НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра математичного моделювання та аналізу даних

«До захисту допущено»
В.о. завідувача кафедри
_____Іван ТЕРЕЩЕНКО
«__»_____2023 р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності: 113 Прикладна математика

на тему: **«Алгоритми машинного навчання для прогнозування відтоку користувачів»**

Виконав: студент 4 курсу, групи ФІ-92
Янушев Антон Олександрович

Керівник:доктор філософії, старший викладач кафедри ММАД Г. О. Яйлимова _____

Рецензент: д.т.н., професор кафедри ММЗІ Кудін А. М. _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань

Студент _____

Київ — 2023

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра математичного моделювання та аналізу даних

Рівень вищої освіти — перший (бакалаврський)

Спеціальність: 113 Прикладна математика,

Освітня програма: «математичні методи моделювання, розпізнавання образів та комп'ютерного зору»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____Іван ТЕРЕЩЕНКО
«__»_____2023 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Янушев Антон Олександрович

1. Тема роботи: «*Алгоритми машинного навчання для прогнозування відтоку користувачів*»,

керівник: с.в. Яйлимова Г.О.,

затверджені наказом по університету №____від «__»_____2023 р.

2. Термін подання студентом роботи: «____»_____2023 р.

3. Вихідні дані до роботи: *опубліковані джерела за тематикою дослідження.*

4. Об'єкт дослідження. Методи машинного навчання для прогнозування відтоку клієнтів (churn) в компанії.

5. Предмет дослідження. Різні алгоритми машинного навчання, що використовуються для прогнозування відтоку клієнтів в різних компаніях та галузях.

6. Мета дослідження. Дослідити ефективність різних алгоритмів машинного навчання у прогнозуванні відтоку клієнтів та визначити найбільш ефективні алгоритми для застосування у бізнесі.

7. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): *презентація доповіді.*

8. Дата видачі завдання: 15 листопада 2022 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	15-30 листопада 2022 р.	Виконано
2	Огляд опублікованих джерел за тематикою дослідження	Листопад-Грудень 2022 р.	Виконано
3	Вибір моделі та метрик	Січень-лютий 2023 р.	Виконано
4	Дослідження моделі XGBoost	15 січня - 10 лютого 2023 р.	Виконано
5	Реалізація та тестування моделі	Лютий 2023 р.	Виконано
6	Аналіз та оформлення результатів	Березень 2023 р.	Виконано

Студент _____ Янушев А.О.

Керівник _____ Яйлимова Г.О.

РЕФЕРАТ

Кваліфікаційна робота містить: 55 стор., 25 рисунки, 1 таблиця, 7 джерел.

Мета роботи полягає в створенні конкурентної моделі машинного навчання для прогнозування відтоку користувачів для бізнесу різних сфер.

Об’єктом дослідження є методи машинного навчання для прогнозування відтоку клієнтів (churn) в компанії.

У даному дослідженні проаналізовано різні алгоритми машинного навчання, що використовуються для прогнозування відтоку клієнтів в різних компаніях та галузях. Досліджено ефективність різних алгоритмів машинного навчання у прогнозуванні відтоку клієнтів та визначено найбільш ефективні алгоритми для застосування у бізнесі.

Практичне значення роботи полягає в оцінці відтоку користувачів що дає змогу своєчасно реагувати та утримувати клієнтів.

MACHINE LEARNING, CHURN, XGBOOST, DECISION TREE

ABSTRACT

Qualification work contains: 55 pages, 25 figures, 1 table, 7 sources.

The aim of the work is to create a competitive machine learning model for predicting customer churn for businesses in various industries.

The object of the research is machine learning methods for predicting customer churn in a company.

In this research, various machine learning algorithms used for predicting customer churn in different companies and industries have been analyzed. The effectiveness of different machine learning algorithms in predicting customer churn has been investigated, and the most effective algorithms for business applications have been identified.

The practical significance of the work lies in assessing customer churn, which allows for timely response and customer retention.

MACHINE LEARNING, CHURN, XGBOOST, DECISION TREE

ЗМІСТ

ВСТУП

На даний момент однією з головних задач бізнесу є утримання клієнтів та запобігання їх відтоку (churn). Відтік користувачів має негативні як фінансові, так і репутаційні наслідки. Тому розробка ефективних методів прогнозування відтоку є дуже актуальною задачею на сьогоднішній день. Не є виключенням і сфера комп'ютерних ігор яку надалі ми будемо розглядати.

Під час проходження переддипломної практики був проведений огляд літератури, досліджені основні аспекти відтоку гравців, розглянуто існуючі підходи, моделі та методи машинного навчання для прогнозування відтоку, а також вибір найбільш ефективної моделі.

Для досягнення цілі, була розроблена методика досліджень, що містить збір та передобробку даних, вибір та побудова найбільш точної моделі прогнозування, її навчання та оцінку за декількома метриками.

Дане дослідження має певні обмеження, що будуть описані надалі. Проте воно дає можливість для подальших досліджень у сфері прогнозування відтоку гравців в комп'ютерні ігри та розвитку більш точних та ефективних моделей машинного навчання.

РОЗДІЛ 1

Теоретичні відомості. Поняття churn та churn rate

Що таке churn?

Churn – це метрика, яка вимірює кількість користувачі, які перестали платити і, за розширенням, використовувати ваш продукт або послугу протягом певного періоду часу. Це поширений термін у компаніях SaaS та інших підприємствах, що працюють на основі підписки, де користувачі платять регулярно за використання продукту або послуги**.**

Що таке churn rate?

Churn rate визначається як відсоток користувачів, виміряний на початку певного періоду часу, наприклад, місяця, кварталу або року.

Higher churn rate означає, що більше клієнтів залишає ваш бізнес. Зворотно, низький показник відтоку означає збереження більшої кількості клієнтів, ніж вже є. Розуміння цієї різниці може бути корисним при прийнятті кращих стратегічних рішень для вашого бізнесу.

Чому churn і churn rate важливі?

Churn є постійною проблемою для компаній, що надають послуги на основі підписки, де конкурують кілька компаній і легко змінити одну на іншу. Розрахунок та відстеження показника відтоку можуть допомогти організаціям оцінити, як вони впораються з конкурентами і внести корективи в стратегії для зміцнення лояльності передплатників.

Згідно з різними дослідженнями, приваблення нових клієнтів коштує в п'ять-сім разів більше, ніж збереження поточних клієнтів. Існуючі клієнти набагато більш ймовірно купують продукти, ніж нові клієнти; згідно зі стандартною метрикою, ймовірність продажу продукту поточному клієнту становить 60-70%, в той час як для нового клієнта ймовірність продажу становить лише 5-20%. Відстеження показників відтоку клієнтів в електронній комерції може допомогти компаніям розробляти стратегії для підвищення та збереження клієнтів, тим самим знижуючи витрати, які зазвичай пов'язані з приваблення нових клієнтів.

Переваги і недоліки churn rate.

Перевагою розрахунку churn rate компанії полягає в тому, що це дає чіткість щодо того, наскільки добре бізнес утримує клієнтів, що є відображенням якості наданої компанією послуги та її корисності.

Якщо компанія бачить, що показник відтоку збільшується з періоду на період, то вона розуміє, що фундаментальний компонент її бізнес-процесів є недосконалим. Можливо, компанія надає непридатний продукт, має поганий сервіс для клієнтів, або її продукт не привабливий для осіб, які вважають, що його вартість не виправдовує його корисність.

Показник відтоку покаже компанії, що їй потрібно зрозуміти, чому клієнти покидають її, та визначити напрямки для виправлення бізнесу. Витрати на приваблення нових клієнтів значно вищі, ніж витрати на збереження поточних клієнтів, тому важливо, щоб клієнти, на привернення яких було витрачено багато зусиль, залишалися платоспроможними клієнтами. Розуміння якості бізнесу стає раціональним кроком.

Недоліки (фактори відтоку)

Одним з обмежень churn rate є те, що він не враховує типи клієнтів, які покидають. Втрата клієнтів переважно спостерігається серед найновіших отриманих клієнтів.

1.1. Фактори, що впливають на відтік

Можливо, компанія проводила нещодавню акцію, яка привернула нових клієнтів. Однак, як тільки акція закінчилася, або навіть якщо переваги акції не припинилися, клієнти, які випробовували продукт, можуть вирішити, що він не підходить для них і скасувати підписку.

Втрата нових клієнтів порівняно з довгостроковими клієнтами має вирішальне значення. Нові клієнти є тимчасовими, тоді як старі клієнти є стійкими і вже насолоджувалися вашим продуктом, тому для їх втрати повинно бути більш серйозне обґрунтування. Високий показник відтоку за один період

може бути свідченням високого темпу зростання у попередньому періоді, а не оцінкою якості бізнесу.

Churn rate також не надає справжнього порівняння між компаніями в одній галузі. Більшість нових компаній матимуть високий показник залучення нових клієнтів, оскільки нові люди випробовують бізнес, але вони також матимуть вищий показник відтоку, оскільки ці нові клієнти покидають.

Компанія, яка має десятиліття досвіду, матиме низький показник відтоку, оскільки її клієнти вже встановлені, але її показник залучення нових клієнтів також буде нижчим. Порівняння показників відтоку для таких компаній буде некоректним, оскільки контекст та умови їх діяльності значно відрізняються від нових компаній.

Крім того, показник відтоку не враховує причини втрати клієнтів. Він надає загальну статистику без деталей про те, чому саме клієнти вирішили покинути компанію. Це обмеження утруднює компанії розуміння суті проблеми та розробку ефективних стратегій для поліпшення ситуації.

Крім того, показник відтоку не враховує контекстуальних факторів, які можуть вплинути на відтік клієнтів. Наприклад, економічні чинники, конкуренція на ринку або зміни в споживацькому поведінці можуть значно вплинути на рішення клієнтів про покидання компанії. Без врахування цих факторів порівняння показників відтоку між компаніями може бути неповним та не об'єктивним.

1.2. Існуючі підходи прогнозування відтоку

У даному розділі дослідження ми звернемо увагу на різні існуючі підходи до прогнозування відтоку клієнтів. Відток є однією з найбільш актуальних проблем для багатьох компаній, оскільки утримання існуючих клієнтів має більшу цінність, ніж приваблення нових. У зв'язку з цим, розробка ефективних моделей прогнозування відтоку стала невід'ємною частиною стратегії утримання клієнтів.

У даному розділі ми розглянемо різні підходи до прогнозування відтоку, включаючи регресійні моделі, дерева прийняття рішень, алгоритми градієнтного бустингу та нейронні мережі. Кожен з цих підходів має свої особливості, переваги і обмеження, і вибір оптимального методу залежить від конкретного контексту та цілей дослідження.

1.2.1 Логістична регресія

Логістична регресія є одним із найпопулярніших алгоритмів машинного навчання, який відноситься до техніки навчання під керівництвом. Він використовується для прогнозування категоріальної залежної змінної з використанням заданого набору незалежних змінних.

Логістична регресія передбачає вихід категоріальної залежної змінної. Тому результат повинен бути категоричним або дискретним значенням. Це може бути «Yes» або «No», «0» або «1», «True» або «False» тощо, але замість того, щоб давати точне значення як 0 і 1, воно дає ймовірнісні значення, які знаходяться між 0 і 1.

Логістична регресія багато в чому схожа на лінійну регресію, за винятком того, як вони використовуються. Лінійна регресія використовується для вирішення проблем регресії, тоді як логістична регресія використовується для вирішення проблем класифікації.

Використання логістичної регресії для churn prediction також спирається на результат «0» або «1», тобто вона використовує логістичну функцію для класифікації клієнтів на «Churn» або «No Churn».

Кожна регресійна модель може бути записана у вигляді формули:

$$y = F(x_1, x_2, \ldots, x_n)$$

У множинній лінійній регресії передбачається, що залежна змінна є лінійною функцією незалежних змінних, тобто:

$$y = a + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n)$$

Для вирішення проблему регресії можна сформулювати інакше:

Замість прогнозування бінарної змінної, ми прогнозуємо неперервну змінну зі значеннями на відрізку [0,1]

$$P = \frac{1}{1+e^{-y}}$$

Де P - ймовірність того, що станеться подія що нас цікавить

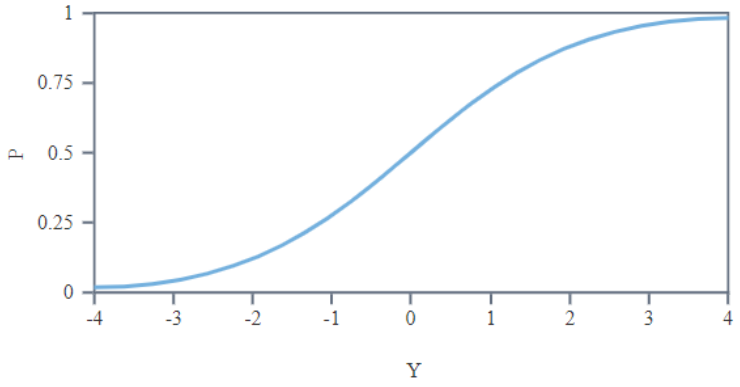


рис 1.1

Робота логістичної регресії базується на наступних кроках:

1) Введення даних: Завантаження та передобробка датасетів де кожен клієнт представлений набором ознак таких як довжина сессій, регулярність використання ресурсу, гри, продукту, тощо.

2) Навчання моделі: Модель навчається на основі підготованих даних шляхом знаходження оптимальних значень кожної ознаки. Це може бути реалізовано за допомогою методу максимальної правдоподібності чи градієнтного спуску.

3)Прийняття рішень: Після навчання моделі можна використовувати для передбачення ймовірності відтоку нових клієнтів. Якщо передбачена ймовірність перевищує граничне значення, наприклад, 0.5, клієнт буде класифіковано як "відтік", інакше - як "не відтік".

Переваги та недоліки використання логістичної регресії для Churn Prediction:

Переваги:

- Простота реалізації, використання та інтерпретованість
- Ефективність на великих наборах даних

Недоліки:

- Лінійний границя ухвалення рішень
- Обмеженість ознакового простору

1.2.2 Дерево прийняття рішень

Дерево рішень — це непараметричний контрольований алгоритм навчання, який використовується як для завдань класифікації, так і для регресії. Він має ієрархічну структуру дерева, яка складається з кореневого вузла, гілок, внутрішніх вузлів і листових вузлів.

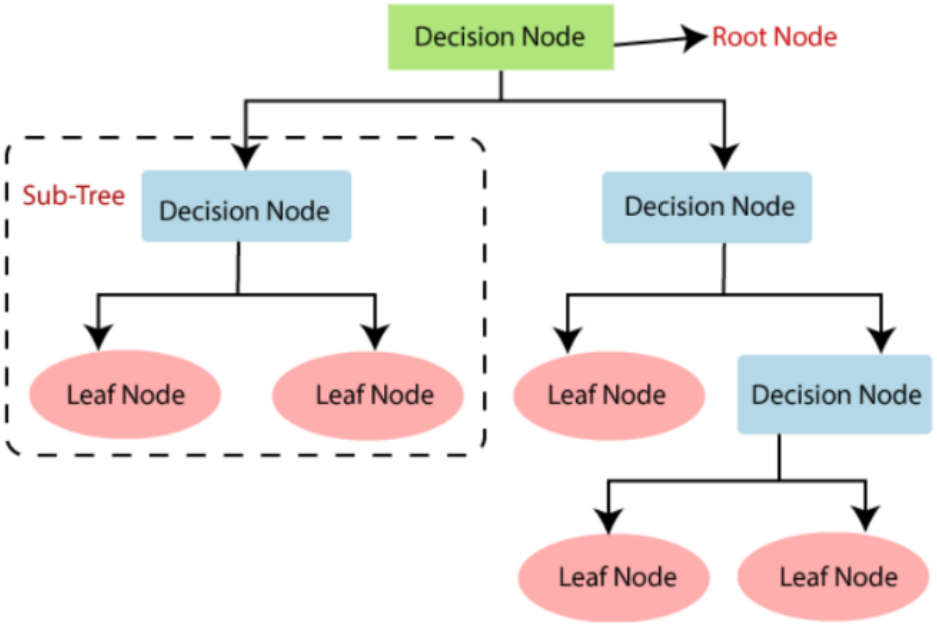


рис 1.2

Основні причини використання дерева рішень:

- Дерево рішень зазвичай імітує здатність людини мислити під час прийняття рішення, тому це легко зрозуміти.
- Логіку дерева рішень можна легко зрозуміти, оскільки воно демонструє деревоподібну структуру.

Термінології дерева рішень:

Кореневий вузол: кореневий вузол – це місце, де починається дерево рішень. Він представляє весь набір даних, який далі ділиться на два або більше однорідних наборів.

Листовий вузол: листові вузли є кінцевим вихідним вузлом, і дерево не може бути відокремлено далі після отримання листкового вузла.

Розділення: Розділення — це процес поділу вузла рішення/кореневого вузла на підвузли відповідно до заданих умов.

Гілка/піддерево: дерево, утворене розщепленням дерева.

Обрізка: обрізка - це процес видалення непотрібних гілок з дерева.

Батьківський/дочірній вузол: кореневий вузол дерева називається батьківським вузлом, а інші вузли називаються дочірніми вузлами.

Дерева прийняття рішень є дуже популярним вибором для прогнозування відтоку. Вони будують дерево на основі ознак клієнтів та їх значень.

Теоретико-інформаційний критерій

Як випливає з назви, критерій заснований на поняттях теорії інформації, а саме — інформаційної ентропії:

$$H = -\sum_{i=1}^{n} \frac{N_i}{N} \log\left(\frac{N_i}{N}\right)$$

Статистичний підхід

В основі статистичного підходу лежить використання індексу Джині

Індекс Джини фактично показує відстань між двома розподілами — розподілом цільових значень і розподілом передбачень моделі. Відповідно, чим менше дана відстань, тим краще працює модель.

Індекс Джини може бути розрахований за формулою:

$$Gini(Q) = 1 - \sum_{i=1}^{n} p_i^2$$

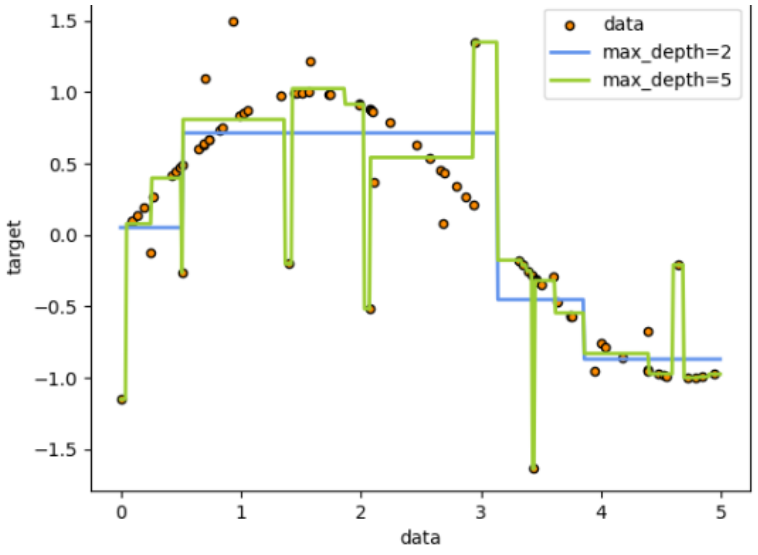


рис 1.3

Робота дерева прийняття рішень базується на наступних кроках:

1) Введення даних: передобробка та представлення даних в виді ознак і цільової змінної (відтік чи не відтік).

2) Розбиття даних: Дерево починається з кореневого вузла в якому здійснюється розбиття даних на основі певної ознаки та її значення.

3) Побудова дерева: рекурсивно продовжується розбиття даних в кожному вузлі дерева за різними ознаками поки не буде досягнуто критерій зупинки, до прикладу: мінімальна кількість об’єктів в вузлу, максимальна глибина дерева.

4) Прийняття рішень: Коли дерево збудовано, клієнти можуть бути класифіковані шляхом проходження через дерево від кореневого вузла до листового вузла, де приймається кінцеве рішення щодо відтоку.

Переваги та недоліки використання дерев прийняття рішень

Переваги:

- Обробка нелінійних залежностей
- Стійкість до шуму та викидів даних
- Логіка та інтерпретованість прийняття кожного рішення

Недоліки:

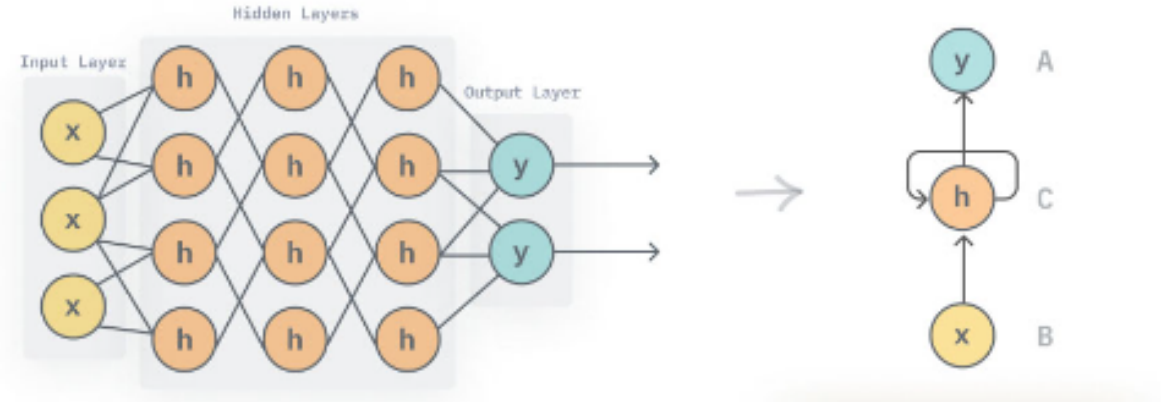
- Недостатня узагальнююча здатність(цю проблему вирішують такі моделі як випадковий ліс або градієнтний бустинг, про них поговоримо далі).

1.2.3 Рекурентні нейронні мережі

Рекурентні нейронна мережа (RNN) — це тип штучної нейронної мережі, яка використовує послідовні дані або дані часових рядів. Ці алгоритми глибокого навчання зазвичай використовуються для порядкових або часових проблем.

Вони відрізняються своєю «пам’яттю», оскільки вони беруть інформацію з попередніх вхідних даних, щоб впливати на поточні вхідні та вихідні дані. Хоча традиційні глибокі нейронні мережі припускають, що входи та виходи незалежні один від одного, вихід рекурентних нейронних мереж залежить від попередніх елементів у послідовності. Хоча майбутні події також були б корисними для визначення результату даної послідовності, односпрямовані рекурентні нейронні мережі не можуть врахувати ці події у своїх прогнозах.

Рекурентні нейронні мережі також використовуються для аналізу послідовностей даних та широко застосовуються в задачах прогнозування відтоку. Вони можуть вловлювати залежності між послідовними подіями та зберігати контекстну інформацію.



Рекурентні нейронні мережі один з найкращих виборів для Churn prediction, вони можуть аналізувати часові ряди без передобробки, такий формат даних найчастіше використовується для прогнозування відтоку

Робота RNN базується на наступних кроках:

1) Введення послідовних даних: Підготовка даних, таких як послідовність дій клієнтів у часі.

2) Рекурентні зв’язки: RNN використовує рекурентні шари для передачі інформації про попередні стани в наступні.

3) Зворотне поширення помилки: ід час навчання RNN використовує зворотне поширення помилки для коригування ваги та поліпшення прогнозів.

4) Прогнозування відтоку: після навчання RNN може бути використана для передбачення відтоку нових послідовностей дій клієнтів.

Переваги та недоліки використання рекурентних нейронних мереж:

Переваги:

- Облік послідовної інформації
- Моделювання залежностей у часі
- Автоматичне вилучення ознак
- Гнучкість у роботі з різними типами даних

Недоліки:

- Високі обчислювальні вимоги
- Потреба в дуже великій кількості даних для навчання
- Проблема загасаючого/вибухового градієнта
- Довгострокова залежність
- Не інтерпретованість

1.3 Gradient Boosting

Gradient Boosting — це популярний алгоритм підвищення в машинному навчанні, який використовується для завдань класифікації та регресії. Підвищення — це один із методів ансамблевого навчання, який послідовно навчає модель, і кожна нова модель намагається виправити попередню модель. Він поєднує кількох слабких учнів у сильних. Існує два найпопулярніших алгоритму підвищення:

- AdaBoost
- Gradient Boosting

Gradient Boosting — це потужний алгоритм підвищення, який поєднує кількох слабких учнів у сильних, у якому кожна нова модель навчена мінімізувати функцію втрат, таку як середня квадратична помилка або крос-ентропія попередньої моделі за допомогою градієнтного спуску. На кожній ітерації алгоритм обчислює градієнт функції втрат щодо прогнозів поточного ансамблю, а потім навчає нову слабку модель, щоб мінімізувати цей градієнт. Прогнози нової моделі потім додаються до ансамблю, і процес повторюється, доки не буде виконано критерій зупинки.

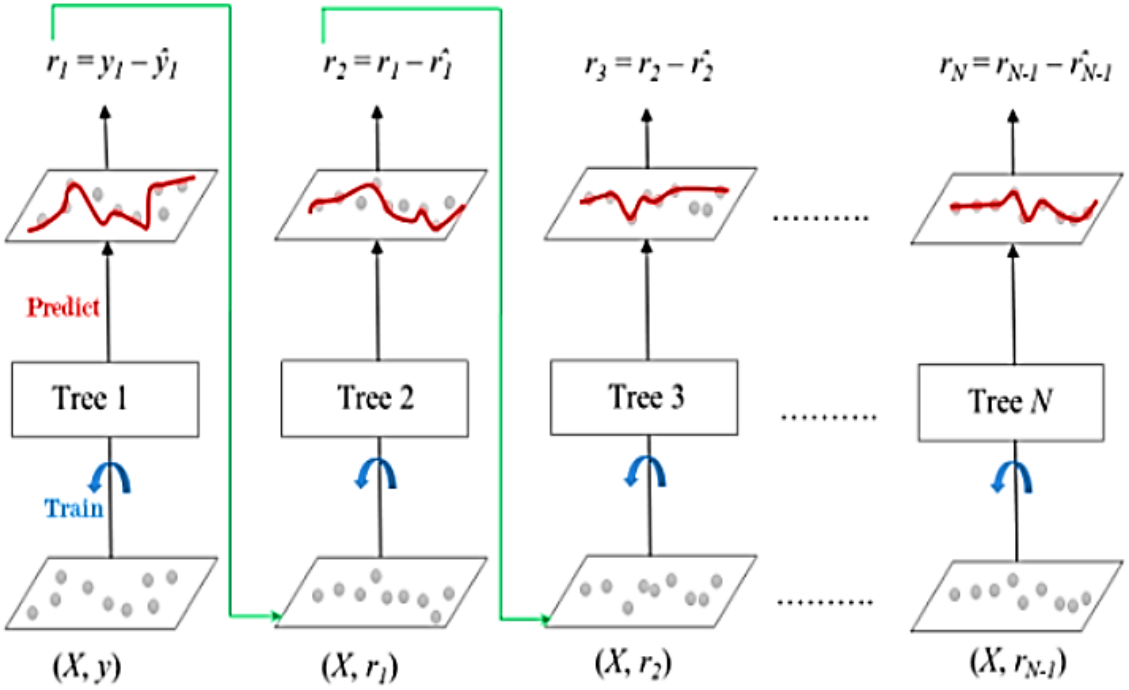


рис 1.4

На відміну від AdaBoost, ваги екземплярів навчання не змінюються, замість цього кожен предиктор навчається з використанням залишкових помилок попередника як міток. Існує техніка під назвою Gradient Boosted Trees, базовим учням якої є CART (дерева класифікації та регресії). На наведеній нижче діаграмі пояснюється, як підсилені градієнтом дерева навчаються для проблем регресії.

Gradient Boosting є одним з найкращих рішень для прогнозування відтоку. Він комбінує слабкі моделі, такі як дерева рішень, і дозволяє обробляти великі обсяги даних із великою кількістю ознак.

Робота Gradient Boosting базується на наступних кроках:

1) Введення даних:Починається з підготовки та представлення даних у вигляді ознак і цільової змінної (відтік або не відтік)

2) Ініціалізація моделі: Gradient Boosting починається з ініціалізації базової моделі. Зазвичай це проста модель, наприклад, одне дерево рішень

3) Навчання моделі:

- Передбачення та залишки
- Навчання наступної моделі
- Оновлення передбачень
- Вагові оновлення

4) Фінальне передбачення: Після навчання всіх моделей у послідовності, фінальне передбачення відтоку нових клієнтів виходить шляхом комбінування передбачень всіх моделей з урахуванням їх ваг.

Процес навчання Gradient Boosting продовжується до досягнення певного критерія зупинки таких як: максимальна кількість дерев або досягнення оптимальної кількості передбачень

Переваги та недоліки використання Gradient Boosting

Переваги:

- Висока точність
- Можливість обробляти великі обсяги даних
- Стійкість до шуму та викидів даних
- Наочність результатів

Недоліки:

- Можливість перенавчання
- Обчислювальна складність
- Необхідність налаштування гіперпараметрів

Тож, час досконало розібратися як працює градієнтний бустинг та XGBoost, адже саме ці моделі ми і будемо використовувати для вирішення проблеми Churn prediction.

1.3.1 Алгоритм Шапіро

Для початку розглянемо класичний алгоритм Шапіро

Процедура бустингу для задачі класифікації

Вхідні дані: вибірка розміру N: M = $\{m_1, m_2, \ldots, m_N\}$ де $m_i = (x_i, y_i)$, $x_i \in \chi$ та $y_i \in$ {-1,1}

Вихід: Класифікатор H: $\chi \rightarrow \{-1, 1\}$

- Довільно вибрати без заміни L_1 < N вибірок із M та отримати M_1^* .
- Використовуючи слабкого учня для M_1^* , отримаємо класифікатор H_1
- Виберемо L_2 < N вибірок із M для половини вибірок, неправильно класифікованих за H_1, щоб отримати M_2^*.
- Використовуючи слабкого учня для M_2^* , отримаємо класифікатор H_2.
- Виберемо всі зразки з M, на яких H_1 та H_2 не збігаються, отримуючи M_3^*.
- Використовуючи слабкого учня для M_3^* , отримаємо класифікатор H_3.
- Отримаємо остаточний класифікатор більшістю голосів:

$$H(x) = \text{sign}\left(\sum_{b=1}^{3} H_b(x)\right).$$

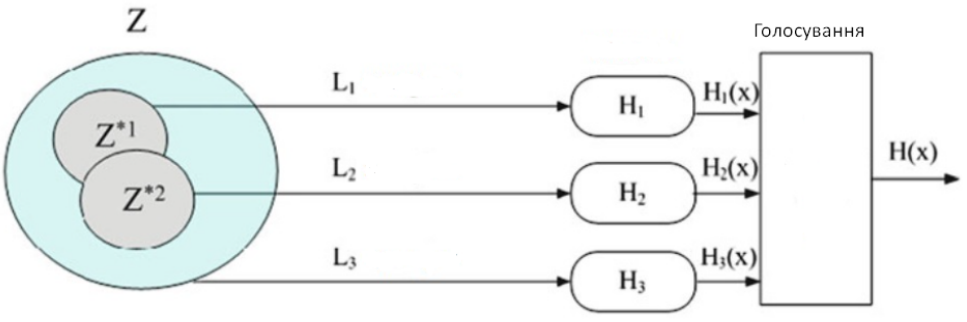


рис 1.5

Як видно з алгоритму Шапіро, навчальний набір випадковим чином поділяється без повернення на три множини, M_1^*, M_2^*, M_3^*. Якщо перші два класифікатори (H_1 і H_2) згодні з міткою класу при визначенні приналежності елемента, то це є остаточним рішенням. Множина прикладів, щодо яких вони не згодні, визначає розбиття M_3^*, яке використовується для навчання H_3. Шапіро показав, що цей метод навчання дозволяє побудувати потужний класифікатор.
Цього достатньо для розуміння роботи бустингу для задачі класифікації.

1.3.2 Градієнтний бустинг

Розглянемо загальну задачу навчання з учителем.
По вибірці $\{(x_i, y_i)\}_{i=1}^n$ потрібно встановити залежність $y = f(x)$. Розв'язок $\widehat{f}$ будемо шукати з умови мінімізації середньої помилки

$$\varepsilon(f) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)),$$

де, $L()$ - функція втрат(до прикладу, середньоквадратична похибка), тоді
$\widehat{f}=arg\ min\ \varepsilon(f)\ \ f \in F$

тут F якийсь клас функцій, званий класом моделі, що може бути лінійними методами та методами локальної регресії.

Серед методів вирішення оптимізаційних завдань, одними з найпопулярніших є градієнтний спуск.
Розглянемо як цей метод можна адаптувати до задачі бустингу.
Розглянемо ансамбль моделей

$$f(x) = \sum_{m=1}^{M} f_m(x),$$

які можуть бути представлені як моделі адаптивних базисних функцій

$$f(x) = \theta_0 + \sum_{m=1}^{M} \theta_m \phi_m(x),$$

де $f_0(x) = \theta_0$, $f_m(x) = \theta_m \phi_m(x)$, m=1,...,M та ϕ_m - послідовне додавання базисних функцій поліпшення відповідності поточної моделі. Це безліч слабких класифікаторів (учнів).

Надалі залишається використати алгоритм для мінімізації помилки.Спочатку використовуємо градієнтний спуск, тобто шукаємо градієнт помилки і використовуючи ітераційні оцінки θ_m будемо рухатись вздовж нього.

1.3.3 Алгоритм посилення градієнта

Функція втрат: L

Число ітерацій: M

Характеристика швидкості навчання: η

1. Ініціалізація

$$\hat{f}^{(0)}(x) = \hat{f}_0 = \theta_0 = argmin \sum_{i=1}^{n} L(y_i, \theta).$$

2. $for\ m = 1\ to\ M$:

   Порахуємо

$$\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f^{(m-1)}(x)},$$

$$\hat{\phi}_m = argmin_{\beta} \sum_{i=1}^{n} [(-\hat{g}_m(x_i)) - \beta\phi(x_i)]^2, \phi \in \Phi,$$

$$\hat{p}_m = argmin_{p} \sum_{i=1}^{n} L(y_i, f^{(m-1)}(x_i) + p\hat{\phi}_m(x_i)),$$

$$\hat{f}_m(x) = \eta\hat{p}_m\hat{\phi}_m(x),$$

$$\hat{f}^{(m)}(x) = \hat{f}^{(m-1)}(x) + \hat{f}_m(x)$$

3. Класифікатор що вийшов в результаті: $\widehat{f}(x) = \widehat{f}^{(M)}(x) = \sum_{m=0}^{M} \widehat{f}_m(x)$.

$\widehat{g}_m(x_i)$ - *від'ємний градієнт*

$\widehat{\phi}_m$ - *слабкий класифікатор*

$\widehat{p}_m$ - *довжина кроку градієнтного спуску*

До речі, коефіцієнт навчання η змінює сходимість

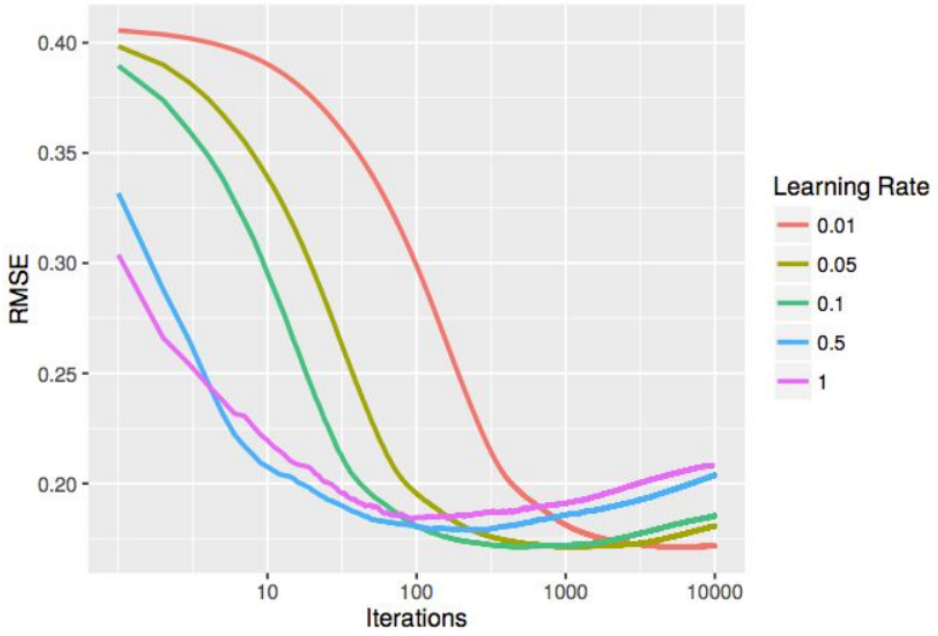


рис 1.6

Одним з найбільш популярних методів використання бустинга є побудова дерев з його застосуванням.

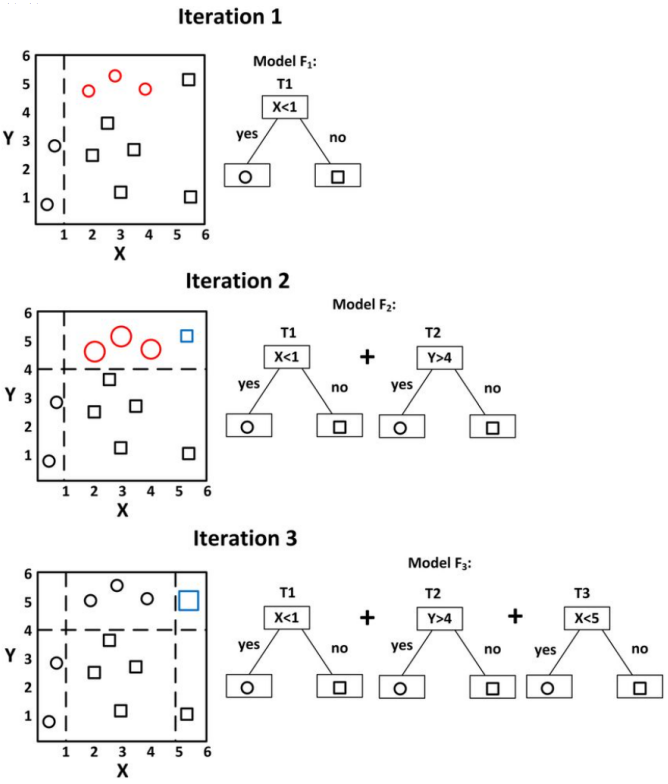


рис 1.7

Ми дослідили як працює градієнтний бустинг, в свою чергу XGBoost різними засобами покращує градієнтний бустинг, фокусуючись саме на швидкості обчислення та масштабах моделі.

Головні покращення що вносить XGBoost:

1. Робота з різними функціями втрат: XGBoost дозволяє використовувати різні функції втрат залежно від завдання (наприклад, бінарна класифікація, багатокласова класифікація, регресія).
2. Регуляризація: XGBoost підтримує L1 та L2 регуляризацію для контролю складності моделі та запобігання перенавчанню.
3. Оптимізації продуктивності: XGBoost використовує різні техніки оптимізації, такі як алгоритми наближеного навчання, паралельне навчання на кількох ядрах CPU, підтримка розподіленого навчання на кластерах та GPU.
4. XGBoost пропонує багатий набір гіперпараметрів, які можна налаштувати для досягнення найкращої продуктивності моделі. Однак, хоча XGBoost має безліч переваг і зазвичай забезпечує хорошу точність, він також вимагає

акуратного налаштування гіперпараметрів і може бути більш вимогливим до ресурсів

1.3.4 Математика XGBoost

$L^{(t)} = \sum_{i=1}^{n} l(y_i, \widehat{y}_i^{(m-1)} + f_t(x_i)) + \Omega(f_t)$ - функція для оптимізації градієнтного бустингу, де:

l - функція втрат.

$y_i, \widehat{y}_i^{t}$ - значення і-го елемента навчальної вибірки та сума передбачень перших t дерев відповідно.

f_t - функція (у нашому випадку дерево), яку ми хочемо навчити на кроці t.

$f_t(x_i)$ - передбачення на і-му елементі навчальної вибірки.

$\Omega(f_t)$ - регулювання функції f.

$\Omega(f) = \gamma T + \frac{1}{2}\lambda||\omega||^2$, де де T - кількість вершин у дереві, ω - значення в листі, а γ та λ - параметри регуляризації.

Далі за допомогою розкладання Тейлора до другого члена можемо наблизити функцію $L^{(t)}$, що оптимізується наступним виразом:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + g_i f_t(x_i) + 0.5 h_i f_t^2(x_i)) + \Omega(f_t)$$

$$g_i = \frac{\partial l(y_i, \hat{y}_i^{t-1})}{\partial \hat{y}_i^{t-1}}, h_i = \frac{\partial^2 l(y_i, \hat{y}_i^{t-1})}{\partial^2 \hat{y}_i^{t-1}}$$

Оскільки ми хочемо мінімізувати помилку моделі на навчальній вибірці, нам потрібно знайти мінімум $L^{(t)}$ для кожного t.

Мінімум цього виразу щодо $f_t(x_i)$ знаходиться в точці $f_t(x_i) = \frac{-g_i}{h_i}$.

1.3.5 Модифікації моделі XGBoost

1. **Тюнинг гіперпараметрів**

Гіперпараметри – це зовнішні змінні зміни, за допомогою яких фахівці за даними керують навчанням моделі МН. Гіперпараметри налаштовуються вручну перед навчанням моделі. Вони відрізняються від параметрів внутрішніх параметрів, що одержуються автоматично в процесі навчання і не налаштовуються фахівцями за даними.

Прикладами гіперпараметрів є кількість вузлів і шарів у нейронній мережі, а також кількість гілок у дереві прийняття рішень. Від гіперпараметрів залежать ключові функції, наприклад, архітектура моделі, темп навчання і складність моделі.

Гіперпараметри можна налаштовувати вручну і намагатися знвйти найбільшу точність, проте це не досить ефективно є певні методи налаштування:

- **Байєсівська оптимізація** - це метод, що базується на використанні теореми Байєса для пошуку оптимальних значень гіперпараметрів моделі. Використовуючи ймовірнісну модель, метод пропонує набір гіперпараметрів, які можуть оптимізувати певний показник якості. Застосовуючи регресійний аналіз, алгоритм ітеративно оновлює ймовірностну модель та вибирає набір гіперпараметрів з найвищою ймовірністю досягнення кращих результатів.

- **Пошук по сітці** - це метод, в якому задається сітка значень гіперпараметрів та обчислюється показник якості для кожної комбінації. Хоча цей підхід є достатньо простим, він може бути обчасливим і часом вимагати значних обчислювальних ресурсів, особливо при наявності багатьох гіперпараметрів.

- **Випадковий пошук** - цей метод також базується на випробуванні різних комбінацій гіперпараметрів, проте замість систематичного перебору він вибирає гіперпараметри випадковим чином. Випадковий пошук може бути ефективним, особливо коли невелика кількість гіперпараметрів має значний вплив на якість моделі.

Обидва методи - пошук по сітці та випадковий пошук - є популярними варіантами для пошуку оптимальних гіперпараметрів, проте вибір конкретного методу залежить від конкретних умов і обмежень дослідження.

2. **Керування класовим дисбалансом**

Незбалансовані дані зазвичай стосуються проблем із класифікацією, коли класи не представлені однаково.
Наприклад, у нас може бути завдання класифікації двох класів (бінарна) зі 100 екземплярами. Загалом 80 екземплярів мають клас 1, а решта 20 екземплярів — клас 2.
Це незбалансований набір даних і співвідношення екземплярів класу 1 і класу 2 складає 80:20 або точніше 4:1.

Для керування класовим дисбалансом існують різні підходи:

- **Зміна порога класифікації**: Можлива зміна порога ймовірності чи вирішального правила визначення класу. Це може бути корисним, якщо важливо підвищити чутливість до меншого класу.

- **Використання зважування класів**: У багатьох алгоритмах машинного навчання можна задати ваги класів. Задаючи більшу вагу меншому класу, можна зробити його більш значущим під час навчання моделі.

Існують ще декілька релевантних підходів для керування класовим дисбалансом, але вони вимагають використання декількох різних моделей на одних даних, зо не підходить для нашого дослідження.

3. **Додавання нових ознак**

Ми просто можемо розробити та додати нові ознаки для об'єктів дослідження, чим більше ознак буде, тим точніше модель навчиться прогнозувати відтік.

4. **Збільшення розміру навчальної вибірки**

За аналогією з попереднім пунктом, чи більше даних ми можемо надати моделі для навчання тим більш точні будуть її прогнози

1.3.6 Метрики оцінки якості моделі

У даному розділі розглянемо важливий аспект оцінки якості алгоритмів прогнозування відтоку - метрики. Одним з ключових завдань в аналізі відтоку є оцінка ефективності моделей, здатних передбачати ймовірність відтоку клієнтів. Метрики надають нам числові показники, які допомагають оцінити продуктивність моделі та порівняти її з іншими алгоритмами.

Кожна метрика має свої особливості та призначена для оцінки різних аспектів моделі. Ми розглянемо Точність, Повнота, F1-міра та AUC-ROC

Точність (Accuracy) - Точність типічна метрика в машинному навчанні.
Це метрика для оцінки продуктивності моделі для класифікацій задач, і вона настільки добре відома, що часто використовується як синонім загальної продуктивності автономної та онлайн-моделі.
Точність - це відсоток правильних класифікацій, яких досягає навчена модель машинного навчання, тобто кількість правильних прогнозів, поділена на загальну кількість прогнозів у всіх класах. Його часто позначають абревіатурою ACC.

ACC повідомляється як значення між [0,1] або [0, 100] залежно від обраної шкали. Точність 0 означає, що класифікатор завжди передбачає неправильну мітку, тоді як точність 1 або 100 означає, що він завжди передбачає правильну мітку.
Хороша характеристика цієї метрики полягає у тому, що вона має прямий зв'язок із усіма значеннями матриці плутанини. Це чотири стовпи контрольованої оцінки машинного навчання: справжні позитивні результати, хибні позитивні результати, справжні негативні значення та хибні негативні результати.

Повнота (Recall) та Точність (Precision):
Повнота, також відомий як істинно позитивний рівень (TPR), є відсотком вибірок даних, які модель машинного навчання правильно ідентифікує як належні класу, що цікавить — «позитивному класу» — від загальної кількості вибірок для цього класу.
Повнота — це метрика, яка використовується для класифікації у навчанні з учителем, і ми можемо поглянути на бінарну класифікацію, щоб краще зрозуміти її.

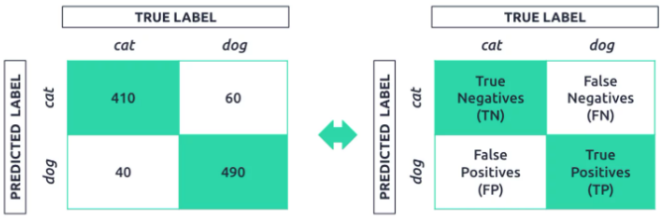


рис 1.8

Матриця плутанини повідомляє інформацію про істинно негативні (TN), хибнонегативні (FN), хибнопозитивні (FP) і істинно позитивні (TP).
Відкликання машинного навчання розраховується на основі цих значень шляхом поділу справжніх позитивних результатів (TP) на все, що мало бути передбачене як позитивне (TP + FN). Формула відкликання у машинному навчанні:

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

Це дає уявлення про чутливість моделі або, простіше кажучи, про можливість того, що фактичний позитивний результат буде позитивним.

Точність (Precision) системи в межах класу – це частка документів, що дійсно належать даному класу щодо всіх документів, які система віднесла до цього класу.

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

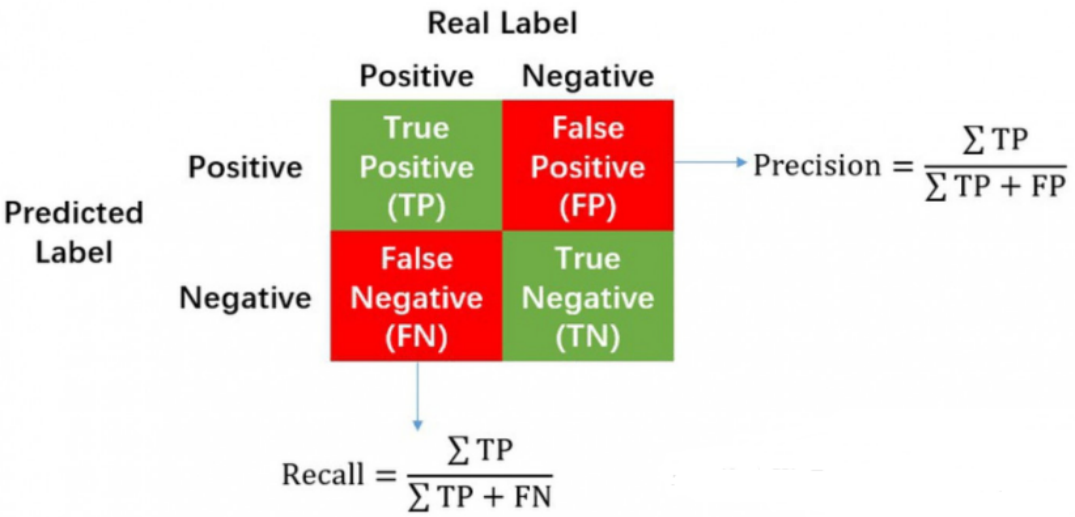


рис 1.9

F1-міра (F1-Score)

Оцінка F1 поєднує точність і повноту класифікатора в єдину метрику, взявши їхнє середнє гармонійне. Він переважно використовується для порівняння продуктивності двох класифікаторів. Припустимо, що класифікатор A більш висока повнота, а класифікатор B більш висока точність. У цьому випадку можна використовувати оцінки F1 для обох класифікаторів, щоб визначити, який із них дає найкращі результати.

Оцінка моделі класифікації F1 розраховується наступним чином:

$$F1 = \frac{2(P*R)}{P+R}$$

AUC-ROC

Коли справа доходить до проблеми класифікації, ми можемо розраховувати на криву AUC-ROC. Коли нам потрібно перевірити або візуалізувати продуктивність завдання класифікації кількох класів, ми використовуємо криву AUC (площа під

кривою) ROC (Робочі характеристики приймача). Це одна з найважливіших оціночних метрик для перевірки продуктивності будь-якої моделі класифікації. Крива AUC - ROC є вимірювання продуктивності для завдань класифікації при різних порогових значеннях. ROC являє собою криву ймовірності, а AUC є ступенем або мірою роздільності. Він каже, наскільки модель здатна розрізняти класи. Чим вище AUC, тим краще модель передбачає 0 класів як 0 і 1 клас як 1.

Крива ROC побудована з TPR залежно від FPR, де TPR знаходиться на осі у, а FPR– на осі х.

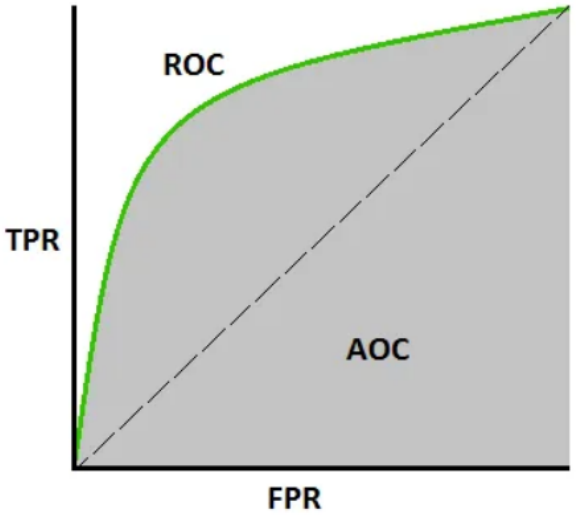


рис 1.10

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

$$Specificity = \frac{TN}{TN+FP}$$

$$FPR = 1 - Specificity = \frac{TP}{TP+FN}$$

Відмінна модель має AUC, буде близькою до 1, що означає добрий ступінь роздільності. Погана модель має AUC близько 0, що означає, що вона має найгірший показник роздільності. Насправді це означає, що він відповідає взаємністю на результат.

Висновки до розділу 1

У результаті вивчення існуючих підходів прогнозування відтоку клієнтів, ми виявили, що різні методи та алгоритми машинного навчання можуть бути ефективними в вирішенні цієї задачі. Регресійні моделі, такі як логістична регресія, дозволяють передбачати ймовірність відтоку на основі набору ознак, що корисно для оцінки важливості факторів, що впливають на відтік. Дерева прийняття рішень надають просту та інтерпретовану модель, засновану на послідовному розбитті даних на підгрупи. Бустінг та нейронні мережі, зі свого боку, надають складніші та потужні моделі, здатні ухопити складні взаємозв'язки між ознаками.

Важливо відзначити, що вибір оптимального підходу залежить від конкретних умов та вимог задачі прогнозування відтоку. Деякі методи можуть бути більш придатними для великого обсягу даних, тоді як інші можуть краще працювати з обмеженою кількістю ознак. Також слід враховувати дисбаланс класів, який часто присутній у даних відтоку, і застосовувати відповідні методи для управління дисбалансом.

У подальшому дослідженні ми зосередимось на алгоритмі градієнтного бустінгу, оскільки він демонструє високу продуктивність та широко використовується в прогнозуванні відтоку. Ми будемо поліпшувати та налаштовувати модель XGBoost, використовуючи різні техніки, такі як налаштування гіперпараметрів, управління дисбалансом класів та оптимізація. Такий підхід дозволить нам отримати найкращі результати та забезпечити точне прогнозування відтоку клієнтів.

В наступному розділі ми розглянемо практичне застосування алгоритму XGBoost для прогнозування відтоку клієнтів на основі наявних даних та оцінимо його ефективність за допомогою різних метрик оцінки якості моделі.

РОЗДІЛ 2.

У даному практичному розділі роботи ми зосередимося на реалізації та оцінці ефективності алгоритму XGBoost у задачі прогнозування відтоку клієнтів. Відтік є однією з найбільш актуальних проблем для багатьох компаній, оскільки збереження існуючих клієнтів має велику цінність порівняно з приваблennям нових. Розробка ефективних моделей прогнозування відтоку стала неодмінною частиною стратегії збереження клієнтів.

У даному розділі ми проведемо конкретні кроки, починаючи з підготовки даних та реалізації алгоритму XGBoost, і закінчуючи оцінкою та аналізом отриманих результатів. Ми також проаналізуємо різні модифікації алгоритму XGBoost, щоб визначити їх вплив на точність передбачень відтоку.

Метою даної практичної роботи є дослідження можливостей та потенціалу алгоритму XGBoost у контексті прогнозування відтоку клієнтів. Ми прагнемо не лише побудувати точну модель, але й зрозуміти, які фактори та параметри можуть вплинути на її продуктивність. Отримані результати дозволять нам зробити висновки щодо застосовності XGBoost у задачі прогнозування відтоку та оцінити його переваги та обмеження.

Далі ми розглянемо деталі кожного етапу, починаючи з підготовки даних, і продемонструємо застосування алгоритму XGBoost для прогнозування відтоку клієнтів.

2.1 Підготовка даних

Для своїх досліджень я використовував декілька наборів даних для розуміння в яких ситуаціях модель діє більш ефективно і що на це впливає.

Тож для початку ми проводимо завантаження даних та переводимо в формат таблиці

	CustomerID	Count	Country	State	City	Zip Code	Lat Long	Latitude	Longitude	Gender	...	Contract	Paperless Billing	Payment Method	Monthly Charges	Total Charges	Churn Label	Churn Value	Churn Score	CLTV	Churn Reason
0	3668-QPYBK	1	United States	California	Los Angeles	90003	33.964131, -118.272783	33.964131	-118.272783	Male	...	Month-to-month	Yes	Mailed check	53.85	108.15	Yes	1	86	3239	Competitor made better offer
1	9237-HQITU	1	United States	California	Los Angeles	90005	34.059281, -118.30742	34.059281	-118.307420	Female	...	Month-to-month	Yes	Electronic check	70.70	151.65	Yes	1	67	2701	Moved
2	9305-CDSKC	1	United States	California	Los Angeles	90006	34.048013, -118.293953	34.048013	-118.293953	Female	...	Month-to-month	Yes	Electronic check	99.65	820.5	Yes	1	86	5372	Moved
3	7892-POOKP	1	United States	California	Los Angeles	90010	34.062125, -118.315709	34.062125	-118.315709	Female	...	Month-to-month	Yes	Electronic check	104.80	3046.05	Yes	1	84	5003	Moved
4	0280-XJGEX	1	United States	California	Los Angeles	90015	34.039224, -118.266293	34.039224	-118.266293	Male	...	Month-to-month	Yes	Bank transfer (automatic)	103.70	5036.3	Yes	1	89	5340	Competitor had better devices

5 rows × 33 columns

рис 2.1

	Age	FrequentFlyer	AnnualIncomeClass	ServicesOpted	AccountSyncedToSocialMedia	BookedHotelOrNot	Target
0	34	No	Middle Income	6	No	Yes	0
1	34	Yes	Low Income	5	Yes	No	1
2	37	No	Middle Income	3	Yes	No	0
3	30	No	Middle Income	2	No	No	0
4	30	No	Low Income	1	No	No	0

рис 2.2

Головне на що ми маємо звернути увагу це звісно стовбчик відтоку(Churn, Target) Він, звісно, потрібен для навчання моделі.

Після завантаження даних проведемо їх огляд та видалимо деякі ознаки які не допоможуть нам в прогнозуванні(також ми можемо додавати ознаки що допоможуть моделі стати більш ефективною)

Після видадення кількість ознак значно зменшилась

	City	Zip_Code	Latitude	Longitude	Gender	Senior_Citizen	Partner	Dependents	Tenure_Months	Phone_Service	...	Device_Protection	Tech_Support	Streaming_TV	Streaming_Movies	Contract	Paperless_
0	Los_Angeles	90003	33.964131	-118.272783	Male	No	No	No	2	Yes	...	No	No	No	No	Month-to-month	
1	Los_Angeles	90005	34.059281	-118.307420	Female	No	No	Yes	2	Yes	...	No	No	No	No	Month-to-month	
2	Los_Angeles	90006	34.048013	-118.293953	Female	No	No	Yes	8	Yes	...	Yes	No	Yes	Yes	Month-to-month	
3	Los_Angeles	90010	34.062125	-118.315709	Female	No	Yes	Yes	28	Yes	...	Yes	Yes	Yes	Yes	Month-to-month	
4	Los_Angeles	90015	34.039224	-118.266293	Male	No	No	Yes	49	Yes	...	Yes	No	Yes	Yes	Month-to-month	

5 rows × 24 columns

рис 2.3

Загалом для забезпечення якісної роботи алгоритму XGBoost необхідно провести передобробку даних. Цей крок включає заповнення пропущених значень, перетворення категоріальних ознак у числовий формат, масштабування числових ознак і видалення непотрібних стовпців.

XGBoost, може і самостійно обробляти відсутні дані, він буде рахувати їх за 0, проте іноді, 0 може бути важливим значенням для певної ознаки, тому просто перетворювать усі пусті значення на 0 ми не можемо, тож будемо використовувати два підходи

- Заміна середнім значенням або медіаною для числових ознак
- Заміна модою для категоріальних ознак

Вибір конкретного методу заповнення пропущених значень залежить від характеру даних, контексту завдання та професійного розсуду дослідника. Важливо враховувати можливі наслідки заповнення пропущених значень та проводити аналіз впливу цього кроку на результати аналізу та моделювання даних.

Після виключення деяких ознак та заповнення пропущених значень ми будуємо матрицю кореляції.

Цей крок має в собі декілька цілей:

- Ідентифікація сильних зв’язків: Матриця кореляції дозволяє ідентифікувати пари змінних, які мають сильний лінійний зв'язок. Це може бути корисно при аналізі даних та розумінні взаємодій між змінними.
- Відбір значущих змінних: Під час виконання завдання прогнозування можна використовувати матрицю кореляції для відбору найбільш значимих змінних. Якщо дві змінні сильно корельовані між собою, то одну з них можна виключити з моделі, щоб уникнути мультиколінеарності.
- Візуалізація: Матрицю кореляції можні візуалізувати в форматі теплової мапи, це дозволяє візуально досліджувати залежності даних і виявляти найбільш значущі зв'язки.

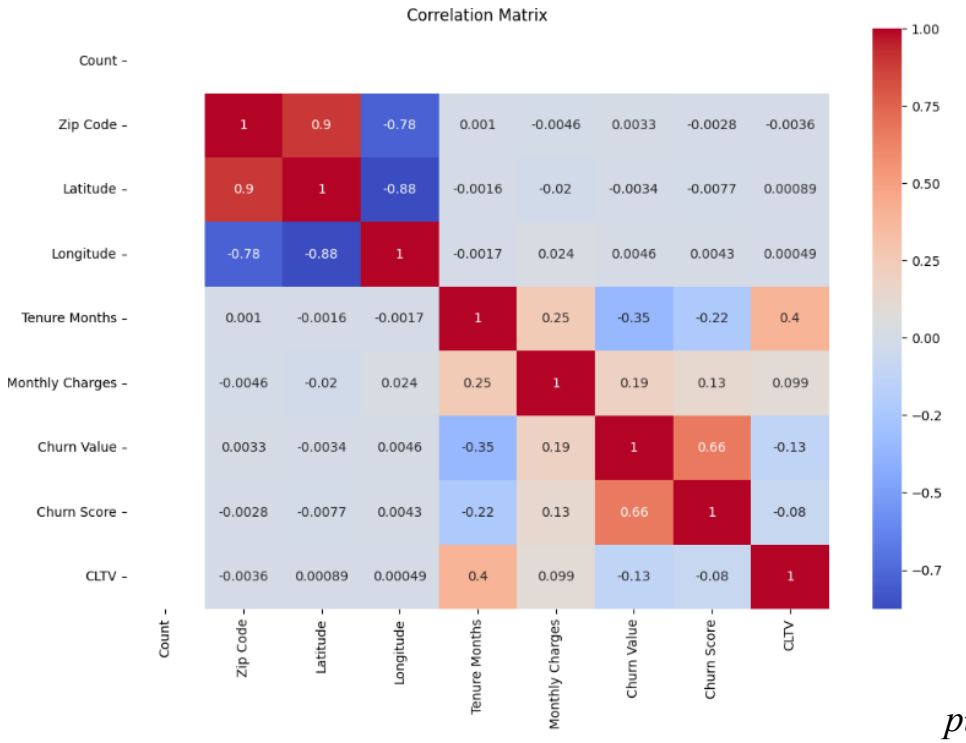


рис 2.4

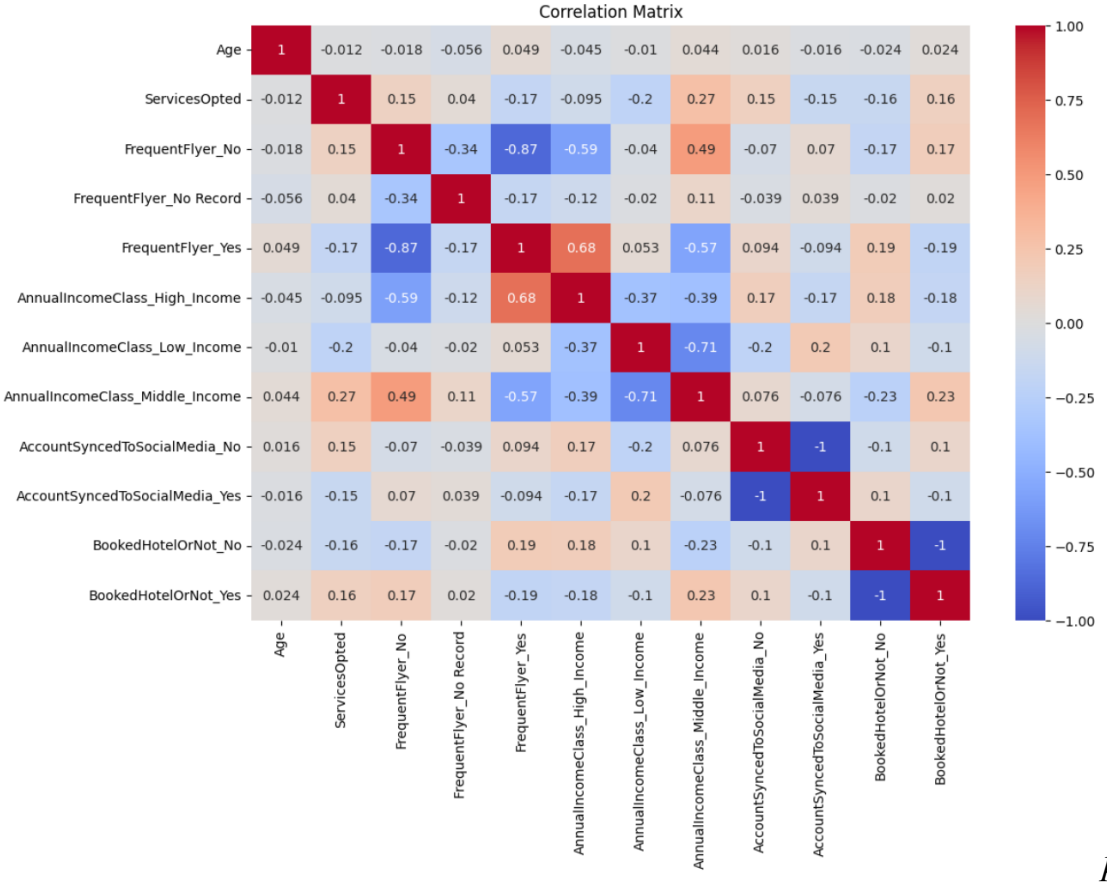


рис 2.5

Можемо помітити, що і в першому і в другому випадку побудова даної матриці не має значущих результатів

Звісно матриця колінеарності буде враховувати лише числові значення ознак, але так як модель XGBoost не може обробляти категоріальні ознаки, ми будемо змінювати їх на числові, тож матриця буде коректною.
Єдиною проблемою є те що часто можна зіткнутися с наборами даних де ознак 100 і більше(в одному з випадків що ми розглянемо їх буде 1178), в таких випадках використання матриць колінеарності буде не доречним.

Після побудови матриці ми знову видаляємо деякі ознаки і можемо переходити до наступного етапу.
Візуалізувати дані ми особливо не будемо, тому-що буквально нічого, ми працюємо не з часовими рядами а з ознаками, тож на графіку можемо побачити лише розподіл клієнтів на тих хто пішли і не пішли від компанії

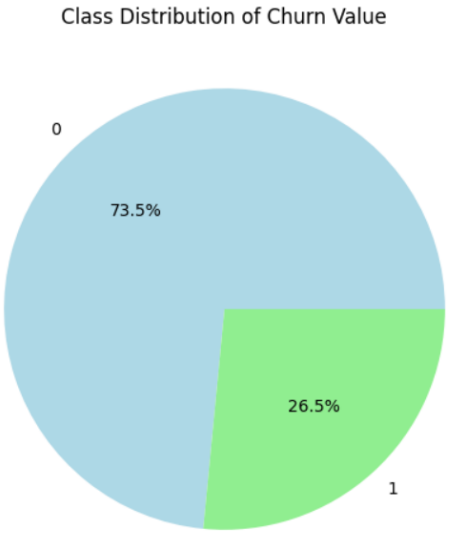
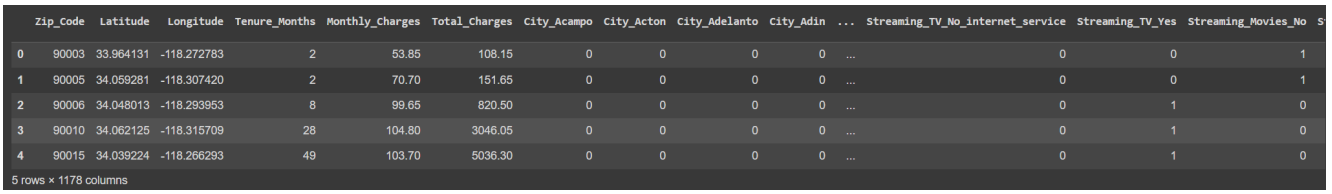


рис 2.6

2.2 Кодування категоріальних ознак

Алгоритм XGBoost вимагає від нас числових даних, але багато ознак має тип даних “object“ тож ми будемо використовувати метот One-Hot Encoding або одноразове кодування, для перетворення ознак в числові значення
Процес One-Hot Encoding полягає у створенні нових бінарних ознак (стовпців) для кожного унікального значення категоріальної ознаки. Кожен новий ознака

представлятиме одне унікальне значення і міститиме значення 1, якщо вихідна ознака відповідає цьому значенню, і 0 в іншому випадку. Таким чином, кожне унікальне значення перетворюється на окремий стовпець з бінарними значеннями. Для прикладу, наш набір даних мав 24 стовбчики ознак, після One-Hot Encoding отримаємо датасет з більш ніж 1000 ознак.



	Zip_Code	Latitude	Longitude	Tenure_Months	Monthly_Charges	Total_Charges	City_Acampo	City_Acton	City_Adelanto	City_Adin	...	Streaming_TV_No_internet_service	Streaming_TV_Yes	Streaming_Movies_No	S
0	90003	33.964131	-118.272783	2	53.85	108.15	0	0	0	0	...	0	0	1	
1	90005	34.059281	-118.307420	2	70.70	151.65	0	0	0	0	...	0	0	1	
2	90006	34.048013	-118.293953	8	99.65	820.50	0	0	0	0	...	0	1	0	
3	90010	34.062125	-118.315709	28	104.80	3046.05	0	0	0	0	...	0	1	0	
4	90015	34.039224	-118.266293	49	103.70	5036.30	0	0	0	0	...	0	1	0	

5 rows × 1178 columns

рис 2.7

Перевагою One-Hot Encoding є те, що він дозволяє алгоритму враховувати різницю між категоріями без введення порядку або значень, що може бути корисним при роботі з номінальними даними. Однак важливо враховувати, що за наявності великої кількості унікальних значень категоріальної ознаки One-Hot Encoding може призвести до збільшення розмірності даних і можливих проблем з перенавчанням моделі. У таких випадках може бути розглянутий альтернативний підхід, наприклад, використання методу кодування за допомогою чисел (label encoding) або категоріальних ознак з урахуванням порядку (ordinal encoding).
На даний момент передобробка даних може бути завершена, вони готові для подальшої роботи.

2.3 Поділ на навчальну та тестову вибірки

Наступним кроком буде Поділ на навчальну та тестову вибірки: Для оцінки якості моделі ми розділимо дані на навчальну та тестову вибірки. Навчальна вибірка використовуватиметься для навчання моделі, а тестова вибірка - з метою оцінки її продуктивності нових даних. Розбиваємо дані в пропорції 1 до 3.

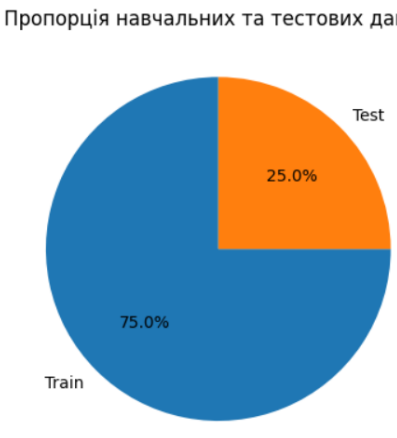


рис 2.8

XGBoost

Розглянемо загальну теорію роботи алгоритму градієнтного бустингу.

$a_n(x) = h_1(x) + \ldots + h_n(x)$

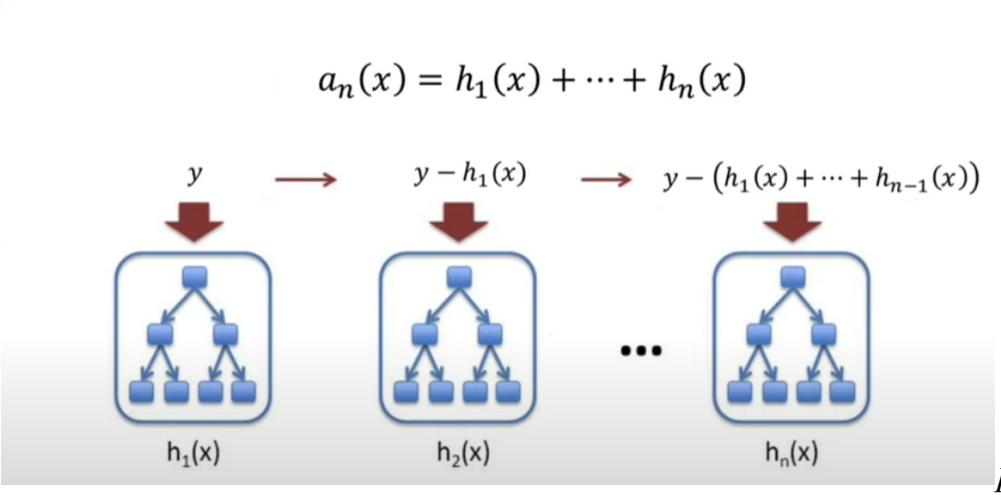


рис 2.9

Як і інші методи підвищення, підвищення градієнта об'єднує слабких «учнів» одного сильного учня ітеративним чином. Це найпростіше пояснити за умов регресії методом найменших квадратів , де мета у тому, щоб «навчити» модель F для передбачення значень $\widehat{y} = F(x)$ мінімізуючи середньоквадратичну помилку

$\frac{1}{n}\sum_i (\widehat{y}_i - y_i)^2$

$\widehat{y}_i$ - прогнозоване значення $F(x_i)$

y_i - спостережуване значення

n - кількість зразків в y

Тепер давайте розглянемо алгоритм підвищення градієнту з M етапів.

На кожному етапі m ($1 \leq m \leq M$) підвищення градієнту. Маємо недосконалу модель F_m. Щоб покращити F_m наш алгоритм повинен додати нову оцінку $h_m(x_i)$

Таким чином,

$$F_{m+1}(x_i) = F_m(x_i) + h_m(x_i) = y_i$$

або

$$h_m(x_i) = y_i - F_m(x_i)$$

Тож підвищення градієнту h_m до залишку $y_i - F_m(x_i)$

Як і в інших варіантах бустингу, кожен F_{m+1}, намагається виправити помилки свого попередника F_m

$$L_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - F(x_i))^2$$

$$-\frac{\partial L_{MSE}}{\partial F(x_i)} = \frac{2}{n}(y_i - F(x_i)) = \frac{2}{n}h_m(x_i)$$

Таким чином, підвищення градієнта може бути адаптовано до алгоритму градієнтного спуску.

XGBoost працює як метод Ньютона-Рафсона у функціональному просторі, на відміну від підвищення градієнта, що працює як градієнтний спуск у функціональному просторі. У функції втрат використовується апроксимація Тейлора другого порядку зв'язку з методом Ньютона-Рафсона.

2.4 Алгоритм

Вхідні дані

$\{(x_i, y_i)\}_{i=1}^{N}$ - тренувальний набір

$L(y, F(x))$ - диференційована функція втрат

M - кількість слабких учнів

α - швидкість навчання

1. Ініціалізуємо модель з постійним значенням:

   $$\widehat{f}_{(0)}(x) = argmin_{\theta}\sum_{i=1}^{N} L(y_i, \theta).$$

2. Для $m = 1$ до M

   a. Рахуєм градієнти та гесіани

$$\widehat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f^{(m-1)}(x)}$$

$$\widehat{h}_m(x_i) = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2}\right]_{f(x)=f^{(m-1)}(x)}$$

b. Відповідайте базовому учню (або слабому учню, наприклад, дереву), використовуючи навчальний набір $\left\{x_i, -\frac{\widehat{g}_m(x_i)}{\widehat{h}_m(x_i)}\right\}_{i=1}^{N}$ шляхом рішення задачі оптимізації:

$$\widehat{\phi}_m = argmin_{\phi\in\Phi} \sum_{i=1}^{N} \frac{1}{2}\widehat{h}_m(x_i)\left[-\frac{\widehat{g}_m(x_i)}{\widehat{h}_m(x_i)} - \phi(x_i)\right]^2.$$

$$\widehat{f}_m(x) = \alpha\widehat{\phi}_m(x).$$

c. Оновити модель:

$$\widehat{f}_{(m)}(x) = \widehat{f}_{(m-1)}(x) + \widehat{f}_m(x).$$

3. Вихід: $\widehat{f}(x) = \widehat{f}_{(M)}(x) = \sum_{m=0}^{M} \widehat{f}_m(x)$.

Таким чином ми можемо побудувати модель XGBoost для прогнозування відтоку користувачів.

2.5 Практична реалізація моделі

Кроки реалізації:

1. Імпорт необхідних бібліотек
2. Завантаження даних
3. Попередня обробка даних
4. Поділ даних на навчальний та тестовий набори
5. Побудова моделі
6. Навчання моделі
7. Оцінка продуктивності моделі

8. Візуалізація результатів

Після проведення передобробки даних які ми будемо використовувати для навчання те тестування моделі, а також розбиття цих даних на учбові та тестові ініціалізуємо модель XGBoost, та стикаємось з основною задачею під час побудови моделі було визначення та налаштування гіперпараметрів для моделі, адже це запорука найбільш точної та ефективної роботи алгоритму.

Для цього ми використаємо два методи:

(Grid Search): Цей метод передбачає перебір всіх можливих комбінацій гіперпараметрів із заданого діапазону. Для кожної комбінації гіперпараметрів виконується крос-валідація та оцінюється якість моделі. Наприкінці вибираються гіперпараметри, що забезпечують найкращу продуктивність моделі.

```
Найкращі гіперпараметри: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 200}
Найкраща продуктивність: 0.8937062937062936
```

рис 2.10

(Random Search): У цьому методі вибираються випадкові комбінації гіперпараметрів із заданих діапазонів. Для кожної комбінації виконується крос-валідація та оцінюється продуктивність моделі. У міру виконання випадкового пошуку можна отримати уявлення про простір гіперпараметрів та їх вплив на продуктивність моделі.

```
Найкращі гіперпараметри: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 249}
Найкраща продуктивність: 0.8909090909090909
```

рис 2.11

Як ми можемо побачити, ми знайшли оптимальні гіперпараметри двома методами, можемо використовувати будь-який з варіантів а також їх середнє значення, бачимо що різниці продуктивності майже немає, на грані похибки.

Після налаштування гіперпараметрів, будуємо модель навчаємо її на тестових даних та робимо прогнозування відтоку користувачів, та візуалізуємо це

Візуалізуємо у вигляді матриці помилок:

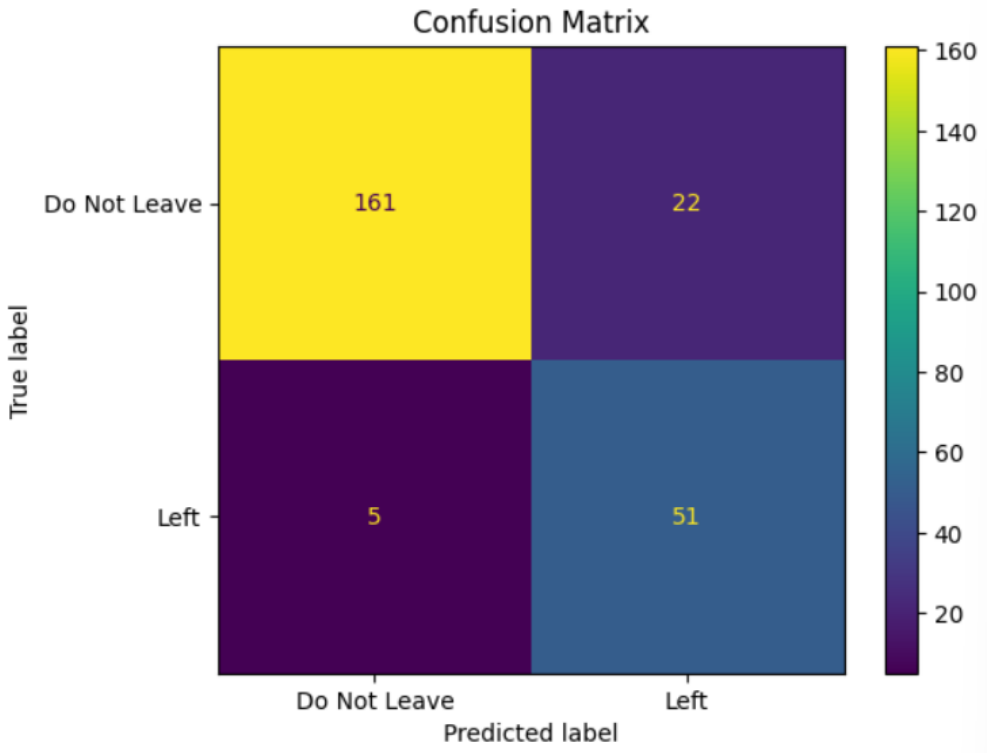


рис 2.12

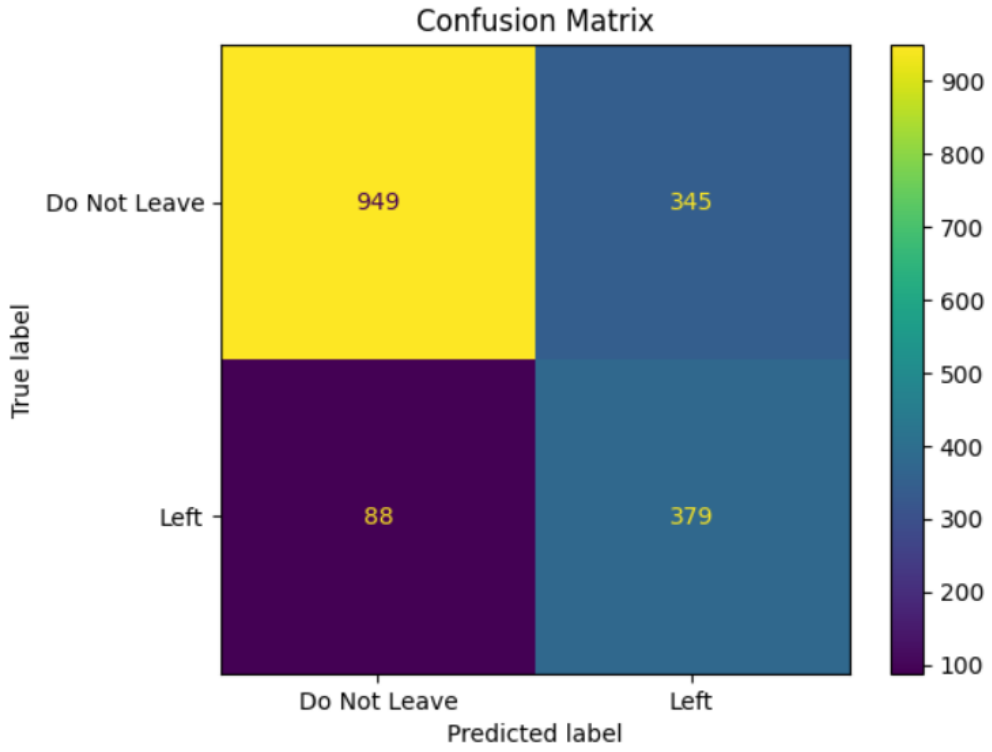


рис 2.13

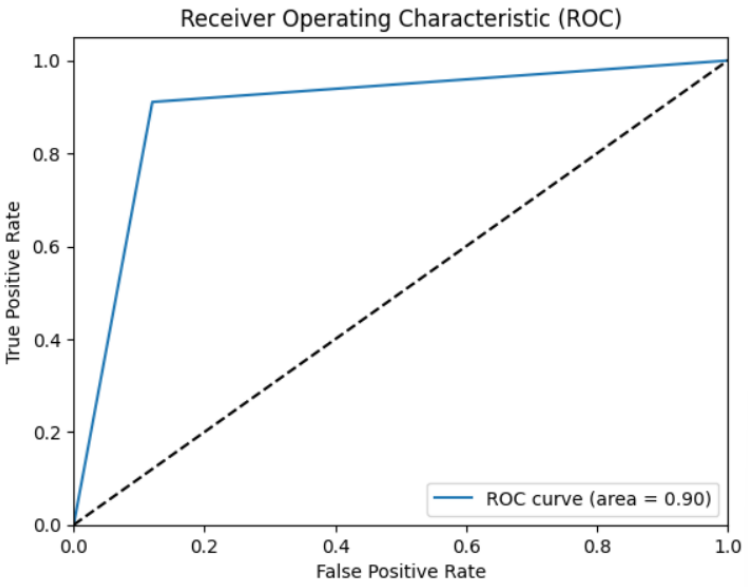
Як можемо помітити модель працю досить точно, далі перевіримо ефективність моделі за декількома метриками.

При цьому, при використанні другого датасету ми бачимо що не завжди більша кількість параметрів запорука більш точної роботи алгоритму, в першому випадку ознак 7 у другому 24.

2.6 Оцінка роботи алгоритму

Оцінювання алгоритму будемо проводити за декількома метриками (гіперпараметри встановлені на ті значення що ми визначили раніше), а саме AUC-ROC, Precision, Recall, F1-score

Для першої моделі:

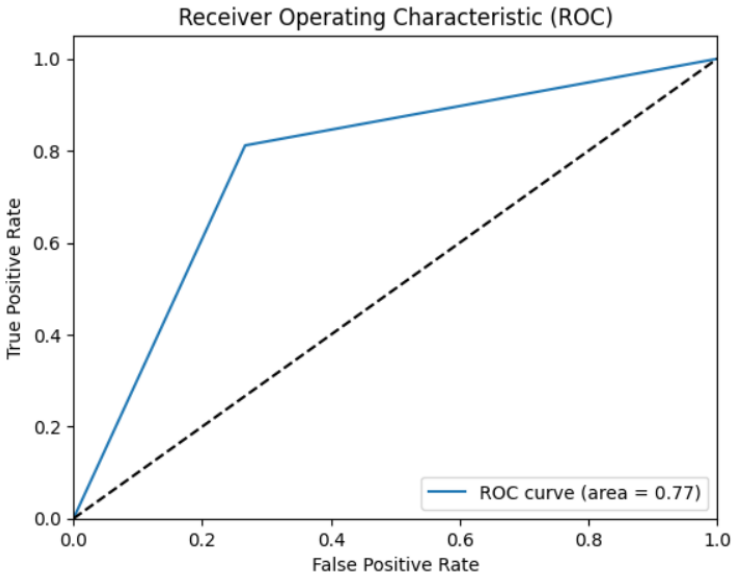



```
Precision: 0.6986
Recall: 0.9107
F1-score: 0.7907
```

рис 2.14

Як можемо побачити, показники ефективності величезні

Перевіримо для другої моделі:




```
Precision: 0.5235
Recall: 0.8116
F1-score: 0.6364
```

рис 2.15

Результати також вражаючи, особли якщо прийняти до уваги що в другому наборі даних структура цих даних набагато складніша, а зв’язки між ознаками не тривіальні.

Введемо ще декілька метрик для повної оцінки ефективності моделі

- MAE (Mean Absolute Error) - це середня абсолютна помилка, яка вимірює середнє абсолютне відхилення між прогнозованими значеннями і фактичними значеннями. Ця метрика вимірює величину помилки в одиницях вихідної змінної і дозволяє оцінити розмах помилок.
- MSE (Mean Squared Error) - це середня квадратична помилка, яка вимірює середнє квадратичне відхилення між прогнозованими значеннями і фактичними значеннями. Вона вимірює величину помилки в квадратних одиницях і видає вагу великим відхиленням.
- RMSE (Root Mean Squared Error) є метрикою, яка використовується для вимірювання точності прогнозування моделі. Вона представляє собою квадратний корінь з середнього квадратичного значення помилок між прогнозованими значеннями моделі і спостережуваними значеннями вихідної змінної.

Створимо таблицю порівняння декількох моделей з різними параметрами та оцінемо їх за MAE, MSE, RMSE та AUC-ROC

№	Налаштування	MAE	MSE	RMSE	AUC-ROC
1	gamma=0.25, learn_rate=0.2, max_depth=4, reg_lambda=10, colsample_bytree=0.5	0.142	0.142	0.377	0.864
2	gamma=0.25, learn_rate=0.05, max_depth=3, reg_lambda=10, colsample_bytree=0.6	0.125	0.125	0.354	0.899
3	gamma=0.25, learn_rate=0.15, max_depth=4, reg_lambda=10, colsample_bytree=0.6	0.117	0.117	0.342	0.898
4	gamma=0.25, learn_rate=0.25, max_depth=5, reg_lambda=10, colsample_bytree=0.7	0.108	0.108	0.329	0.904
5	gamma=0.25, learn_rate=0.1, max_depth=4,	0.104	0.104	0.323	0.906

	reg_lambda=10, colsample_bytree=0.7				

Все ж таки нам вдалось ще трохи покращити роботу моделі завдяки зміні гіперпараметрів, значно покращити результат вже не отримується.

Висновок до розділу 2:

У практичній частині нашого дослідження була реалізована модель прогнозування відтоку клієнтів з використанням алгоритму XGBoost. Ми провели попередню обробку даних, включаючи видалення зайвих ознак, заповнення пропущених значень та кодування категоріальних змінних.

Для побудови моделі ми використали класифікатор XGBoost та навчили його на навчальному наборі даних. Для вибору оптимальних гіперпараметрів моделі ми застосували метод перехресної перевірки та пошуку по сітці. Потім ми оцінили продуктивність моделі за допомогою різних метрик, включаючи точність, повноту, F-меру та AUC-ROC.

Результати показали, що розроблена модель має високу здатність прогнозувати відтік клієнтів. Вона досягла високих значень метрик, що підтверджує її ефективність. Важливим результатом дослідження є визначення найбільш значимих ознак, які мають найбільший вплив на прогнозування відтоку.

Заключно, ця практична частина дипломної роботи підтверджує ефективність використання алгоритму XGBoost для прогнозування відтоку клієнтів. Отримана модель може бути використана компаніями для прийняття заходів щодо утримання клієнтів та оптимізації бізнес-процесів.

Висновки :

У результаті дослідження та використання різних алгоритмів машинного навчання для прогнозування відтоку користувачів, надано переконливі докази того, що ці методи є потужними та ефективними інструментами для вирішення виклику, що ставиться перед бізнесом. Розроблені моделі відрізняються високою точністю та здатністю передбачати відтік з високою ймовірністю, що робить їх цінними для компаній уникнути втрат та прийняти належні заходи для збереження клієнтів.

Отримані результати та аналіз підтверджують значимість та вплив алгоритмів машинного навчання в сфері прогнозування відтоку. Вони надають компаніям цінні інсайти та можливості для управління клієнтською базою та оптимізації бізнес-процесів. Застосування таких моделей дозволяє компаніям не тільки ефективно виявляти клієнтів, які піддаються ризику відтоку, але й уникати непотрібних витрат на збереження стійких клієнтів. Крім того, використання алгоритмів машинного навчання сприяє удосконаленню стратегій залучення та утримання клієнтів, підвищенню рівня задоволеності клієнтів та стабільному зростанню бізнесу.

Узагальнюючи, використання алгоритмів машинного навчання для прогнозування відтоку користувачів є невід'ємною складовою ефективного управління клієнтською базою та досягнення успіху на ринку. Ці алгоритми виявляються потужними інструментами, які сприяють збереженню клієнтів, вдосконаленню стратегій бізнесу та досягненню стійкого зростання компанії.

Перелік використаних джерел:

1. Provost, F., & Fawcett, T. (2013). Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking. O'Reilly Media.
2. Verbeke, W., Dejaeger, K., Martens, D., & Hur, J. (2012). New insights into churn prediction in the telecommunication sector: A profit driven data mining approach. European Journal of Operational Research.
3. Dhiman, G., & Gupta, A. (2018). Predicting customer churn in the insurance industry using machine learning algorithms. Procedia Computer Science.
4. "Hands-On Machine Learning with Scikit-Learn and TensorFlow" by Aurélien Géron
5. "Pattern Recognition and Machine Learning" by Christopher M. Bishop
6. Hadden, C. (2019). "Churn Prediction with Machine Learning: Strategies and Tips." Towards Data Science.
7. Nguyen, T. T., & Nguyen, T. T. (2015). "A novel feature selection approach for churn prediction in telecom industry." Journal of Computational Science.

Додаток А (текст програми)

```
import pandas as pd
import numpy as np
import seaborn as sns

import matplotlib.pyplot as plt
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import balanced_accuracy_score, roc_auc_score, make_scorer
from sklearn.model_selection import  GridSearchCV
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import roc_curve

df = pd.read_csv('Telco_customer_churn.csv')
df.head()

# Матриця кореляції
correlation_matrix = df.corr()

# Візуалізація
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
```

```
plt.show()

df.drop(['Churn Label', 'Churn Score', 'CLTV', 'Churn Reason'],
        axis=1, inplace=True)
df.head()

df['Count'].unique()
df['Count'].unique()
df['State'].unique()

df.drop(['CustomerID', 'Count', 'Country', 'State', 'Lat Long'],
        axis=1, inplace=True)
df.head()

df['City'].replace(' ','_', regex=True, inplace=True)
df.head()
df['City'].unique()[0:10]

df.columns = df.columns.str.replace(' ','_')
df.head()
df.dtypes

len(df.loc[df['Total_Charges'] == ' '])
df.loc[df['Total_Charges'] == ' ']

df.loc[(df['Total_Charges'] == ' '), 'Total_Charges'] = 0
df['Total_Charges'] = pd.to_numeric(df['Total_Charges'])
df.dtypes

df.replace(' ', '_', regex=True, inplace=True)
```

```
df.head()

X = df.drop('Churn_Value', axis=1).copy()
y = df['Churn_Value'].copy()
y.head()

df['Churn_Value'] = y
plt.figure(figsize=(8, 6))
sns.countplot(x='Churn_Value', data=df, palette='Blues')
plt.xlabel('Churn Value')
plt.ylabel('Count')
plt.title('Distribution of Churn Value')
plt.show()

plt.figure(figsize=(6, 6))
df['Churn_Value'].value_counts().plot(kind='pie', autopct='%1.1f%%',
colors=['lightblue', 'lightgreen'])
plt.ylabel('')
plt.title('Class Distribution of Churn Value')
plt.show()
X.dtypes

y.unique()

# Матриця кореляції
correlation_matrix = df.corr()

# Візуалізація
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
```

```
plt.title('Correlation Matrix')
plt.show()

X_encoded = pd.get_dummies(X, columns = ['City',
                                         'Gender',
                                         'Senior_Citizen',
                                         'Partner',
                                         'Dependents',
                                         'Payment_Method','Phone_Service', 'Multiple_Lines',
                                         'Internet_Service',
                                         'Online_Security',
                                         'Online_Backup',
                                         'Device_Protection',
                                         'Tech_Support',
                                         'Streaming_TV',
                                         'Streaming_Movies',
                                         'Contract',
                                         'Paperless_Billing'])
X_encoded.head()

X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, random_state=42, stratify=y)

sum(y_train)/len(y_train)
sum(y_test)/len(y_test)


train_size = len(X_train) / len(df)
test_size = len(X_test) / len(df)
```

```
labels = ['Train', 'Test']
sizes = [train_size, test_size]
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
plt.title('Пропорція навчальних та тестових даних')
plt.show()

# Тестова модель
clf_xgb = xgb.XGBClassifier(objective='binary:logistic', missing=None)
clf_xgb.fit(X_train,
            y_train,
            verbose=True,
            early_stopping_rounds=10,
            eval_metric='aucpr',
            eval_set=[(X_test, y_test)])

count_negative = len(y_train[y_train == 0])
count_positive = len(y_train[y_train == 1])
scale_pos_weight = count_negative / count_positive
print(scale_pos_weight)

# Модель XGBoost
clf_xgb = xgb.XGBClassifier(seed=42,
                            objective='binary:logistic',
                            gamma=0.25,
                            learn_rate=0.1,
                            max_depth=4,
                            reg_lambda=10,
                            scale_pos_weight=scale_pos_weight,
                            subsample=0.9,
                            colsample_bytree=0.5,
```

```
                booster='gbtree',
                n_estimators=10
                )

clf_xgb.fit(X_train,
        y_train,
        verbose=True,
        early_stopping_rounds=20,
        eval_metric='aucpr',
        eval_set=[(X_test, y_test)])

y_pred = clf_xgb.predict(X_test)

#Матриця помилок
cm = confusion_matrix(y_test, y_pred)

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
        xticklabels=['Do not leave', 'Leave'],
        yticklabels=['Do not leave', 'Leave'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')

plt.show()

auc = roc_auc_score(y_test, y_pred)
print("AUC-ROC:", auc)

fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % auc)
```

```
plt.plot([0, 1], [0, 1], 'k--')  # Випадкове вгадування
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
plt.show()



y_pred = clf_xgb.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Do Not Leave", "Left"])
disp.plot()
plt.title('Confusion Matrix')
plt.show()

precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Precision: {:.4f}".format(precision))
print("Recall: {:.4f}".format(recall))
print("F1-score: {:.4f}".format(f1))
```