МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Кафедра прикладної математики

Дипломна робота

Аналіз даних та прогнозування на основі методів класифікації

Виконав: студент групи ПМП-42
спеціальності
113 - прикладна математика

_____Гураль В.Р._____
(прізвище та ініціали)

Керівник __Щербатий М.В. _____
(прізвище та ініціали)

Рецензент_____
(прізвище та ініціали)

Львів – 2023

Зміст

Вступ

Тема моєї бакалаврської роботи “Аналіз даних та прогнозування на основі методів класифікації”. У рамках цієї теми буде проведений аналіз методів класифікації, таких як “k – найближчих сусідів” та “метод опорних векторів”. В роботі також розглянемо застосування цих методів для аналізу даних та прогнозування в різних сферах.

Тема дипломної роботи є актуальною, оскільки машинне навчання та прогнозування даних є однією з найбільш поширеною та актуальною задачею в сучасному світі. Завдяки швидкому розвитку інформаційних технологій та збереженню великих обсягів даних в різноманітних галузях людської діяльності стало можливим застосовувати складні методи аналізу даних і прогнозувати для підтвердження гіпотез, підтримки прийняття рішень та отримання нових знань. Тут слід виділити методи класифікації, які є одними з найпопулярніших та ефективніших інструментів для аналізу даних і прогнозування. Це все тому, що методи класифікації дозволяють автоматично розподіляти об’єкти за певними категоріями на основі їх характеристик. Також слід замітити що за даною темою не існує багато досліджень, присвячених порівняльному аналізу різних методів класифікації і їх застосування в різних сферах

Метою дипломної роботи є аналіз різних методів класифікації і їх застосування для аналізу даних в різних сферах. Для досягнення поставленої мети слід визначити наступні завдання:

1) Дослідити теоретичні основи методів класифікації;
2) Проаналізувати переваги та недоліки різних методів класифікації;
3) Вибрати найбільш доцільні методи класифікації для аналізу даних та прогнозування в різних сферах;
4) Провести експериментальне застосування вибраних методів на основі реальних даних;
5) Оцінити якість та точність отриманих результатів і зробити висновки.

Об’єктом дослідження є методи аналізу даних та прогнозування.

Предметом дослідження є методи класифікації, та їх застосування в різних галузях.

Наукова новизна полягає в тому, що вона проводить порівняльний аналіз різних методів класифікації за різними критеріями. Такий аналіз дозволяє вибрати найбільш оптимальні методи класифікації для конкретних задач

аналізу даних і прогнозування. Практична значимість роботи полягає в описі та експериментального застосування Результати експериментального застосування методів класифікації можуть бути використані дослідниками та фахівцями, які займаються аналізом даних і прогнозування в різних галузях.

У роботі застосовані такі методи досліджень:

1) Аналіз наукової літератури за темою роботи;
2) Порівняльний аналіз різних методів класифікації
3) Експериментальне застосування вибраних методів класифікації на реальних даних;
4) Статистична обробка і аналіз отриманих результатів
5) Формулювання висновків і рекомендацій

В даній роботі для застосування вибраних методів класифікації були застосовані такі засоби як Python та його бібліотеки. На мою думку дані засоби дозволять якісно оцінити вибрані методи класифікації.

Структура роботи складається з вступу, двох розділів, висновків та списком використаних джерел. В вступі описана тема й мета моєї дипломної роботи. В першому розділі описані теоретичні відомості по темі. В другому приведені приклади практичного застосування. В висновку підбив підсумки моєї роботи.

Розділ 1 Теоретичні відомості

1.1 Машинне навчання

Машинне навчання – це галузь штучного інтелекту, яка вивчає комп'ютерні алгоритми та моделі для автоматичного виконування завдань без явного програмування. Воно ґрунтується на ідеї, що комп'ютерні системи можуть навчатись та робити передбачення з даних, тобто покращувати свою продуктивність за допомогою даних та досвіту[7]. Машинне навчання почало активно розвиватись з 50-тих років минулого століття, а точніше з появою перших обчислюваних машин. Сам термін «машинне навчання» започаткував Артур Семюель у 1959 році. Алгоритмів машинного навчання, як і методів їх навчання на даний час є дуже багато, і вони продовжують розвиватись. Алгоритми навчання можна розділити на такі основні категорії:

- навчання з вчителем;
- навчання без вчителя;
- навчання з підкріплення.

Одним з найбільш використовуваних алгоритмів навчання є метод навчання з вчителем (Supervised Learning). Даний алгоритм полягає в використанні попередньо-підготовлених даних. Такі дані називаються навчальною вибіркою. Слід зауважити що підготовка таких даних – це складний процес, який потребує роботи, від однієї до групи людей. Зважаючи на це цікаво поглянути як проблему з розміткою даних вирішують великі компанії. Наприклад корпорація Google для збору таких даних реалізувала ReCapcha, яка просить вибрати зображення різних типів. Під час вибору зображень ми власноруч підготовлюємо дані, та формуємо навчальну вибірку, яку потім корпорація використовує у власних цілях. Тобто при використанні алгоритму навчання з вчителем ми користуємось навчальною вибіркою, та можемо явно сказати правильно чи неправильно алгоритм визначив значення, та наскільки правильно він це зробив. Після завершення процесу навчання модель може бути використана для передбачення відповідей до нових, раніше не використовуваних вхідних даних. Успішність моделі оцінюється на основі її здатності до точного передбачення вихідних значень для нових прикладів даних, котрі не брали участь у процесі навчання.

Алгоритм навчання без учителя (Unsupervised Learning) – це кардинально протилежний відхід у машинному навчанні. В цьому випадку ми

використовуємо нерозмічені дані, а сама модель робить висновки та знаходить структуру даних без явних вихідних міток. В більшості даний алгоритм навчання використовується для виявлення прихованих закономірностей вхідних даних. Хорошим прикладом застосування алгоритму навчання без учителя є задачі кластеризації, де модель аналізує нерозмічені дані та групує їх відповідно до схожих ознак. Іншим прикладом є задача зменшення розмірності, де модель знаходить нові представлення даних з меншою кількістю змінних, зберігаючи при цьому основну інформацію.

Останнім з виділених методів навчання є метод навчання з підкріпленням (Reinforcement Learning). Концепція навчання з підкріпленням полягає в виконанні певних дій що призведуть до максимального числа нагороди. Для цього ми використовуємо алгоритми які прораховують винагороду та покарання яку агент отримає при взаємодії з навколишнім середовищем. Прикладом застосування навчання з підкріпленням є автономні системи, робототехніка, ігри та інші, де важливо вибирати оптимальні стратегії при взаємодії з навколишнім середовищем .

Дані алгоритми навчання основні, проте також існують і інші. Також задачі машинного навчання класифікуються за бажаним виходом системи з машинним навчанням. За даним критерієм задачі машинного навчання поділяються на:

- задачі класифікації;
- задачі регресії;
- задачі ранжування.

Задача класифікації полягає в призначенні вхідних даних до певного класу. Наприклад приведення зображення до певного класу, розпізнання певного слова або на основі медичних даних видати діагноз.

Задача регресії робить прогнози відносно вхідних даних. Зазвичай використовується при оцінці курсу валют відносно попередніх даних, прогнозу продажу товару певного виду або побудуванні прогнозів деякої функції відносно емпіричних даних.

Задача ранжування полягає в порядкуванню по певному критерію вхідного набору даних. Класичним прикладом системи ранжування є пошукові системи, які ранжують вибірку по цінності для кожної конкретної людини та кожного конкретного запиту.

Всі ці задачі зазвичай вирішуються з застосуванням алгоритмів машинного навчання через те, що вхідні дані мають достатньо складну структуру і щоб їх

розв’язати слід побудувати нетривіальні критерії для їх обробки. Для людини це достатньо складно, тому ми і приходимо до методів машинного навчання.

Методи класифікації в більшості представлені методами навчання з вчителем, для яких необхідна навчальна вибірка.

1.2 Навчальна вибірка

Навчальна вибірка – це сукупність розмічених даних. Розмічені дані в свою чергу складаються з складається з вхідних образів та цільових виходів. Навчальна вибірка представлена набором певних чисел, які зручно представляти в вигляді векторів, так як ми працюємо з обчислювальними машинами, які в свою чергу сприймають тільки числа. Для прикладу розпишемо вектор вхідних даних:

$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \dots \\ x_{in} \end{bmatrix} = [x_{i1}, x_{i2}, ..., x_{in}]^T \tag{1.1}$$

В даному векторі (1.1) ми представили заміри певного об’єкта і з певною кількістю ознак n. Для прикладу для зображення n – це кількість пікселів, для людей цими ознаками можуть бути зріст та вага, та інше. Для представлення великої кількості об’єктів будемо використовувати матрицю:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_l \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ . & . & . & . \\ x_{l1} & x_{l2} & \dots & x_{ln} \end{bmatrix} \tag{1.2}$$

В даній матриці (1.2) вже записані вхідні дані l об’єктів з n кількістю ознак. Для попередніх прикладів l - це кількість зображень, або кількість людей для наступного прикладу. Проте навіть дана матриця ще не навчальна вибірка. Для того щоб побудувати навчальну вибірку нам також потрібний вектор вихідних даних:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix} = [y_1, y_2, ..., y_m]^T \tag{1.3}$$

Даний вектор (1.3) показує наші цільові значення. Даний вектор для задачі класифікації матиме вигляд:

- $y \in \{-1,1\}$ - при бінарній класифікації;
- $y \in \{1,2,...,M\}$ - при М-класовій класифікації(в якій класи не перетинаються)
- $y \in \{0,1\}^M$ - при М-класовій класифікації з класами, що перетинаються, де М – це розмірність простору.

Тепер маючи як вхідні, та вихідні дані ми можемо записати навчальну вибірку:

$$X^l = \{(x_i, y_i)_{i=1}^l\} \tag{1.4}$$

В записі (1.4) записана сукупність вхідних значень та вихідних значень які формують розмічені дані або навчальну вибірку. Такі дані мають велике значення для методів машинного навчання, так як їх формування це найбільш тривалий, та складний процес, який потребує участі однієї, або групи людей.

При моделюванні зображень та звуків для яких властива велика кількість даних до появи глибокий нейронних мереж інженерам приходилось власноруч визначати вторинні ознаки. Вторинні ознаки – використовувались як підготовчий крок задля оптимізації та усунення надлишкових ознак та побудови нового зменшеного набору ознак для полегшення навчання. Побудова вторинних ознак це досить творчий процес. На прикладі зображень, інженери виділяли границі певні об’єкти або області на зображенні, або просто ділили зображення хаотичними лініями і вимірювали різні співвідношення площ, і тому подібне. На рівні математики витяг ознак з сухих вхідних даних можна представити в виді функціонального перетворення:

$$\{f_1(x), f_2(x),..., f_k(x)\} \tag{1.5}$$

Ми на основі даних з вектора (1.1) формуємо певні вторинні ознаки $f_k(x)$, при тому k завжди буде меншим за попередню кількість вхідних ознак. Після чого ми отримуємо новий ознаковий простір:

$$F = \left\| f_j(x_i) \right\|_{l*n} = \begin{bmatrix} f_1(x_1) & ... & f_n(x_1) \\ ... & ... & ... \\ f_1(x_l) & ... & f_n(x_l) \end{bmatrix} \tag{1.6}$$

Ознаковий простір (1.5) дозволяє нам спростити роботу машинного навчання.

1.3 Постановка задачі машинного навчання

Постановка задачі машинного навчання полягає в підборі такого моделі, яка видаватиме бажані виходи a_i. Бажані виходи повинні в середньому бути як можна ближче до наших цільових виходів y_i. Для пошуку бажаних виходів ми і застосовуємо нашу навчальну вибірку, так як для неї цільові виходи вже є відомі.

Такий підбір ми називаємо навчанням моделі в результаті якого формується функція $a_i = a(x_i)$ - функція прийняття рішень, яка буде наближувати бажані виходи a_i до цільових виходів y_i на всій множинні вхідних даних. Після процесу навчання модель повинна вміти працювати з нерозміченими даними такого ж типу як і під час навчання. Для прикладу модель навчена класифікувати зображення тварин повинна вміти класифікувати тварин і за межами навчальної вибірки.

Графічне представлення задачі показане на рис. 1.

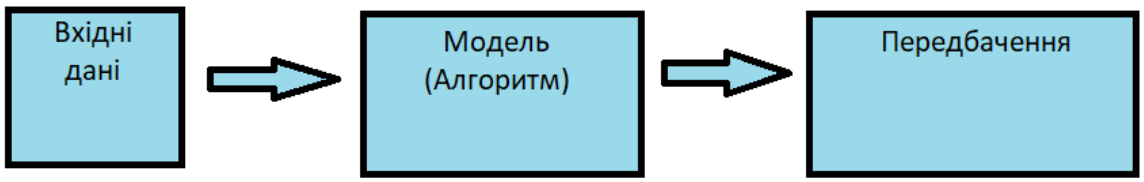


Рисунок 1.1 Постановка задачі машинного навчання

В такому представленні даної задачі ми не можемо явно показати правило функціонального перетворення $a(x)$. Для цього нам необхідна конкретизація. В сумі весь курс машинного навчання зводиться до вирішення даної проблеми. Одним з найпростіших, і найбільш використовуваних способів вирішення цієї задачі є представлення функціонала $a(x)$ в деякій параметричній функції $g(x,\theta)$

яка залежить від набору параметрів $\theta = [\theta_1, \theta_2, ..., \theta_n]^T$. Таким чином ми зводимо задачу навчання до пошуку даних параметрів. Для визначення параметрів θ ми і використовуємо навчальну вибірку. Представлена функція g може бути настільки складною наскільки потрібно, а також складатись з композиції інших більш простих функцій. Функція g повинна відображати характер та природу зміни даних між входом та виходом.

Тепер ми можемо представити нашу модель у вигляді параметричної функції:

$$a(x) = g(x, \theta) \tag{1.7}$$

Після цього нам необхідно для нашої функції (1.7) знайти значення параметрів θ на множині входів і виходів $X \times Y$ нашої навчальної вибірки. Ці параметри треба вибрати так, щоб зменшити похибку між заданими виходами та виходами моделі. Математично це можна записати як:

$$a(x) - y(x) \quad \forall x \tag{1.8}$$

Саму похибку незручно використовувати у якості оптимізованої величини, тому що в нулі вона не утворює точки екстремуму[1]. Математично краще використовувати функцію яка б збільшувалась коли б збільшувалась похибка, та зменшувалась коли похибка зменшується. Ці функції називаються функціями втрат. Наведу декілька прикладів таких функцій.

$$L(a, x) = | a(x) - y(x) | \tag{1.9}$$

$$L(a, x) = (a(x) - y(x))^2 \tag{1.10}$$

Функція (1.9) – це ніщо інше як абсолютна похибка, а функція (1.10) – це квадратична похибка. Хоч я і навів два приклади насправді функцій втрат набагато більше, але всіх їх об’єднує властивість описана вище.

Функція втрат – це випадкова величина, що залежить від алгоритму та значення x, тому оптимізувати одне конкретне значення функції неправильно. Необхідно зробити так, щоб на всій навчальній множині похибка була б мінімальною. В результаті ми приходимо до поняття середнього емпіричного ризику, тобто шукаємо середнє арифметичне функції втрат на всій множині навчальних даних і отримуємо одне число для всієї навчальної вибірки. Середній емпіричний ризик – це критерій якості при навчанні моделі, та записується наступним чином:

$$Q(a, X^l) = \frac{1}{l}\sum_{i=1}^{l} L(a, x_i) \qquad (1.11)$$

Тепер сумуючи все попередньо сказане, можна виділити наступні чотири пункти:

- Загальна задача машинного навчання визначається як пошук моделі $a(x)$, яка найкращим способом опише природу залежностей між вхідними даними $\{x_i\}$ та цільовими вихідними значеннями $\{y_i\}$.
- Задачу пошуку найкращої моделі найчастіше зводять до задачі параметричної оптимізації функції виду $a(x) = g(x, \theta)$.
- Для знаходження параметрів θ вводяться функції втрат $L(a, x)$ і визначають середній емпіричний ризик $Q(a, X^l)$. Мінімізуючи цей показних якості, отримуємо набір параметрів θ з навчальної вибірки.
- На основі знайденої залежності $a(x) = g(x, \theta)$ в подальшому визначаються вихідні значення $a_k = a(x_k^{'})$ при застосуванні нового вхідного вектора $x_k^{'}$ тієї ж природи, що й при процесі навчання.

Дані чотири етапи і лежать в основі всіх алгоритмів машинного навчання.

1.4 Задача класифікації

Задача класифікації – це одна з основних задач машинного навчання та штучного інтелекту, що полягає в поділі безлічі спостережень на групи, або як їх ще називають класи, які опираються на певні критерії даних об’єктів або на основі аналізу їх формального опису[4]. Класифікація – це процес групування моделлю даних на основі заздалегідь розміченої вибірки, тобто використовується метод навчання з вчителем, або навчання з підкріпленням. Слід зазначити, що існують також методи при яких обчислювальна машина самостійно знаходить спільні характеристики, за якими і групується нерозмічена вибірка, тобто тут уже використовується метод навчання без вчителя. Такі методи називаються методами непідконтрольної класифікації, або по іншому методами кластеризації.

Класифікація отримала широкий спектр застосувань в сучасному світі. Задачі класифікації використовуються при розпізнаванні облич, розпізнаванні відбитків пальців у власників смартфонів, доступ до захищених даних по сітківці ока чи голосу, визначення злоякісних пухлин на рентгенівських знімках

та інші. Типовою задачею класифікації є створення спам фільтр. Для реалізації даної задачі передусім потрібно задати навчальну вибірку, яка буде поділена на два класи: листи які дійсно є спамом та звичайні листи які не відносяться до спам-розсилки. Ідея створення даного алгоритму в тому, щоб навчитись за певними ознаками фільтрувати спам-листи з навчальної вибірки, та в подальшому працювати уже з нерозміченим набором даних.

Необхідно визначити, чому класифікація на сьогоднішній день настільки популярна. Насамперед це зумовлено простотою використання з великими обсягами даних, які в наш час далеко не рідкість. Класифікація допомагає легко згрупувати великий обсяг даних по певним ознакам, і знайти закономірності. Також слід зазначити що класифікація – це те чим займаємось ми з вами кожен день, на відміну від машинного навчання, де класифікація вимагає використання деколи складних алгоритмів. Для людини не буде проблемою при поході в магазин одягу досить точно згрупувати товари за певними ознаками (штани, футболки, шапки, куртки, сорочки і т.д.). Задача класифікації полягає в тому, щоб навчити обчислювальну машинну проводити ті ж маніпуляції з класифікацією об’єктів.

Найбільш популярними серед алгоритмів машинного навчання це алгоритми на основі навчання з вчителем. Це зумовлено такими причинами як:

- Доступність даних: Можливість зібрати набір розмічених даних є легшою та доступнішою, ніж зібрати великий набір нерозмічених даних чи навчальних сигналів.
- Легкість оцінювання: Навчання з учителем передбачає оцінювання моделі порівнюючи передбачені вихідні дані з розміченими вихідними даними.
- Широкий спектр застосувань: Гнучкість та здатність адаптуватись до різних завдань роблять привабливим цей підхід для багатьох дослідників та практиків. Також це відображається на можливості застосування в різноманітних сферах, включаючи обробки голосу та зображень.
- Прогрес у глибокому навчанні: За останні роки глибоке навчання, зокрема глибокі нейронні мережі, зазнало великого прогресу у навчанні з вчителем. Цей прогрес забезпечив сильні результати та підтримує популярність цього підходу.

В загальному, підхід навчання з вчителем є широко використовуваним завдяки своїй ефективності та доступності даних.

Існує багато різних видів задач класифікації, які представленні в машинному навчанні, також для розв’язання цих задач були розроблені спеціальні підходи до моделювання.

Найбільш популярними є такі види класифікаційних задач як:

- Бінарна класифікація (Binary Classification)
- Мультикласова класифікація (Multi-Class Classification)
- Класифікація за декількома мітками (Multi-Label Classification)
- Незбалансована класифікація (Imbalanced Classification)

Розглянемо дані види класифікації більш детально.

Бінарна класифікація (Binary Classification)

Бінарна класифікація – це завдання в машинному навчанні при якому вхідні дані діляться на два класи. Ці класи зазвичай позначають як “позитивний” і “негативний” або “клас 1” і “клас 0”. Метою бінарної класифікації є побудова моделі, яка зможе приймати рішення на основі вхідних ознак та визначати до якого класу належить кожен екземпляр даних. Для цього використовуємо навчальну вибірку, яка включає вхідні приклади та відповідні мітки класів. Під час навчання модель поступово прогресує та вчиться знаходити залежності між вхідними ознаками та класами, шукаючи оптимальні параметри для прийняття рішень. Після завершення навчання модель можна застосовувати для класифікації нових даних, які не брали участі в процесі навчання. Приклади задач бінарної класифікації є: розпізнавання спам-повідомлень, класифікації зображень за наявністю об’єкта певного типу, визначення ризику втрати клієнта та багато інших. Алгоритмів, що застосовують бінарну класифікацію є велика кількість. До них включають логістичну регресію, метод опорних векторів, рішаючи дерева, навчання на основі глибоких нейроних мереж, метод k – сусідніх елементів та інші. Вибір даних алгоритмів залежить від характеристик даних та вимог задачі.

Мультикласова класифікація (Multi-Class Classification)

Мультикласова класифікація мало чим відрізняється від задачі бінарної класифікації. Основна відмінність полягає в тому, що для мультикласової класифікації може бути більше двох класів.

Класифікація за декількома мітками (Multi-Label Classification)

Класифікація за декількома мітками є задачею в машинному навчанні, де модель повинна визначити кілька можливих міток або категорій, які відповідають вхідному екземпляру даних. У відмінність від класифікації з

однією міткою, де кожен екземпляр даних належить тільки до одного класу, у класифікації за декількома мітками може бути декілька одночасних міток. Приклади класифікації за декількома мітками включають розпізнавання об'єктів на зображеннях, визначення тематичних категорій для текстів, прогнозування декількох характеристик або властивостей об'єктів та багато інших задач. У класифікації за декількома мітками модель навчається визначати наявність або відсутність кожної мітки для кожного екземпляра даних. Для цього використовується навчальна вибірка, яка містить вхідні дані та набори міток для кожного екземпляра. Модель може використовувати різні алгоритми, такі як методи на основі глибоких нейронних мереж, ансамблі моделей та інші. Після завершення навчання модель може класифікувати нові дані, визначаючи наявність або відсутність кожної мітки для кожного екземпляра. Для оцінки продуктивності моделі використовуються метрики, такі як точність, відповідність, F-мера та інші, які враховують правильність класифікації з урахуванням декількох міток.

Незбалансована класифікація (Imbalanced Classification)

Незбалансована класифікація поміж інших виділяється тим, що кількість об’єктів в кожному класі роз приділяється нерівномірно. В таких випадках один клас (називається меншим класом або позитивним класом) може мати значно менше прикладів, ніж інші класи (негативний клас або більші класи). Приклади незбалансованої класифікації включають виявлення рідкісних подій, виявлення шкідливих програм, виявлення фроду, медичні діагнози рідкісних захворювань та багато інших. Проблема незбалансованої класифікації полягає в тому, що моделі, навчені на незбалансованих даних, мають тенденцію схильніше класифікувати більш численні класи, ігноруючи менші класи. Це може призводити до недооцінки або нездатності правильно класифікувати менші класи.

Для вирішення проблеми незбалансованої класифікації можуть використовуватись різні підходи, такі як:

- Застосування алгоритмів, які враховують вагу класів або мають вбудовані методи для роботи з незбалансованими даними, наприклад, зважена логістична регресія, збалансовані дерева рішень.
- Використання евристик або алгоритмів, спрямованих на точніше виявлення меншого класу, таких як збільшення порогу прийняття рішення для більшого класу або використання алгоритмів з розпізнаванням аномалій.

Метрики, такі як точність, відповідність, F-мера, AUC-ROC, часто використовуються для оцінки продуктивності моделі в задачах незбалансованої класифікації. Однак, варто також звертати увагу на специфічні метри, які враховують незбалансованість класів, такі як показники чутливості, специфічності та інші.

Для кожної з цих задач існують популярні, та найбільш підходящі алгоритми для побудови моделей. Назвемо найпопулярніші алгоритми для побудови моделей в задачах класифікації такі, як:

- Логістична регресія (Logistic Regression)
- К – найближчих сусідів (k-Nearest Neighbors)
- Дерева прийняття рішень (Decision Trees)
- Метод опорних векторів (Support Vector Machines)
- Наївний Баєс (Naive Bayes)

Більш детально деякі алгоритми будуть розглянуті в наступних розділах.

1.5 Проблеми машинного навчання

Крім проблем в машинному навчанні при побудові моделі слід відмітити дві основні проблеми з якими так чи інакше стикнеться кожен хто почне вивчати машинне навчання. Дані проблеми пов’язані безпосередньо з навчанням та даними на яких вона навчається, та називаються недонавчання та перенавчання. В цьому параграфі ми дамо їм визначення, навчимось з ними боротись при потребі та як уникнути цих проблем в майбутньому.

Спочатку дамо визначення для перенавчання моделі. Перенавчання (overfitting) – це явище в машинному навчанні, коли модель занадто добре пристосувалась до навчальних даних, проте показує погані результати на нових, раніше небачених даних. В такому випадку модель запам’ятовує тренувальні дані замість узагальнення паттернів, що в свою чергу негативно впливає на здатність моделі проводити узагальнення. Основними ознаками перенавчання є низька помилка на навчальних даних, і навпаки дуже велику похибку на тестових даних. Це стається через те, що модель запам’ятовує непотрібні деталі або шум в навчальних даних. Модель вважає, що цей шум є значущими даними, і коли ми використовуємо модель з новими вхідними даними, то ми отримуємо недостовірні результати. Також до ознак перенавчання відносять занадто складну модель, недостає, або неправильне

регулювання, недостатній обсяг тренувальних даних або їх неправильне розподілення.

Проблема перенавчання найчастіше виникає у нелінійних моделей, тому що вони є більш гнучкими при вивчені цільової функції. Тому багато алгоритмів передбачають методи які допомагають обмежити степінь деталізації об'єкта для запобігання проблемі перенавчання.

Для запобігання перетренування можна вжити певних заходів, а саме:

- Збільшити обсяг навчальних даних: Це може допомогти моделі краще узагальнювати певні ознаки в даних.
- Використовувати регуляризацію: Додавання штрафу до функцій втрати за складність моделі, такої як $L1$ та $L2$ регуляризації допоможе уникнути перетренування моделі.
- Використання методів зниження розмірності.
- Використання перехресної перевірки: Використання кросвалідації або перехресної перевірки допоможе оцінити продуктивність моделі на незалежних даних, та виявляти перенавчання.
- Спрощення моделі: Використання простіших моделей з меншою кількістю параметрів або з нижчою гнучкістю, можуть допомогти уникнути перенавчання.

Варто зазначити що дана помилка не критична, так як не дає моделі виконувати свою основну функцію, а також ніяк не вирішується. Все що ми можемо це мінімізувати значення перенавчання моделі.

Почнемо з використання методів повторної перевірки. Найпопулярнішим з таких методів є к-кратна перехресна перевірка (k-fold cross validation). Суть методу полягає в розбиванні всієї вибірки на віддільні наглядові групи. Після цього ми по черзі забираємо 1 групу як тестову вибірку, а інші залишаємо як навчальну вибірку. Перебравши таким чином всю вибірку ми отримаємо масив моделей $\{a_1(x), a_2(x), a_3(x), ..., a_k(x)\}$ потім на основі даних груп повинна бути сформована одна спільна модель: $a(x) = F(a_1, a_2, ..., a_k)$. Даний підхід дуже схожий на перехресну перевірку з виключеннями, проте на відміну від перехресної перевірки з виключеннями к-кратна перехресна перевірка групує свої дані, що дозволяє бути більш оптимізованим за свого конкурента. Для того щоб сформувати спільну модель існує декілька способів найбільш очевидних і найбільш вживаних підходів. Перший спосіб полягає в виборі однієї моделі з масиву, яка має найбільші виміряні ознаки якості, тобто точність на навчальній та відкладеній вибірці. Основною проблемою такого способу є те, що дана

модель може показувати себе вдало на навчальній вибірці, проте бути неактуальною при введені нових даних. Завжди є шанс що саме для цієї моделі була вдало розбита вибірка і показники якості виявились кращими за інші, проте її узагальнюючі властивості далекі від ідеалу. Другий підхід передбачає, що ми залишаємо всі моделі і на моменті експлуатації при пропускаємо вхідні дані через всі моделі і вибираємо те значення, яке частіше всього фігурувало. Такий підхід дає стійкіші результати хоча збільшується кількість обчислень, так як ми використовуємо зразу декілька моделей.

Графічне представлення методу к-кратної перехресної перевірки можна побачити на рисунку 1.2.

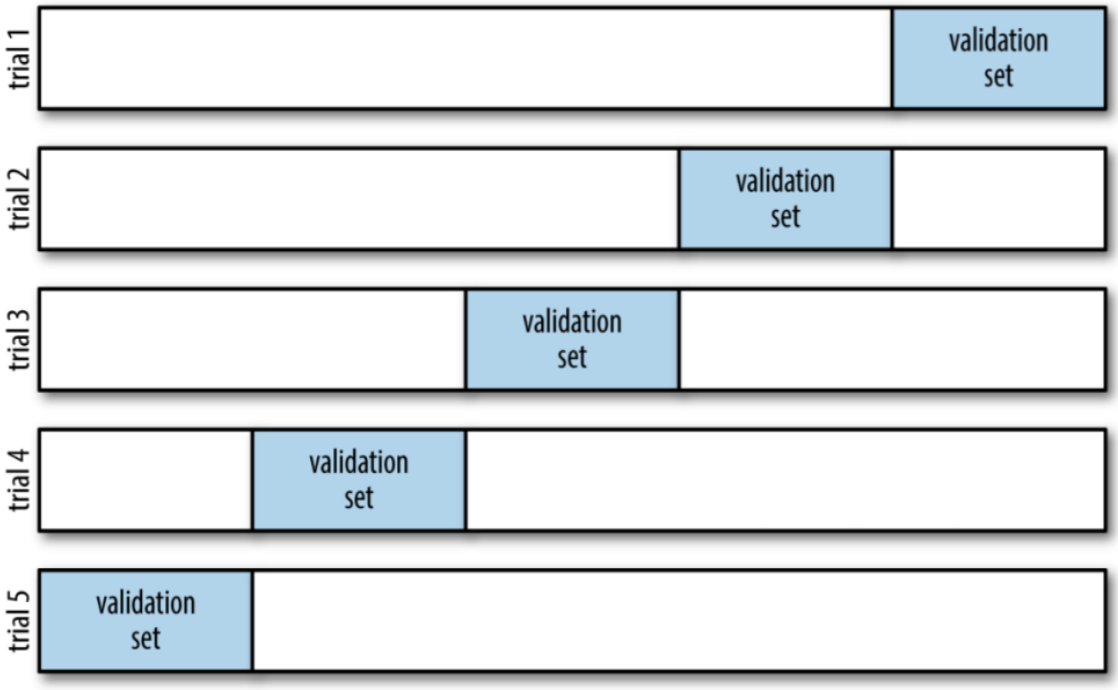


Рисунок 1.2 Графічне представлення к-кратної перехресної перевірки

Наступним популярним підходом є оцінка по відкладеній вибірці (hold-out). Суть цього методу полягає в випадковому розбитті розмічених даних на навчальну та тестову здебільшого в співвідношенні 70% на 30%, тобто 70% - навчальна вибірка, 30% - тестова вибірка. Далі, після проходження процесу навчання знаходимо два значення $Q_1(a, X^l)$ та $Q_2(a, X^t)$, де $Q_1(a, X^l)$ – це значення якості моделі на навчальній вибірці, а $Q_2(a, X^t)$ – це значення якості моделі на тестовій вибірці. Порівнявши ці два значення і якщо вони близькі між собою, то

можна сказати, що дана модель не перенавчена. Проте і в цього метода є свої мінуси. В залежить від випадкового розбиття розмічених даних може вийти так, що якісна величина навчальної вибірки завжди буде наближена до якісної величини тестової вибірки, і ми знову стикнемося з перенавчанням моделі.

На рисунку 1.3 показане графічне представлення оцінки по відкладеній вибірці.

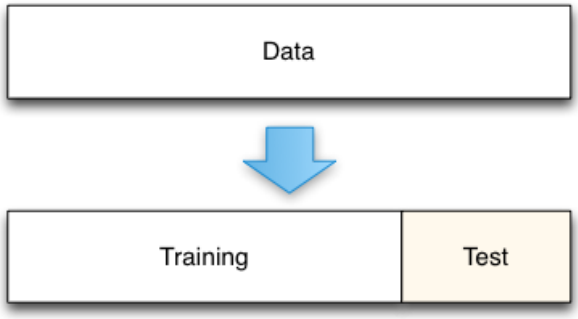


Рисунок 1.3 Графічне представлення оцінки по відкладеній вибірці

На даному етапі ми вже розглянули як правильно перевіряти натреновану модель, як поділити розмічені дані на навчальну та тестову вибірку для запобігання перенавчання моделі. Тепер слід перейти до визначення як саме тренувати модель, щоб запобігти перенавчанню.

Існує багато методів, які запобігають перенавчанню моделей. Серед них цікавим методом є регуляризація. Регуляризація – це метод, які запобігають перенавчання моделі шляхом додавання штрафу до функції втрати. Після цих дій модель стає більш узагальненою, що допомагає їй точніше прогнозувати нові вхідні дані[2]. В основі роботи методу регуляризації стоїть дуже простий алгоритм. Фактично регуляризація зменшує коефіцієнти моделі штрафуючи вагові матриці. В такому випадку ми можемо прийти до ситуації, коли деякі вагові коефіцієнти деяких вузлів дорівнюватимуть нулю. Це означатиме, що дані коефіцієнти вже не зможуть впливати на прогнозування. Таким чином і спрощується модель за допомогою регуляризації. Як завжди в машинному навчанні необхідно пам’ятати, що надмірна регуляризація шкодить не менше. При надмірній регуляризації модель може стати лінійною і втратити свою ефективність в роботі з реальними даними, тому процес вибору регуляризації також не є простим.

На даним момент є багато різних методів регуляризації, проте одним з найпоширеніших є $L1$ та $L2$ регуляризації. Суть як і $L1$ так і $L2$ регуляризації полягає в оновленні функції витрат з використанням спеціального доданка. Таким чином тепер функція витрат схематично матиме вигляд:

$$Cost\_function = Loss + Regularization \tag{1.12}$$

$L2$ регуляризація відрізняється від $L1$ регуляризації лише доданком, який використовується при регуляризації. При $L2$ функція витрат матиме наступний вигляд:

$$Cost\_function = Loss + \frac{\lambda}{2}\sum_{i=0}^{n}\|\omega\|^2 \tag{1.13}$$

Де λ – це гіперпараметр регуляризації. З його допомогою регулюється, як сильно будуть змінюватись вагові матриці. В свою чергу функція витрат $L1$ регуляризації приймає такий вигляд:

$$Cost\_function = Loss + \frac{\lambda}{2}\sum_{i=0}^{n}\|\omega\| \tag{1.14}$$

Принцип вибору виду регуляризації залежить від кожної конкретної задачі. Для більш складних даних застосовуємо $L2$ регуляризацію, для більш простих за структурою даних краще використовувати $L1$ регуляризацію.

1.6 Метод k – найближчих сусідів

В 1936 році Рональд Фішер[5], спеціаліст математичної статистики, представив свою роботу по аналізу даних квітів ірисів. Дані були зібрані ботаніком Едгаром Андерсоном, а Фішер знайшов спосіб по цим даним класифікувати ці квіти на три види. Пізніше названі “Іриси Фішера” складаються з даних 150 екземплярів і 4-ох мірний ознаковий простір. Для візуалізації найчастіше використовують так звану “Діаграму розсіювання Фішера” яка представлена на рисунку 4.

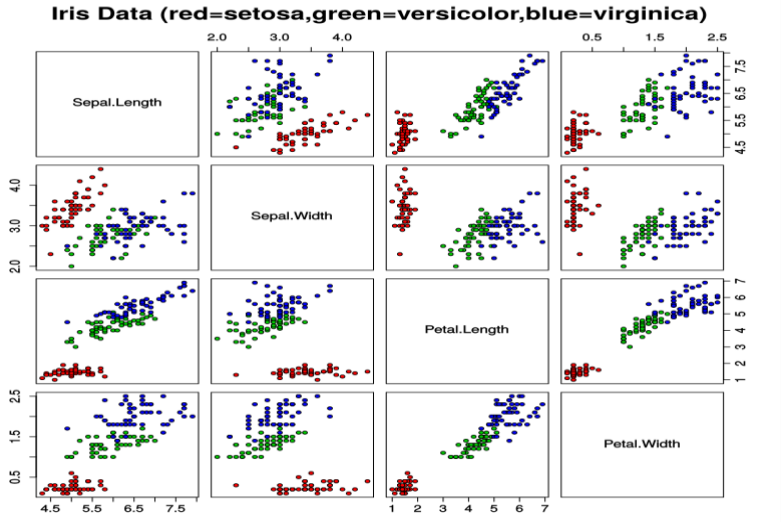


Рисунок 4 Діаграма розсіювання Фішера

Дана діаграма відображає представлення всіх попарних наборів ознак. З даної діаграми прекрасно видно, що об'єкти результати обчислень одного і того ж класу розташовані близько один до одного. Згодом була гіпотеза яка звучить так:

Гіпотеза компактності:

"Координати образів одного і того ж класу в ознаковому просторі концентруються в геометрично близькі точки, утворюють компактні згустки."

Слід зазначити, що не для всіх об'єктів дана гіпотеза виконується. Однак, якщо аналіз ознакового простору показує, що гіпотеза компактності в цілому виконується, то в такому випадку для задач класифікації ми можемо використовувати метричні методи.

Метричні методи – це методи що базуються на відстанях від конкретного образу x до образів інших класів в ознаковому просторі. Вирішуючи дану задачу ми стикаємось з певними проблемами, а саме:

- Як вимірювати відстань між об'єктами в ознаковому просторі;
- Як вирішити на основі відстаней приналежність об'єкта до тієї чи іншої групи точок.

На дані питання немає єдиної чіткої відповіді, тому існують досить велика кількість різних метричних методів класифікації. Одним з таких є метод k – найближчих сусідів.

Метод k – найближчих сусідів – це як ми швидше визначили параметричний метод навчання з вчителем. Даний метод використовується як для класифікації, так і для задач регресії. Принцип роботи цього методу полягає в знаходженні k

найближчих навчальних даних для вхідних даних. Для задач класифікації метод k – найближчих сусідів буде результатом буде приналежність точки до того чи іншого класу. Приналежність точки до класу визначається шляхом голосів k найближчих точок.

Для математичного представлення методу k – найближчих сусідів запишемо вектор ваги. Вектор ваги необхідний для того, щоб в рівній мірі враховувати при визначені при наближені, і вин матиме вигляд:

$$\omega(i, x) = [i \le k] \tag{1.15}$$

Після визначення вектора ваги необхідно визначити відстань вхідної точки до кожної точки з вхідних даних. Першим що приходить в голову це просто використати евклідову відстань, але, зважаючи на те що ми вже описали вектор ваги, доцільніше застосувати метрику Мінковського. Тоді відстань матиме вигляд:

$$p(x, x_i) = \left(\sum_{j=1}^{n} \omega_j * \left| x^j - x_i^j \right| \right)^{\frac{1}{p}} \tag{1.16}$$

Тепер, маючи відстань яка в рівній мірі рахуватиме вхідні дані, ми можемо записати написати напишемо наш алгоритм класифікації, котрий матиме вигляд:

$$a(x; X^T) = \arg\max \sum_{i=1}^{k} [y^{(i)} = y] \tag{1.17}$$

Метод k – найближчих сусідів це один з найпопулярніших та одним з простіших серед алгоритмів для задач класифікації. В нього є свої переваги та недоліки.

Постараємось визначити основні плюси та мінуси методу k – найближчих сусідів.

Ось деякі переваги методу:

- Даний метод легко реалізувати і зрозуміти, що робить його доступним для початківців.
- Метод не залежить від певного розподілу даних, тож може працювати у випадках, коли дані не мають чіткого математичного опису.
- Метод k – найближчих сусідів може використовуватись не тільки для бінарної класифікації, а й для мультикласової класифікації та регресії що вказує на його гнучкість.

- Добре працює з невеликими наборами даних, де більш складні алгоритми стикались з проблемами з узагальненням.

Недоліки методу k – найближчих сусідів:

- Для класифікації нового об’єкта необхідно порівняти його з всіма тренувальними прикладами, що дуже витратно при великій тренувальній вибірці.
- Даний метод дуже чутливий до шуму і непотрібних ознак, оскільки він враховує всі непотрібні ознаки, що призводить до неправильної класифікації.
- Вимога до вибору оптимального k, так як надто мале k загрожує привести до перенавчання програми, а завелике до недостатньої уваги до локальних паттернів.
- Метод k – найближчих сусідів не надає пояснення, які ознаки є важливими для класифікації, що може бути проблемою для деяких застосувань.

Необхідно збалансувати переваги та недоліки KNN при використанні його для конкретних задач класифікації та урахувати особливості вхідних даних.

1.7 Метод опорних векторів

Метод опорних векторів (Support Vector Machines) також є одним з популярних алгоритмів машинного навчання. Даний алгоритм відноситься до методу навчання з учителем і використовується як і в задачах класифікації, так і в задачах регресії.

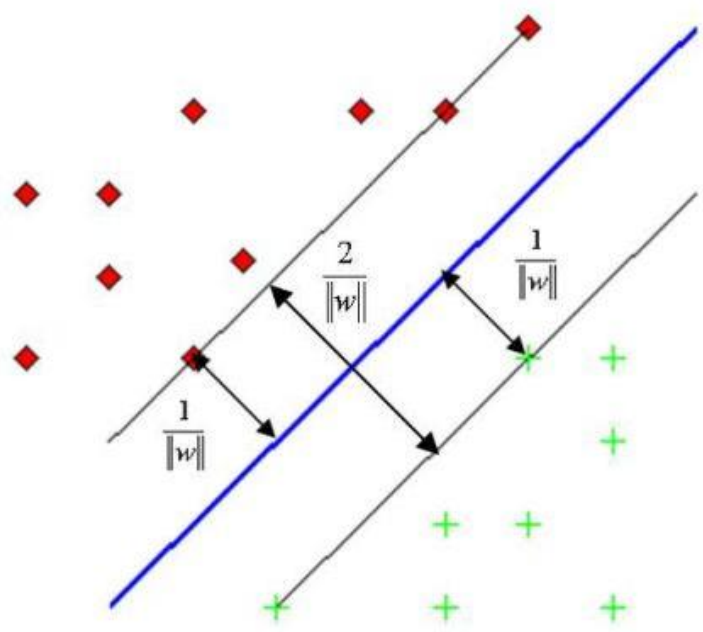
Основна ідея методу опорних векторів полягає в тому, щоб побудувати гіперплощину, за допомогою якої відбувається класифікація нових даних. Для прикладу в двомірному просторі гіперплощина – це лінія, яка розділяє площину на дві частини, в кожній з яких знаходиться відповідний клас. Ціль методу – це побудова даної гіперплощини так, щоб відстань від точок, по різні сторони гіперплощини, була максимальною[3].

Дослідимо детальніше цей метод, та визначимо принцип його роботи. Об'єкти, що розташовані найближче до заданої гіперплощини, називають опорними векторами. Метод непогано працює не тільки в бінарній класифікації, а й при більшій кількості класів. Для цього необхідно змінити ядро, але до цього ми повернемось, коли будемо описувати ядра.

Для кращого розуміння принципів роботи алгоритму досліджуватимемо алгоритм саме на бінарній класифікації. Розглянемо вектор $\vec{w}$ – вектор нормалі, приведений до роздільної гіперплощини. Розділимо всі об'єкти по гіперплощині, і ті, що відносяться до позитивного класу позначимо, як x_+, а негативного до x_-. Тепер шукаємо проекцію вектора, кінцями якого є опорні вектори з обох класів на вектор $\vec{w}$. Міра цієї проекції буде параметром, який буде максимізувати алгоритм при навчанні. Ця міра відповідає за ширину роздільної полоси між класами. Запишемо математичну інтерпретацію даного алгоритму, а саме:

$$\left\langle (x_+ - x_-), \frac{\vec{w}}{\left\|\vec{w}\right\|} \right\rangle = \frac{1}{\left\|\vec{w}\right\|} * (\left\langle x_+, \vec{w} \right\rangle - \left\langle x_-, \vec{w} \right\rangle = \frac{2}{\left\|\vec{w}\right\|} \to max \to \left\|\vec{w}\right\| \to min \leftrightarrow \frac{(\overrightarrow{w^T}\ \vec{w})}{2} \to min \quad (1.18)$$

На рисунку 5 можна побачити принцип роботи алгоритму



Рисунку 5 Принцип будування максимальної роздільної полоси

Метод опорних векторів за своєю структурою є нелінійним узагальненням лінійного класифікатора, який за допомогою спеціальних

ядрових функцій збільшує розмірність вихідного простору. За допомогою цього алгоритм може будувати роздільні поверхні різної форми.

При відображенні результатів у просторах більшої розмірності, вихідна множина точок може змінити розподіл та стати лінійно розподіленою. Саме тому використовуються різні ядрові функції[6]. При їх використанні роздільну площину можна знайти підбором коефіцієнта a_i для наступної формули:

$$\sum_{i=0}^{p} a_i K(x, x_i) \tag{1.19}$$

Це дозволяє звести задачу до задачі квадратичної оптимізації за допомогою множників Лагранжа.

Найбільш популярними ядровими функціями є:

- Лінійне ядро:

$$K(x_i, x_j) = x_i^T x_j \tag{1.20}$$

- Поліноміальне ядро зі степеню p:

$$K(x_i, x_j) = (1 + x_i^T x_j)^p \tag{1.21}$$

- Ядро Гаусса з радіально-базисною функцією:

$$K(x_i, x_j) = exp(\gamma \left\| x_i - x_j \right\|^2) \tag{1.22}$$

- Сигмоїдне ядро:

$$K(x_i, x_j) = tanh(\gamma x_i^T x_j + \beta_0) \tag{1.23}$$

Алгоритм методу опорних векторів має як і переваги так і певні недоліки. Розглянемо плюси та мінуси цього методу.

Переваги методу опорних векторів:

- Метод опорних векторів працює, у випадку коли ми не маємо уявлення про дані.
- Виконує свої функції навіть з неструктурованими та напівструктурованими даними, такими як зображення, дерева та текст.
- Може бути модифікований відповідною функцією ядра, що дозволить вирішувати складні задачі.
- Ризик виникнення перенавчання в методі опорних векторів значно нижчий відносно інших моделей.

- Непогано масштабується під різну кількість даних.

Виділимо недоліки методу опорних векторів:

- Складний вибір підходящої функції ядра.
- Довге навчання при великих об’ємах даних
- Остаточну модель складно побачити, що не дозволяє робити невеликі калібрування моделі.

Метод опорних векторів популярний через його доступність. Для його реалізації складено безліч бібліотек як для R так і для Python. Також даний метод непоганий для оцінки навчальної вибірки, якщо ви робите це вперше.

1.8 Висновок по розділу

В даному розділі було сформовано визначення машинного навчання, та розглянута класична постановка задачі машинного навчання. Також було розглянуто задачу класифікації, сформовано, що саме являється задачею класифікації. В цьому розділі були розглянуті класичні методи реалізації методів машинного навчання. Було розглянуто як саме будується навчальна та тестова вибірка, як від даного процесу залежить майбутня модель, що буде тренуватись. та принцип побудови процесу навчання з вчителем, а також були названі і інші види навчання. Розглянули принцип побудови процесу навчання з вчителем, без вчителя, навчання з підкріпленням, та їх найбільш доречні приклади застосування. Також розглянули проблеми задач машинного навчання, та способи їх усунення.

Розділ 2 Програмна реалізація та аналіз результатів

2.1 Інструменти для реалізації моделі

Класичні моделі класифікації для на масиві великої кількістю вхідних параметрів була розроблена з використанням теоретичних відомостей, та методів к – найближчих сусідів і опорних векторів. Для реалізації також були використанні дані з кількох дата сетів.

Для реалізації програмного продукту було вирішено використовувати мову програмування Python 3.10 та бібліотеку для машинного навчання scikit-learn (sklearn).

Надав перевагу Python через його гнучкість при виконані різноманітних завдань. Саме для цієї мови програмування було розробленно велику кількість бібліотек, за допомогою, яких можна доволі легко та просто реалізовувати алгоритми що нас цікавлять.

Для роботи з великим обсягом даних, яких можна представити у вигляді таблиці використовую бібліотеку pandas. Дана бібліотека дозволяє швидко доступатись до данних. Так як в даній бібліотеці реалізований власний тип даних, це забезпечує швидку обробку великих масивів даних, на відміну від стандартних рішень. Основною причиною швидкодії даної бібліотеки в застосуванні DataFrame, що індексується.

Найпопулярнішою бібліотекою для реалізації різних алгоритмів машинного навчання, а також їх тестування є sklearn[9]. Дана бібліотека дозволяє реалізовувати алгорим навчання з вчителем, так і алгоритм навчання без учителя. Бібліотека реалізована для мови програмування Python, хоча її також можна використовувати для мови програмування C/C++.

У представленій бібліотеці реалізовано безліч класичних алгоритмів машинного навчання. Серед них є лінійна регресія, метод опорних векторів, дерева рішень, метод к – наближених сусідів, та багато інших. Завдяки тому, що алгоритми вже реалізовані, досліднику достатньо лиш правильно підібрати гіперпараметри для відповідного алгоритму та доволі швидко отримати результати роботи.

Для виведення графіків використав бібліотеку matplotlib. Дана бібліотека є однією з найпопулярніших бібліотек для візуалізації даних у мові

програмування Python. Вона надає потужний інструментарій для створення різноманітних графіків, діаграм та інших видів візуалізації.

2.2 Побудова задачі регресії, та їх розв'язання

Почнемо дослідження з методу k – найближчих сусідів. Спочатку для цього необхідно сформувати навчальну та тестового набору даних. Для цього використали вибірку цін на житло в Каліфорнії[8]. Набір складається з таких атрибутів:

- MedInc – середній дохід з багатоповерхівки
- HouseAge – вік багатоповерхівки
- AveRooms – середня кількість кімнат
- AveBedrms – середня кількість спалень
- Population – кількість людей що живе в багатоповерхівці
- AveOccup – середня кількість управляючих
- Latitude – широта (розташування будинку)
- Longitude – довгота (розташування будинку
- MedHouseVal – середня вартість будинку для округу Каліфорнії(сотні тисяч доларів)

Цей набір даних має наступний вигляд:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

Рисунок 2.1 Вигляд вхідних даних

Тепер нам необхідно виділити параметр яку будемо прогнозувати та розбити наші дані на навчальні та тестові. В цій задачі ми будемо прогнозувати параметр MedHouseVal. Наступним буде підготовка набору даних. Зважаючи на те, що ми будемо передбачати середню вартість будинку, для чого перенесли його з масиву вхідних даних, ми можемо відкинути її. Наступним, що необхідно виконати, це поглянути на відмінності у вимірюваннях диних. Для цього застосуємо descried() спосіб перевірки, та отримаємо такі результати:

	count	mean	std	min	25%	50%	75%	max
MedInc	20640.0	3.870671	1.899822	0.499900	2.563400	3.534800	4.743250	15.000100
HouseAge	20640.0	28.639486	12.585558	1.000000	18.000000	29.000000	37.000000	52.000000
AveRooms	20640.0	5.429000	2.474173	0.846154	4.440716	5.229129	6.052381	141.909091
AveBedrms	20640.0	1.096675	0.473911	0.333333	1.006079	1.048780	1.099526	34.066667
Population	20640.0	1425.476744	1132.462122	3.000000	787.000000	1166.000000	1725.000000	35682.000000
AveOccup	20640.0	3.070655	10.386050	0.692308	2.429741	2.818116	3.282261	1243.333333
Latitude	20640.0	35.631861	2.135952	32.540000	33.930000	34.260000	37.710000	41.950000
Longitude	20640.0	-119.569704	2.003532	-124.350000	-121.800000	-118.490000	-118.010000	-114.310000

Рисунок 2.2 Виводи метода describe()

На рисунку 2.2 по колонці mean ми бачимо наскільки сильно відрізняються між собою вхідні дані.

Зважаючи на те, що ми користуємось алгоритмом, який заснований на відстані, сильно страждають від даних, які мають різний масштаб. Це може спотворити реальну відстань між значеннями.

В бібліотеці sklearn є метод що дозволить нам масштабувати функцію. Даний метод має назву StandardScaner. Однак перед масштабуванням щоб запобігти витоку даних необхідно розділити масив даних на підготовчий та тестувальний. Для цього застосую метод train_test_split метод з класу sklearn. Цей метод розбиває вибірки X і Y випадковим чином та розділяє ці вибірки за певним відсотком. Тепер треба зробити, щоб процес був відтворюваним(щоб метод завжди відбирав ті самі точки даних), встановимо аргумент random_state певну константу.

Для даної задачі ми вибрали 70% даних для навчання та 30% для тестування. Тепер ми можемо масштабувати наші дані не боячись витоку даних з тестової в нашу навчальну вибірку. Після масштабування наших даних, ми можемо використати функцію describe() за спостереженнями в полях mean та std. І це нам дасть:

	count	mean	std	min	25%	50%	75%	max
MedInc	14448.0	-1.003258e-16	1.000035	-1.772846	-0.687546	-0.177333	0.462602	5.839437
HouseAge	14448.0	-5.286776e-17	1.000035	-2.186232	-0.838437	0.033665	0.667922	1.857152
AveRooms	14448.0	3.786807e-16	1.000035	-1.854194	-0.403180	-0.083841	0.253748	55.623328
AveBedrms	14448.0	3.491731e-16	1.000035	-1.708893	-0.203919	-0.108476	0.005128	54.836403
Population	14448.0	6.786745e-17	1.000035	-1.249733	-0.558617	-0.227969	0.262302	30.042538
AveOccup	14448.0	1.180304e-17	1.000035	-0.196144	-0.055673	-0.024415	0.012923	100.233841
Latitude	14448.0	-1.464068e-15	1.000035	-1.451850	-0.801000	-0.646481	0.968938	2.949583
Longitude	14448.0	3.856150e-15	1.000035	-2.379545	-1.106366	0.536284	0.785927	2.633284

Рисунок 2.3 Виводи метода describe() для масштабованих даних

Як видно з рисунка 2.3 усі стандартні відхилення тепер длизькі до 1, а середні значення стали меншими. Це робить наші дані більш однорідними, і тепер підготувавши нашу вибірку ми можемо навчити, та оцінити регресом на основі KNN.

Тепер необхідно навчити нашу модель на основі тренувальної вибірки, та застосувати навчену модель на тестовій вибірці, щоб поглянути та проаналізувати результати.

Зазвичай для оцінки якості проведення регресії застосовують середню абсолютну похибку (MAE), середню квадратичну похибку (MSE), середньоквадратична похибка (RMSE) і коефіцієнт детермінації (R2)

1. Середня абсолютна похибка (MAE): для цього віднімаємо прогнозовані значення від фактичних, отримуємо похибки, підсумовуємо абсолютні значення цих похибок і отримуємо їх середнє значення. Цей показних дає уявлення про загальну помилку для кожного передбачення моделі, чим менше, тим краще:

$$mae = \left(\frac{1}{n}\right)\sum_{i=1}^{n}\left|Actual - Predicted\right| \tag{2.1}$$

2. Середня квадратична похибка (MSE): схожа на метрику, але зводить у квадрат абсолютні значення помилок. Як і в MAE, чим менше, або ближче до нуля тим краще.

$$mse = \sum_{i=1}^{D}\left(Actual - Predicted\right)^2 \tag{2.2}$$

3. Середньоквадратична похибка (RMSE): намагається вирішити проблему з інтерполяцією, що виникає з MSE, шляхом отримання квадратного кореня з його остаточного значення, щоб масштабувати його назад до тих самих одиниць даних. Це легше для інтерполяції та добре, коли нам потрібно відобразити або показати фактичне значення даних з помилкою. Чим ближче до нуля, тим краще.

$$rmse = \sqrt{\sum_{i=1}^{D}\left(Actual - Predicted\right)^2} \tag{2.3}$$

Для обчислення цих показників використаємо методи mean_absolute_error() та mean_squared_error(), sklearn.matrics. Підставивши в ці формули значення отримаємо:

```
MAE: 0.44652109916020666
MSE: 0.42949402014873317
RMSE: 0.6553579328494721
```

Рисунок 2.4 Вивід похибок

А R2 можна обчислити безпосередньо за допомогою score() методу, та вивести значення, в нашому випадку R2 = 0.6727764602454027.

Результати показали, що загальна та середня похибка нашого алгоритму KNN становить близько 0.44, і 0.43. Зі значеннями 0.67 ми бачимо що модель пояснює 67% дисперсії даних. Це вже більше ніж 50, але ми можемо спробувати. Якщо згадати, що ми весь час застосовували попередньо визначене k зі значенням 5, тому ми і вибирали 5 сусідів для прогнозування наших цілей, що не обов'язково є ідеальним варіантом. Тому, щоб зрозуміти яке ідеальне значення числа k, ми можемо проаналізувати помилки нашого алгоритму і вибрати k, що буде мінімізувати втрати.

Шукати найкраще k будемо опираючись на середню абсолютну похибку. Для цього запишемо цикл for і запустимо моделі, які мають від 1 до X сусідів, Під час кожної взаємодії будемо обчислювати MAE і будемо будувати графік кількості ks разом із результатом MAE. На виході нас буде очікувати графік на якому можна визначити найоптимальніше k:

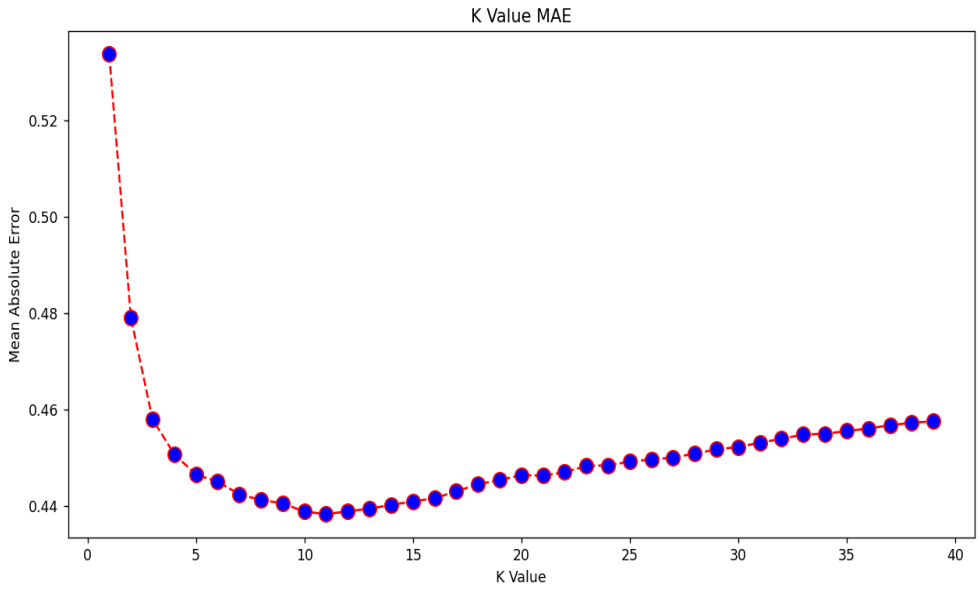


Рисунок 2.5 Графік пошуку найкращого k

З графіку видно, що оптимальне число k сусідів дорівнює 12. Тепер підставивши це значення в нашу модель ми отримаємо наступні похибки:

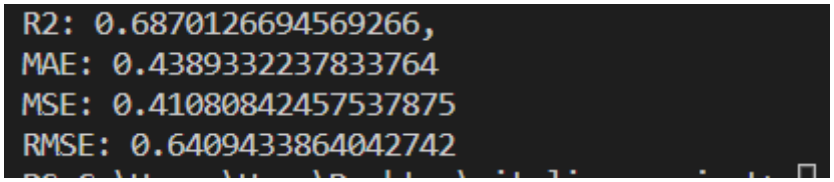


Рисунок 2.6 Вивід кінцевих похибок

Як видно на рисунку 2.6 похибки кардинально не помінялись, хоча тепер наша KNN модель пояснює 69% дисперсій. Це не дуже велике, але все таки покращення.

Висновки до розділу 2

В даному розділі визначили як застосовувати задачі класифікації для вирішування задач регресії. Навчились корегувати вхідні дані для наших потреб, та навчились розв’язувати задачу регресії.

Висновки

Метою дослідження моєї бакалаврської роботи було вивчення та систематизація знань про методи класифікації. На даний момент існують безліч методів класифікації від простих, до дуже складних та громістких. Методи класифікації мають безліч застосувань в різних галузях людського життя. Вони використовуються від камери в телефоні, яка фокусується на обличчі власника, до знаходження аномальних утворень на рентгенівських знімках. На мою думку якомога більше людей повинні познайомитись з методами класифікації, та й з машинним навчанням.

При написані бакалаврської роботи я систематизував власні знання за темою “Машинне навчання та методи класифікації”. Шукаючи матеріал за темою моєї дипломної роботи, я впевнився в актуальності цієї теми. Озираючись на тему дипломної роботи, я розумію, що дослідив ще не все, і ця тема набагато більша ніж я уявляв спочатку.

Мною були розглянуті такі алгоритми, як метод опорних векторів (SVM) та метод k – найближчих сусідів (KNN). Дані алгоритми, хоч і не без недоліків, проте їхня простота в реалізації та легкість в застосуванні дозволяє їх і надалі бути популярними. Я показав як переваги, так і недоліки цих методів. SVM метод показав себе краще для задач класифікації з лінійною та нелінійною роздільністю. Проте сам алгоритм може вимагати налаштування гіперплощини, та й обмеження на розмір вибірки. З іншого боку KNN є простим в реалізації та не вимагає в навчанні, як той же метод опорних векторів, але на практиці він залежить від правильно розміченої вибірки.

З завершенням роботи над дипломною роботою в мене залишився великий інструментарій, з методів класифікації. Які виявляються корисними навіть при повсякденному життю. А з розвитком глибоких нейронних я вважаю що можна очікувати збільшення якості методів класифікації.

Список використаної літератури

1. Anita C. Faul A Concise Introduction to Machine Learning, 2020. 123c.
2. Corinna Cortes, Mehryar Mohri, Afshin Rostamizadeh. L2 Regularization for Learning Kernels. 2012. 65 c.
3. Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani An Introduction to Statistical Learning with Applications in R, 2015. 144 c.
4. Jain, Murty, Flynn Data clustering: a review, 1999.
5. John R. Koza Forrest H. BennettIII David Andre Martin A. Keane Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming, 1996.
6. Krzysztof Jajuga, Krzysztof Najman, Marek Walesiak Data Analysis and Classification, 2021. 132 c.
7. Max Kuhn, Kjell Johnson "Applied Predictive Modeling" 2013. 40 c.
8. Pace, R. Kelley, and Ronald Barry, "Sparse Spatial Autoregressions," Statistics and Probability Letters, Volume 33, Number 3, May 5 1997 . 291-297 c.
9. scikit-learn Documentation URL: https://scikitlearn.org/stable/documentation.html
10. Xin Rong. Parameter Learning Explained. 2016. 87 c.