МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інформаційних управляючих систем та технологій

СТЕФАНІВ ОЛЕГ ТАРАСОВИЧ

ЗАСТОСУВАННЯ МЕТОДІВ МАШИНОГО НАВЧАННЯ В ЗАДАЧАХ НА ВИЖИВАННЯ

122 Комп’ютерні науки

Дипломна робота на здобуття освітнього ступеня
бакалавра

Науковий керівник:
Ніколенко Володимир
Володимирович
к.ф-м.н., доц.

Ужгород – 2025

Реєстрація _____

(номер)

«____» червня 2025 р. _____ Рижак Е.М.

(підпис)

Дипломна робота допущена до захисту

Завідувач кафедри

_____ Міца О.В.

(підпис)

доктор технічних наук, професор

«____» червня 2025 р.

Рецензент _____ Мич І.А.

(підпис)

к.ф-м.н., доц,

ПІБ: Стефанів Олег Тарасович

Назва: Застосування методів машинного навчання в задачах на виживання

Факультет: Інформаційних технологій

Спеціальність: 122 «Комп'ютерні науки»

Науковий керівник: к.ф-м.н., доц. Ніколенко В. В.

За темою роботи опубліковано 0 статей.

Анотація

У цій дипломній роботі було досліджено різні методи машинного навчання. Методи досліджувалися на задачах на виживання. Методи створювалися на мові програмування Python.

Робота складається з двох розділів. У першому розділі описано загальні відомості про машинне навчання, наведено їх типи, а також розказано про методи машинного навчання які були використані в даній роботі. В другому розділі описана структура дипломної роботи, також описані процеси підготовки даних, та створення моделей. Наведені робоче середовище, мова програмування та використані бібліотеки. Проведено порівняльний результатів роботи моделей на різних даних.

Дана дипломна робота містить: сторінок — 64, розділів — 2, рисунків — 68, кількість використаних джерел — 14.

Ключові слова: KNN, Random forest, Logistic Regression, XGboost, машинне навчання.

Abstract

This thesis explores various machine learning methods applied to survival prediction tasks. The models were developed using the Python programming language.

The work consists of two chapters. The first chapter provides general information about machine learning, its types, and a detailed description of the methods used in this research. The second chapter describes the structure of the thesis, the data preprocessing steps, and the model development process. It also outlines the development environment, programming language, and the libraries used. A comparative analysis of the model performance on different datasets is presented.

This thesis includes: 64 pages, 2 chapters, 68 figures, and 14 references.

Keywords: KNN, Random Forest, Logistic Regression, XGBoost, Machine Learning

ЗМІСТ

ВСТУП

У сучасному світі велика увага приділяється розробці методів прогнозування в різних сферах, включаючи задачі на виживання. Прогнозування виживання має важливе значення в медицині, фінансах, страхуванні, а також в інших галузях, де важливо визначити ймовірність виживання певної події або об'єкта.

Метою даного проєкту є дослідження ефективності різних методів машинного навчання в задачах на виживання, шляхом їх реалізації, тестування та порівняння на основі двох реальних наборів даних. У процесі дослідження планується реалізувати моделі логістичної регресії, KNN, Random Forest та XGBoost, а також провести аналіз їх результатів за відповідними метриками.

Для досягнення даної мети було визначено наступні завдання:

Аналіз вимог до дипломного проєкту: сформульовано чітку мету, завдання дослідження та обґрунтовано актуальність використання методів машинного навчання у задачах виживання.

Огляд існуючих підходів та технологій: здійснено аналіз теоретичних основ задач виживання та сучасних методів машинного навчання, що використовуються для їх вирішення.

Підготовка даних та реалізація моделей: буде проведено попередню обробку даних, створено відповідні моделі в середовищі Python з використанням бібліотек scikit-learn, XGBoost та інших.

Навчання та тестування моделей: буде виконано тренування моделей на даних, а також буде проведено тестування їх ефективності.

Візуалізація та порівняльний аналіз результатів: будуть побудовані графіки результатів, оцінено ефективність моделей за метриками точності, повноти, F1-міри, ROC-кривими та показниками важливості ознак.

РОЗДІЛ 1. ОСНОВНІ ВІДОМОСТІ ПРО МАШИННЕ НАВЧАННЯ

1.1 Поняття про машинне навчання

Машинне навчання (МН) – це підгалузь штучного інтелекту (ШІ), яка зосереджується на розробці алгоритмів, що дозволяють комп'ютерним системам навчатися на основі даних без явного програмування для кожної конкретної задачі. Замість того, щоб слідувати жорстко заданим інструкціям, системи машинного навчання використовують статистичні методи для виявлення закономірностей у великих обсягах даних і роблять на їх основі прогнози або приймають рішення.

Основна ідея машинного навчання полягає в тому, щоб надати системі можливість "навчатися" на прикладах (навчальній вибірці даних) і потім узагальнювати отримані знання для обробки нових даних. Цей процес навчання може включати розпізнавання образів, класифікацію об'єктів, прогнозування майбутніх значень або прийняття оптимальних рішень у складних середовищах.

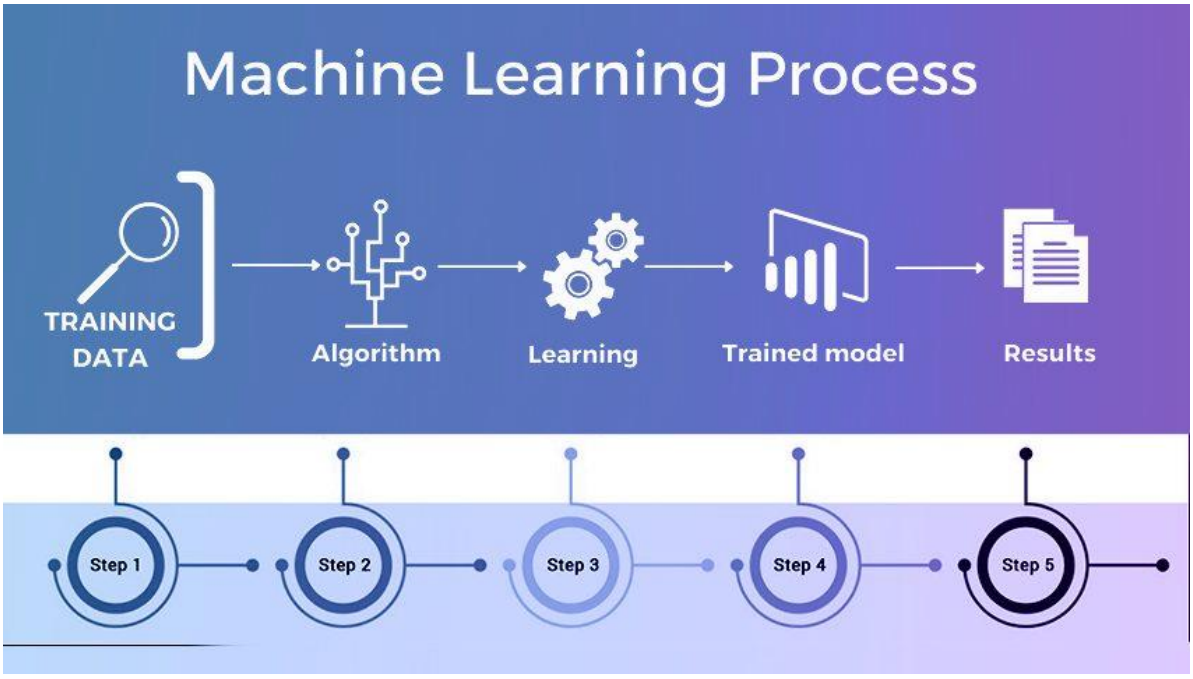

Рисунок. 1.1 Схематичне зображення процесу машинного навчання

Ключовими компонентами будь-якої системи машинного навчання є:

- Дані: Це основа, на якій відбувається навчання. Якість, кількість та репрезентативність даних безпосередньо впливають на ефективність моделі.
- Ознаки (Features): Це характеристики вхідних даних, котрі використовуються для навчання моделі. Наприклад, при розпізнаванні зображень котів ознаками можуть бути колір шерсті, форма вух, наявність вусів тощо.
- Алгоритм (Модель): Це математична функція або набір правил, які система використовує для відображення вхідних даних на вихідні прогнози або рішення. Вибір відповідного алгоритму залежить від типу задачі та характеру даних.
- Процес навчання: Це процес налаштування параметрів моделі на основі навчальних даних з метою мінімізації помилки прогнозування або максимізації певної метрики якості.
- Оцінка: Після навчання модель оцінюється на тестових даних, щоб перевірити її здатність до узагальнення.

1.2 Класифікація типів машинного навчання

Для прогнозування в задачах використовуються різні типи машинного навчання. За типом задач, машинне навчання поділяють: навчання з вчителем, навчання без вчителя, навчання з частковим залученням вчителя, навчання з підкріпленням.

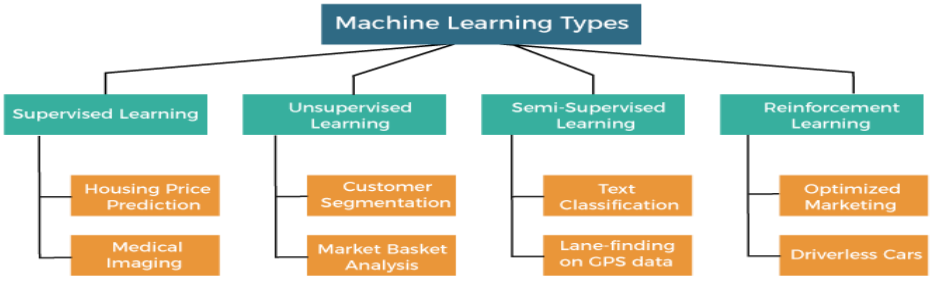

Рисунок. 1.2. Типи машинного навчання

Навчання з вчителем (Supervised Learning): передбачає використання повного набору даних для тренування моделі на всіх етапах її розробки. Основна мета даного підходу полягає в тому, щоб навчитися прогнозувати мітки для нових, ще не розмічених даних. У процесі навчання подається набір тренувальних прикладів, який називають навчальний або тренувальний набір даних. Завдання полягає в тому, щоб застосувати вже відомі відповіді до нових ситуацій, які зазвичай представлені у вигляді тестового набору даних. Наявність повного набору даних означає, що кожному прикладу в навчальному наборі відповідає певне рішення (мітка), яке алгоритм має навчитися визначати.

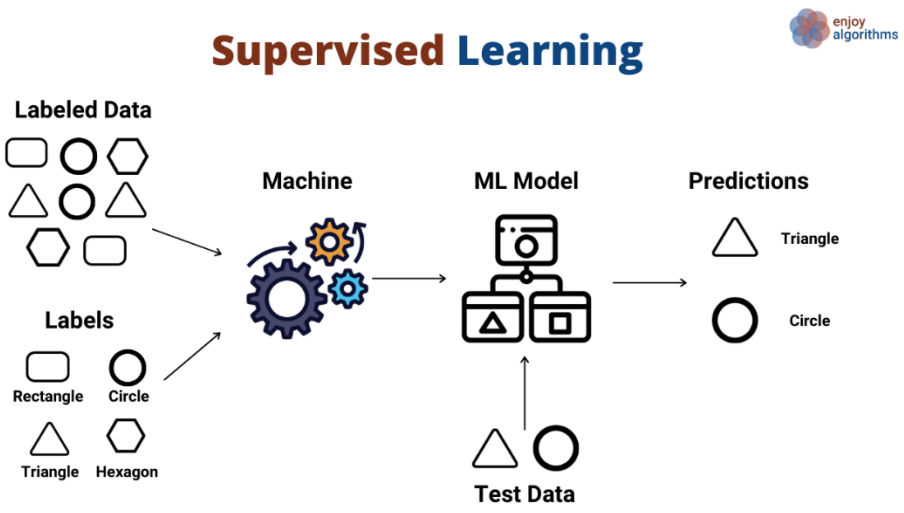

Рисунок 1.3 Навчання з вчителем (Supervised Learning)

Зазвичай навчання з учителем застосовується для вирішення таких типів задач:

- Класифікації – мета задачі полягає в віднесенні об'єктів до певних категорій або класів на основі вхідних ознак.
- Регресії – мета задачі полягає в прогнозування безперервної числової змінної на основі вхідних даних.
- Ранжирування – мета задачі полягає в впорядковуванні об'єктів у певному порядку за заданим критерієм.

Навчання без учителя (Unsupervised Learning): На відміну від навчання з вчителем, цей підхід працює з даними, які не містять заздалегідь заданих міток або відповідей. Головна задача навчання без учителя це самостійне розпізнання прихованих структур, тенденції, аномалії або взаємозв'язків прямо у вихідному масиві даних. Алгоритм опрацьовує дані, роблячи спроби об'єднати схожі об'єкти в групи чи скоротити кількість атрибутів без жодних підказок стосовно того, які саме мають бути "правильні" результати.

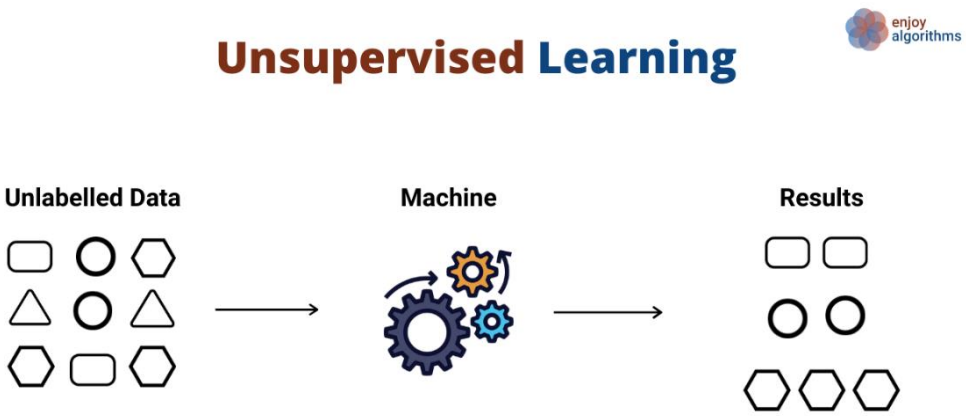

Рисунок 1.4 Навчання без учителя (Unsupervised Learning)

Зазвичай навчання без учителя застосовується для вирішення таких типів задач:

- Кластеризація – мета задачі полягає в об'єднанні схожих об'єктів у групи (кластери) на основі їхніх характеристик.
- Зменшення розмірності – мета задачі полягає в скороченні кількості вхідних ознак (змінних) при збереженні найбільш важливої інформації.
- Виявлення аномалій – мета задачі полягає в ідентифікації об'єктів або подій, які суттєво відрізняються від загальної маси даних і можуть свідчити про нетипову поведінку або помилки.

Навчання з підкріпленням (Reinforcement Learning): даний підхід машинного навчання суттєво відрізняється від навчання з учителем та навчання без вчителя.

У цьому випадку система, яка називається агентом, вивчає, як приймати рішення, взаємодіючи з середовищем. За кожну виконану дію агент отримує зворотний зв'язок у формі винагороди (позитивний) або штрафу (негативний). Головна ціль агента – навчитися стратегії, яка забезпечить максимальну винагороду протягом тривалого часу. Агент не має заздалегідь визначених "правильних" відповідей для кожного випадку, він навчається методом спроб і помилок.

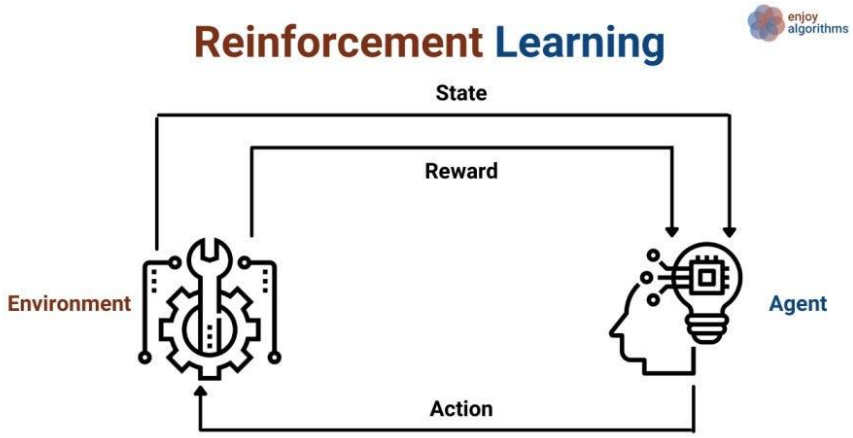

Рисунок 1.5 Навчання з підкріпленням (Reinforcement Learning)

Ключовими елементами навчання з підкріпленням є:

- Агент (Agent): Агент за свої дії може получати чисельний сигнал – винагороду або штраф.
- Середовище (Environment): Середовище з яким агент взаємодіє.
- Стан (State): Поточна ситуація або конфігурація, в якій перебуває агент у середовищі.
- Винагорода (Reward): Чисельний сигнал від середовища, що оцінює якість дії агента. Може бути як позитивним так і негативним.

- Стратегія (Policy): Стратегія, яку використовує агент для вибору дій у різних ситуаціях.

Зазвичай навчання з підкріпленням найбільше застосовується в таких областях:

- Безпілотне керування – керування автомобілями, керування роботами для виконання складних операцій.
- Система рекомендацій – персональні рекомендації користувачу в різних сферах повсякденного життя.

Напівконтрольоване навчання (Semi-Supervised Learning): Цей тип навчання розташовується посередині між навчанням із вчителем та навчанням без учителя. Його застосовують у випадках, коли наявна велика кількість даних, але лише незначна його частина має відомі відповіді, тоді як більшість даних залишається без відповіді.

Основна ціль напівконтрольованого навчання – використати інформацію як з даних які мають відповіді, так і з даних які їх не мають для створення точнішої моделі, порівняно з тією, яку можливо отримати, застосовуючи лише один із цих типів даних окремо. Сенс полягає в тому, що нерозмічені дані, хоча й не мають чітких міток, все одно містять інформацію про структуру та розподіл даних, що може бути корисним для процесу навчання.

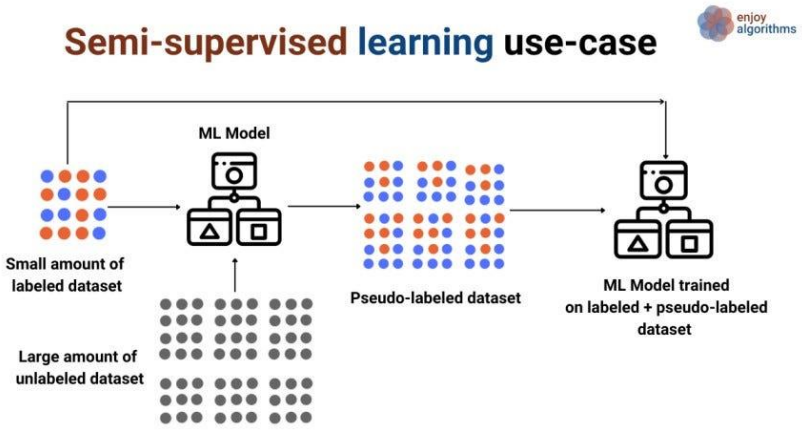

Рисунок 1.6 Напівконтрольоване навчання (Semi-Supervised Learning)

Напівконтрольоване навчання використовується для розв'язання задач, які нагадують ті, що розв'язуються у навчанні з учителем, але при частковій наявності розмічених даних:

- Класифікації – Розпізнавання об'єктів на зображеннях, де не на всіх зображеннях вказано який там об'єкт.
- Регресії – Прогнозування числових значень, коли лише частково відомо точні значення.

1.3 Огляд методів машинного навчання

K-найближчих сусідів (KNN) – це алгоритм машинного навчання з вчителем, що зазвичай використовується для класифікації, хоча його також можна використати й для розв'язання регресійних задач. Його принцип полягає у відшуканні "k" найближчих точок даних (сусідів) до певного вхідного значення та наступному здійсненні прогнозу, враховуючи клас більшості (для класифікації) або середнє значення (для регресії).

Значення K в алгоритмі KNN – це число, яке вказує, яку кількість найближчих сусідів слід враховувати під час прийняття рішення.

Вибір правильного значення k є важливим для отримання добрих результатів. Якщо дані містять багато шуму, використання більшого значення k може зробити прогноз більш стабільним, але якщо k буде занадто велике, модель стане занадто простою та пропустити важливі закономірності, внаслідок чого передбачення буде нестабільне і неправильне.

Для того, щоб обрати значення k існує декілька способів:

- Перехресна перевірка: Перехресна перевірка – чудовий спосіб визначити оптимальне значення k, застосовуючи k-кратну перехресну валідацію. Суть полягає у поділі вибірки на k сегментів. Модель тренується на кількох з них та оцінюється на інакших. Цей алгоритм повторюється для кожного сегмента. Значення k, яке показує найвищу

середню точність під час цих випробувань, вважається найкращим для використання.

- Метод ліктя: В методі ліктя будується графік, на якому відображено коефіцієнт помилок або точність для різних значень k. З збільшенням значення k помилка, зазвичай, спочатку зменшується. Але, після певної межі, зменшення сповільнюється. Точка, де крива змінює свою форму і графік починає показувати різке зменшення точності, утворюючи "лікоть", зазвичай вважається найкращим вибором для k.
- Непарні значення для k: Рекомендується використовувати непарні значення для k, особливо у задачах класифікації. Це допомагає запобігти нічийним результатам при визначенні домінуючого класу серед сусідів.

Також для того, щоб знайти найближчих сусідів потрібно використовувати певні метрики відстані. Розглянемо декілька метрик:

- Евклідова відстань: Визначається як пряма між двома точками на площині. Дану відстань можна уявити як шлях від точки А до точки Б при умові що в нас немає ніяких перешкод.
- Відстань до Манхеттена: Визначається як найкоротший шлях від точки А до точки Б при умові що можна рухатися тільки горизонтально та вертикально. Дану відстань можна уявити як шлях від будинку А до будинку Б в місті. Оскільки існують перешкоди у вигляді будинків, людина мусить рухатися вулицями між ними.

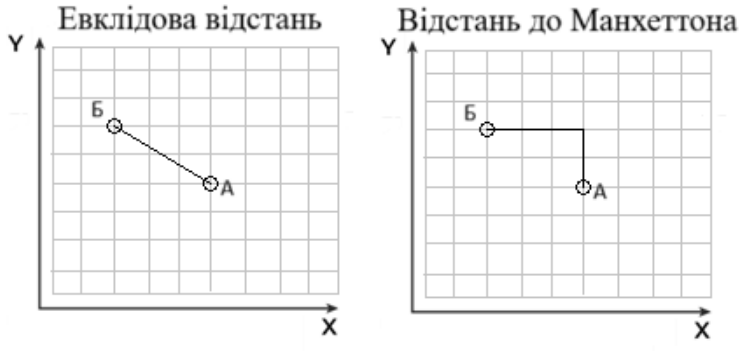

Рисунок 1.7 Метрики відстані

Розглянемо покрокову роботу алгоритму KNN. Алгоритм KNN базується на подібності.

Крок 1: Визначення оптимального значення k

- k – це кількість найближчих точок, яких потрібно враховувати під час процесу прогнозування.

Крок 2: Обчислення дистанції

- Для визначення подібності між тестовими та тренувальними точками даних застосовується евклідова відстань. Обчислення відбувається між елементами набору даних та цільовою точкою.

Крок 3: Ідентифікація найближчих сусідів

- k точок, що знаходяться найближче до цільової точки, спираючись на мінімальну відстань, вважаються її найближчими "сусідами".

Крок 4: Голосування для класифікації або використання середнього значення для регресії

- Коли необхідно віднести точку даних до певної категорії, наприклад, "Вижив" чи "не вижив", алгоритм KNN аналізує k сусідів з набору даних. Алгоритм визначає, до якої категорії належить більшість сусідів, та обирає її як відповідну.
- У регресії алгоритм також вибирає k найближчих точок. Проте, замість голосування за клас, він обчислює середнє арифметичне значення сусідів.

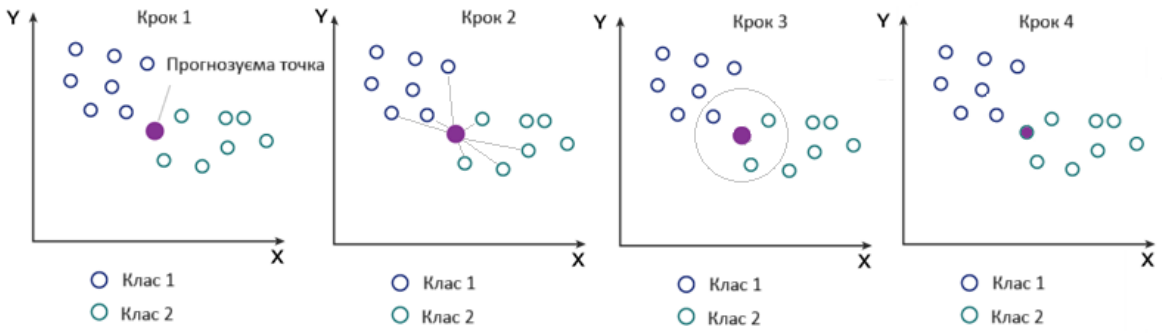

Рисунок 1.8 Покрокова робота алгоритму KNN

Алгоритм KNN має як переваги та недоліки.

Переваги:

- Простий у реалізації.
- Не потребує навчання.

Недоліки:

- Мала швидкість передбачення з великими даними.
- Витрачає багато пам'яті
- Сильно чутливий до шуму, особливо при малому значенні k

Random forest (RF)– це алгоритм машинного навчання, що оперує кількома деревами рішень для генерації передбачень. Цей підхід сприяє зростанню точності та мінімізації помилок.

В основі Random Forest лежить створення великої кількості дерев рішень, котрі функціонують як незалежні "експерти". Кожне з дерев навчається на випадковій підмножині вихідних даних.

Розглянемо покрокову роботу алгоритму RF.

Крок 1: Формування багатьох дерев рішень

- Створюється велика кількості дерев рішень, які працюють незалежно один від одного. Кожне з дерев навчається на випадковій підмножині вихідних даних.
- Random Forest також вводить випадковість у виборі ознак (стовпців даних) під час створення кожного дерева. На кожному етапі розгалуження дерева замість розгляду всіх доступних ознак для пошуку найкращого розділу, алгоритм випадковим чином обирає тільки підмножину ознак. Серед цієї випадкової підмножини згодом здійснюється пошук оптимальної ознаки для розділення.

Крок 2: Кожне дерево робить прогноз

- Після того, як усі дерева у Random Forest побудовані, кожне з них готове зробити власний індивідуальний прогноз.

Крок 3: Об’єднання прогнозів

- Кінцевий прогноз Random Forest є наслідком об'єднання індивідуальних прогнозів кожного дерева. Спосіб об'єднання залежить від типу задачі:
- Для задач класифікації Random Forest використовує принцип голосування більшістю. Кожне дерево голосує за певну категорію, і остаточна відповідь – це та категорія, за яку проголосувало найбільше дерев.
- Для задач регресії, Random Forest обчислює середнє значення всіх прогнозів, зроблених окремими деревами. Це дозволяє отримати плавний та точний числовий прогноз, усереднюючи можливі відхилення окремих дерев.

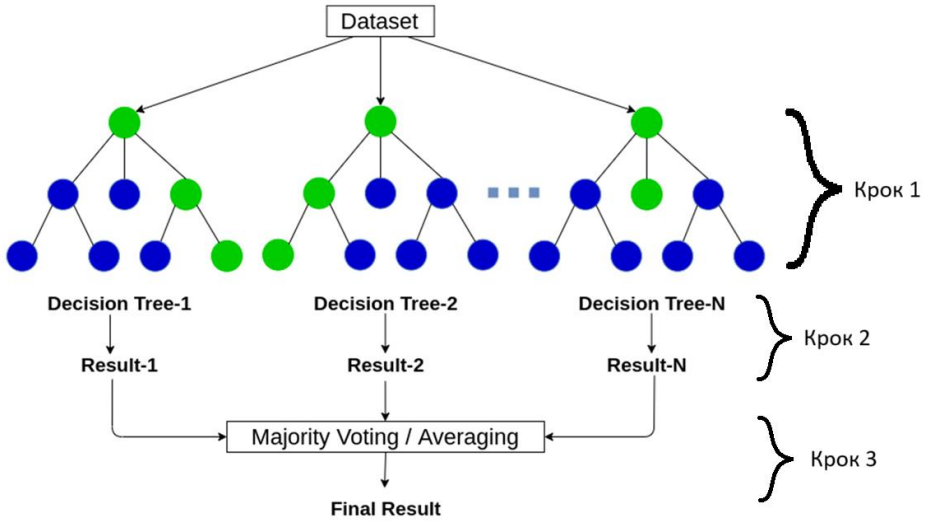

Рисунок 1.9 Покрокова робота алгоритму RF

Алгоритм RF має як переваги та недоліки.

Переваги:

- Може добре обробляти дані навіть якщо частина з них відсутні.
- Забезпечує велику точність прогнозу для великих даних.
- Кожне дерево може будуватися паралельно, що забеспечує пришвидшення процесу навчання.

Недоліки:

- Може знадобитися велика кількість пам'яті при великій кількості дерев.
- Важко заглянути всередину моделі і сказати чому було зроблене те чи інше передбачення.

XGBoost (eXtreme Gradient Boosting) – це передовий алгоритм градієнтного бустингу, що працює за принципом послідовного створення дерев рішень. Кожне наступне дерево розроблене для виправлення похибок, допущених попередніми. Такий метод дає змогу алгоритму з часом підвищувати точність прогнозів, приділяючи особливу увагу важким випадкам, тобто тим, які раніше були неправильно класифіковані або спрогнозовані.

Розглянемо покрокову роботу алгоритму XGBoost.

Крок 1: Початковий базовий прогноз

- Спершу XGBoost ініціалізує процес з простої, "базової" моделі, яка робить початковий прогноз для всього набору даних.
- У випадках регресії, ця базова модель зазвичай представляє собою середнє значення цільової змінної.
- Для задач класифікації початковий прогноз може бути пов'язаний з логарифмічними шансами цільової змінної.

Крок 2: Обчислення помилок

- Після формування базового прогнозу, алгоритм проводить розрахунок похибок, або псевдо-залишків. Це розбіжність між реальними показниками цільової змінної та прогнозами поточної ансамблевої моделі. Дані похибки перетворюються на нову цільову змінну для тренування наступного дерева.

Крок 3: Навчання наступного дерева на помилках

- Наступне дерево рішень вивчає не вихідні дані безпосередньо, а помилки, обчислені попереднім деревом. Завдання цього нового

дерева – відшукати в цих помилках певні закономірності та створити прогноз, щоб з максимальною точністю їх передбачити.

- Отже, кожне наступне дерево намагається виправити недоліки, що виникли в результаті роботи всіх попередніх дерев разом.

Крок 4: Повторення процесу

- Кроки 2 та 3, повторюються циклічно. Кожне наступне дерево вираховує помилки, які залишилися від загальної сукупності всіх попередніх дерев, і вчиться на цих нових залишках, намагаючись звести їх до мінімуму.

Крок 5: Об’єднання прогнозів

- Кінцевий прогноз XGBoost для будь-якого нового вхідного прикладу є сумою прогнозів усіх дерев у ансамблі.
- Для регресії: Це просто сума прогнозів, кожен з яких масштабується параметром швидкості навчання.
- Для класифікації: Сума прогнозів дерев перетворюється аби отримати ймовірності належності до певних класів.

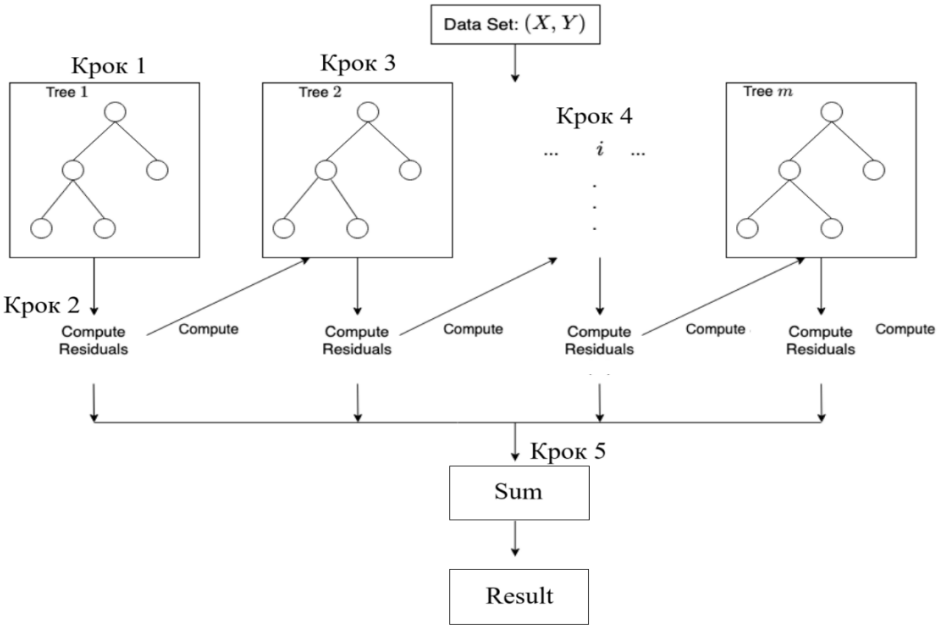

Рисунок 1.10 Покрокова робота алгоритму RF

Алгоритм XGBoost має як переваги та недоліки.

Переваги:

- Висока точність.
- Оптимізація та швидкість.

Недоліки:

- Складність налаштування.
- Потреба в обчислювальних ресурсах.

Logistic Regression (LR) – це ефективний та широко застосовуваний статистичний алгоритм, розроблений для вирішення завдань класифікації. На відміну від лінійної регресії, яка передбачає безперервні числові значення, логістична регресія визначає ймовірність того, що конкретне спостереження відноситься до певної категорії. Дана модель особливо добре зарекомендувала себе в задачах бінарної класифікації (наприклад "вижив" чи "не вижив") проте може бути адаптована і для багатокласової класифікації.

Розглянемо покрокову роботу алгоритму Logistic Regression.

Крок 1: Обчислення лінійної комбінації параметрів

- Так само, як і лінійна регресія, логістична регресія стартує з обчислення зваженої суми вхідних характеристик. Кожна характеристика множиться на свій відповідний ваговий коефіцієнт, а далі ці добутки підсумовуються разом з вільним членом, або ж зміщенням.

Крок 2: Використання логістичної функції

- На відміну від лінійної регресії, де лінійна сума безпосередньо використовується для передбачення, логістична регресія застосовує логістичну функцію до цієї суми, щоб отримати результат.

Крок 3: Винесення вердикту щодо класифікації

- Найбільш вживаний поріг – 0.5. Коли обчислена ймовірність дорівнює або перевищує 0.5, модель передбачає, що спостереження належить

до позитивного класу. Якщо ймовірність менша за 0.5, робиться прогноз негативного класу.

Крок 4: Тренування моделі

- Процес тренування логістичної регресії полягає у відшуканні найкращих значень вагових коефіцієнтів, які найточніше описують взаємозв'язок між вхідними ознаками та ймовірностями класів.

Алгоритм LR має як переваги та недоліки.

Переваги:

- Добра для бінарної класифікації.
- Ефективність та швидкість.

Недоліки:

- Вимагає даних великої розмірності.
- Працює тільки з бінарними змінними.

РОЗДІЛ 2. ПРАКТИЧНЕ ЗАВДАННЯ

2.1 Вибір середовища розробки

Для успішного виконання проєкту, в якому передбачено завдання машинного навчання, аналіз даних і створення моделей класифікації, було обрано Visual Studio Code (VS Code) як головне середовище розробки. Такий вибір аргументовано його гнучкістю, великими можливостями розширення та чудовою підтримкою мови Python, що є критичним для цього проєкту.

Visual Studio Code – це крос платформний текстовий редактор, розроблений компанією Microsoft для операційних систем Windows, Linux та macOS. Це "легкий" інструмент, орієнтований на розробку різноманітних застосунків. Для завдань, пов'язаних із машинним навчанням, VS Code пропонує вбудовані можливості та потужну інтеграцію з інструментами, що значно полегшують робочий процес.

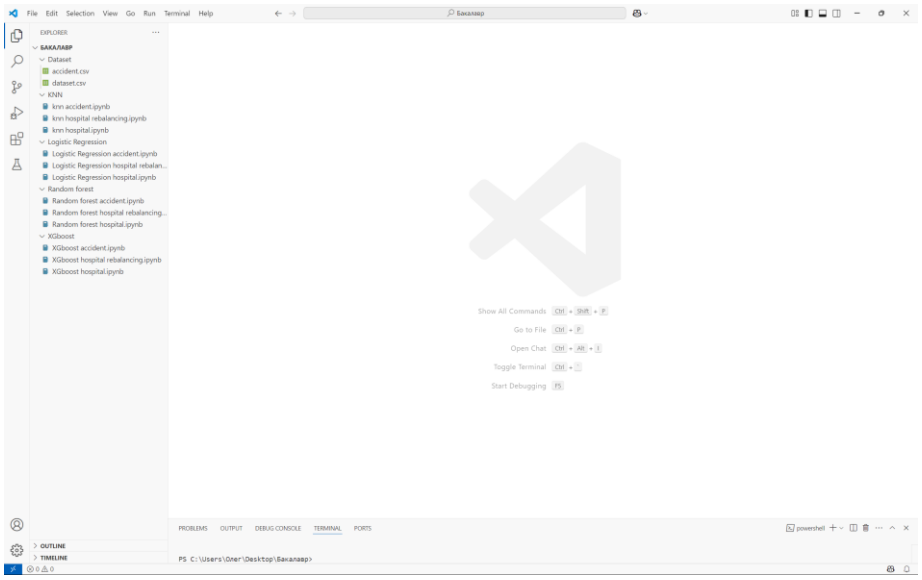

Рисунок. 2.1 Програмне середовище VS Code

Ключові аспекти VS Code для даної роботи:

Підтримка Python та спеціалізованих доповнень: VS Code реалізовано підтримку Python, що включає виділення синтаксису, діагностику помилок та форматування.

Підтримка бібліотек NumPy, Pandas, Scikit-learn.

Вбудований відладчик: Наявність потужного відладчика для Python є незамінною для виявлення та усунення помилок у складному коді моделей та алгоритмів.

Гнучкість робочого середовища: VS Code дозволяє відкривати один чи декілька каталогів як робочі області. Це надає можливість гнучко організовувати файли проекту, зокрема, блокноти Jupyter (.ipynb), набори даних та файли результатів (.csv). Непотрібні файли та папки можуть бути виключені з структури проекту, використовуючи конфігураційні налаштування, забезпечуючи таким чином впорядкований робочий простір.

2.2 Вибір мови програмування

Для розробки дипломного проекту, зосередженого на машинному навчанні, аналізі даних та створенні класифікаційних моделей, було віддано перевагу мові Python. Таке рішення пояснюється неабиякою популярністю Python у цих сферах, наявністю великої екосистеми бібліотек та інших засобів.

Python – це потужна, високо інтерпретована мова програмування загального призначення. Дана мова дає змогу розробникам продуктивно втілювати в життя складні алгоритми, працювати з великими масивами інформації та наочно представляти результати.

Ключові аспекти використання Python у проєкті:

Кількість бібліотек: Python може похвалитися великою кількістю бібліотек, які є ключовими для машинного навчання та аналітики:

NumPy і Pandas для ефективної взаємодії з числовими даними та керування табличними даними (схожими на дані, що зберігаються у файлах .csv).

Scikit-learn для застосування широкого спектру алгоритмів машинного навчання, включаючи моделі класифікації (такі як Логістична

регресія, Випадковий ліс, XGBoost), інструменти для попередньої обробки даних та оцінювання моделей.

Matplotlib та Seaborn для потужної візуалізації даних та результатів аналізу.

Файли та середовище розробки:

Основний формат файлів, який використовуватиметься для розробки та проведення експериментів – Jupyter Notebook (.ipynb). Ці інтерактивні файли надзвичайно зручні для поетапного виконання коду, розміщення коментарів, візуалізацій та кінцевих результатів. Вони є важливими для досліджень у сфері машинного навчання.

Файли .csv будуть використовуватися для зберігання вихідних наборів даних, а також для експорту отриманих результатів аналізу та передбачень моделей класифікації.

Для комфортної та продуктивної розробки передбачено використання середовища Visual Studio Code (VS Code), в комбінації з розширенням Jupyter. Це надасть всі необхідні інструменти для роботи з Python, запуску ноутбуків, налагодження коду та інтеграції з системою контролю версій.

2.3 Оцінка основних бібліотек

Основною мовою для реалізації дипломного проєкту є Python із використанням сучасних бібліотек машинного навчання та аналізу даних. Для роботи з даними застосовуються pandas – зручна бібліотека для обробки табличних даних, яка дозволяє ефективно проводити попередню обробку, фільтрацію та об’єднання датасетів.

Для побудови моделей машинного навчання використовуються бібліотеки scikit-learn та XGBoost.

- scikit-learn – це потужна і широко розповсюджена бібліотека, що пропонує готові реалізації базових та розширених моделей машинного навчання, а також інструменти для масштабування даних,

кодування категоріальних ознак і оцінки якості моделей через метрики та крос-валідацію.

- XGBoost – популярна бібліотека для градієнтного бустингу, що дозволяє отримати високоточні прогнози за допомогою ансамблю слабких моделей.

Для візуалізації результатів і аналізу використовуються бібліотеки matplotlib та seaborn, що дозволяють будувати графіки якості моделей, криві ROC, важливості ознак та інші наочні діаграми.

Переваги використання цього стеку бібліотек:

- Гнучкість і модульність: Кожен етап – від обробки даних до побудови і оцінки моделей – реалізовано за допомогою спеціалізованих бібліотек, що оптимізовані для відповідних задач.
- Велика спільнота та документація: Усі бібліотеки мають велику кількість прикладів і добре підтримуються, що спрощує розробку та налагодження.
- Можливість інтеграції: Використання відкритих і популярних бібліотек дозволяє легко додавати нові методи або змінювати існуючі без переписування великої частини коду.
- Ефективність: Бібліотеки оптимізовані під сучасні вимоги машинного навчання і підтримують багатопоточність та швидке виконання

2.4 Структура дипломної роботи

Структура проекту містить велику кількість папок, в кожній папці міститься декілька файлів.

Існує папка Dataset, котра містить 2 бази даних:

- accident.csv – база даних автомобільних аварій з невеликою кількістю ознак і рядків.
- dataset.csv – база даних пацієнтів в реанімації з великою кількістю ознак і рядків.

В проекті є 4 папки для моделей, в кожній папці знаходиться по 3 файли. В кожному файлі міститься код, який виконує:

- Зчитування файлу датасета.
- Виконує процес підготовки даних перед навчанням моделі.
- Проводиться навчання моделі.
- Виконується розрахунок точності моделі.

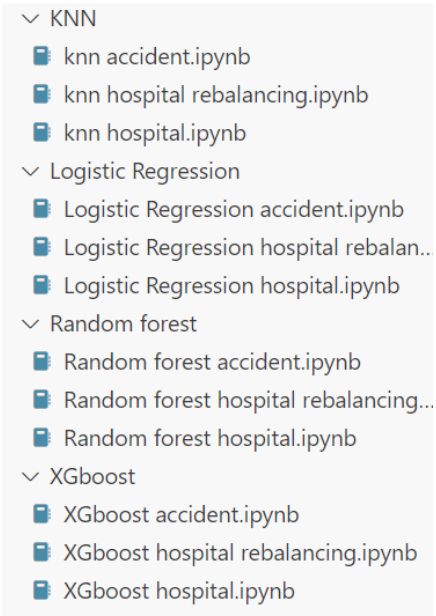

Рисунок 2.2 Структура папок для моделей

Загалом в проекті використовується 5 папок, 2 .csv файла, та 9 .ipynb файлів з кодом.

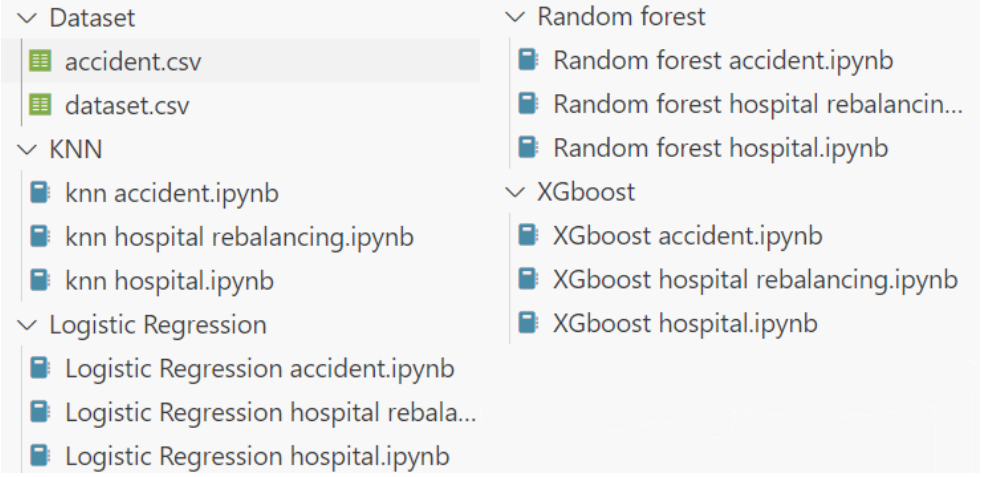

Рисунок 2.3 Загальна структура дипломної роботи

2.5 Процес підготовки даних перед навчанням моделей

В проекті використовується два набори даних:

- В accident.csv описані дані автомобільних аварій. Містить ~3000 рядків даних. Даний набір даних містить такі ознаки: Age, Gender, Speed_of_Impact, Helmet_Used, Seatbelt_Used, Survived.
- В dataset.csv описані медичні дані пацієнтів. Містить ~ 91000 рядків даних. Даний набір даних містить велику кількість ознак, а точніше 83 ознаки

Розглянемо процес підготовки даних в accident.csv:

- Спершу заповнимо пропуски в даних. Для цього в стовбцях Gender, Speed_of_Impact, Helmet_Used, Seatbelt_Used пропущені значення заповнюються модою, тобто найчастішим значенням у стовпці.
- Дальше розділимо ознаку Age та ознаку Speed_of_Impact на декілька ознак. Age розділимо на Child, Adult, Old. Speed_of_Impact розділимо на Low_Speed, Medium_Speed, High_Speed, Very_High_Speed.
- В кінці перетворимо всі категоріальні ознаки на числові.

```
df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])
df['Speed_of_Impact'] = df['Speed_of_Impact'].fillna(df['Speed_of_Impact'].mean())
df['Helmet_Used'] = df['Helmet_Used'].fillna(df['Helmet_Used'].mode()[0])
df['Seatbelt_Used'] = df['Seatbelt_Used'].fillna(df['Seatbelt_Used'].mode()[0])

df['Child'] = (df['Age'] < 18).astype(int)
df['Adult'] = ((df['Age'] >= 18) & (df['Age'] <= 55)).astype(int)
df['Old'] = (df['Age'] > 55).astype(int)
df = df.drop(columns=['Age'])

df['Low_Speed'] = (df['Speed_of_Impact'] < 40).astype(int)
df['Medium_Speed'] = ((df['Speed_of_Impact'] >= 40) & (df['Speed_of_Impact'] < 80)).astype(int)
df['High_Speed'] = ((df['Speed_of_Impact'] >= 80) & (df['Speed_of_Impact'] < 120)).astype(int)
df['Very_High_Speed'] = (df['Speed_of_Impact'] >= 120).astype(int)
df = df.drop(columns=['Speed_of_Impact'])

label_encoders = {}
for col in ['Gender', 'Helmet_Used', 'Seatbelt_Used']:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

Рисунок 2.4 Підготовка даних в accident.csv

Розглянемо процес підготовки даних в dataset.csv:

- Спершу видалимо ознаки, які не мають е несуть аналітичної цінності для навчання моделі.

- Заповнимо пропуски в даних. Для цього в категоріальних стовбцях пропущені значення заповнюються модою, тобто найчастішим значенням у стовпці. Для числових ознак пропущені значення заповнюються середнім значенням.
- В кінці перетворимо всі категоріальні ознаки на числові.

```
df = df.drop(columns=['encounter_id', 'patient_id', 'hospital_id', 'Unnamed: 83'])

for col in df.columns:
    if df[col].dtype == 'object':
        df[col] = df[col].fillna(df[col].mode()[0])
    else:
        df[col] = df[col].fillna(df[col].mean())

label_encoders = {}
for col in df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

Рисунок 2.5 Підготовка даних в dataset.csv

Після всіх перетворень, отримані дані поділяються на ознаки (X) та цільову змінну (y). Потім відбувається розбиття даних на тренувальний набір (80%) та тестовий набір (20%) за допомогою функції train_test_split.

Дальше відбувається масштабування числових ознак. Для цього використовується StandardScaler з бібліотеки sklearn.preprocessing.

Також було виявлено, що дані з dataset.csv мали великий дисбаланс класів. Для більш об'єктивної оцінки, було створено копію кожної моделі, котра працювала з dataset.csv. В кожній копії було проведено балансування класів, шляхом збільшенням кількості не домінуючого класу.

```
df_0 = df[df.hospital_death == 0]
df_1 = df[df.hospital_death == 1]

df_1_upsampled = resample(df_1, replace=True, n_samples=len(df_1) * 10, random_state=42)
df_0_downsampled = resample(df_0, replace=False, n_samples=len(df_1_upsampled), random_state=42)

df_balanced = pd.concat([df_1_upsampled, df_0_downsampled]).sample(frac=1, random_state=42)

X = df_balanced.drop(columns='hospital_death')
y = df_balanced['hospital_death']

print("Розподіл після балансування:")
print(y.value_counts())
```

Рисунок 2.6 Балансування класів в dataset.csv

2.6. Процес створення моделей

В даній дипломній роботі було використано чотири моделі машинного навчання для класифікації виживання в задачах на виживання

Модель K-найближчих сусідів (KNN) базується на концепції класифікації об'єктів за найближчими сусідами у просторі характеристик, орієнтуючись на відстань.

Спершу проводиться підбір гіперпаметрів.

Для початку створимо словник param_grid, який містить гіперпараметри:

- 'n_neighbors' – кількість найближчих сусідів, в даному випадку перевіряються всі значення від 1 до n.
- 'metric' – метрика відстані. Тут вказується метрика визначення близькість сусідів.

Дальше виконується крос-валідація. Для цього спершу налаштується метод крос-валідації:

- StratifiedKFold – це тип крос-валідації, який забезпечує, щоб в кожному розбитті буде зберігатися така сама пропорція класів, як і в всьому наборі даних. Це потрібно для незбалансованих наборів даних, щоб в одному розбиті були присутні обидві представники класу.
- n_splits – визначає, скільки раз модель буде навчатися, і скільки буде розбитів для навчання і тестування.
- shuffle – визначає чи будуть перемішуватися дані перед розбиттям.
- random_state – даний параметр потрібен, якщо потрібно зафіксувати випадкове розбиття. Це може знадобитися, якщо потрібно порівняти моделі, або провести повторне навчання з іншими параметрами і подивитися, чи було покращення результатів.

Тепер створюється об'єкт котрий буде виконувати всю роботу. Для цього треба вказати певні параметри:

- estimator – тут вказується модель до якої буде вказуватися певні параметри.
- param_grid – тут передається словник з гіперпараметрами.
- cv – тут вказується метод крос-валідації.
- scoring – вказуємо метрикою, за якою буде оцінюватися ефективність.
- n_jobs – вказує скільки ядер буде використовуватися пристроєм для паралельного розрахунку.

Тепер запускається процес пошуку, отримується найкраща модель і виводяться найкращі параметри.

```
param_grid = {
    'n_neighbors': list(range(1, 100)),
    'metric': ['euclidean', 'manhattan']
}

knn = KNeighborsClassifier()

cv = StratifiedKFold(n_splits=36, shuffle=True, random_state=42)

grid_search = GridSearchCV(estimator=knn, param_grid=param_grid, cv=cv, scoring='accuracy', n_jobs=-1, verbose=1)
grid_search.fit(X_train_scaled, y_train)

best_knn = grid_search.best_estimator_
print("Найкращі параметри:", grid_search.best_params_)
```

```
Fitting 36 folds for each of 198 candidates, totalling 7128 fits
Найкращі параметри: {'metric': 'euclidean', 'n_neighbors': 68}
```

Рисунок 2.7 Підбір найкращих параметрів в KNN

Логістична регресія – це лінійна модель, призначена для бінарної класифікації. Дана модель обчислює ймовірність віднесення об’єкта до конкретного класу, використовуючи логістичну функцію активації.

Спершу проводиться підбір гіперпаметрів.

Для початку створимо словник param_grid, який містить гіперпараметри:

- 'C' – параметр котрий контролює силу регуляризації. Чим менше значення ти буде більш сильної регуляризація.
- 'penalty' – визначає тип регуляризації.

- 'solver' – визначає алгоритм оптимізації, який використовується для пошуку оптимальних значень.
- 'max_iter' – визначає максимальну кількість ітерації який алгоритм буде виконувати.

Дальше виконується крос-валідація. Для цього спершу налаштовується метод крос-валідації, та створюється об'єкт котрий буде виконувати всю роботу.

Тепер запускається процес пошуку, отримується найкраща модель і виводяться найкращі параметри.

```
param_grid = {
    'C': [0.01, 0.1, 1.0, 10.0],
    'penalty': ['l1', 'l2'],
    'solver': ['liblinear'],
    'max_iter': [500]
}

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

log_reg = LogisticRegression(random_state=42)

grid_search = GridSearchCV(estimator=log_reg, param_grid=param_grid, cv=cv, n_jobs=-1, scoring='accuracy', verbose=1)
grid_search.fit(X_train_scaled, y_train)

best_model = grid_search.best_estimator_
print("Найкращі параметри:", grid_search.best_params_)
```

✓ 1.7s

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
Найкращі параметри: {'C': 0.01, 'max_iter': 500, 'penalty': 'l2', 'solver': 'liblinear'}
```

Рисунок 2.8 Підбір найкращих параметрів в Logistic Regression

Метод випадкового лісу (Random Forest) – це ансамбль багатьох дерев рішень, кожне з яких навчається на випадковій вибірці даних та характеристик.

Спершу проводиться підбір гіперпаметрів.

Для початку створимо словник param_grid, який містить гіперпараметри:

- 'n_estimators' – визначає кількість дерев рішень, що будуть використовуватися при навчанні.
- 'max_depth' – визначає максимальну глибину дерев.

- 'min_samples_split ' – мінімальна кількість зразків, при якій вузол зможе ділитися дальше.
- 'min_samples_leaf ' – мінімальна кількість зразків, які повинні бути в останньому вузлі (листку).

Дальше виконується крос-валідація. Для цього спершу налаштовується метод крос-валідації, та створюється об'єкт котрий буде виконувати всю роботу.

Тепер запускається процес пошуку, отримується найкраща модель і виводяться найкращі параметри.

```
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

rf = RandomForestClassifier(random_state=42)

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=cv, scoring='accuracy', n_jobs=-1, verbose=1)
grid_search.fit(X_train_scaled, y_train)

best_rf = grid_search.best_estimator_

print("Найкращі параметри:", grid_search.best_params_)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Найкращі параметри: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
```

Рисунок 2.9 Підбір найкращих параметрів в Random forest

XGBoost – реалізація градієнтного бустингу, який послідовно комбінує слабкі моделі (дерева рішень), оптимізуючи функцію втрат.

Спершу проводиться підбір гіперпаметрів.

Для початку створимо словник param_grid, який містить гіперпараметри:

- 'n_estimators' – визначає кількість дерев рішень, що будуть використовуватися при навчанні.
- 'max_depth' – визначає максимальну глибину дерев.
- 'learning_rate' – визначає коефіцієнт навчання, чим менше значення тим повільніше навчання, але менший шанс перенавчання.

- 'subsample' – визначає процент тренувального набору даних для кожного дерева.

Дальше виконується крос-валідація. Для цього спершу налаштовується метод крос-валідації, та створюється об'єкт котрий буде виконувати всю роботу.

Тепер запускається процес пошуку, отримується найкраща модель і виводяться найкращі параметри.

```
param_grid = {
    'n_estimators': [200, 500],
    'max_depth': [6, 10],
    'learning_rate': [0.01, 0.1, 0.05],
    'subsample': [0.8, 1.0]
}

xgb = XGBClassifier(random_state=42, eval_metric='logloss', verbosity=0)

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

grid_search = GridSearchCV(estimator=xgb, param_grid=param_grid, cv=cv, n_jobs=-1, scoring='recall', verbose=1)
grid_search.fit(X_train_scaled, y_train)

best_model = grid_search.best_estimator_
print("Найкращі параметри:", grid_search.best_params_)
```

✓ 3m 56.8s

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
Найкращі параметри: {'learning_rate': 0.1, 'max_depth': 10, 'n_estimators': 500, 'subsample': 0.8}
```

Рисунок 2.10 Підбір найкращих параметрів в XGboost

2.7. Результати

На заключному етапі було здійснено оцінювання точності кожної з втілених моделей класифікації за допомогою метрики ROC-крива, Precision-Recall крива та точності (Accuracy). Остання відображає відсоток правильно розпізнаних прикладів серед загальної кількості.

2.7.1 Оцінювання моделі KNN

Спершу розглянемо результати модель KNN на даних автомобільних аварій accident.csv.

Після навчання моделі KNN, розраховується її точність, і вона склала: 0.7783, або 77,83%. Це означає, що модель правильно спрогнозувала 77,83% випадків автомобільних аварій.

```
Точність (Accuracy): 0.7783
Звіт про класифікацію:
              precision    recall  f1-score   support

           0       0.63      0.50      0.56       169
           1       0.82      0.89      0.85       431

    accuracy                           0.78       600
   macro avg       0.73      0.69      0.71       600
weighted avg       0.77      0.78      0.77       600
```

Рисунок 2.11 Точність моделі KNN на наборі даних accident.csv

Для кращого розуміння продуктивності моделі, буде проаналізовано матрицю невідповідності. Вона краще демонструє, як модель класифікує кожен клас, як їх розділяє на правильні та неправильні передбачені випадки.

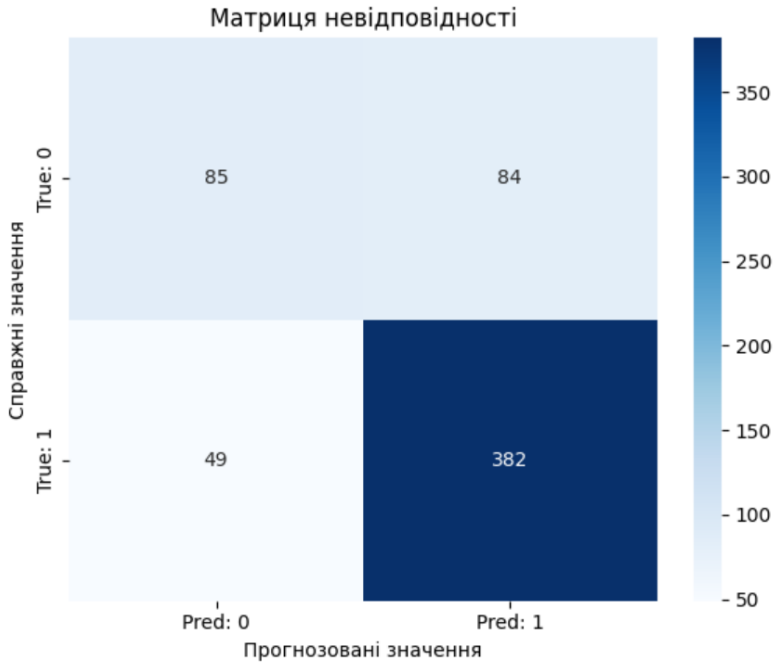

Рисунок 2.12 Матриця невідповідності KNN на наборі даних accident.csv

Розглянемо детальніше матрицю невідповідності:

- True Positive (TP): [382] – випадки, які було передбачено як Позитивний клас (1) і були насправді Позитивний клас (1).
- True Negative (TN): [85] – випадки, які було передбачено як Негативний клас (0) і були дійсно Негативний клас (0).

- False Positive (FP): [84] – випадки, які було передбачено як Позитивний клас (1), але дійсно були Негативний клас (0).
- False Negative (FN): [49] – випадки, які було передбачено як Негативний клас (0), але дійсно були Позитивний клас (1).

З даних значень, випливає, що модель добре передбачує позитивний клас (1), в той час як негативний клас (0) модель правильно передбачає близько 50% випадків.

Для кращої оцінки здатності моделі розрізняти класи, була побудована ROC-крива, та визначене значення AUC (площі під ROC-кривою). ROC-крива відображає відношення між часткою вірно класифікованих позитивних випадків (True Positive Rate) та часткою помилково класифікованих негативних випадків (False Positive Rate) при зміні порогу класифікації.

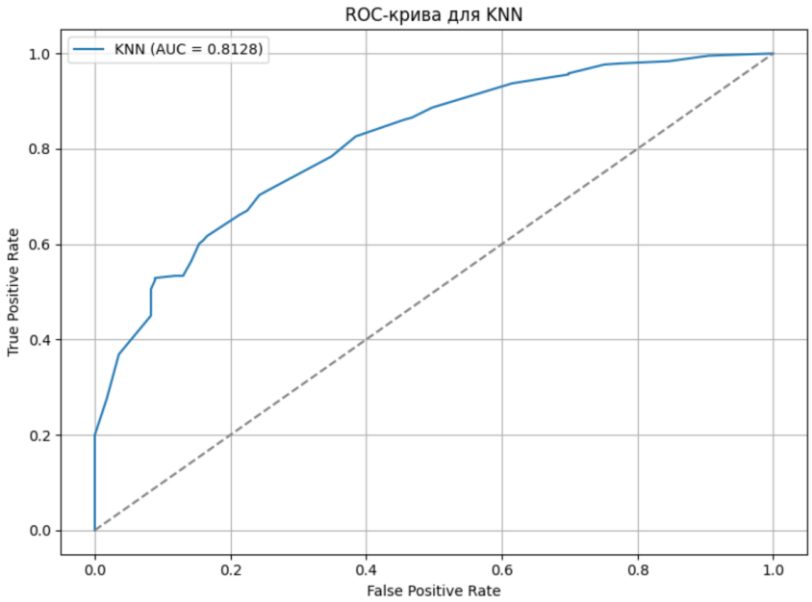

Рисунок 2.13 ROC-крива для KNN на наборі даних accident.csv

Як видно з графіка, значення AUC = 0.8128, що свідчить про добру здатність моделі відрізняти клас, оскільки значення 1 є ідеальним, а значення 0.5 вгадуванням.

Для додаткової оцінки якості роботи моделі слід збудувати графік Precision-Recall та визначити значення середньої точності (AP). Даний графік стає особливо корисним, коли спостерігається дисбаланс класів, тобто, кількість екземплярів одного класу переважає над кількість екземплярів іншого.

У подібних випадках висока загальна точність моделі може вводити в оману, адже точність може бути досягнута за рахунок коректної класифікації лише більшості випадків одного класу. Precision-Recall крива відображає співвідношення між точністю (Precision) та повнотою (Recall) при зміні порогу класифікації.

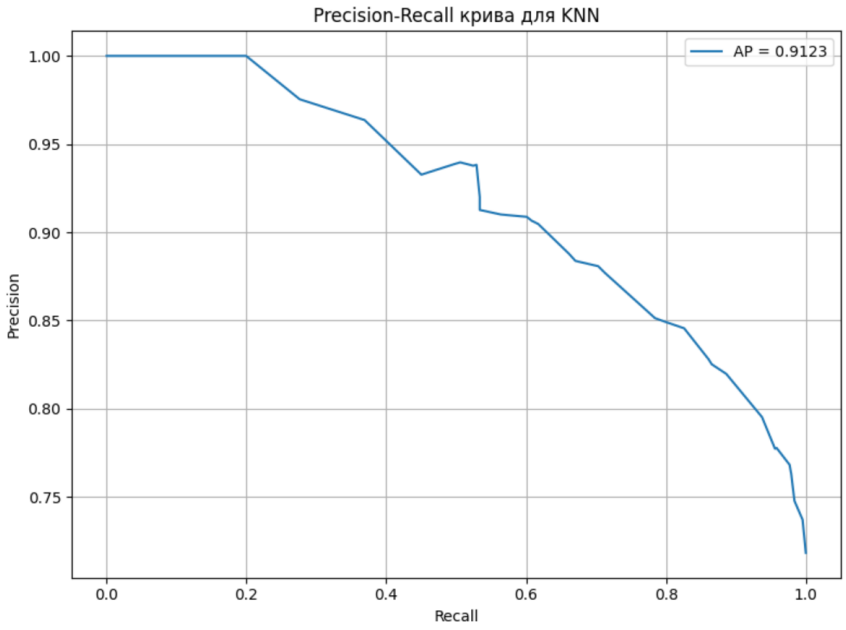

Рисунок 2.14 Precision-Recall -крива для KNN на наборі даних accident.csv

Як видно з графіка, значення AP = 0.9123, що свідчить про високу здатність моделі відрізняти клас, оскільки значення 1 є ідеальним, а значення 0.5 вгадуванням.

Тепер розглянемо результати модель KNN на даних пацієнтів реанімації dataset.csv.

Після навчання моделі KNN, розраховується її точність, і вона склала: 0.9227, або 92,27%.

```
Точність (Accuracy): 0.9227
Звіт про класифікацію:
              precision    recall  f1-score   support

           0       0.93      1.00      0.96     16760
           1       0.75      0.16      0.26      1583

    accuracy                           0.92     18343
   macro avg       0.84      0.58      0.61     18343
weighted avg       0.91      0.92      0.90     18343
```

Рисунок 2.15 Точність моделі KNN на наборі даних dataset.csv

Для кращого розуміння продуктивності моделі, буде проаналізовано матрицю невідповідності.

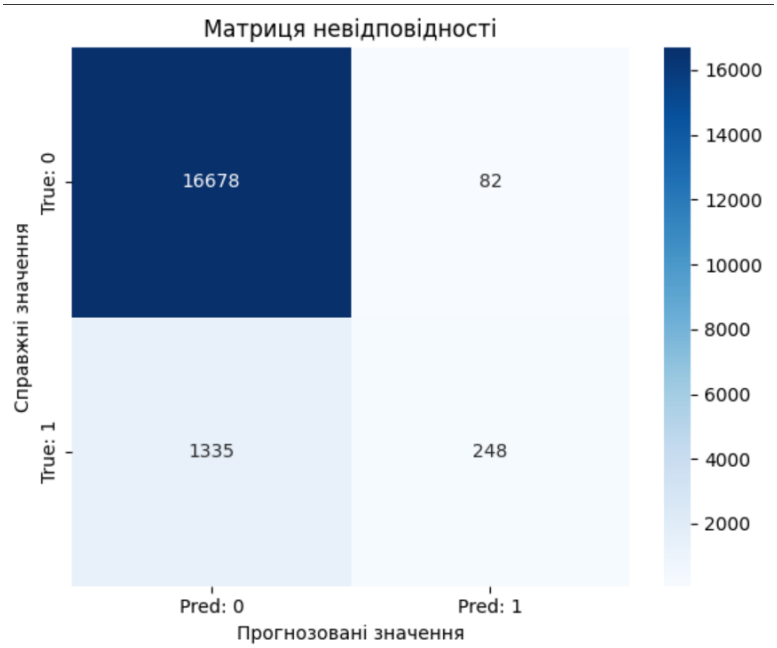

Рисунок 2.16 Матриця невідповідності KNN на наборі даних dataset.csv

Розглянемо детальніше матрицю невідповідності:

- True Positive (TP): 248.
- True Negative (TN): 16678.
- False Positive (FP): 82.
- False Negative (FN): 1335.

З даних значень можна зробити висновок, що дані мають великий дисбаланс класів. У таких умовах загальна точність моделі може бути

оманливою, адже модель правильно класифікує лише приклади домінуючого класу.

Для кращої оцінки побудуємо ROC-криву та обчислимо AUC.

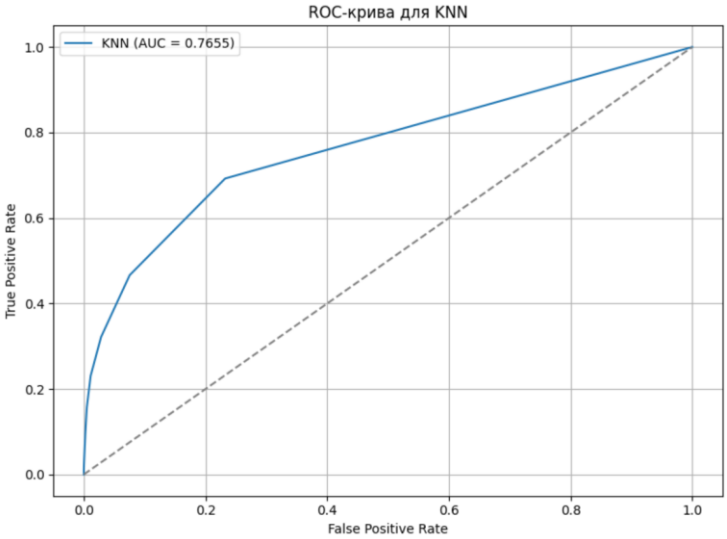

Рисунок 2.17 ROC-крива для KNN на наборі даних dataset.csv

Як видно з графіка, значення AUC = 0.7655, що свідчить про помірну здатність моделі відрізняти клас.

Також слід побудувати криву Precision-Recall та знайти значення AP, оскільки дана метрика краще відображає якість моделі при дисбалансі класів.

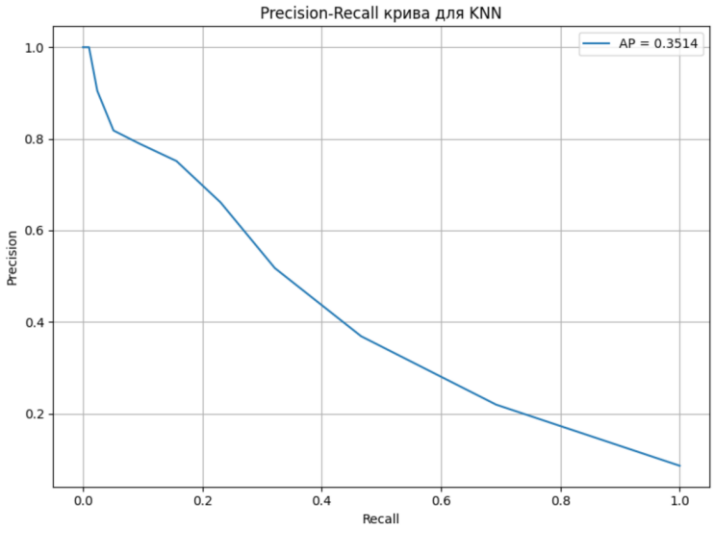

Рисунок 2.18 Precision-Recall -крива для KNN на наборі даних dataset.csv

Як видно з графіка, значення AP = 0.3514, що свідчить про труднощі моделі передбачати значення недомінуйочого класу.

В такому випадку слід провести балансування класів, після чого повторно оцінити модель на даних.

Після балансування класів, та повторного навчання моделі KNN, розраховується її точність, і вона склала: 0.9805, або 98,05%.

```
Точність (Accuracy): 0.9805
Звіт про класифікацію:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     16760
           1       0.82      1.00      0.90      1583

    accuracy                           0.98     18343
   macro avg       0.91      0.99      0.94     18343
weighted avg       0.98      0.98      0.98     18343
```

Рисунок 2.19 Точність моделі KNN на наборі даних dataset.csv

Для кращого розуміння продуктивності моделі, буде проаналізовано матрицю невідповідності.

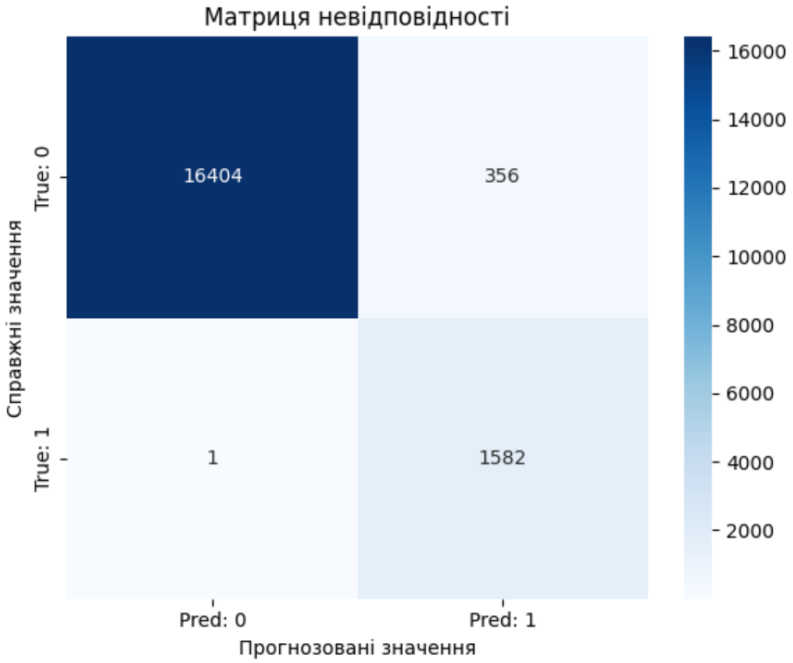

Рисунок 2.20 Матриця невідповідності після балансування класів в dataset.csv

Розглянемо детальніше матрицю невідповідності:

- True Positive (TP): 1582.
- True Negative (TN): 16404.
- False Positive (FP): 356.
- False Negative (FN): 1.

Після балансування класів, значення точності збільшилося, як видно з матриці невідповідності, дана модель стала краще розпізнавати випадки позитивного класу.

Для кращої оцінки побудуємо ROC-криву та обчисляємо AUC.

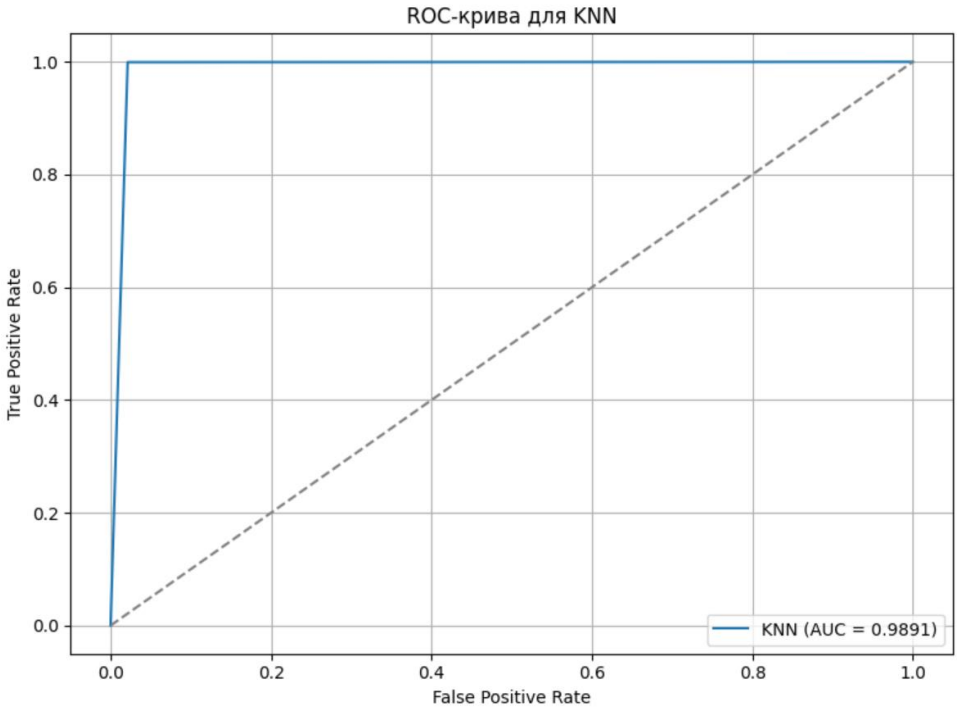

Рисунок 2.21 ROC-крива після балансування класів в dataset.csv

Як видно з графіка, значення AUC = 0.9891, що свідчить про чудову здатність моделі відрізняти клас.

Також слід побудувати криву Precision-Recall та знайти значення AP, оскільки дана метрика краще відображає якість моделі при дисбалансі класів.

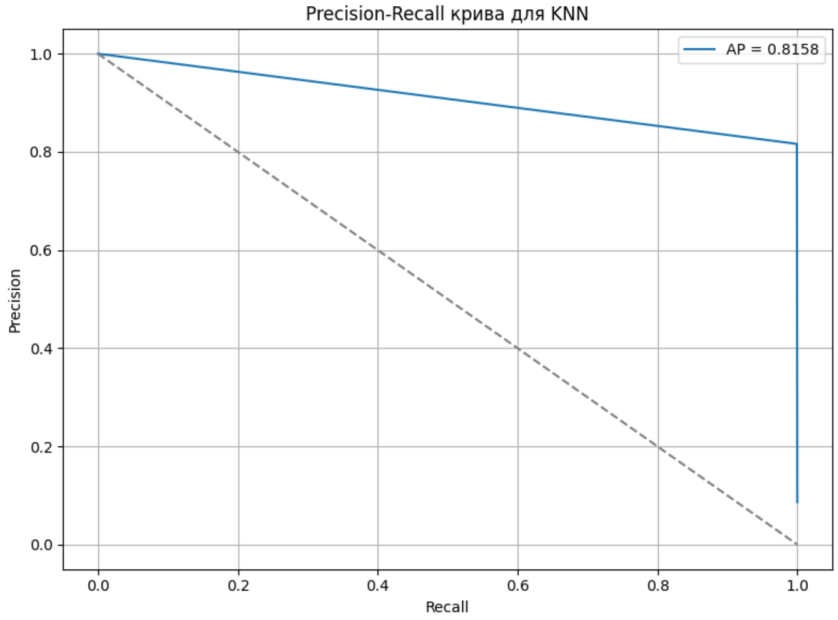

Рисунок 2.22 Precision-Recall крива після балансування класів в dataset.csv

Як видно з графіка, значення AP = 0.8158, що свідчить про добру здатність моделі передбачати значення недомінуйочого класу.

2.7.2 Оцінювання моделі Random forest

Спершу розглянемо результати модель Random forest (RF) на даних автомобільних аварій accident.csv.

Після навчання моделі RF, її точність, склала: 0.7833, або 78,33%. Це означає, що модель правильно спрогнозувала 78,33% випадків автомобільних аварій.

```
Точність (Accuracy): 0.7833
Звіт про класифікацію:
              precision    recall  f1-score   support

           0       0.64      0.52      0.58       169
           1       0.83      0.89      0.85       431

    accuracy                           0.78       600
   macro avg       0.73      0.70      0.71       600
weighted avg       0.77      0.78      0.78       600
```

Рисунок 2.23 Точність моделі RF на наборі даних accident.csv

Для кращого розуміння продуктивності моделі, буде проаналізовано матрицю невідповідності.

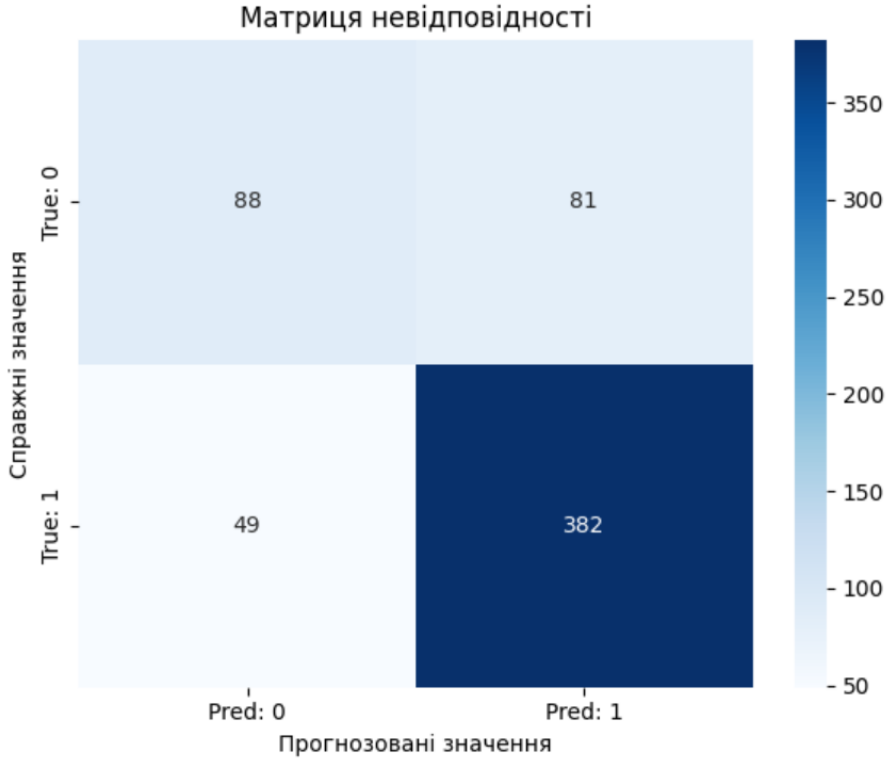

Рисунок 2.24 Матриця невідповідності RF на наборі даних accident.csv

Розглянемо детальніше матрицю невідповідності:

- True Positive (TP): 382
- True Negative (TN): 88
- False Positive (FP): 81
- False Negative (FN): 49

З даних значень, випливає, що модель добре передбачує позитивний клас, в той час як негативний клас модель правильно передбачає близько 50% випадків.

Для кращої оцінки здатності моделі розрізняти класи, побудуємо ROC-криву, та визначимо значення AUC (площі під ROC-кривою).

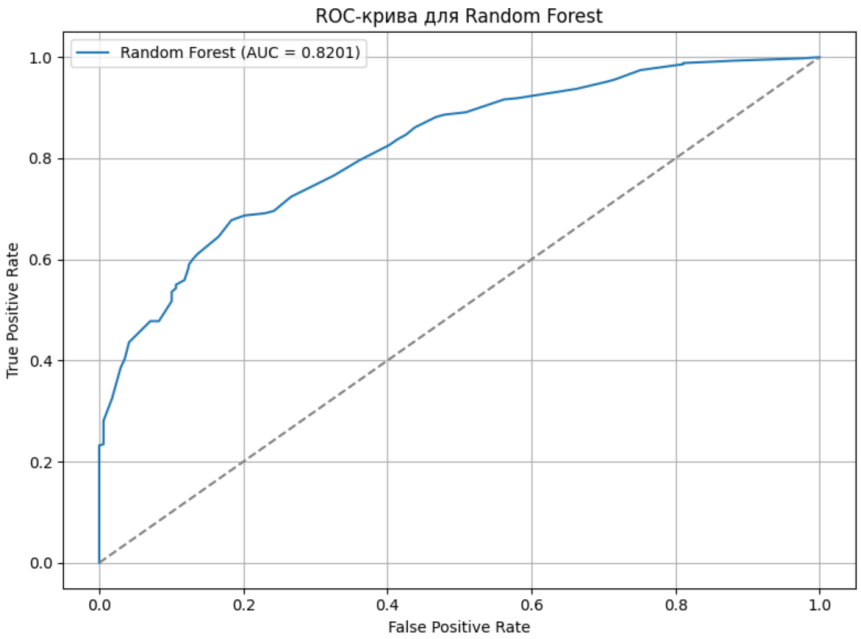

Рисунок 2.25 ROC-крива для RF на наборі даних accident.csv

Як видно з графіка, значення AUC = 0.8201, що свідчить про добру здатність моделі відрізняти клас, оскільки значення 1 є ідеальним, а значення 0.5 вгадуванням.

Також слід побудувати криву Precision-Recall та знайти значення AP, оскільки дана метрика краще відображає якість моделі при дисбалансі класів.

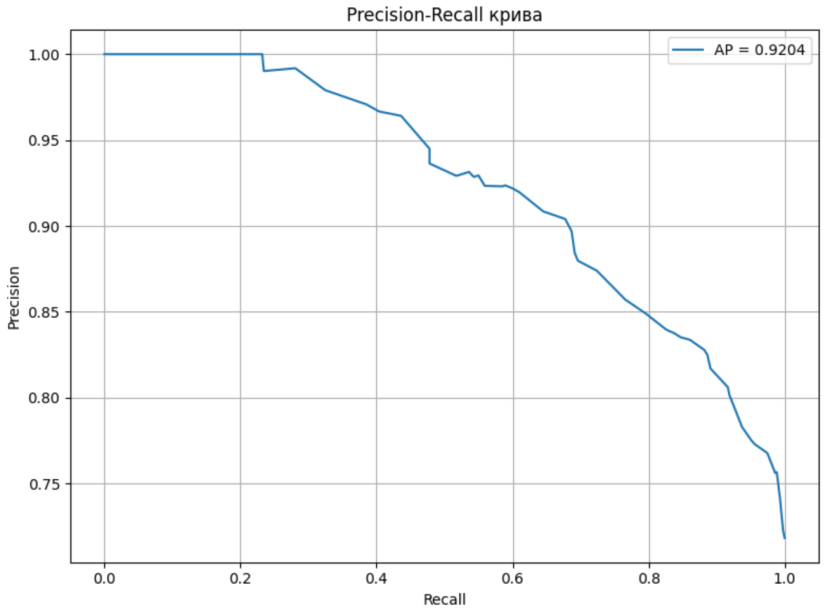

Рисунок 2.26 Precision-Recall -крива для RF на наборі даних accident.csv

Як видно з графіка, значення AP = 0.9204, що свідчить про високу здатність моделі відрізняти клас, оскільки значення 1 є ідеальним, а значення 0.5 вгадуванням.

Тепер розглянемо результати модель RF на даних пацієнтів реанімації dataset.csv.

Після навчання моделі RF, розраховується її точність, і вона склала: 0.9296, або 92,96%.

```
Точність (Accuracy): 0.9296
Звіт про класифікацію:
              precision    recall  f1-score   support

           0       0.94      0.99      0.96     16760
           1       0.75      0.28      0.40      1583

    accuracy                           0.93     18343
   macro avg       0.84      0.63      0.68     18343
weighted avg       0.92      0.93      0.91     18343
```

Рисунок 2.27 Точність моделі RF на наборі даних dataset.csv

Для кращого розуміння продуктивності моделі, буде проаналізовано матрицю невідповідності.

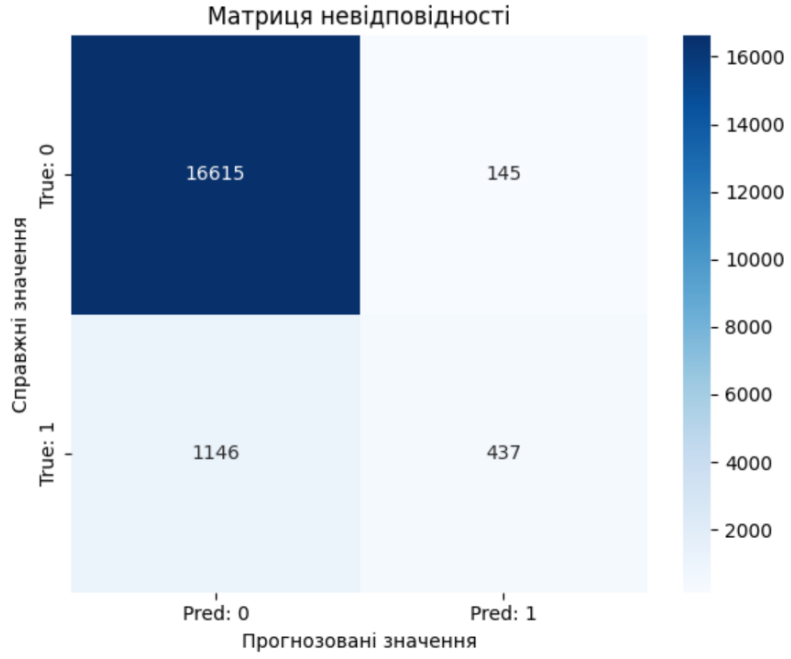

Рисунок 2.28 Матриця невідповідності RF на наборі даних dataset.csv

Розглянемо детальніше матрицю невідповідності:

- True Positive (TP): 437.
- True Negative (TN): 16615.
- False Positive (FP): 145.
- False Negative (FN): 1146.

З даних значень можна зробити висновок, що дані мають великий дисбаланс класів. У таких умовах загальна точність моделі може бути оманливою, адже модель правильно класифікує лише приклади домінуючого класу.

Для кращої оцінки побудуємо ROC-криву та обчислимо AUC.

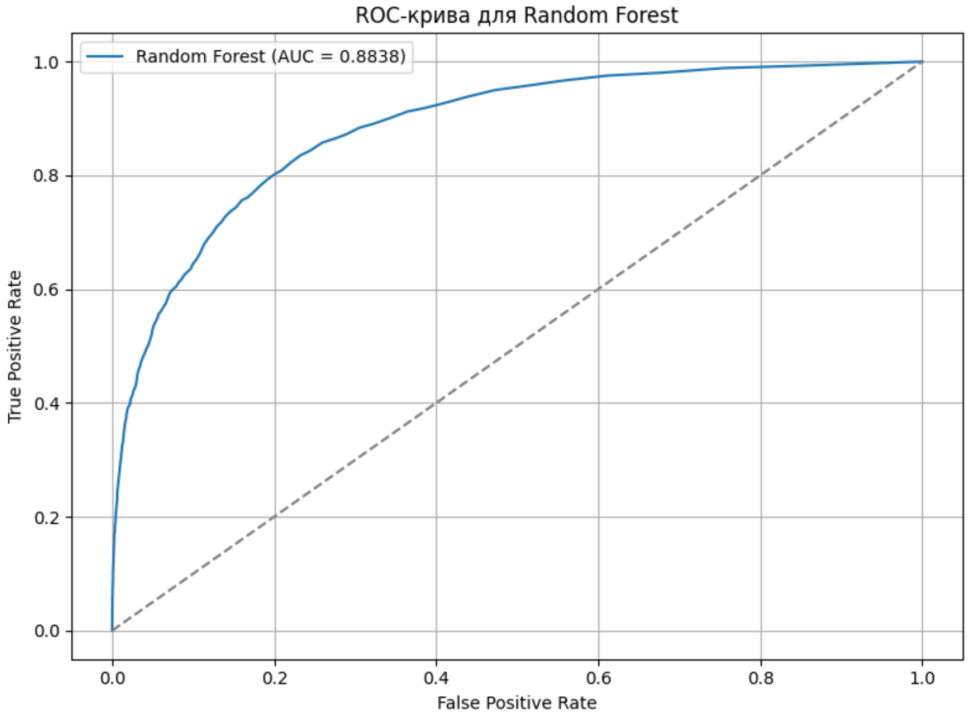

Рисунок 2.29 ROC-крива для RF на наборі даних dataset.csv

Як видно з графіка, значення AUC = 0.8838, що свідчить про добру здатність моделі відрізняти клас.

Також слід побудувати криву Precision-Recall та знайти значення AP, оскільки дана метрика краще відображає якість моделі при дисбалансі класів.

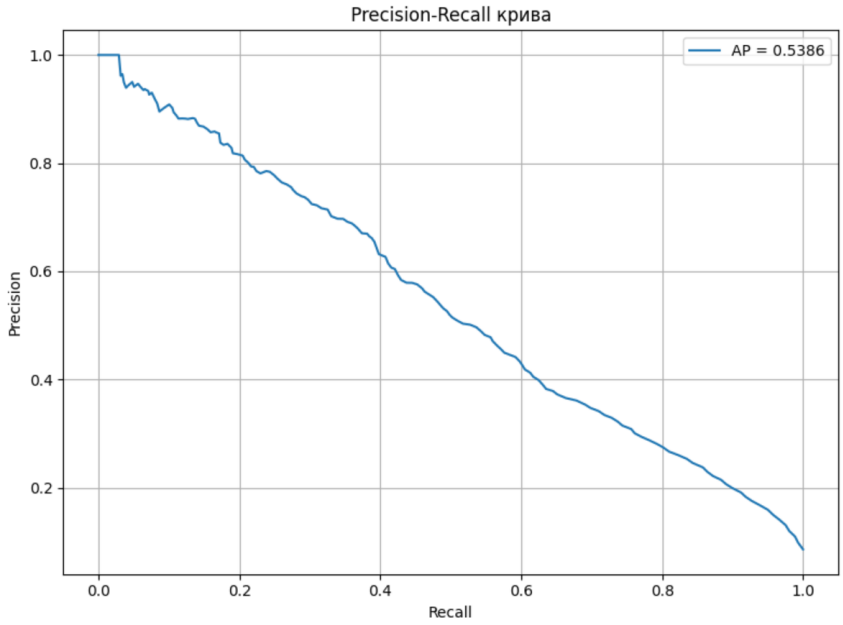

Рисунок 2.30 Precision-Recall -крива для RF на наборі даних dataset.csv

Як видно з графіка, значення AP = 0.5386, що свідчить про труднощі моделі передбачати значення не домінуйочого класу.

В такому випадку слід провести балансування класів, після чого повторно оцінити модель на даних.

Після балансування класів, та повторного навчання моделі RF, її точність склала: 0.9916, або 99,16%.

```
Точність (Accuracy): 0.9916
Звіт про класифікацію:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00     16760
           1       0.91      1.00      0.95      1583

    accuracy                           0.99     18343
   macro avg       0.96      1.00      0.97     18343
weighted avg       0.99      0.99      0.99     18343
```

Рисунок 2.31 Точність моделі RF після балансування класів в dataset.csv

Для кращого розуміння продуктивності моделі, проаналізуємо матрицю невідповідності.

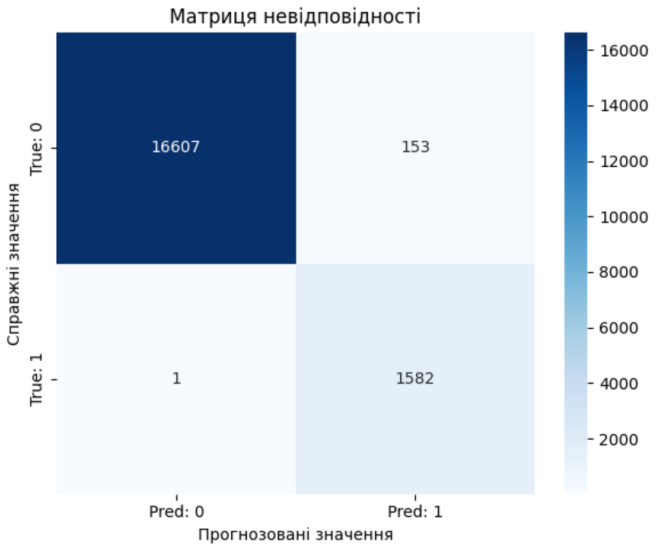

Рисунок 2.32 Матриця невідповідності після балансування в dataset.csv

Розглянемо детальніше матрицю невідповідності:

- True Positive (TP): 1582.
- True Negative (TN): 16607.
- False Positive (FP): 153.
- False Negative (FN): 1.

Після балансування класів, значення точності збільшилося, як видно з матриці невідповідності, дана модель стала краще розпізнавати випадки позитивного класу.

Для кращої оцінки побудуємо ROC-криву та обчисляємо AUC.

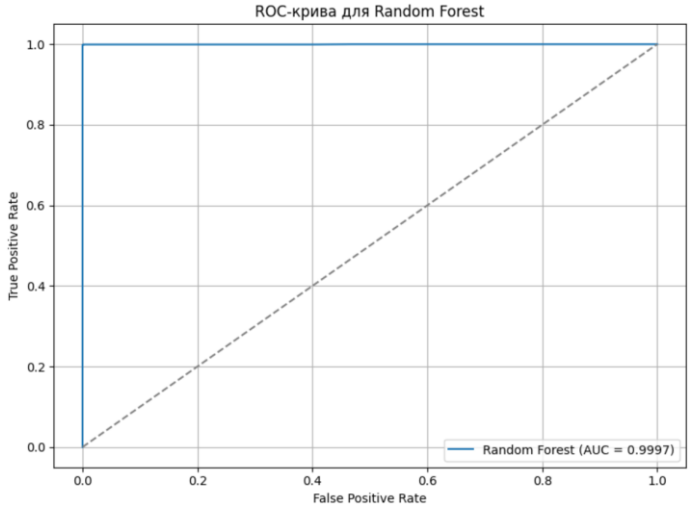

Рисунок 2.33 ROC-крива після балансування класів в dataset.csv

Як видно з графіка, значення AUC = 0.9997, що свідчить про чудову здатність моделі відрізняти клас.

Також слід побудувати криву Precision-Recall та знайти значення AP, оскільки дана метрика краще відображає якість моделі при дисбалансі класів.

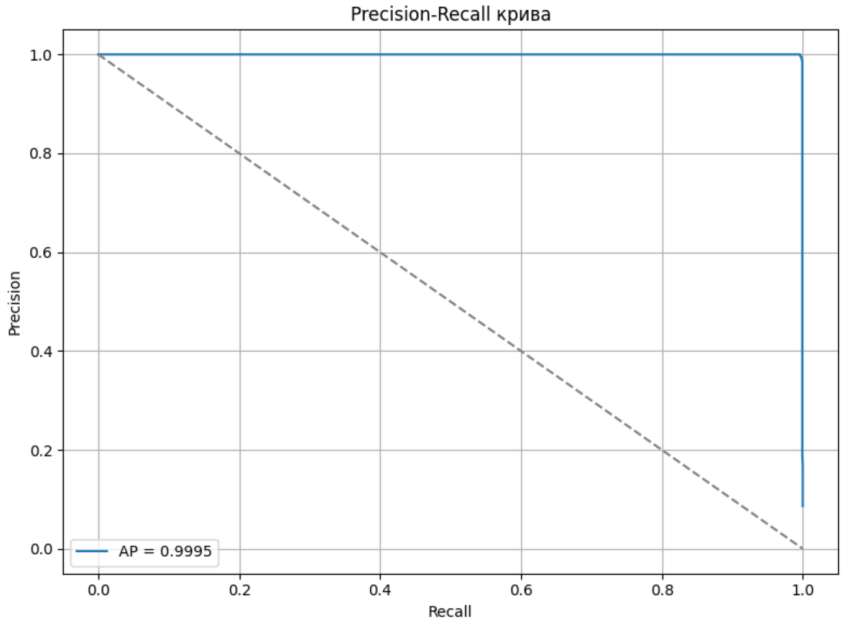

Рисунок 2.34 Precision-Recall крива після балансування класів в dataset.csv

Як видно з графіка, значення AP = 0.9995, що свідчить про чудову здатність моделі передбачати значення не домінуйочого класу.

2.7.3 Оцінювання моделі Logistic Regression

Спершу розглянемо результати модель Logistic Regression (LR) на даних автомобільних аварій accident.csv.

Після навчання моделі LR, її точність, склала: 0.7733, або 77,33%. Це означає, що модель правильно спрогнозувала 77,33% випадків автомобільних аварій.

```
Точність (Accuracy): 0.7733
Звіт про класифікацію:
              precision    recall  f1-score   support

           0       0.60      0.57      0.59       169
           1       0.84      0.85      0.84       431

    accuracy                           0.77       600
   macro avg       0.72      0.71      0.72       600
weighted avg       0.77      0.77      0.77       600
```

Рисунок 2.35 Точність моделі LR на наборі даних accident.csv

Для кращого розуміння продуктивності моделі, буде проаналізовано матрицю невідповідності.

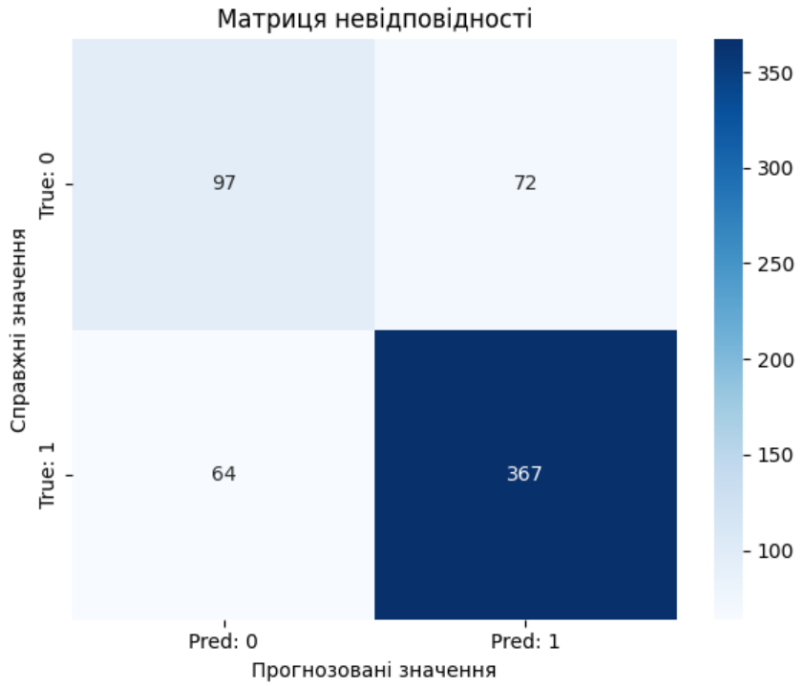

Рисунок 2.36 Матриця невідповідності LR на наборі даних accident.csv

Розглянемо детальніше матрицю невідповідності:

- True Positive (TP): 367
- True Negative (TN): 97
- False Positive (FP): 72
- False Negative (FN): 64

З даних значень, випливає, що модель добре передбачує позитивний клас, в той час як негативний клас правильно передбачає близько 57% випадків.

Для кращої оцінки здатності моделі розрізняти класи, побудуємо ROC-криву, та визначимо значення AUC (площі під ROC-кривою).

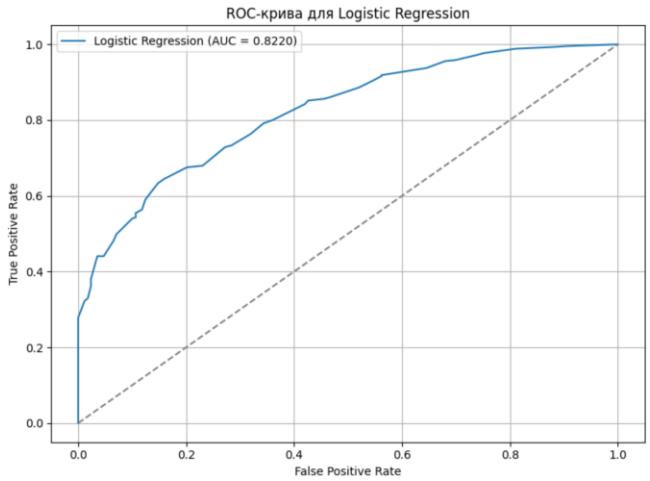

Рисунок 2.37 ROC-крива для LR на наборі даних accident.csv

Як видно з графіка, значення AUC = 0.8220, що свідчить про добру здатність моделі відрізняти клас, оскільки значення 1 є ідеальним, а значення 0.5 вгадуванням.

Також слід побудувати криву Precision-Recall та знайти значення AP, оскільки дана метрика краще відображає якість моделі при дисбалансі класів.

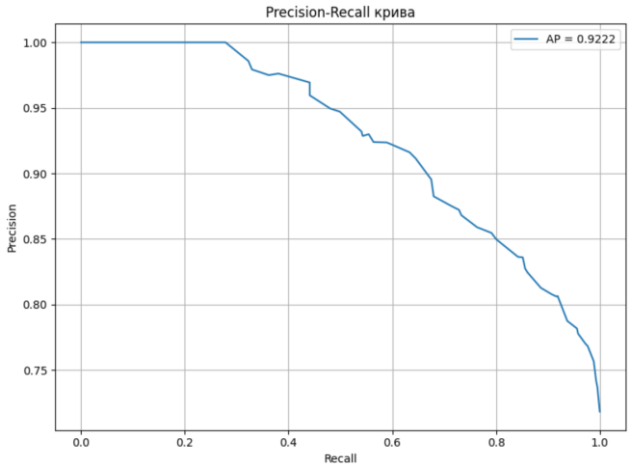

Рисунок 2.38 Precision-Recall -крива для LR на наборі даних accident.csv

Як видно з графіка, значення AP = 0.9222, що свідчить про високу здатність моделі відрізняти клас, оскільки значення 1 є ідеальним, а значення 0.5 вгадуванням.

Тепер розглянемо результати модель LR на даних пацієнтів реанімації dataset.csv.

Після навчання моделі RF, розраховується її точність, і вона склала: 0.9256, або 92,56%.

```
Точність (Accuracy): 0.9256
Звіт про класифікацію:
              precision    recall  f1-score   support

           0       0.93      0.99      0.96     16760
           1       0.67      0.27      0.38      1583

    accuracy                           0.93     18343
   macro avg       0.80      0.63      0.67     18343
weighted avg       0.91      0.93      0.91     18343
```

Рисунок 2.39 Точність моделі LR на наборі даних dataset.csv

Для кращого розуміння продуктивності моделі, буде проаналізовано матрицю невідповідності.

Рисунок 2.40 Матриця невідповідності LR на наборі даних dataset.csv

Розглянемо детальніше матрицю невідповідності:

- True Positive (TP): 426.
- True Negative (TN): 16553.
- False Positive (FP): 207.
- False Negative (FN): 1157.

З даних значень можна зробити висновок, що дані мають великий дисбаланс класів. У таких умовах загальна точність моделі може бути оманливою, адже модель правильно класифікує лише приклади домінуючого класу.

Для кращої оцінки побудуємо ROC-криву та обчислимо AUC.

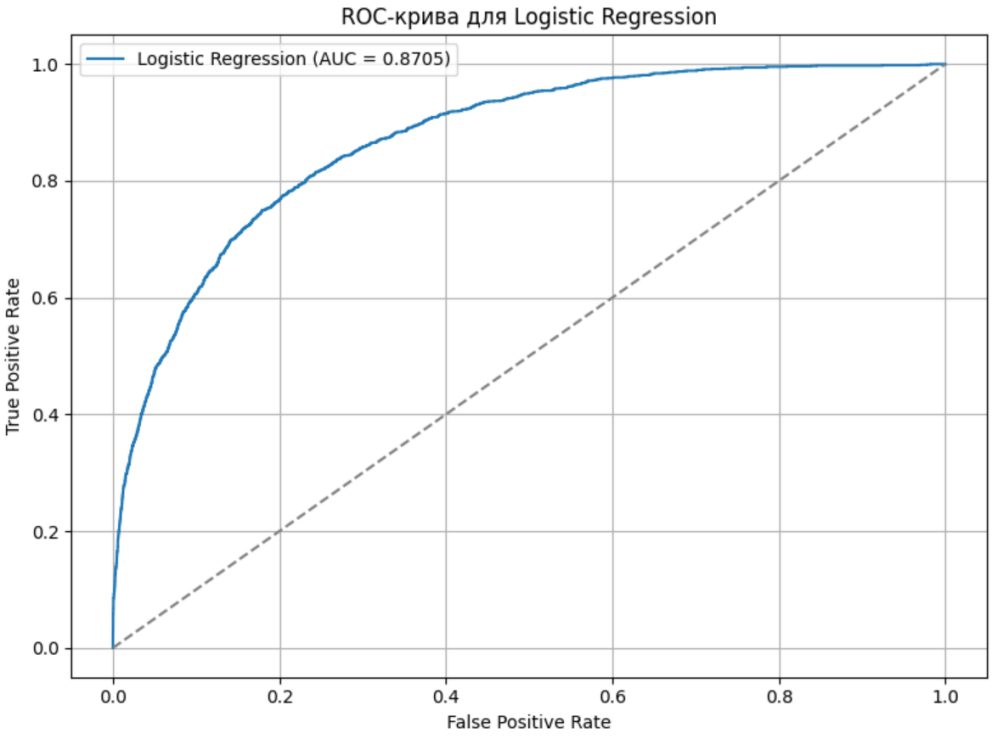

Рисунок 2.41 ROC-крива для LR на наборі даних dataset.csv

Як видно з графіка, значення AUC = 0.8705, що свідчить про добру здатність моделі відрізняти клас.

Також слід побудувати криву Precision-Recall та знайти значення AP, оскільки дана метрика краще відображає якість моделі при дисбалансі класів.

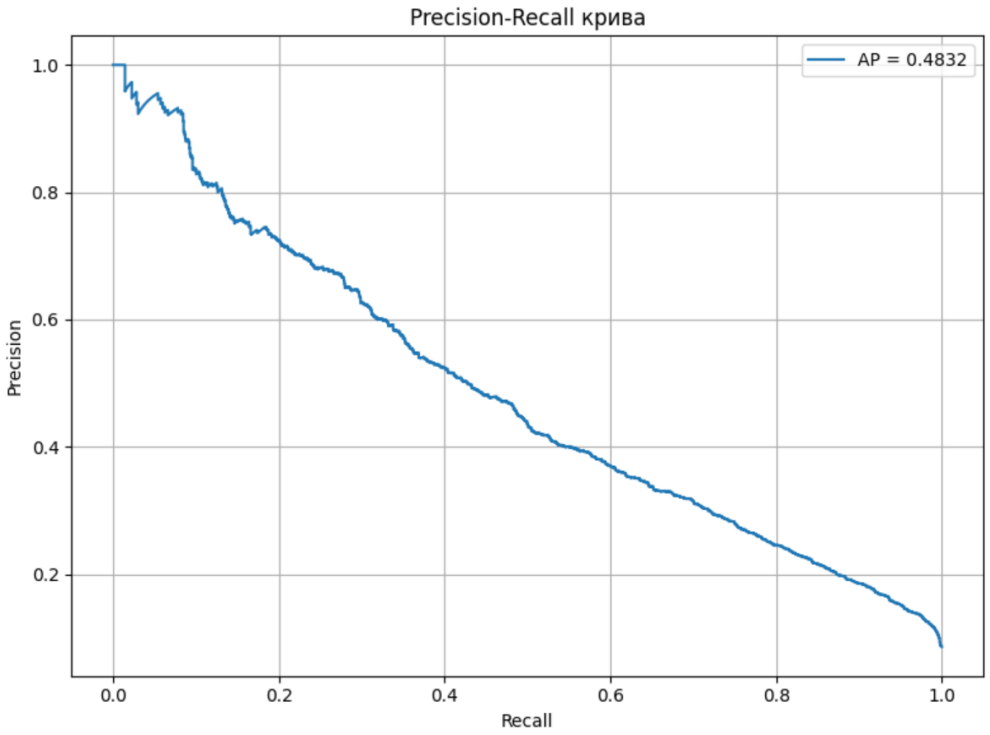

Рисунок 2.42 Precision-Recall -крива для LR на наборі даних dataset.csv

Як видно з графіка, значення AP = 0.4832, що свідчить про труднощі моделі передбачати значення не домінуйочого класу.

В такому випадку слід провести балансування класів, після чого повторно оцінити модель на даних.

Після балансування класів, та повторного навчання моделі RF, її точність склала: 0.8020, або 80,20%.

```
Точність (Accuracy): 0.8020
Звіт про класифікацію:
              precision    recall  f1-score   support

           0       0.97      0.81      0.88     16760
           1       0.27      0.76      0.40      1583

    accuracy                           0.80     18343
   macro avg       0.62      0.78      0.64     18343
weighted avg       0.91      0.80      0.84     18343
```

Рисунок 2.43 Точність моделі LR після балансування класів в dataset.csv

Для кращого розуміння продуктивності моделі, проаналізуємо матрицю невідповідності.

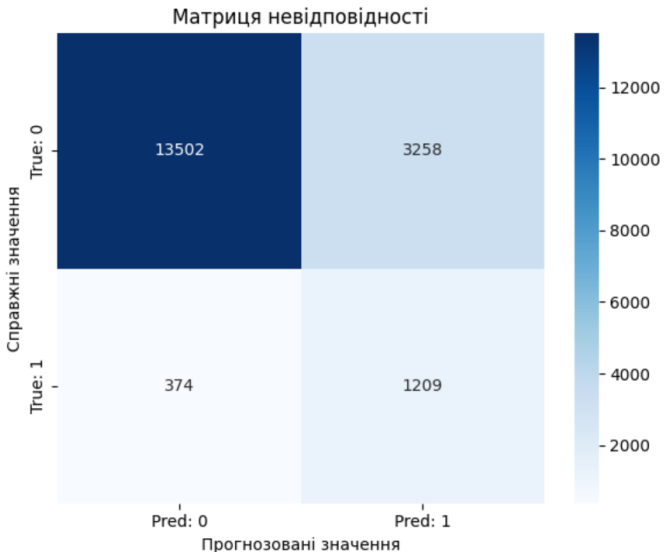

Рисунок 2.44 Матриця невідповідності після балансування в dataset.csv

Розглянемо детальніше матрицю невідповідності:

- True Positive (TP): 1209.
- True Negative (TN): 13502.
- False Positive (FP): 3258.
- False Negative (FN): 374.

Після балансування класів, значення точності зменшилося. Як видно з матриці невідповідності, дана модель стала краще розпізнавати випадки позитивного класу, за рахунок гіршого розпізнавання негативного класу.

Для кращої оцінки побудуємо ROC-криву та обчисляємо AUC.

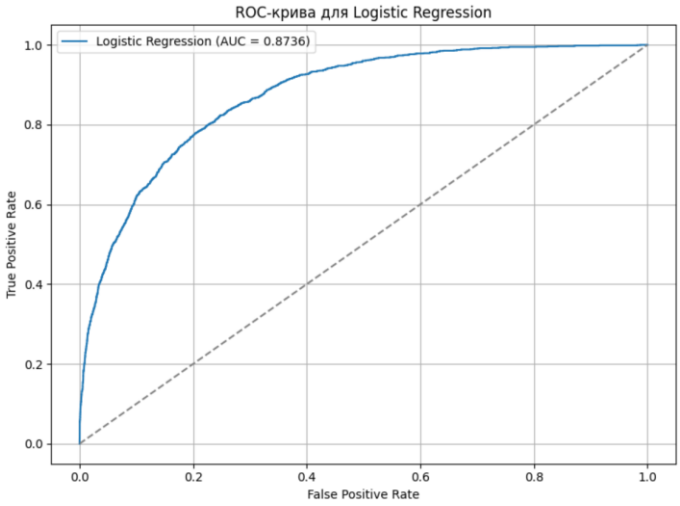

Рисунок 2.45 ROC-крива після балансування класів в dataset.csv

Як видно з графіка, значення AUC = 0.8736, що свідчить про добру здатність моделі відрізняти клас.

Також слід побудувати криву Precision-Recall та знайти значення AP, оскільки дана метрика краще відображає якість моделі при дисбалансі класів.

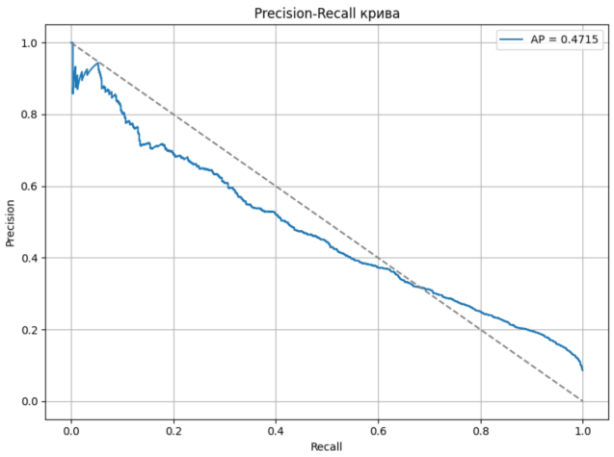

Рисунок 2.46 Precision-Recall крива після балансування класів в dataset.csv

Як видно з графіка, значення AP = 0.4715, оскільки зросла кількість хибно позитивних спрацювань (FP).

2.7.4 Оцінювання моделі XGboost

Спершу розглянемо результати модель XGboost на даних автомобільних аварій accident.csv.

Після навчання моделі XGboost, її точність, склала: 0.7833, або 78,33%. Це означає, що модель правильно спрогнозувала 78,33% випадків автомобільних аварій.

```
Точність (Accuracy): 0.7833
Звіт про класифікацію:
              precision    recall  f1-score   support

           0       0.64      0.52      0.58       169
           1       0.83      0.89      0.85       431

    accuracy                           0.78       600
   macro avg       0.73      0.70      0.71       600
weighted avg       0.77      0.78      0.78       600
```

Рисунок 2.47 Точність моделі RF на наборі даних accident.csv

Для кращого розуміння продуктивності моделі, буде проаналізовано матрицю невідповідності.

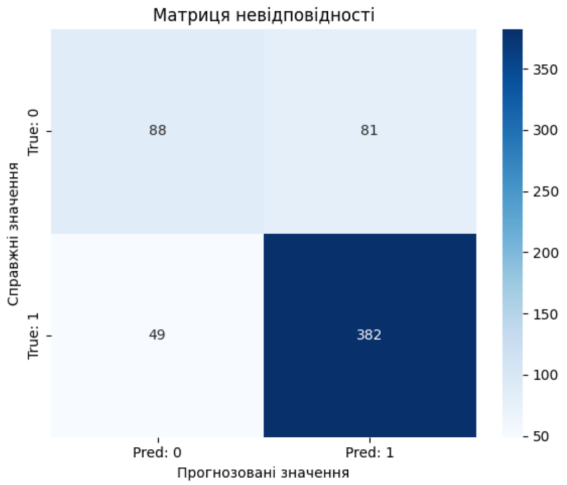

Рисунок 2.48 Матриця невідповідності XGboost на даних accident.csv

Розглянемо детальніше матрицю невідповідності:

- True Positive (TP): 382
- True Negative (TN): 88
- False Positive (FP): 81
- False Negative (FN): 49

З даних значень випливає, що модель добре передбачує позитивний клас, але негативний клас правильно передбачує близько 50% випадків.

Для кращої оцінки здатності моделі розрізняти класи, побудуємо ROC-криву, та визначимо значення AUC (площі під ROC-кривою).

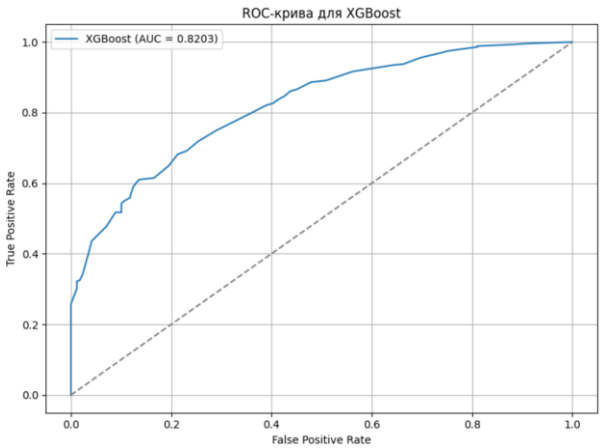

Рисунок 2.49 ROC-крива для XGboost на наборі даних accident.csv

Як видно з графіка, значення AUC = 0.8203, що свідчить про добру здатність моделі відрізняти клас, оскільки значення 1 є ідеальним, а значення 0.5 вгадуванням.

Також слід побудувати криву Precision-Recall та знайти значення AP, оскільки дана метрика краще відображає якість моделі при дисбалансі класів.

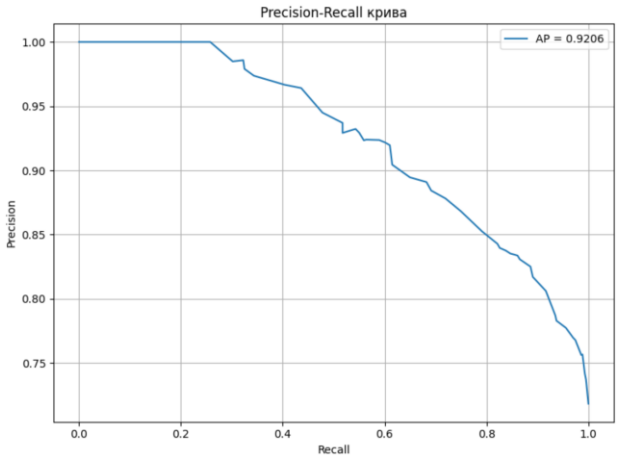

Рисунок 2.50 Precision-Recall -крива для XGboost на наборі даних accident.csv

Як видно з графіка, значення AP = 0.9206, що свідчить про високу здатність моделі відрізняти клас, оскільки значення 1 є ідеальним, а значення 0.5 вгадуванням.

Розглянемо результати модель XGboost на даних пацієнтів dataset.csv.

Після навчання моделі XGboost, розраховується її точність, і вона склала: 0.9325, або 93,25%.

```
Точність (Accuracy): 0.9325
Звіт про класифікацію:
              precision    recall  f1-score   support

           0       0.94      0.99      0.96     16760
           1       0.72      0.35      0.47      1583

    accuracy                           0.93     18343
   macro avg       0.83      0.67      0.72     18343
weighted avg       0.92      0.93      0.92     18343
```

Рисунок 2.51 Точність моделі XGboost на наборі даних dataset.csv

Для кращого розуміння продуктивності моделі, буде проаналізовано матрицю невідповідності.

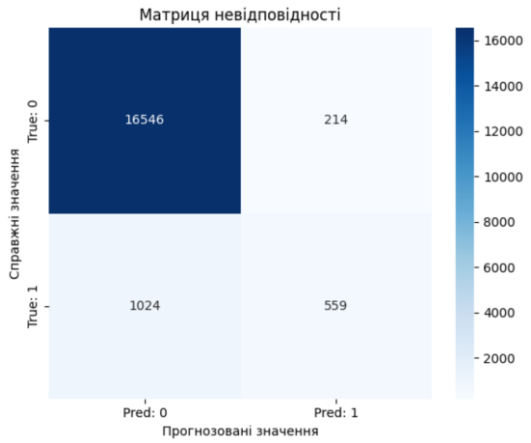

Рисунок 2.52 Матриця невідповідності XGboost на наборі даних dataset.csv

Розглянемо детальніше матрицю невідповідності:

- True Positive (TP): 559.
- True Negative (TN): 16546.
- False Positive (FP): 214.
- False Negative (FN): 1024.

З даних значень можна зробити висновок, що дані мають великий дисбаланс класів. У таких умовах загальна точність моделі може бути оманливою, адже модель правильно класифікує лише приклади домінуючого класу.

Для кращої оцінки побудуємо ROC-криву та обчислимо AUC.

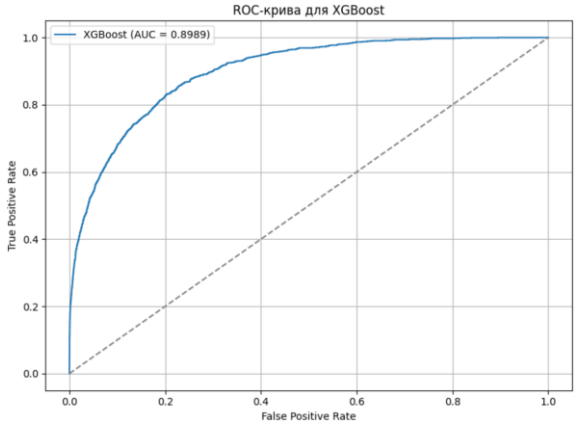

Рисунок 2.53 ROC-крива для XGboost на наборі даних dataset.csv

Як видно з графіка, значення AUC = 0.8989, що свідчить про добру здатність моделі відрізняти клас.

Також слід побудувати криву Precision-Recall та знайти значення AP, оскільки дана метрика краще відображає якість моделі при дисбалансі класів.

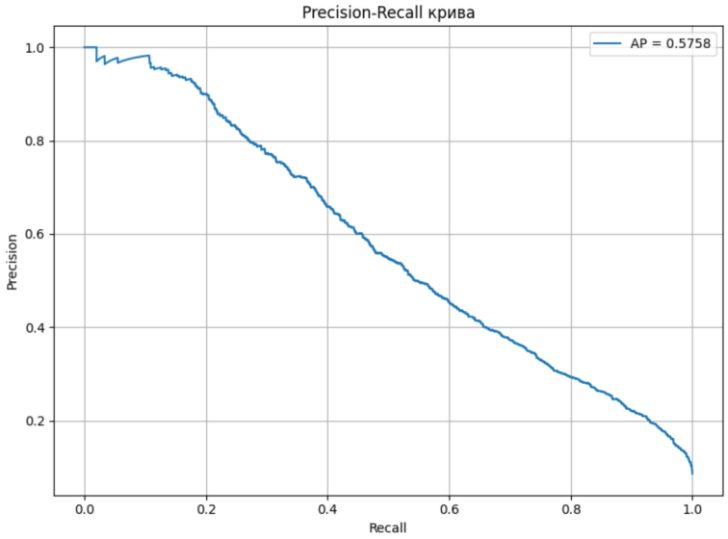

Рисунок 2.54 Precision-Recall -крива для XGboost на даних dataset.csv

Як видно з графіка, значення AP = 0.5758, що свідчить про труднощі моделі передбачати значення не домінуйочого класу.

В такому випадку слід провести балансування класів, після чого повторно оцінити модель на даних.

Після балансування класів, та повторного навчання моделі RF, її точність склала: 0.9882, або 98,82%.

```
Точність (Accuracy): 0.9882
Звіт про класифікацію:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99     16760
           1       0.88      1.00      0.94      1583

    accuracy                           0.99     18343
   macro avg       0.94      0.99      0.96     18343
weighted avg       0.99      0.99      0.99     18343
```

Рисунок 2.55 Точність моделі XGboost після балансування в dataset.csv

Для кращого розуміння продуктивності моделі, проаналізуємо матрицю невідповідності.

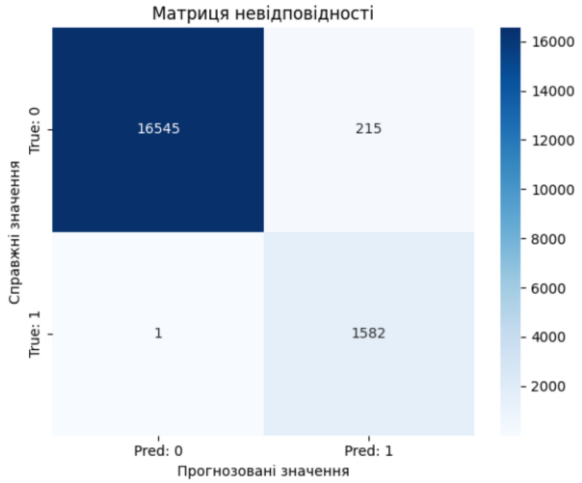

Рисунок 2.56 Матриця невідповідності після балансування в dataset.csv

Розглянемо детальніше матрицю невідповідності:

- True Positive (TP): 1582.
- True Negative (TN): 16545.
- False Positive (FP): 215.
- False Negative (FN): 1.

Після балансування класів, значення точності збільшилося, як видно з матриці невідповідності, модель стала краще розпізнавати випадки позитивного класу.

Для кращої оцінки побудуємо ROC-криву та обчисляємо AUC.

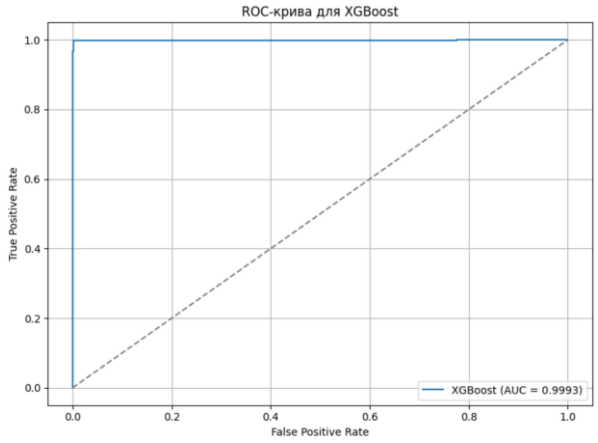

Рисунок 2.57 ROC-крива після балансування класів в dataset.csv

Як видно з графіка, значення AUC = 0.9993, що свідчить про чудову здатність моделі відрізняти клас.

Також слід побудувати криву Precision-Recall та знайти значення AP, оскільки дана метрика краще відображає якість моделі при дисбалансі класів.

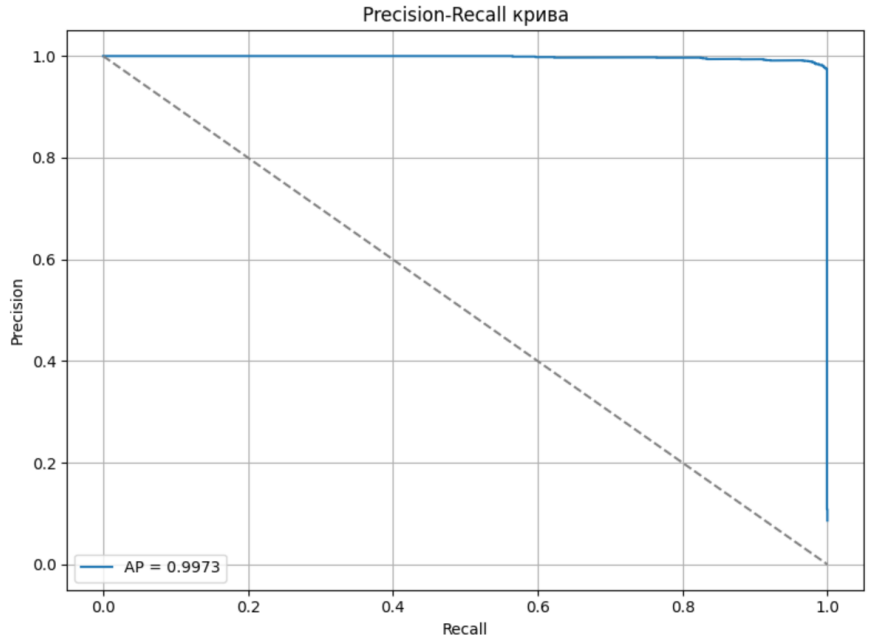

Рисунок 2.58 Precision-Recall крива після балансування класів в dataset.csv

Як видно з графіка, значення AP = 0.9973, що свідчить про чудову здатність моделі передбачати значення не домінуйочого класу.

2.7.5 Підсумок результатів

Аналіз ефективності розроблених моделей був проведений на датасеті accident.csv, а також на датасеті dataset.csv до і після балансування, після чого результати були записані в таблиці. Дані таблиці містять такі метрики, як: точність (accuracy), точність класифікації (precision), повнота (recall), F1-показник (f1-score), площа під ROC-кривою (AUC) та середня точність (AP).

Аналіз був проведений для моделей: KNN, Logistic Regression, Random forest та XGBoost.

Спершу розглянемо результати оцінки моделей на датасеті accident.csv. Деталі оцінки моделей, були записані в таблицю 2.1.

Таблиця 2.1 Результати моделей на датасеті accident.csv

Метрики (accident.csv)		Моделі			
		KNN	Logistic Regression	Random forest	XGboost
accuracy		0.7783	0.7733	0.7833	0.7833
precision	0	0.63	0.60	0.64	0.64
	1	0.82	0.84	0.83	0.83
recall	0	0.50	0.57	0.52	0.52
	1	0.89	0.85	0.89	0.89
f1-score	0	0.56	0.59	0.58	0.58
	1	0.85	0.84	0.85	0.85
AUC		0.8128	0.8220	0.8201	0.8203
AP		0.9123	0.9222	0.9204	0.9206

Загалом, всі моделі показали задовільні результати. Найкраще себе показали моделі Random forest та XGBoost, котрі показали найвищу точності в 0.7833. Також слід зазначити, що метрика AUC, яка визначає здатність моделі розрізняти класи, виявилася найкращою в моделі Logistic Regression з показником 0.8220. Також значення середньої точності (AP), у Logistic Regression також виявилася найкращою 0.9222.

Тепер розглянемо результати оцінки моделей на датасеті dataset.csv до балансування класів. Деталі оцінки моделей, були записані в таблицю 2.2.

Таблиця 2.2 Результати моделей на датасеті dataset.csv до балансування

Метрики (dataset.csv до балансування)		Моделі			
		KNN	Logistic Regression	Random forest	XGboost
accuracy		0.9227	0.9256	0.9296	0.9325
precision	0	0.93	0.93	0.94	0.94
	1	0.75	0.67	0.75	0.72
recall	0	1.00	0.99	0.99	0.99
	1	0.16	0.27	0.28	0.35
f1-score	0	0.96	0.96	0.96	0.96
	1	0.26	0.38	0.40	0.47
AUC		0.7655	0.8705	0.8838	0.8989
AP		0.3514	0.4832	0.5386	0.5759

Загалом, всі моделі показали чудові результати, якщо дивитися на показник точності accuracy. Найкраще себе показала модель XGBoost, з точністю в 0.9325. Метрики AUC та AP також виявилися найкращими в моделі XGBoost з показниками 0.8989 та 0.5759 відповідно.

Слід зазначити, що не зважаючи на велике значення точності в XGBoost, значення AP значно менше норми. Це означає, що модель передбачила в основному представників домінуючого класу, та погано передбачила представників не домінуючого класу.

Тепер розглянемо результати оцінки моделей на датасеті dataset.csv після балансування. Деталі оцінки моделей, були записані в таблицю 2.3.

Таблиця 2.3 Результати моделей на датасеті dataset.csv після балансування

Метрики (dataset.csv після балансування)		Моделі			
		KNN	Logistic Regression	Random forest	XGboost
accuracy		0.9805	0.8020	0.9916	0.9882
precision	0	1.00	0.97	1.00	1.00
	1	0.82	0.27	0.91	0.88
recall	0	0.98	0.81	0.99	0.99
	1	1.00	0.77	1.00	1.00
f1-score	0	0.99	0.88	1.00	0.99
	1	0.90	0.40	0.95	0.94
AUC		0.9891	0.8737	0.9997	0.9993
AP		0.8158	0.4713	0.9995	0.9973

Більшість моделей показали чудові результати. Найкраще себе показала модель Random forest, з точностю в 0.9916. Метрики AUC та AP також виявилися найкращими в моделі Random forest з показниками 0.9997 та 0.9995 відповідно.

Слід зазначити, що хоча значення точності в Logistic Regression, значно зменшилося, на ділі модель почала краще прогнозувати випадки не домінуючого класу, внаслідок зменшення передбачень домінуючого класу.

ВИСНОВКИ

Виконання дипломної роботи стало ключовим етапом у професійному шляху майбутнього фахівця з інформатики, оскільки дозволило поєднати теоретичні знання з практичними вміннями. В рамках дослідження були зосередженні зусилля на розробці та аналізі моделей машинного навчання в задачах на виживання.

В процесі створення дипломної роботи було покращено розуміння сучасних підходів до створення моделей прогнозування, зокрема методів KNN, Random Forest, XGBoost та Logistic Regression. Було виконано повний цикл моделювання: від збору та попередньої обробки даних до навчання моделей та оцінки їхньої ефективності.

Окрему увагу було наданно практичному опануванню популярних бібліотек Python, таких як scikit-learn, XGBoost та matplotlib. Це дало змогу не тільки забезпечити технічну реалізацію завдання, але й візуалізувати результати.

Завдяки виконанню дипломної роботи було здобуто не лише цінні технічні навички, а й сформовано цілісне бачення процесу розробки систем інтелектуального аналізу даних. Отриманий досвід є важливою базою для подальшого професійного росту та ефективної роботи в галузі інформаційних технологій, зокрема в напрямку машинного навчання.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ ТА ДЖЕРЕЛ

1. IT-Enterprise. Машинне навчання [Електронний ресурс] – Режим доступу: https://www.it.ua/knowledge-base/technology-innovation/machine-learning. – Дата звернення: 10.05.2025.

2. GoIT. Машинне навчання: що це таке, як працює і для чого використовується [Електронний ресурс] – Режим доступу: https://goit.global/ua/articles/mashynne-navchannia-shcho-tse-take-iak-pratsiuie-i-dlia-choho-vykorystovuietsia/ – Дата звернення: 10.06.2025.

3. DeNovo. Що таке машинне навчання? [Електронний ресурс] – Режим доступу: https://denovo.ua/resources/what-is-machine-learning. – Дата звернення: 10.05.2025.

4. IT-Enterprise. Машинне навчання (Machine Learning, ML) [Електронний ресурс] – Режим доступу: https://www.it.ua/knowledge-base/technology-innovation/machine-learning. – Дата звернення: 10.05.2025.

5. Unite.AI. Короткий посібник із розуміння алгоритму KNN [Електронний ресурс] – Режим доступу: https://www.unite.ai/uk/a-quick-guide-to-knn-algorithm/ – Дата звернення: 17.05.2025.

6. IT Wiki. KNN Models [Електронний ресурс] – Режим доступу: https://itwiki.dev/data-science/ml-reference/ml-glossary/knn-models. – Дата звернення: 17.05.2025.

7. W3Schools. Python Machine Learning – K Nearest Neighbors (KNN) [Електронний ресурс] – Режим доступу: https://www.w3schools.com/python/python_ml_knn.asp – Дата звернення: 17.06.2025.

8. IT Wiki. Random Forests [Електронний ресурс] – Режим доступу: https://itwiki.dev/data-science/ml-reference/ml-glossary/random-forests. – Дата звернення: 18.05.2025.

9. Kaggle. Decision Trees and Random Forest [Електронний ресурс] – Режим доступу: https://www.kaggle.com/code/emstrakhov/decision-trees-and-random-forest-ua – Дата звернення: 18.06.2025.

10. IT Wiki. XGBoost [Електронний ресурс] – Режим доступу: https://itwiki.dev/data-science/ml-reference/ml-glossary/xgboost. – Дата звернення: 24.05.2025.

11. Sonawane P. XGBoost: How does this work? [Електронний ресурс] – Режим доступу: https://medium.com/@prathameshsonawane/xgboost-how-does-this-work-e1cae7c5b6cb – Назва з екрана. – Дата звернення: 24.06.2025.

12. Medium. XGBoost Regression – In-Depth [Електронний ресурс] – Режим доступу: https://medium.com/@fraidoonomarzai99/xgboost-regression-in-depth-cb2b3f623281 – Дата звернення: 24.06.2025.

13. IT Wiki. Логістична регресія (Logistic Regression) [Електронний ресурс] – Режим доступу: https://itwiki.dev/data-science/ml-reference/ml-glossary/logistic-regression. – Дата звернення: 25.05.2025.

14. Spiceworks. What is Logistic Regression? [Електронний ресурс] – Режим доступу: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/ – Назва з екрана. – Дата звернення: 25.06.2025.