібліотеками sklearn, numpy, plotly, seaborn, UMAP. Для класифікації було обрано алгоритм KNN, як метричний і швидкий метод, з підтримкою мультикласової класифікації. Для оцінки результатів використовувалися confusion matrix, precision, recall, accuracy.

Дані представлені, як матриця (4601,57), тобто є 57 функцій які підлягають аналізу. Для того, щоб навчити класифікатор, по перше робився аналіз даних, та предметної області, по друге видалялися порожні місця, та пропуски в даних заповнювалися середнім значенням за для того, щоб уникнути проблеми пов’язані з подальшою обробкою. Дані стандартизувалися, та розбивались на навчальну вибірку та тестову вибірку в співвідношенні 70 на 30.

Алгоритми, які використовувалися для аналізу зменшення розмірності:

1. PCA (Principal component analysis)
2. T-sne (t-distributed Stochastic Neighbor Embedding)
3. UMAP (Uniform Approximation and Projection)
4. MDS (Multidimensional scaling)

Набір даних представлений на рисунку 4.1:

	word_freq_make	word_freq_address	word_freq_all	word_freq_3d	word_freq_our	word_freq_over	word_freq_remove	word_freq_internet	word_freq_order
0	0.00	0.64	0.64	0.0	0.32	0.00	0.00	0.00	0.00
1	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07	0.00
2	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12	0.64
3	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31
4	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31
5	0.00	0.00	0.00	0.0	1.85	0.00	0.00	1.85	0.00
6	0.00	0.00	0.00	0.0	1.92	0.00	0.00	0.00	0.00
7	0.00	0.00	0.00	0.0	1.88	0.00	0.00	1.88	0.00
8	0.15	0.00	0.46	0.0	0.61	0.00	0.30	0.00	0.92
9	0.06	0.12	0.77	0.0	0.19	0.32	0.38	0.00	0.06
10	0.00	0.00	0.00	0.0	0.00	0.00	0.96	0.00	0.00
11	0.00	0.00	0.25	0.0	0.38	0.25	0.25	0.00	0.00
12	0.00	0.69	0.34	0.0	0.34	0.00	0.00	0.00	0.00
13	0.00	0.00	0.00	0.0	0.90	0.00	0.90	0.00	0.00
14	0.00	0.00	1.42	0.0	0.71	0.35	0.00	0.35	0.00
15	0.00	0.42	0.42	0.0	1.27	0.00	0.42	0.00	0.00
16	0.00	0.00	0.00	0.0	0.94	0.00	0.00	0.00	0.00
17	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00
18	0.00	0.00	0.55	0.0	1.11	0.00	0.18	0.00	0.00
19	0.00	0.63	0.00	0.0	1.59	0.31	0.00	0.00	0.31

Рисунок 4.1 – Набір даних

Далі було проведено опис усіх даних за допомогою data.describe(), тобто описової статистики. Описова статистика (метод) включає ті методи, що підсумовують центральну тенденцію, дисперсію та форму розподілу набору даних, за винятком NaN значень. Аналізує числові та об'єктні серії, а також DataFrame стовпці змішаних типів даних як показано на рисунку 4.2.

	count	mean	std	min	25%	50%	75%	max
word_freq_make	4601.0	0.104553	0.305358	0.0	0.000	0.000	0.000	4.540
word_freq_address	4601.0	0.213015	1.290575	0.0	0.000	0.000	0.000	14.280
word_freq_all	4601.0	0.280656	0.504143	0.0	0.000	0.000	0.420	5.100
word_freq_3d	4601.0	0.065425	1.395151	0.0	0.000	0.000	0.000	42.810
word_freq_our	4601.0	0.312223	0.672513	0.0	0.000	0.000	0.380	10.000
word_freq_over	4601.0	0.095901	0.273824	0.0	0.000	0.000	0.000	5.880
word_freq_remove	4601.0	0.114208	0.391441	0.0	0.000	0.000	0.000	7.270
word_freq_internet	4601.0	0.105295	0.401071	0.0	0.000	0.000	0.000	11.110
word_freq_order	4601.0	0.090067	0.278616	0.0	0.000	0.000	0.000	5.260
word_freq_mail	4601.0	0.239413	0.644755	0.0	0.000	0.000	0.160	18.180
word_freq_receive	4601.0	0.059824	0.201545	0.0	0.000	0.000	0.000	2.610

Рисунок 4.2 – Описова статистика

Далі було проведено опис класів присутніх в датасеті, на рисунку 4.3.

Рисунок 4.3 – Класи задачі

Для того, щоб порівняти як працює класифікатор на стандартизованих вхідних даних, було вирішено зробити попередній аналіз результатів knn без використання налаштування та алгоритмів пониження розмірності. За допомогою методу train_test_split() розділили вибірку в співвідношенні 70 на 30.

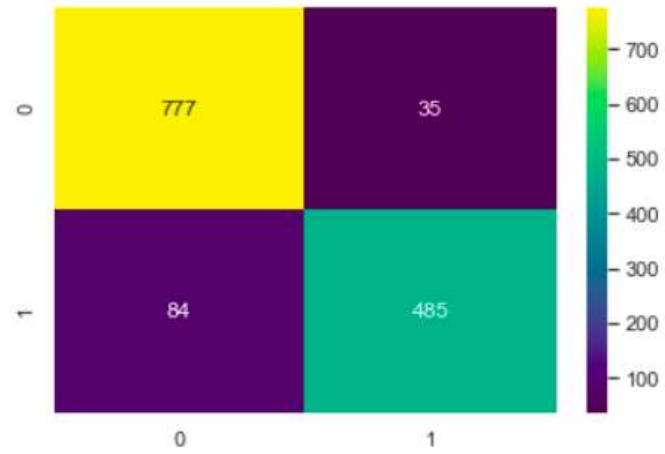
Алгоритм зі стандартними параметрами спрацював досить непогано він отримав 89.7% точності на тестовому наборі за 0.18 секунд, але зрозуміло, що це не кращий результат, бо кількість функцій дуже велика.. Більш докладно, оцінити роботу можна за допомогою confusion matrix на рисунку 4.4.

	precision	recall	f1-score	support
0	0.91	0.92	0.91	812
1	0.89	0.86	0.87	569
accuracy			0.90	1381
macro avg	0.90	0.89	0.89	1381
weighted avg	0.90	0.90	0.90	1381

Рисунок 4.4 – confusion matrix KNN без налаштування

Досягається компроміс між precision та recall, але алгоритм все ж дає хибні спрацювання, але результат досить не поганий. Метрика f1 це середнє гармонійне між precision та recall, саме ця міра показує наскільки добре алгоритм працює на даних.

Далі аналізується налаштований KNN на рисунку 4.5. Як видно з рисунку, алгоритм дав кращі результати, ніж KNN без налаштування. Підбір параметрів було проведено в GridSearchCV(), це сітка пошуку параметрів. В результаті отримали хороші прогнози, досягли результату між precision та recall, а також метрика f1 значно покращилась. Параметри підібрано автоматизовано. Параметри оцінювача, який використовується для застосування цих методів, оптимізуються за допомогою перехресного пошуку по сітці параметрів.



	precision	recall	f1-score	support
0	0.90	0.96	0.93	812
1	0.93	0.85	0.89	569
accuracy			0.91	1381
macro avg	0.92	0.90	0.91	1381
weighted avg	0.91	0.91	0.91	1381

Рисунок 4.5 – confusion matrix KNN з налаштуванням

Далі переходимо до алгоритмів зниження розмірності, а саме PCA. Для якості дослідження використаємо налаштований KNN для аналізу усіх методів пониження розмірності даних. Результат класифікації за допомогою PCA на рисунку 4.6.

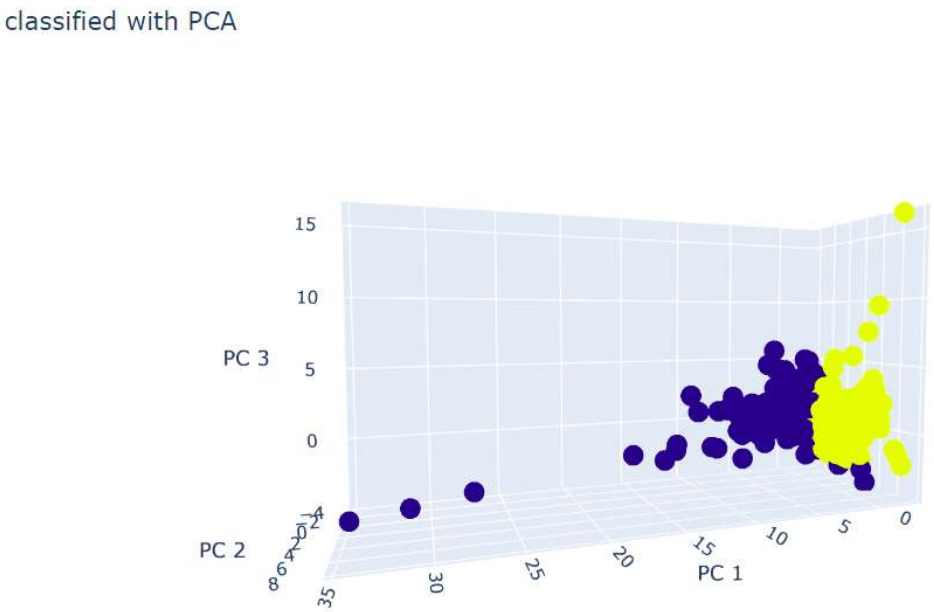
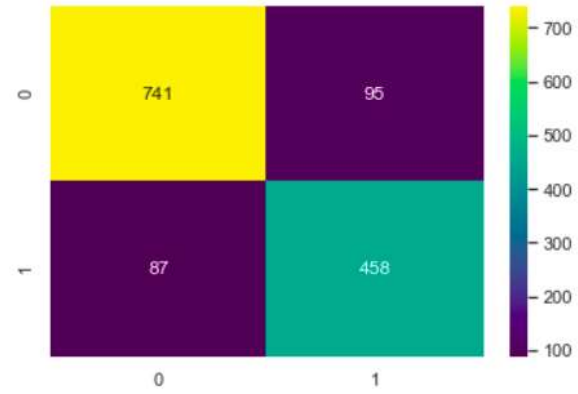


Рисунок 4.6 – Класифіковані дані з PCA

З рисунку видно, що результатом PCA є 3 головні компоненти, та проекції точок на ці компоненти в бік максимізації дисперсії на компонентах. Далі робимо класифікацію на основі налаштованого KNN та нового набору даних отриманого з алгоритму PCA. Далі проаналізуємо confusion matrix класифікатора на основі PCA.



	precision	recall	f1-score	support
0	0.89	0.89	0.89	836
1	0.83	0.84	0.83	545
accuracy			0.87	1381
macro avg	0.86	0.86	0.86	1381
weighted avg	0.87	0.87	0.87	1381

Рисунок 4.7 – Класифікація з PCA

Видно, що похибка на даних велика, бо алгоритму треба описати більшість дисперсії, але обмежилися трьома компонентами за для кращого аналізу. Взагалі алгоритм спрацював не дуже добре порівняно навіть зі стандартним методом, але цей алгоритм найшвидший серед усіх. Він виконав задачу за 0,16 секунд.

T-sne метод машинного навчання використовується для зменшення розмірності та візуалізації даних великої розмірності у просторі меншої розмірності. T-sne працює шляхом присвоєння кожній точці даних відповідне місце розташування на двох чи трьох вимірній сітці. Метою зменшення розмірності є збереження якомога більшої частини значної структури багатовимірних даних на карті низької розмірності. Класифікуємо дані за допомогою t-sne, отримаємо результат класифікації згідно рисунку 4.8. Видно що t-sne виділив багатовид з вхідного набору даних. Ознак залишилося 3 штуки, тому алгоритм спрацював доволі добре, це можна побачити з візуалізації.

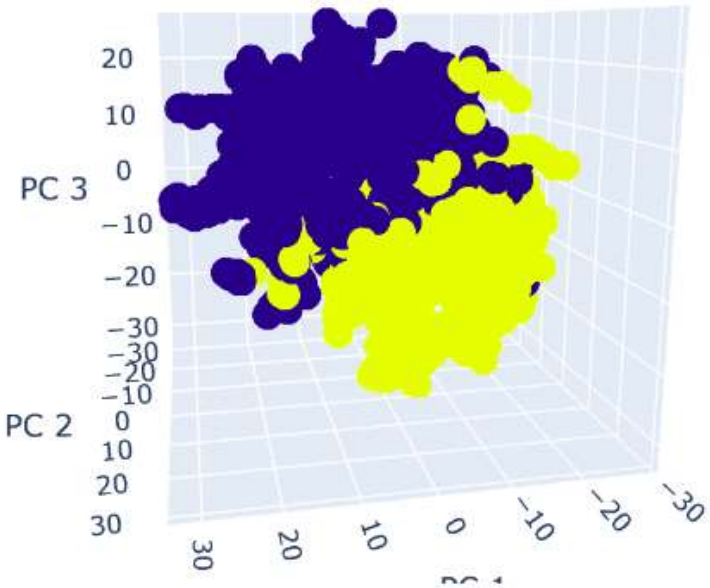
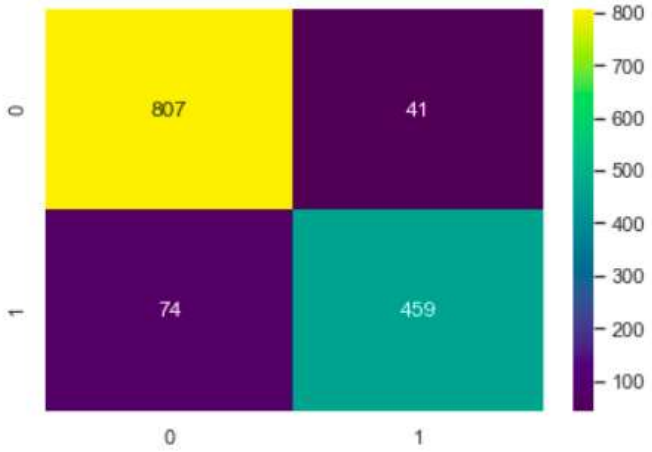


Рисунок 4.8 – Класифіковані дані з T-sne

Далі проаналізуємо наші результати класифікації за confusion matrix.



	precision	recall	f1-score	support
0	0.92	0.95	0.93	848
1	0.92	0.86	0.89	533
accuracy			0.92	1381
macro avg	0.92	0.91	0.91	1381
weighted avg	0.92	0.92	0.92	1381

Рисунок 4.9 – confusion matrix з T-sne

Отримали найкращі результат. Точність класифікатор 92%, час спрацювань 118 секунд, що дуже багато як для обробки даних, але все ж ми покращили результат.

UMAP це метод зменшення розмірності, який можна використовувати для візуалізації аналогічно t-SNE, але також і для загального нелінійного зменшення розмірності. Вкладення знаходиться шляхом пошуку низьковимірної проекції даних, яка має найближчу можливу еквівалентну нечітку топологічну структуру.

Цей алгоритм який вийшов в 2018 році, його характеризували, як швидший аналог t-sne. Результати візуалізації класифікації зображено на рисунку 4.10.

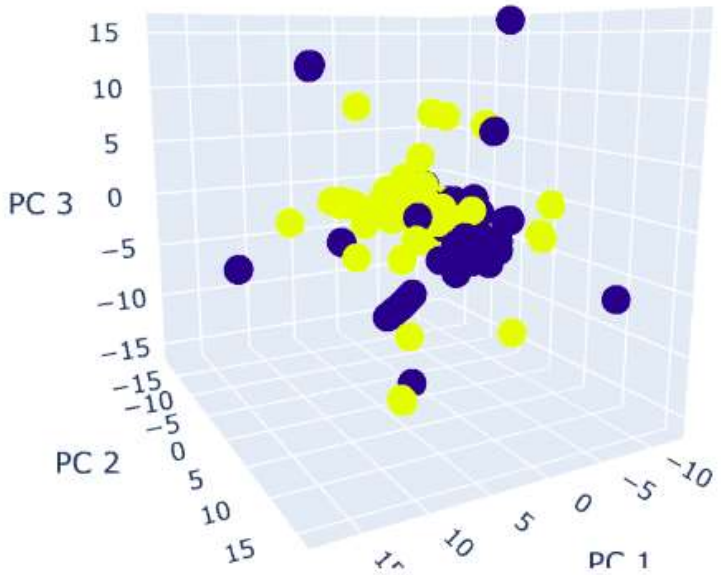
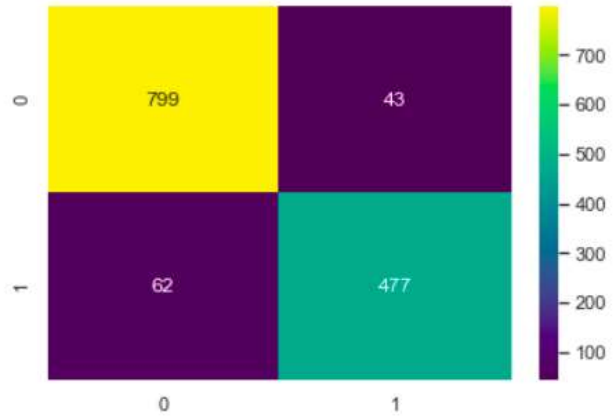


Рисунок 4.10 – Класифіковані дані з UMAP



	precision	recall	f1-score	support
0	0.93	0.95	0.94	842
1	0.92	0.88	0.90	539
accuracy			0.92	1381
macro avg	0.92	0.92	0.92	1381
weighted avg	0.92	0.92	0.92	1381

Рисунок 4.11 – confusion matrix UMAP

Отримали ще кращі результати за t-sne, майже 93% за 20 секунд. Це кращий результат який ми отримували. Це один з найкращих алгоритмів зменшення розмірності даних.

Наступний алгоритм це MDS. На рисунку 4.12 зображено класифіковані дані в просторі меншої вимірності, саме в трьох вимірах:

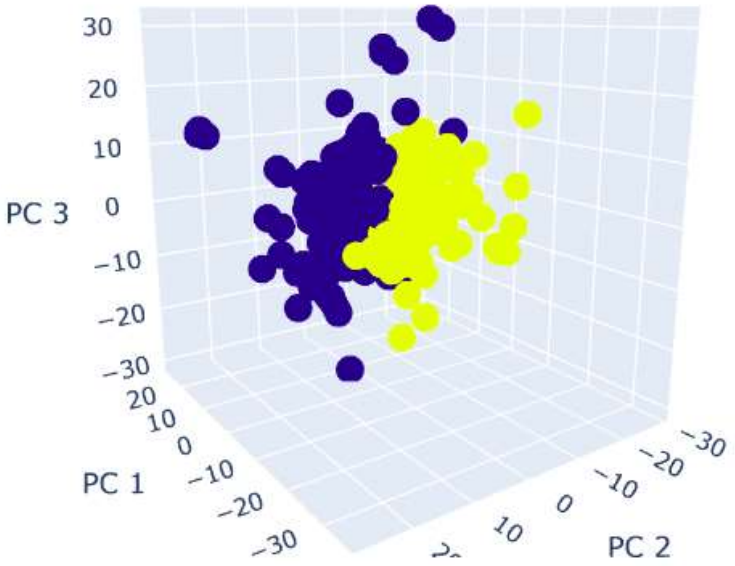
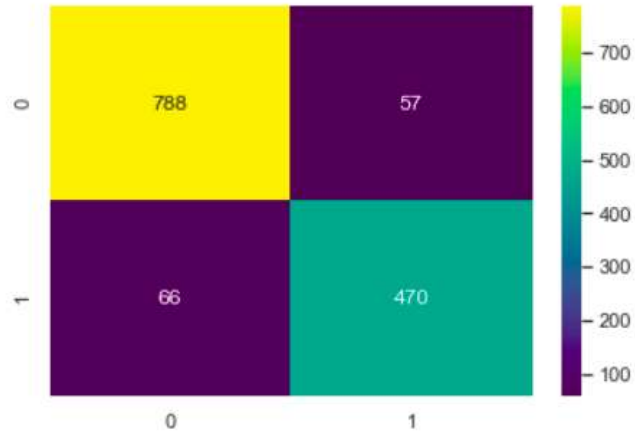


Рисунок 4.12 – класифіковано з MDS



	precision	recall	f1-score	support
0	0.92	0.93	0.93	845
1	0.89	0.88	0.88	536
accuracy			0.91	1381
macro avg	0.91	0.90	0.91	1381
weighted avg	0.91	0.91	0.91	1381

Рисунок 4.13 – Confusion matrix MDS

MDS застосовується при візуалізації інформації, для відображення інформації, що міститься в матриці відстані. Це форма нелінійного зменшення розмірності. Виходячи з матриці відстаней між кожною парою об’єктів в наборі даних, та кількості вимірів, MDS розміщує кожний об’єкт, в простір меншої вимірності таким чином, щоб відстані між об’єктами якомога краще зберігалися. Тому бачимо, що результат у MDS отримали дуже гарний, але потрібно багато часу для обчислення та проекції у новий підпростір.

Далі, щоб наявно показати які результати отримали. Швидкість кожного алгоритму вказана в таблиці 4.1.

Таблиця 4.1 — Швидкість алгоритмів зменшення розмірності даних

	PCA	T-SNE	UMAP	MDS
Time/sec	**0.053**	**118.01**	**20.8**	**781.9**

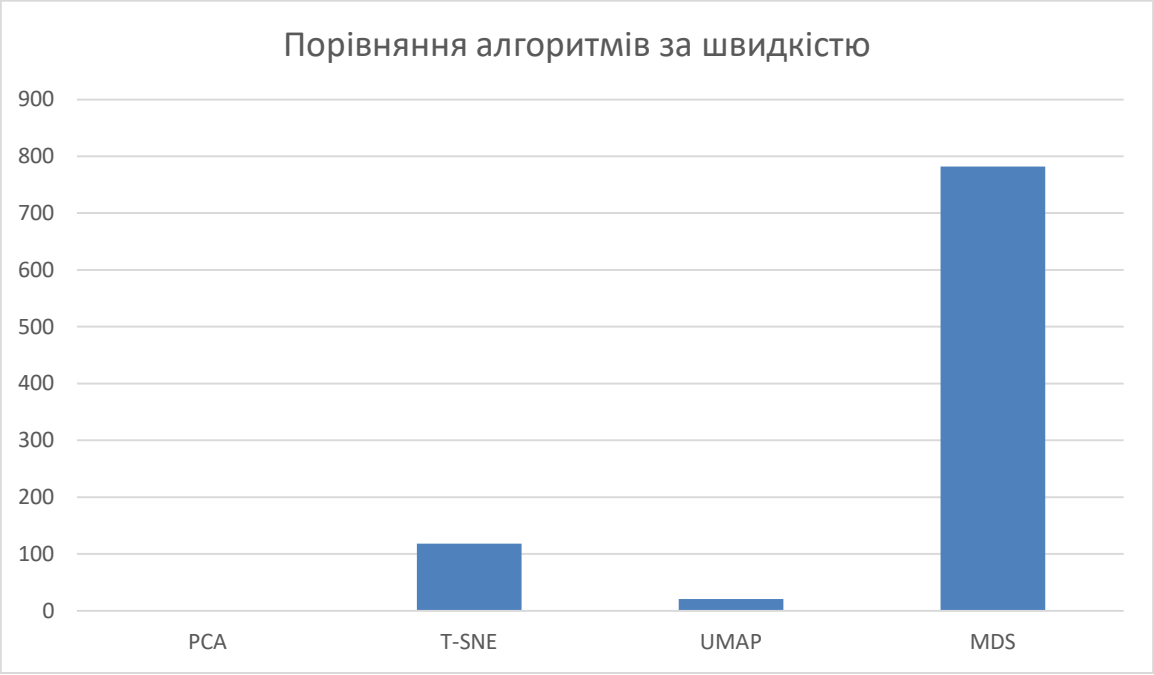


Рисунок 4.14 – Гістограма швидкості алгоритмів

Найкращий результат по швидкості показав нам алгоритм PCA, другий за швидкістю UMAP, третій T-SNE, четвертий MDS. Усі алгоритми працюють добре, але є ті, які працюють краще, а вибір який використовувати залежить від задачі. В наступній таблиці 4.2 представлено результати класифікації за допомогою алгоритмів зменшення розмірності. Тільки KNN навчається на стандартизованих вхідних даних, інші на даних зменшеного простору.

Таблиця 4.2 — Результати класифікатора на даних зменшених ознак

	KNN	**PCA**	**T-SNE**	**UMAP**	**MDS**
Accuracy	0.9138	0.8682	0.9167	0,924	0.9123
Precision	0.91	0.87	0.92	0.92	0.91
Recall	0.90	0.87	0.91	0.92	0.90
F-1	0.91	0.87	0.91	0.92	0.91
Time_total/sec	100.28	18,4	135,21	37,5	807,58

Як видно з таблиці 4.2 алгоритм, що найкраще показав себе це UMAP. Він має найкраще співвідношення якості та швидкості. Це дуже універсальний та швидкий алгоритм, який вміє працювати з нечіткими множинами. В документації до бібліотеки UMAP можна знайти приклади дуже цікавих задач, які він може вирішувати. Це дуже потужний алгоритм, який може допомогти вирішувати безліч цікавих задач.

Висновки до розділу 4

Було опрацьовано та досліджено методи пониження розмірності. Продемонстровано метод вкладення різних методів машинного навчання. Проаналізовано та знайдено найкращі за декількома критеріями, для фіксованої задачі аналізу спаму в електронних листах.

ВИСНОВКИ

В дипломній роботі досліджувалися алгоритми зниження розмірності даних, з використанням класифікатором KNN, щоб побудувати якісну модель на даних зі зменшеним простором ознак.

Зниження розмірності даних це задача навчання без вчителя яка входить до задач штучного інтелекту. Цілі зниження розмірності даних це зменшення кількості змінних в датасеті, для того, щоб покращити роботу алгоритмів, та видалити непотрібні сильно корелюючи дані. У міру збільшення кількості функцій модель стає дедалі складнішою. Чим більше функцій, тим більше шансів на переоснащення. Модель машинного навчання, яка навчається за великою кількістю функцій, стає все більш залежною від даних, на яких вона навчалася, та в свою чергу, переоснащується, що призводить до зниження продуктивності реальних даних, перевищуючи мету дослідження.

У багатьох проблемах реального світу навчальні приклади поширюються не на всі виміри. Багато функцій можуть бути майже константами, тоді як інші сильно корелюють. Як наслідок, усі навчальні приклади насправді лежать у межах (або близько до) значно менших розмірів підпростору багатовимірного простору.

Було побудовано вкладення алгоритмів з урахуванням класифікатора і методів зменшення розмірності даних, що дозволило оцінити як роботу зменшення розмірності, а й те, як працює алгоритм з новими даними меншої розмірності.

У цьому досліджені:

1. Було вивчено та проаналізовано основні типи задач інтелектуального аналізу.
2. Було вивчено різні підходи до машинного навчання.
3. Були продемонстровані різні алгоритми зменшення розмірності даних.
4. Побудовано вкладену модель класифікації, з використанням зменшення розмірності даних.
5. Було порівняно різні моделі зниження розмірності даних.
6. Було досягнуто більшої точності класифікації, саме з допомогою зниження розмірності

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Анализ данных и процессов: учеб. пособие / А. А. Барсегян, М. С. Куприянов, И. И. Холод, М. Д. Тесс, С. И. Елизаров. [Текст] — 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2009. — 512 с.
2. Aurélien Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. [Текст] — 2017 — 564с.
3. Dimensionality_reduction. [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Dimensionality_reduction
4. Machine learning. [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Machine_learning
5. Я. Гудфеллоу. Глубокое обучение. [Текст] — 2018. — 652 с.
6. С. Рашка. Python и машинное обучение. [Текст] — 2017. — 418 с.
7. Николенко. Глубокое обучение, погружение в мир нейронных сетей. [Текст] — 2018. — 477 с.
8. Документація бібліотеки Scikit-learn на Python. [Електронний ресурс]. – Режим доступу: https://scikit-learn.org/stable/index.html.

ДОДАТКИ

```
data = pd.read_csv("spambase.data")
data.head(20)
data.rename(columns={ data.columns[57]: "Decision" }, inplace = True)
data = data.fillna(data.mean())
data.describe().T
print(data['Decision'].value_counts())
p=data['Decision'].value_counts().plot(kind="bar")
X = data.iloc[:,:57]
y = data['Decision']
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scale_X = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(scale_X, y, test_size=0.3)
X_train.shape, X_test.shape
from sklearn.neighbors import KNeighborsClassifier
start_time = time.time()
classifier_knn = KNeighborsClassifier()
steps = [
   ('model', KNeighborsClassifier())
]
knn_pipe = Pipeline(steps)
grid_params = { 'model__algorithm' : ['auto'],
              'model__leaf_size' : [0.5,1,2,5,10,20],
              'model__metric' : ['minkowski'],
              'model__p' : [1,2],
              'model__n_neighbors' : [3,5,10,19,25],
```

```
        'model__weights' : ['uniform', 'distance'],
        'model__n_jobs' : [-1]
}
classifier_knn = GridSearchCV(knn_pipe, grid_params, cv = 5)
classifier_knn = classifier_knn.fit(X_train, y_train.ravel())
y_pred_knn_train = classifier_knn.predict(X_train)
accuracy_knn_train = accuracy_score(y_train, y_pred_knn_train)
print("Training set: ", accuracy_knn_train)
y_pred_knn_test = classifier_knn.predict(X_test)
accuracy_knn_test = accuracy_score(y_test, y_pred_knn_test)
print("Test set: ", accuracy_knn_test)
print("--- %s seconds ---" % (time.time() - start_time))
print('The grid_params: ')
print(classifier_knn.best_params_)
sns.heatmap(confusion_matrix(y_test, y_pred_knn_test), annot=True, cmap = 'viridis',
fmt='.0f')
plt.show()
print(classification_report(y_test, y_pred_knn_test))
from sklearn.decomposition import PCA
start_time = time.time()
pca = PCA(n_components = 3, svd_solver='randomized')
components = pca.fit_transform(scale_X)
#total_var = pca.explained_variance_ratio_.sum() * 100
#print(total_var)
print("--- %s seconds ---" % (time.time() - start_time))
components.shape
total_var = pca.explained_variance_ratio_.sum() * 100
fig = px.scatter_3d(
    components, x=0, y=1, z=2, color=data['Decision'],
    title='PCA visualization',
```

```
    labels={'0': 'PC 1', '1': 'PC 2', '2': 'PC 3'}
)
fig.show()
pca.get_params()
comp_X_train, comp_X_test, comp_y_train, comp_y_test = train_test_split(components, y,
test_size=0.3)
from sklearn.manifold import TSNE
start_time = time.time()
tsne_X = TSNE(n_components=3).fit_transform(scale_X)
print("--- %s seconds ---" % (time.time() - start_time))
fig = px.scatter_3d(
    tsne_X, x=0, y=1, z=2, color=data['Decision'],
    labels={'0': 'PC 1', '1': 'PC 2', '2': 'PC 3'},
    title = 'TSNE visualization'
)
fig.show()
comp1_X_train, comp1_X_test, comp1_y_train, comp1_y_test = train_test_split(tsne_X, y,
test_size=0.3)
start_time = time.time()
reducer = umap.UMAP(n_neighbors=13,
                    min_dist=0.1,
                    metric='manhattan',n_components=3)
embedding = reducer.fit_transform(scale_X)
print("--- %s seconds ---" % (time.time() - start_time))
embedding.shape
fig = px.scatter_3d(
    embedding, x=0, y=1, z=2, color=data['Decision'],
    labels={'0': 'PC 1', '1': 'PC 2', '2': 'PC 3'},
    title = 'UMAP'
)
```

```
fig.show()
start_time = time.time()
y_fit_pca = classifier_knn.fit(comp_X_train, comp_y_train)
z1 = classifier_knn.predict(comp_X_test)
accuracy_knn_test1 = accuracy_score(comp_y_test, z1)
print(accuracy_knn_test1)
print("--- %s seconds ---" % (time.time() - start_time))
print(classifier_knn.best_params_)
start_time = time.time()
tsne_1 = classifier_knn.fit(comp1_X_train, comp1_y_train)
z_tsn = classifier_knn.predict(comp1_X_test)
accuracy_tsn = accuracy_score(comp1_y_test, z_tsn)
print(classifier_knn.best_params_)
print(accuracy_tsn)
print("--- %s seconds ---" % (time.time() - start_time))
start_time = time.time()
umap_1 = classifier_knn.fit(comp2_X_train, comp2_y_train)
z_umap = classifier_knn.predict(comp2_X_test)
accuracy_umap = accuracy_score(comp2_y_test, z_umap)
print(classifier_knn.best_params_)
print(accuracy_umap)
print("--- %s seconds ---" % (time.time() - start_time))
from sklearn.manifold import MDS
start_time = time.time()
embedding = MDS(n_components=3, n_init=1, max_iter=1000)
X_transformed1 = embedding.fit_transform(scale_X)
print("--- %s seconds ---" % (time.time() - start_time))
```