

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Створення веб-додатку прогнозування
потенційних донорів крові на основі машинного
навчання»**

Виконав студент групи ІПЗм-21-1
СМУТНИХ Іван Олександрович

Керівник роботи:
ЄФРЕМОВ Юрій Миколайович

Рецензент:
САВІЦЬКИЙ Роман Святославович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«05» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **СМУТНИХ Іван Олександрович**

Керівник роботи: **ЄФРЕМОВ Юрій Миколайович**

Тема роботи: **«Створення веб-додатку прогнозування потенційних донорів крові на основі машинного навчання»**,

затверджена Наказом закладу вищої освіти від **«05» лютого 2024 р. № 48**

Вихідні дані для роботи: Предметом дослідження є прогнозне моделювання з використанням алгоритмів машинного навчання. Об'єктом дослідження є прогнозування ймовірності повторної здачі крові.

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Єфремов Ю.М.	05.02.2024	05.02.2024
2	Єфремов Ю.М.	05.02.2024	05.02.2024
3	Єфремов Ю.М.	05.02.2024	05.02.2024

РЕФЕРАТ

Кваліфікаційна робота присвячена науковому дослідженню використання методів машинного навчання в медичній сфері. Пояснювальна записка складається з вступу, трьох розділів, висновків та списку використаних джерел.

В першому розділі було проведено аналіз задачі, детально досліджено методи та алгоритми для розв'язання, а також розглянуто існуючі програмні рішення.

В другому розділі було проаналізовано набір інструментів та даних. Описано практичну реалізацію алгоритмів машинного навчання та проведено аналіз отриманих результатів.

В третьому розділі розглянуто інтеграцію алгоритмів машинного навчання у веб-сервіс, наведена структура інтерфейсу та порядок роботи із додатком.

У висновках роботи підсумовуються отримані результати, робиться аналіз ключових висновків кожного розділу та визначається загальна важливість дослідження.

Кваліфікаційна робота складається з програмного додатку та пояснювальної записки. Пояснювальна записка до кваліфікаційної магістерської роботи містить 53 сторінки, 17 рисунків, 2 таблиці та 3 додатки. Список використаних джерел складається із 15 найменувань.

Ключові слова: МАШИННЕ НАВЧАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ, НАБІР ДАНИХ, ПРОГНОЗУВАННЯ, ЗАДАЧА КРОВІ, ДЕРЕВО РІШЕНЬ, ВИПАДКОВИЙ ЛІС.

ABSTRACT

The master's thesis is devoted to the scientific study of the use of machine learning methods in the medical field. The explanatory note consists of an introduction, three chapters, conclusions and a list of references.

In the first chapter, an analysis of the problem was carried out, methods and algorithms for solving were studied in detail, and existing software solutions were considered.

In the second section, a set of tools and data was analyzed. The practical implementation of machine learning algorithms is described and the results obtained are analyzed.

In the third section, the integration of machine learning algorithms into the web service is considered, the structure of the interface and the procedure for working with the application are given.

The conclusions of the work summarize the obtained results, analyze the key conclusions of each section, and determine the overall importance of the study.

The master's thesis consists of a software application and an explanatory note. The explanatory note to the qualifying master's thesis contains 53 pages, 17 figures, 2 tables and 3 appendices. The list of references consists of 15 items.

Keywords: MACHINE LEARNING, ARTIFICIAL INTELLIGENCE, DATA SET, PREDICTION, BLOOD DONATION, DECISION TREE, RANDOM FOREST.

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ	9
1.1 АНАЛІЗ ЗАДАЧІ, ЗАСОБІВ ТА МЕТОДІВ ЇЇ ВИРІШЕННЯ.....	9
1.2 АНАЛІЗ НАЯВНИХ ПРОГРАМНИХ ПРОДУКТІВ ПРОГНОЗУВАННЯ ЗАДАЧІ КРОВІ 10	
1.3 ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ	13
1.4 МЕТОДИ ТА АЛГОРИТМИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ ДЛЯ МАШИННОГО НАВЧАННЯ	15
Висновки до першого розділу	19
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГНОЗУВАННЯ ДОНАЦІЙ	21
2.1 ОПИС ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ МАШИННОГО НАВЧАННЯ.....	21
2.2 ОБРОБКА ТА АНАЛІЗ НАБОРУ ДАНИХ	23
2.3 ПРАКТИЧНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ	26
2.4 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ НА ОСНОВІ ОБРАНИХ МЕТРИК	31
Висновки до другого розділу	36
РОЗДІЛ 3. ІНТЕГРАЦІЯ МАШИННОГО НАВЧАННЯ ТА ОПИС ПРОГРАМНОГО ПРОДУКТУ	37
3.1 ІНТЕГРАЦІЯ МАШИННОГО НАВЧАННЯ У ВЕБ-СЕРВІС.....	37
3.2 СТРУКТУРА ІНТЕРФЕЙСУ ТА ПОРЯДОК РОБОТИ З ПРОГРАМНИМ ПРОДУКТОМ	39
Висновки до третього розділу	46
ВИСНОВКИ.....	47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	48
ДОДАТКИ.....	50

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ІІІ – штучний інтелект

API – Application Programming Interface

ВСТУП

Актуальність теми обумовлена необхідністю оптимізації рішень. Донорство крові є невід’ємною та важливою частиною системи охорони здоров’я. Без донорської крові багато звичних медичних процедур були б нездійсненні. В світі загалом, та в Україні конкретно, враховуючи війну, існує дефіцит донорської крові.

Оскільки донорство крові здійснюється добровільно, важко передбачити скільки крові буде отримано в той чи інший період часу.

Однак сучасні технології машинного навчання дозволяють робити передбачення на основі історичних даних. Таких даних в будь-якому банку крові більш ніж достатньо, адже всі необхідні дані можна отримати із історії здач крові донорів.

Це дослідження дозволить класифікувати потенційних донорів крові та передбачити кількість донорів крові, які мають можливість відвідати захід з донорства крові, що проводиться в певному місці в певний час. Це допоможе прогнозувати потенційний дефіцит крові, що в свою чергу збільшить час для вирішення цієї проблеми і загалом може покращити планування та організацію донорства крові.

В свою чергу створений веб-додаток стане зручним інструментом використання алгоритмів машинного навчання для необхідного прогнозування для некомерційних організацій.

Метою дослідження даної магістерської роботи є аналіз ефективності методів машинного навчання для прогнозування та створення веб-сервісу для їх зручного використання.

Мета роботи передбачає такі **завдання**:

- визначити основні вимоги та бажані кінцеві результати;
- дослідити методи машинного навчання для прогнозування;
- створити веб-сервіс з використанням машинного навчання, який прогнозує потенційні задачі крові.

Об'єктом дослідження є прогнозування потенційних здач крові та їх ймовірність.

Предметом дослідження є алгоритми машинного навчання спрямовані на прогнозування певних подій.

Наукова новизна полягає у дослідженні найбільш ефективних алгоритмів машинного навчання та необхідних характеристик набору даних для вирішення проблеми прогнозування потенційних здач крові.

Публікації. За темою кваліфікаційної магістерської роботи було опубліковано тезу «Доцільність використання штучного інтелекту в сучасній медицині» у збірнику за матеріалами VI Міжнародної студентської наукової конференції «Модернізація та сучасні українські і світові наукові дослідження».

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1 Аналіз задачі, засобів та методів її вирішення

Сучасний спосіб життя, включаючи високі показники нещасних випадків, природні катаклізми та антропогенні катастрофи, війни та різні збройні конфлікти призводять до зростання попиту на переливання крові. Відповідно лікарі повинні постійно мати певний запас донорської крові для задоволення поточних і майбутніх потреб. Донорство крові є добровільним, а отже одним із найважливіших завдань банків крові є утримання поточних та заохочення нових донорів у разі дефіциту. Звісно, це все вимагає зусиль, часу та ресурсів, а отже оптимізація цього процесу є актуальним завданням сьогодення.

Комп'ютерні та інформаційні технології значно вплинули на те, як ми зберігаємо інформацію. Обсяг даних що зберігаються у велетенських базах даних щодня збільшується. Проте, щоб отримати з цього якусь користь, ці дані повинні бути перетворені в інформацію шляхом їх опрацювання та аналізування.

Для цього використовується інтелектуальний аналіз даних. Він є важливим інструментом для сфери охорони здоров'я, адже кров користується величезним попитом. Це зв'язано із тим, що кров неможливо виготовити чи штучно синтезувати, добровільні здачі крові є єдиним джерелом надходження крові для переливання. Також варто зазначити, що тривале зберігання крові неможливе, тому необхідно створити найбільш ефективний та організований підхід для передбачення подальшої поведінки донорів крові.

Інструмент, котрий реалізовував би такий підхід дозволив би банкам крові та волонтерським організаціям систематично планувати ефективну організацію здачі крові та забезпечував би необхідним часом при потенційних дефіцитах донорської крові для їх подолання.

Даного результату можна досягнути за допомогою машинного навчання. Алгоритми машинного навчання дозволяють робити певні прогнозування на основі відповідних історичних даних. Метою прогнозування в машинному навчанні є прогнозування ймовірного набору даних, який пов'язаний з вихідними даними. Щоб отримати найбільш якісне прогнозування у машинному навчанні, організації повинні мати інфраструктуру для підтримки рішень і високоякісні дані для забезпечення алгоритму. І те, і інше доступно сучасній системі охорони здоров'я.

Інтелектуальний аналіз даних може стати ключовим інструментом для оптимізації процесу збору та управління донорською кров'ю. Алгоритми машинного навчання дозволяють не лише аналізувати великі обсяги інформації, але й робити прогнози щодо майбутніх потреб у крові на основі історичних даних та різноманітних факторів, що впливають на цей процес. Таким чином, використання інтелектуального аналізу даних у сфері донорства крові може значно покращити управління ресурсами, забезпечити стабільний доступ до крові для медичних потреб та збільшити ефективність діяльності банків крові та волонтерських організацій.

1.2 Аналіз наявних програмних продуктів прогнозування задачі крові

У 2024 році штучний інтелект трансформуватиме бізнес-прогнозування та прогнозу аналітику. За даними IDC, до 2025 року понад 50% великих виробників перейдуть на інтелектуальну аналітику на основі ШІ, щоб підвищити точність прогнозів на 30% або більше. Ці інструменти прогнозування ШІ не тільки підвищують точність прогнозів, але й спрощують складний процес аналізу даних.

Особливість випадку із прогнозуванням донорства крові полягає у тому, що більшість існуючих рішень не є вузькоспеціалізованими. Більшість із них надає неймовірну кількість простих і складних інструментів для збору, нормалізації та обробки даних, вибору і налаштування моделі машинного

навчання тощо. Такі системи є максимально універсальними, для того щоб можна було покрити якнайбільше завдань і, відповідно, мати якнайбільшу клієнтську базу. Можна зробити висновок, що вони призначенні в першу чергу для спеціалістів машинного навчання, а не для пересічних користувачів.

Вузькоспеціалізовані рішення достатньої популярності просто відсутні, тому розглянемо універсальні сервіси для прогнозування: їх переваги та недоліки.

Одним із найпопулярніших веб-сервісів є H2O AI. Це платформа штучного інтелекту, яка має відкритий код та надає широкий спектр інструментів для аналізу даних і прогнозування моделей. Вона підтримує різноманітні алгоритми та забезпечує широкі можливості для обробки даних, навчання моделей і оцінки їх ефективності. ШІ H2O відомий своєю масштабованістю і продуктивністю, здатністю обробляти великі обсяги даних і ефективно виконувати складні обчислення.

Переваги:

- Будучи платформою з відкритим вихідним кодом, H2O AI отримує переваги від великої спільноти розробників і користувачів, що призводить до постійних удосконалень і оновлень платформи.
- Велика кількість інструментів обробки даних та навчання моделей.
- ШІ H2O є дуже продуктивним, що дозволяє користувачам швидко та ефективно навчати моделі та виконувати складні обчислення.
- Можливість розгортати свої моделі локально, у хмарі або в гібридних середовищах.
- Чудова документація та підтримка.

Недоліки:

- Високий поріг входження, інтуїтивно не зрозумілий інтерфейс.
- Перевантажений інтерфейс – велика кількість надлишкових графіків, метрик, інтуїтивно не зрозумілий інтерфейс.

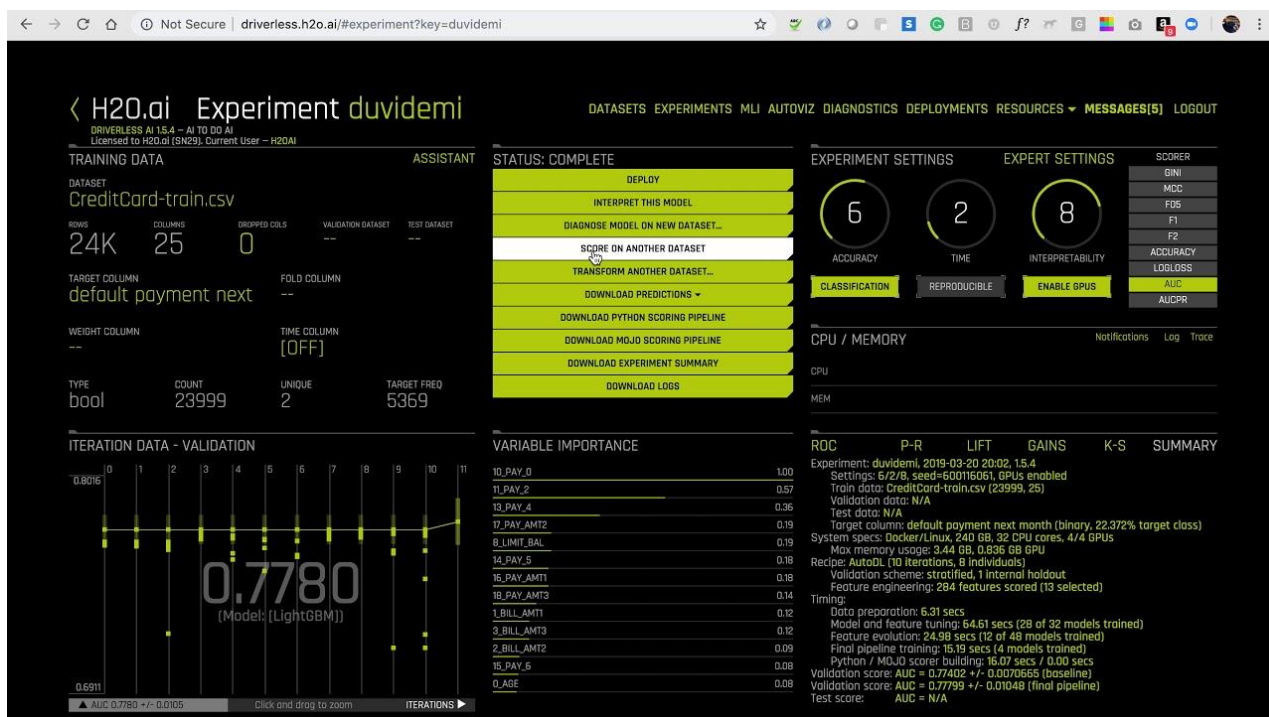


Рис. 1.1. Інтерфейс сервісу H2O AI

Розглянемо конкурента вищезгаданої платформи, а саме Google Cloud AI Platform. Це першу чергу платформа, котра надає комплексний набір інструментів і послуг Google Cloud для розробки, розгортання та керування моделями штучного інтелекту та машинного навчання. Це дозволяє розробникам створювати та масштабувати програми ML в інфраструктурі Google Cloud.

В своїх особливостях даний сервіс є протилежністю H2O AI, переконаймося в цьому розглянувши переваги та недоліки платформи.

Переваги:

- Використання новітніх провідних технологій та алгоритмів машинного навчання.
- Інтуїтивно зрозумілий інтерфейс.
- Висока швидкість навчання моделей та проведення складних обчислень, але лише при стабільному інтернет з'єднанні.
- Чудова документація та підтримка.

- Відносно низький поріг входження. Google Cloud AI Platform легко використовувати без необхідності багато налаштовувати та конфігурувати продукт.

Недоліки:

- Попри інтуїтивно зрозумілий інтерфейс, деякі рішення інтерфейсу досить спантеличливі.
- Серйозні проблеми із захистом даних та доступом до платформи.
- Погана підтримка мов програмування.
- Відносно невелика кількість інструментів обробки даних та навчання моделей.

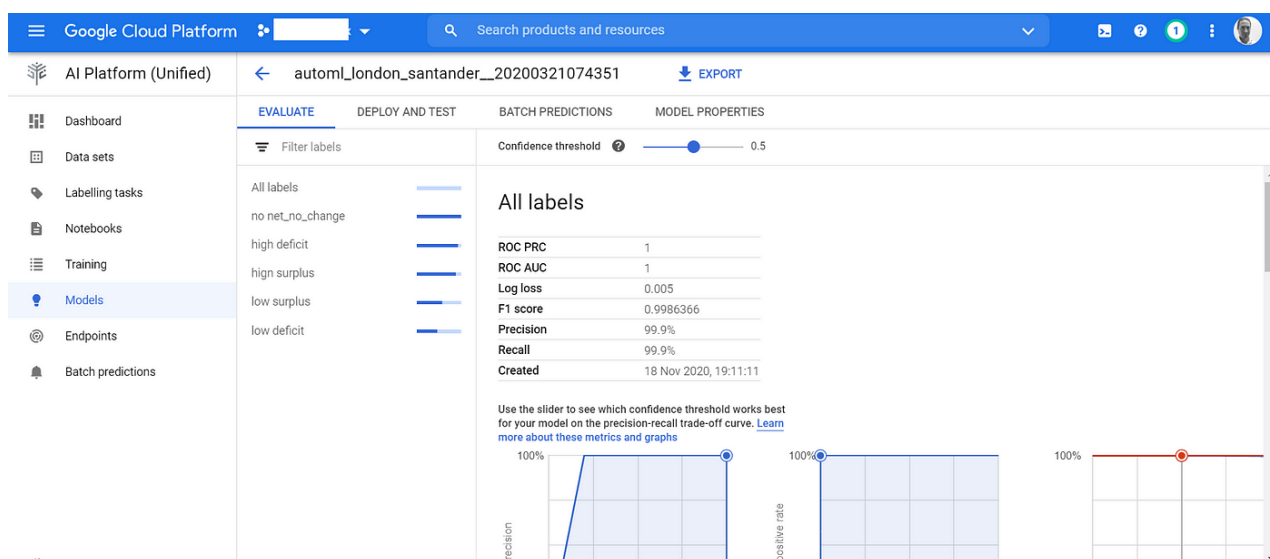


Рис. 1.2. Інтерфейс сервісу Google Cloud AI Platform

1.3 Постановка задачі дослідження

Проведений аналіз переваг та недоліків аналогічного програмного забезпечення дозволив визначити основні аспекти, котрі повинні бути реалізовані в майбутньому додатку. Було визначено концептуальну різницю призначення досліджених платформ та нашого сервісу, котра полягає у переході від універсальності до вузьконаправленості. Сформуємо етапи розробки для кращого розуміння кола задач даної роботи:

- 1) Визначити або синтезувати набори даних на основі яких буде здійснюватися навчання моделей.
- 2) Виокремити важливі та надлишкові характеристики набору даних.
- 3) Провести тренування та тестування навчених моделей.
- 4) Дослідити особливості алгоритмів машинного навчання та набору даних, котрі забезпечують найбільшу точність моделей.
- 5) Розробити серверне API сервісу.
- 6) Інтегрувати алгоритми машинного навчання в серверне API.
- 7) Розробити інтерфейс сервісу.

Варто також наголосити на особливостях додатку, котрі відрізнятимуть його від аналогів. Як було згадано вище, буде здійснено перехід від універсальності до вузьконаправленості. Це цілком реальна задача, адже коло завдань області донорства крові попередньо відоме. Це дозволить нам значно зменшити функціонал та навантаженість інтерфейсу. Ідея полягає в тому, щоб цією платформою могли користуватися звичайні працівники медичної сфери, котрі не мають зовсім ніяких знань і навичок роботи із машинним навчанням. Це надасть банкам крові можливість прогнозування, при цьому не витрачаючи на цю задачу багато додаткового часу, зусиль та ресурсів.

Також для підтримки рівня простоти платформи буде опущений функціонал попередньої обробки даних. Це завдання буде покладене на самих користувачів, яке може бути виконане при отриманні наборів даних. Платформа буде надавати всі необхідні рекомендації для збереження високої точності отриманих результатів.

Проте універсальність платформи буде в деякій мірі збережена. На основі проведених досліджень будуть надані рекомендації щодо набору даних та алгоритмів машинного навчання для застосування для отримання максимальної точності. Однак користувачі все ще матимуть змогу завантажувати довільні набори даних та обирати методи та параметри машинного навчання.

1.4 Методи та алгоритми інтелектуального аналізу даних для машинного навчання

Вибір правильного алгоритму машинного навчання має вирішальне значення для досягнення точних і ефективних результатів у проектах аналізу даних. З такою кількістю алгоритмів на вибір, визначити, який із них найкраще підходить для конкретних потреб, може бути не тривіальним завданням.

В нашому випадку перед нами стоїть задача класифікації, адже нам потрібно спрогнозувати чи буде потенційний донор здавати кров в той чи інший час. Попри те, що перед нами чітко поставлено завдання класифікації, потрібно не тільки розглянути класичні алгоритми класифікації, а й алгоритми що відносяться до інших категорій, наприклад, нейроні мережі та ансамблеві методи. Розглянемо спочатку класичних представників класифікації.

Логістична регресія – це статистичний метод для моделювання ймовірностей для бінарних (логістичних) змінних залежно від одного або декількох змінних. Вона використовує логістичну функцію для оцінки ймовірності того, що деяка подія відбудеться. Однак вона також може бути адаптована для розв'язання багатокласових задач класифікації, використовуючи, наприклад, метод «один проти всіх». Логістична регресія є відносно простим методом, який не вимагає великої кількості обчислювальних ресурсів для навчання. Це робить її привабливим вибором для великих наборів даних або ситуацій з обмеженими обчислювальними ресурсами.

Незважаючи на свої переваги, логістична регресія також має свої обмеження, такі як неспроможність моделювати складні залежності між ознаками або невідповідність для задач з великим числом класів.

Формула логістичної регресії виглядає наступним чином:

$$P(Y = 1 | X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n)}} , \text{ де}$$

- $P(Y = 1 | X)$ – ймовірність того, що подія Y відбудеться, якщо значення ознаки X відоме.

- e – число Ейлера.
- β_0 – константа (зсув).
- x_n – значення ознак.
- β_n – коефіцієнти для ознак.

Після того як модель була навчена (тобто знайдені оптимальні значення коефіцієнтів), можна використовувати отримані ймовірності для класифікації, встановлюючи поріг. Наприклад, якщо ймовірність $P(Y = 1 | X)$ більше порогового значення (зазвичай 0.5), спостереження класифікується як клас 1, в іншому випадку, як клас 0.

Метод опорних векторів – це метод аналізу даних для класифікації та регресійного аналізу за допомогою моделей з керованим навчанням з пов'язаними алгоритмами навчання, які називаються опорно-векторними машинами. Основна ідея полягає в тому, щоб знайти оптимальну гіперплощину, яка найкращим чином розділяє дані на класи. Опорні вектори – це точки даних, які лежать найближче до гіперплощини, і вони визначають положення цієї гіперплощини.

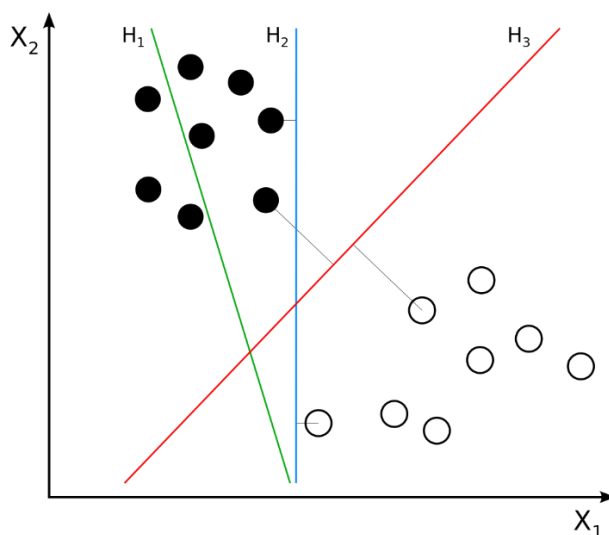


Рис. 1.3. Приклад гіперплощини, де H_1 не розділяє ці класи, H_2 розділяє, але лише з невеликим розділенням, а H_3 – з максимальним

Цей метод добре працює навіть у випадках, коли кількість ознак (або розмірність простору) є великою. Він може ефективно розділяти класи в

просторі високої розмірності, що робить його корисним для багатьох реальних задач.

За допомогою функцій ядра, таких як ядро RBF (Radial Basis Function), алгоритм може розділяти дані, які не є лінійно роздільними в оригінальному просторі ознак. Це дає йому здатність впоратися зі складними взаємозв'язками між ознаками.

Дерева рішень (Decision Tree) є одним з найпростіших і найпопулярніших методів машинного навчання для класифікації. Цей метод використовується для прийняття рішень шляхом створення структури дерева, де кожен вузол представляє певну ознаку (або їх набір), а кожен лист відповідає кінцевому класу.

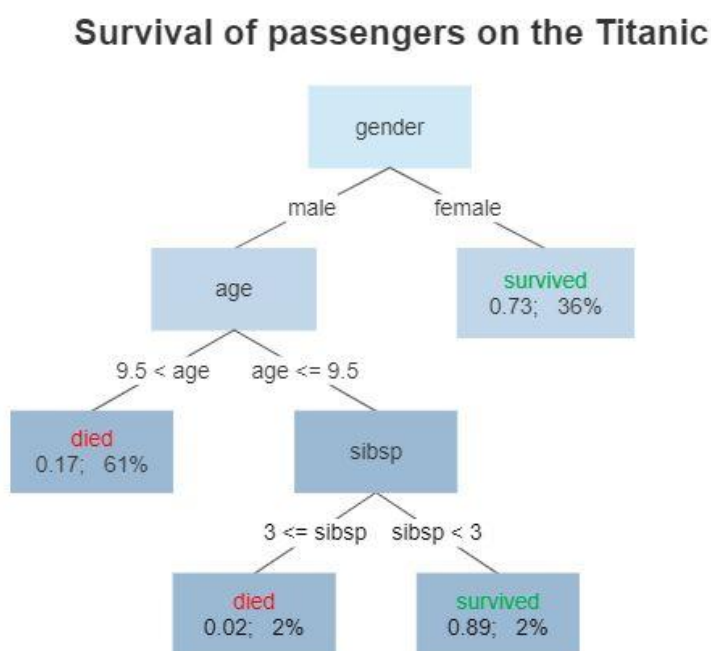


Рис. 1.4. Структура дерева рішень

Дерева рішень можуть працювати з різними типами ознак, включаючи категоріальні та числові, без потреби в попередній обробці даних. Під час навчання моделі дерево рішень автоматично вибирає найважливіші ознаки, що сприяє автоматизації процесу вибору характеристик. Дерева рішень не вимагають нормалізації або стандартизації даних, оскільки вони не чутливі до

масштабування ознак. Вони можуть ефективно працювати з великими наборами даних, оскільки їх можна швидко навчати і використовувати.

Недоліками є тенденція дерев до перенавчання, особливо у випадку глибоких дерев. Також варто враховувати, що невеликі зміни в даних можуть призвести до значних змін у структурі дерева, що може зробити його менш стійким до шуму.

Перейдемо до ансамблевих методів. Найбільш ефективним в задачах класифікації є Random Forest (Випадковий ліс). Це метод машинного навчання, який базується на ідеї злиття декількох дерев рішень для отримання більш точної та стабільної класифікації або регресії. Кожне дерево випадкового лісу навчається незалежно одне від одного, і кінцеве рішення визначається шляхом голосування (у випадку класифікації) або усереднення (у випадку регресії) результатів окремих дерев. Random Forest добре пристосовується до різних типів даних і різних задач машинного навчання.

Random Forest має вбудовану стійкість до перенавчання завдяки використанню багатьох дерев рішень. Оскільки кожне дерево навчається на випадковому підмножині даних і випадковій підмножині ознак, воно менше схильне до перенавчання на конкретних даних.

Незважаючи на свої переваги, Random Forest також має деякі обмеження, включаючи вищу обчислювальну складність порівняно з простішими методами, такими як звичайне дерево рішень. Також, як і у випадку із деревом рішень, якщо набір даних має велику кількість шуму, це може вплинути на точність Random Forest.

Наступним розглянемо алгоритм штучної нейронної мережі — багатошаровий перцептрон (MLP). Це вид штучної нейронної мережі, яка складається з одного або кількох прихованих шарів нейронів, що дозволяє моделі вивчати складні залежності між ознаками у наборі даних, а також входу та виходу. Кожен нейрон в одному шарі з'єднаний з кожним нейроном у наступному шарі, а також має свій ваговий коефіцієнт та функцію активації.

MLP використовує алгоритм зворотного поширення помилки для навчання моделі. Основна ідея цього методу полягає в поширенні сигналів помилки від виходів мережі до її входів, в напрямку, зворотному прямому поширенню сигналів у звичайному режимі роботи. Цей алгоритм дозволяє моделі коригувати вагові коефіцієнти таким чином, щоб мінімізувати помилку між прогнозованими та фактичними значеннями.

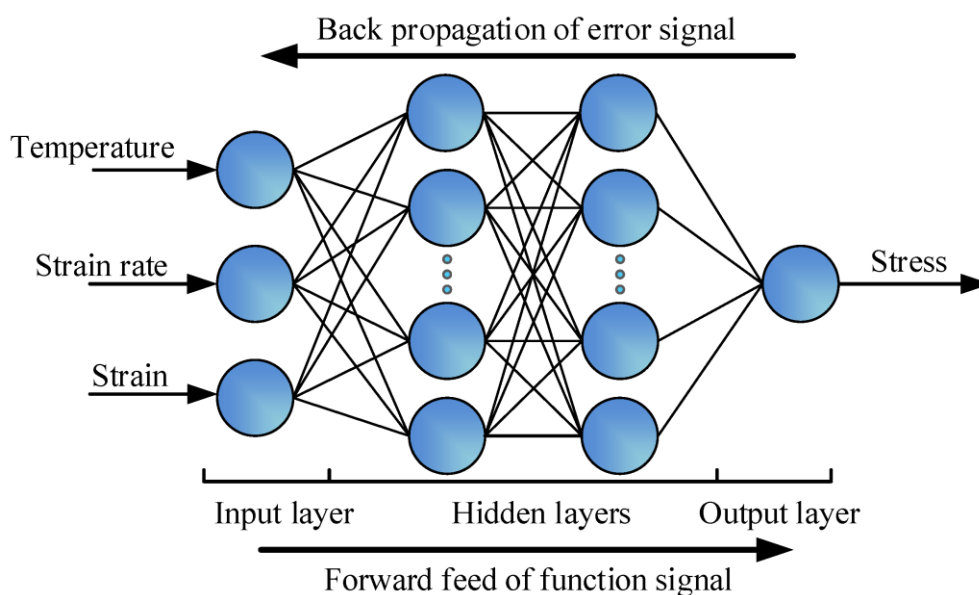


Рис. 1.5. Архітектура нейронної мережі MLP

MLP широко використовується для класифікації, регресії та інших завдань у багатьох областях, таких як комп'ютерний зір, обробка мов, фінансовий аналіз тощо. Він є потужним і гнучким інструментом машинного навчання, який може моделювати складні залежності в даних.

Висновки до першого розділу

В даному розділі було проведено аналіз задачі, її складності та можливих шляхів вирішення. Дослідження показало, що задача має свої особливості, зокрема у складності та обсязі даних, необхідних для аналізу. Аналіз наявних програмних продуктів для прогнозування задачі крові дав змогу визначити їхні переваги та недоліки в рамках визначеної задачі.

Постановка задачі дослідження дозволила чітко визначити цілі та завдання дослідження.

Детальний огляд методів та алгоритмів інтелектуального аналізу даних для машинного навчання свідчить про широкий спектр можливостей для вирішення поставленої задачі. Враховуючи їхню ефективність та застосовність, було обрано та описано підходи, які найбільше відповідають поставленим цілям дослідження.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГНОЗУВАННЯ ДОНАЦІЙ

2.1 Опис інструментальних засобів реалізації машинного навчання

Однією із провідним мов програмування для прогнозової аналітики є Python. Для вирішення поставлених завдань будуть використані наступні потужні бібліотеки:

- `scikit-learn` – одна із найпопулярніших бібліотек, надає інструментарій для створення та тренування різноманітних алгоритмів класифікації, регресії та кластеризації (random forest, лінійна регресія, градієнтний бустинг), працює у зв'язці з бібліотеками NumPy та SciPy.
- `pandas` – бібліотека, призначена для маніпулювання даними та їхнього аналізу. Вона, надає інструментарій для об'єднання, зміни форми, а також очищення даних та функції перегляду даних.
- `Numpy` — бібліотека, для здійснення складним математичних обчислень: починаючи від базових функцій над масивами і матрицями, закінчуючи лінійною алгеброю.
- `matplotlib` – бібліотека для візуалізації даних 2D та 3D графікою.

Важливу частину даної роботи складає дослідження наборів даних та їх характеристик. В деяких випадках потрібно керуватися не так результатами певних операцій, як покладатися на аналітичний аналіз. Тому важливо завжди мати уявлення про природу і особливості наборів даних. З цим нам допоможуть наступні інструменти:

- `head()` – функція, котра виводить перших 5 записів набору даних

```
transfusion.head()
```



	recency	frequency	quantity	time	donation
0	16	1	250	16	0
1	4	16	4200	83	0
2	16	1	250	16	0
3	4	4	1060	35	0
4	11	7	1850	28	0

Рис. 2.1. Перші 5 записів набору даних

- `info()` – функція, котра виводить інформацію про набір даних, включаючи тип даних стовпців, ненульові значення та використання пам'яті

```
transfusion.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  -
0   recency     10000 non-null  int64
1   frequency   10000 non-null  int64
2   quantity    10000 non-null  int64
3   time        10000 non-null  int64
4   donation    10000 non-null  int64
dtypes: int64(5)
memory usage: 390.8 KB
```

Рис. 2.2. Структура набору даних

- `describe()` – функція, що виводить деяку базову статистику для кожного зі стовпців у наборі даних. Він дає вам кількість значень, середнє значення, стандартне відхилення, мінімальне та максимальне значення та деякі проценти

 transfusion.describe()



	recency	frequency	quantity	time	donation
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	9.499000	5.386400	1405.975000	34.287600	0.232800
std	7.472856	4.892554	1164.160752	24.412884	0.422637
min	1.000000	1.000000	250.000000	2.000000	0.000000
25%	2.000000	2.000000	580.000000	16.000000	0.000000
50%	8.000000	4.000000	1115.000000	28.000000	0.000000
75%	14.000000	7.000000	1900.000000	50.000000	0.000000
max	38.000000	37.000000	9440.000000	98.000000	1.000000

Рис. 2.3. Статистика набору даних

- `corr()` – функція, що виводить попарну кореляцію стовпців, за винятком нульових значень.

 transfusion.corr()



	recency	frequency	quantity	time	donation
recency	1.000000	-0.113232	-0.103102	0.126481	-0.265285
frequency	-0.113232	1.000000	0.867931	0.555144	0.144056
quantity	-0.103102	0.867931	1.000000	0.572157	0.154057
time	0.126481	0.555144	0.572157	1.000000	0.036625
donation	-0.265285	0.144056	0.154057	0.036625	1.000000

Рис. 2.4. Кореляція стовпців набору даних

2.2 Обробка та аналіз набору даних

Набори даних є фундаментальним елементом у машинному навчанні. Вони є вихідними матеріалами, на основі яких модель буде вчитися вирішувати завдання. Роль набору даних в машинному навчанні полягає в

тому, щоб постачити моделі необхідні знання та інформацію для розробки прогнозів чи прийняття рішень.

Набір даних може містити різноманітні типи і формати інформації, включаючи числові дані, текст, зображення, відео, аудіо, графи та інше. Важливо, щоб дані були репрезентативними для проблеми і щоб вони були правильно обробленими.

Основні етапи роботи з набором даних включають збір, очищення, підготовку та аналіз даних. Збір даних може включати в себе різноманітні джерела, такі як бази даних, веб-сторінки, сенсори, суб'єктивні оцінки, тощо. Очищення даних передбачає видалення аномальних значень, заповнення пропущених даних та інші маніпуляції для підвищення якості даних. Підготовка даних включає в себе перетворення даних у формат, придатний для подальшого використання моделлю, такий як векторизація тексту, масштабування числових даних, кодування категоріальних ознак та інше. На етапі аналізу даних виконується дослідження структури та властивостей даних, щоб зрозуміти, як вони можуть впливати на результати моделювання.

Почнемо із збору даних. В зв'язку із війною в Україні доступ до даних банків крові та інших платформ став обмеженим. Для отримання даних було здійснено звернення до місцевого банку крові, однак отримана відмова, в зв'язку із вищезгаданою причиною. Проте, певні дані щодо донорства крові можна було знайти у відкритих джерелах.

Отриманий набір даних містить 4 незалежні характеристики «recency» (скільки минуло місяців із останньої здачі крові), «frequency» (скільки разів загалом донор здавав кров), «quantity» (скільки одиниць крові було здано донором) та «time» (скільки минуло місяців із першої здачі крові), а також 1 результуючий стовпець «donation», котрий вказує чи був здійснений збір крові відповідного донора в певний момент часу.

Дослідження показало, що цих характеристик достатньо для прогнозування майбутніх здач крові. Такий невеликий набір характеристик є

визначальним, адже репрезентує постійність та частоту донорства, що є особливостями на основі яких можна робити прогнози щодо майбутнього донорства.

Деякі дослідники також розглядали інші характеристики, такі як стать, вік та група крові донора. Проте ці дані виявилися менш релевантними, оскільки прогноз базується на історичних даних саме про здачі крові, які в свою чергу залежать від інших факторів.

Важливо відзначити, що у такому випадку дані можуть втрачати свою універсальність, оскільки розподіл донорів за статтю, віком та групою крові може відрізнятися в залежності від демографічного складу донорів. Наприклад, в Греції жінок серед донорів значно менше ніж в інших регіонах, через низький рівень заліза у крові. Віковий розподіл в свою чергу залежить від рівня розвиненості країни, що впливає на свідомість громадян.

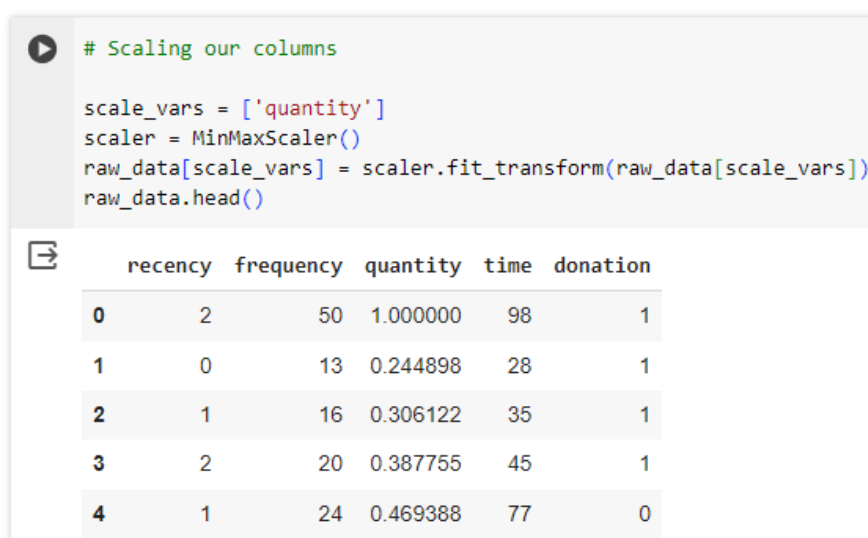
Для підвищення якості даних було проведено процес очищення. Однак не було виявлено жодних аномальних чи пропущених значень, дані виявилися відмінної якості. Натомість, виявлено, що стовпець «quantity» має надто широкий діапазон значень, що може впливати на навчання моделі. Тому була виконана нормалізація цього стовпця для забезпечення рівної ваги всіх характеристик.

В даному наборі даних близько 1400 записів. Здавалося б недостатньо для ефективного машинного навчання. Однак дослідники радять мати в 10 разів більше записів ніж є характеристик. Враховуючи це і кількість характеристик в даному наборі даних можна стверджувати, що даних більш ніж достатньо. Попри це була проведена спроба синтезувати дані, використовуючи різні сервіси та платформи, однак результати були дуже низької якості, оскільки містили шум та дуже погано синтезували випадки здачі крові донором.

2.3 Практична реалізація алгоритмів машинного навчання

Перейдемо до реалізації вищезгаданих алгоритмів машинного навчання мовою програмування Python.

В першу чергу після завантаження даних відбувається нормалізація відповідних стовпців. Нормалізація – це процес приведення значень характеристик до діапазону між 0 та 1 або до іншого заданого діапазону. Один з найпоширеніших методів нормалізації – це MinMaxScaler. MinMaxScaler може бути чутливим до викидів у даних, однак, як було досліджено раніше, аномальних значень чи іншого шуму в наборі даних виявлено не було.



```
# Scaling our columns

scale_vars = ['quantity']
scaler = MinMaxScaler()
raw_data[scale_vars] = scaler.fit_transform(raw_data[scale_vars])
raw_data.head()
```

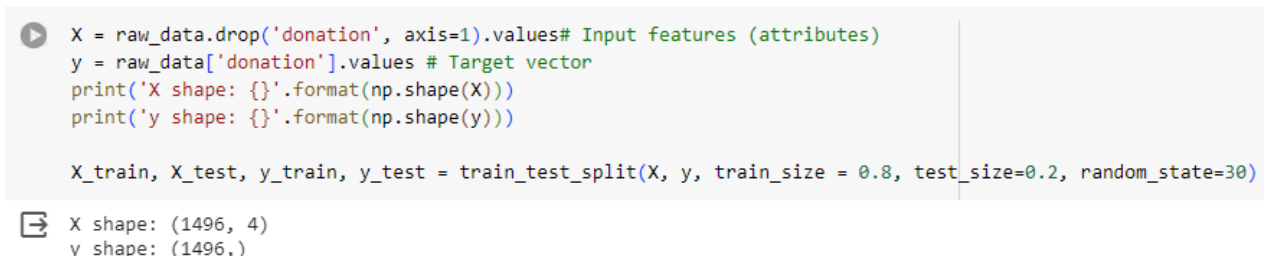
	recency	frequency	quantity	time	donation
0	2	50	1.000000	98	1
1	0	13	0.244898	28	1
2	1	16	0.306122	35	1
3	2	20	0.387755	45	1
4	1	24	0.469388	77	0

Рис. 2.5. Нормалізація даних

Далі відбувається розділення набору даних на навчальний та тестовий. Це є важливим етапом в машинному навчанні. Це допомагає оцінити реальну продуктивність моделі під час тестування на даних, які модель не бачила під час навчання. Відношення розділення між навчальним і тестовим наборами даних зазвичай залежить від розміру загального набору даних та конкретної задачі, але існують загальні рекомендації. Зазвичай рекомендується розділяти набір даних у співвідношенні близько 70-80% для навчання і 20-30% для тестування. Ці співвідношення дозволяють забезпечити достатню кількість даних для ефективного навчання моделі, а також достатньо об'єктивне тестування її продуктивності. Важливо, щоб обидва набори даних були

репрезентативними за розподілом класів або категорій, що дозволить уникнути викривлення результатів тестування.

Саме тому було обрано розділення на 20% і 80%, а також використане перемішування даних.



```
X = raw_data.drop('donation', axis=1).values# Input features (attributes)
y = raw_data['donation'].values # Target vector
print('X shape: {}'.format(np.shape(X)))
print('y shape: {}'.format(np.shape(y)))

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.8, test_size=0.2, random_state=30)
```

X shape: (1496, 4)
y shape: (1496,)

Рис. 2.6. Розділення набору даних на навчальний та тестовий

Після цього ми починаємо навчання моделі. Було навчено 5 моделей використовуючи різні алгоритми. Для налаштування параметрів моделі використовувався GridSearchCV. Це метод оптимізації параметрів моделі шляхом перебору всіх можливих комбінацій значень параметрів заздалегідь визначеного простору параметрів і вибору того, який найкраще підходить за допомогою крос-валідації. Це допомагає знайти оптимальні параметри моделі, які максимізують її продуктивність на навчальних даних та забезпечують хорошу загальну узагальнену здатність на нових даних.

У якості крос-валідації використовується K-fold крос-валідація. Під час K-fold крос-валідації навчальний набір ділиться на K частин. Потім модель навчається K разів, кожного разу використовуючи іншу частину як тестовий набір, а решту даних – як навчальний набір.

Після цього обчислюється середнє значення помилок (або іншої метрики оцінки) для кожного тестового набору. Ці значення середньої помилки можуть бути використані для отримання більш точної оцінки продуктивності моделі.

```

▶ # Random Forest
  param_grid = {
      'n_estimators': [100, 200, 300],
      'min_samples_split': [2, 5, 10],
      'max_features': [None, 'sqrt', 'log2'],
      'max_depth': [None, 5, 10, 15]
  }

  dt = RandomForestClassifier()
  grid_search = GridSearchCV(
      dt, param_grid=param_grid, n_jobs=5, scoring='roc_auc', cv=5, verbose=3)
  grid_search.fit(X_train, y_train)
  best_params = grid_search.best_params_
  dt = RandomForestClassifier(**best_params)
  dt.fit(X_train, y_train)
  print(best_params)

```

 Fitting 5 folds for each of 108 candidates, totalling 540 fits
 {'max_depth': 15, 'max_features': 'log2', 'min_samples_split': 2, 'n_estimators': 100}

Рис. 2.7. Пошук оптимальних параметрів моделі та її навчання

Далі проводиться оцінка важливості характеристик набору даних. Це процес визначення впливу кожної окремої характеристики на результати моделі в машинному навчанні. Цей процес допомагає виявити, які характеристики є найбільш важливими для моделі у вирішенні конкретної задачі.

Важливо відзначити, що оцінка важливості характеристик є лише одним із засобів аналізу моделі і може допомогти в розумінні, які аспекти даних є найбільш істотними для прогнозування. Однак, вона повинна використовуватися разом з іншими методами оцінки моделі для повного розуміння її продуктивності та недоліків.

```

#del final-fi
final-fi = pd.DataFrame();
# Calculating FI
for i, column in enumerate(raw_data.drop('donation', axis=1)):
    print('Importance of feature {}: {:.3f}'.format(column, dt.feature_importances_[i]))

    fi = pd.DataFrame({'Variable': [column], 'Feature Importance Score': [dt.feature_importances_[i]]})
    try:
        final-fi = pd.concat([final-fi, fi], ignore_index = True)
    except:
        final-fi = fi

# Ordering the data
final-fi = final-fi.sort_values('Feature Importance Score', ascending = False).reset_index()
final-fi

```

Importance of feature recency: 0.275
 Importance of feature frequency: 0.150
 Importance of feature quantity: 0.139
 Importance of feature time: 0.436

	index	Variable	Feature Importance Score
0	3	time	0.436353
1	0	recency	0.274517
2	1	frequency	0.150262
3	2	quantity	0.138868

Рис. 2.8. Оцінка важливості характеристик набору даних

Після того як модель навчена, можна провести заміри її ефективності. Для цього в першу чергу використовується матриця невідповідностей. Це інструмент, що використовується для оцінки продуктивності класифікаційних моделей в машинному навчанні. Вона дозволяє візуалізувати та порівнювати фактичні та передбачені класифікації для моделі на основі кількості помилок.

Матриця невідповідностей має кілька основних елементів:

- True Positives (TP): Кількість випадків, коли модель правильно передбачила позитивний клас.
- True Negatives (TN): Кількість випадків, коли модель правильно передбачила негативний клас.
- False Positives (FP): Кількість випадків, коли модель помилково передбачила позитивний клас (коли фактично він є негативним). Це також відомо як помилкова спритність (Type I error).

- False Negatives (FN): Кількість випадків, коли модель помилково передбачила негативний клас (коли фактично він є позитивним). Це також відомо як промах (Type II error).

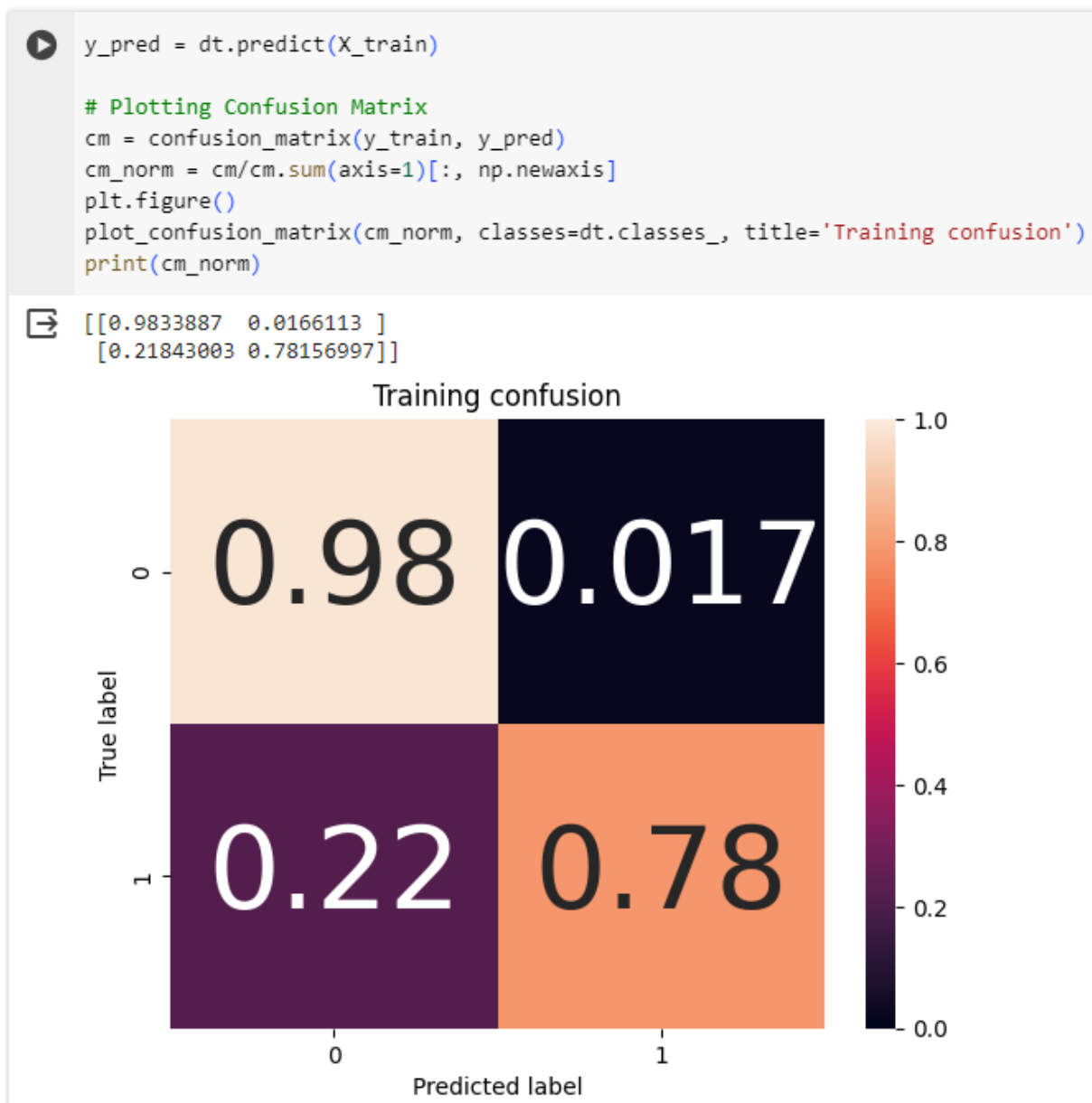


Рис. 2.9. Побудова матриці невідповідності

На основі цієї матриці можна розрахувати різні метрики, такі як точність (accuracy), чутливість (recall), специфічність (specificity), влучність (precision) та інші, які допомагають оцінити продуктивність моделі для конкретної задачі класифікації. Наприклад, точність – це співвідношення правильно класифікованих випадків (TP і TN) до загальної кількості випадків. Чутливість – це співвідношення TP до суми TP і FN, специфічність – це співвідношення

TN до суми TN і FP, а влучність – це співвідношення TP до суми TP і FP. Точність вимірює загальну ефективність моделі, а чутливість є важливою для виявлення позитивних випадків і уникнення пропусків.

```

# Calculating False Positives (FP), False Negatives (FN), True Positives (TP) & True Negatives (TN)
FP = cm.sum(axis=0) - np.diag(cm)
FN = cm.sum(axis=1) - np.diag(cm)
TP = np.diag(cm)
TN = cm.sum() - (FP + FN + TP)

# Sensitivity, hit rate, recall, or true positive rate
TPR = TP/(TP+FN)
print ("Recall per class is: ",TPR)

# Precision or positive predictive value
PPV = TP/(TP+FP)
print ("The Precision per class is: ",PPV)

# Overall accuracy
ACC = (TP+TN)/(TP+FP+FN+TN)
print ("The Accuracy of each class is", ACC)

# Specificity
SPE = (TN)/(TN+FP)
print ("The Specificity of each class is", SPE)
print("")

##Total averages :
print ("The average Accuracy is", round(ACC.sum()/2, 4))
print ("The average Recall is: ", round(TPR.sum()/2, 4))
print ("The average Precision is: ", round(PPV.sum()/2, 4))
print ("The average Specificity is", round(SPE.sum()/2, 4))

```

```

Recall per class is: [0.9833887  0.78156997]
The Precision per class is: [0.93277311 0.93852459]
The Accuracy of each class is [0.93394649 0.93394649]
The Specificity of each class is [0.78156997 0.9833887 ]

The average Accuracy is 0.9339
The average Recall is:  0.8825
The average Precision is: 0.9356
The average Specificity is 0.8825

```

Рис. 2.10. Обчислення метрик для оцінки моделі

Відповідні обчислення було проведено як для навчального, так і для тестового набору даних.

2.4 Аналіз результатів досліджень на основі обраних метрик

Оцінка моделі машинного навчання – це процес визначення ефективності моделі на основі тестових даних. Це один з найважливіших етапів в процесі розробки моделі, оскільки від цього залежить її придатність для використання в практичних задачах. Оцінка дозволяє визначити, наскільки

добре модель узгоджується з реальними даними і наскільки вона може передбачати нові дані.

Основними метриками для задач класифікації є точність (accuracy), чутливість (recall), специфічність (specificity) та влучність (precision). Кожну із цих метрик можна розрахувати для кожного класу даних окремо. Специфічність є оберненою до чутливості, тому при обчисленні середнього для двох класів значення метрики ми отримуємо однакові значення для цих двох метрик. В зв'язку із цим специфічність було опущено при дослідженні ефективності моделі.

Як було згадано вище, для кожної із моделі була використана крос-валідація. Загалом вона не сильно вплинула на результати навчання моделі, однак дослідники рекомендують завжди використовувати її, адже при наборах даних необхідного розміру вона не має жодних негативних впливів. В свою чергу крос-валідація дозволяє отримати більш надійну оцінку моделі. Також вона допомагає виявити перенавчання моделі. Перенавчання виникає, коли модель стає дуже добре пристосованою до тренувальних даних, але не може узагальнити до нових даних. Це може статися, коли модель надмірно складна або коли в тренувальному наборі дуже мало даних.

Отримані при дослідженнях результати наведені в табл. 2.1 та табл. 2.2 (повна таблиця із підібраними параметрами навчання моделі та матрицями невідповідності наведена в додатках).

Табл. 2.1

Порівняння метрик ефективності моделі з крос-валідацією

Name	Train			Test		
	Accuracy	Recall	Precision	Accuracy	Recall	Precision
Logistic Regression	0.7751	0.5423	0.7057	0.77	0.5711	0.7143
SVM	0.7651	0.5	nan	0.75	0.5	nan
ANN	0.8027	0.6676	0.7436	0.8133	0.6779	0.716
Decision Tree	0.9339	0.8693	0.9454	0.87	0.8022	0.8377
Random Forest	0.9348	0.8871	0.9272	0.8867	0.8222	0.8642

Табл. 2.2

Порівняння метрик ефективності моделі без крос-валідації

Name	Train			Test		
	Accuracy	Recall	Precision	Accuracy	Recall	Precision
Logistic Regression	0.7651	0.5504	0.6842	0.7967	0.5508	0.6896
SVM	0.7818	0.5516	0.7509	0.7767	0.5756	0.7512
ANN	0.8043	0.6502	0.7394	0.8067	0.6756	0.7582
Decision Tree	0.9306	0.8622	0.9431	0.87	0.8022	0.8377
Random Forest	0.9348	0.8846	0.9297	0.8933	0.8356	0.8707

На основі отриманих спостережень можна зробити висновок, що найкраще для вирішення нашої задачі класифікації підходять деревоподібні алгоритми. Можна припустити, що це зв'язано із невеликою кількістю характеристик в наборі даних, що уможливорює побудову відносно не глибоких, проте дуже ефективних дерев.

При оцінці ефективності важливо звертати увагу не тільки на точність моделі, а й на збалансованість інших метрик. Адже можливі випадки, коли

загальна точність моделі є високою, проте правильне передбачення мінорного класу є надзвичайно низьким.



Рис. 2.10. Матриця невідповідності моделі Random Forest

Як ми бачимо, модель доволі ефективно передбачає мінорний клас. Взявши до уваги вищезгадане, зроблено висновок, що найкраще справляється алгоритм Random Forest. Його ефективність дещо вища ніж Decision Tree.

Розглянемо також модель, котра зовсім не впоралася із завданням. Це модель навчена за алгоритмом методу опорних векторів (SVM). Як ми бачимо із матриці невідповідності мінорний клас (успішні задачі крові) передбачається неймовірно погано.

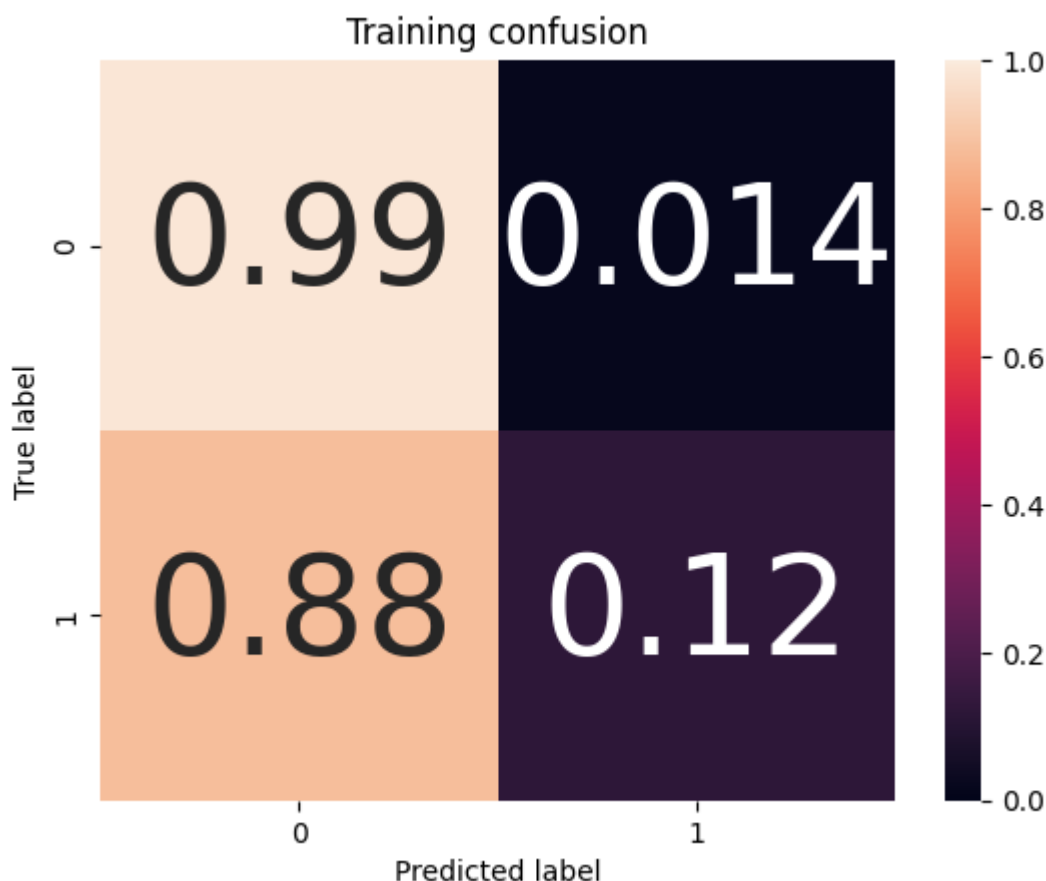


Рис. 2.11. Матриця невідповідності моделі SVM

Це зв'язано із незбалансованістю набору даних, тобто один клас представлений набагато частіше ніж інший. Особливість роботи алгоритму методу опорних векторів така, що модель сильно реагує на незбалансованість такого роду. Однак природа нашого набору даних така, що баланс класів можливо підтримувати лише штучно, адже кількість людей, котрі успішно здали кров в певний момент часу завжди менша ніж кількість тих, хто цього не зробив.

Розглянемо також швидкість навчання моделі. Абсолютними переможцями є алгоритм логістичної регресії та Decision Tree. Навчання моделей Random Forest та ANN трохи поступаються попереднім алгоритмам. І найбільш повільне навчання у SVM.

Висновки до другого розділу

В даному розділі було розглянуто програмні засоби реалізації алгоритмів машинного навчання. Далі проведено аналіз набору даних, розглянуто основні етапи роботи із ним та виявлені основні особливості для отримання найкращого результату.

Наведено приклади реалізації визначених алгоритмів на основі вищерозглянутих інструментів і засобів засоби реалізації алгоритмів машинного навчання.

Визначені основні критерії по яким буде здійснена оцінка ефективності моделі штучного інтелекту. Згідно них, на основі отриманих результатів експериментальних досліджень, були зроблені висновки щодо ефективності передбачення тієї чи іншої моделі. Також визначена доцільність використання відповідних алгоритмів для вирішення поставленої задачі.

РОЗДІЛ 3. ІНТЕГРАЦІЯ МАШИННОГО НАВЧАННЯ ТА ОПИС ПРОГРАМНОГО ПРОДУКТУ

3.1 Інтеграція машинного навчання у веб-сервіс

Для вирішення задач машинного навчання було створено окремий сервіс, за допомогою Python та бібліотеки Flask. Сервіс прийматиме запити двох видів: тренування моделі та передбачення на основі натренованої моделі. Ці ендпоінти можна побачити на рисунку 3.1.

```
@app.route('/train', methods=['POST'])
def execute_train_request():
    classification_method = request.json['method']
    train_data_file_path = request.json['trainDataFilePath']
    target_column = request.json['targetColumn']

    try:
        if classification_method == constants.ClassificationMethod.RANDOM_FOREST.value:
            service = RandomForestTrainService(train_data_file_path, target_column)
        elif classification_method == constants.ClassificationMethod.DECISION_TREE.value:
            service = DecisionTreeTrainService(train_data_file_path, target_column)
        elif classification_method == constants.ClassificationMethod.ANN.value:
            service = AnnTrainService(train_data_file_path, target_column)
        elif classification_method == constants.ClassificationMethod.LOGISTIC_REGRESSION.value:
            service = LogisticRegressionTrainService(train_data_file_path, target_column)
        else:
            return jsonify({'error': 'Classification method is not supported'}), 500

        return json.dumps(service.train_and_save_model(TRAIN_SIZE, TEST_SIZE))

    except Exception as e:
        return jsonify({'error': str(e)}), 500

@app.route('/predict', methods=['POST'])
def execute_predict_request():
    new_data_file_path = request.json['newDataFilePath']
    model_path = request.json['modelPath']

    try:
        service = PredictService(new_data_file_path, model_path)
        return json.dumps(service.predict_and_save_result())

    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

Рис 3.1. Ендрінти АРІ машинного навчання

Розглянемо детальніше, як працюють наведені ендпоінти. У випадку тренування моделі, у запиті до АРІ передаються метод машинного навчання моделі, шлях до файлу з набором даних для тренування моделі (попередньо

збережений веб-сервісом) та цільова колонка. Після цього відбувається навчання моделі та її збереження для подальшого використання. Даний ендпоінт повертає у відповідь на запит шлях до натренованої моделі (особливості побудови архітектури цілої системи).

Інший ендпоінт, котрий відповідає за передбачення, в свою чергу, приймає параметри, такі як шлях до навченої моделі та файл з даними, на основі яких необхідно зробити передбачення, застосовуючи завантажену модель. Повертає ендпоінт схожий на отриманий файл, лише із двома новими колонками, котрі відображають ймовірність що задача крові відбудеться та фінальне передбачення.

	A	B	C	D	E	F	G	H
1		recency	frequency	quantity	time	probability	prediction	
2	0	2	50	12500	98	0.58	1.0	
3	1	0	13	3250	28	0.9086111111111111	1.0	
4	2	1	16	4000	35	0.5477777777777778	1.0	
5	3	2	20	5000	45	0.48	0.0	
6	4	1	24	6000	77	0.13	0.0	
7	5	4	4	1000	4	0.326355383408015	0.0	
8	6	2	7	1750	14	0.1466666666666667	0.0	
9	7	1	12	3000	35	0.4497222222222223	0.0	
10	8	2	9	2250	22	0.09533333333333334	0.0	
11	9	5	46	11500	98	0.20333333333333334	0.0	
12	10	4	23	5750	58	0.1766666666666667	0.0	
13	11	0	3	750	4	0.3327777777777778	0.0	
14	12	2	10	2500	28	0.14	0.0	
15	13	1	13	3250	47	0.4592124542124542	0.0	
16	14	2	6	1500	15	0.12234567901234568	0.0	
17	15	2	5	1250	11	0.14234567901234568	0.0	
18	16	2	14	3500	48	0.32083333333333336	0.0	
19	17	2	15	3750	49	0.4391666666666667	0.0	
20	18	2	6	1500	15	0.12234567901234568	0.0	
21	19	2	3	750	4	0.17221769702032858	0.0	
22	20	2	3	750	4	0.17221769702032858	0.0	
23	21	4	11	2750	28	0.16	0.0	
24	22	2	6	1500	16	0.135679012345679	0.0	
25	23	2	6	1500	16	0.135679012345679	0.0	
26	24	9	9	2250	16	0.0380952380952381	0.0	
27	25	4	14	3500	40	0.4266666666666667	0.0	
28	26	4	6	1500	14	0.19425	0.0	
29	27	4	12	3000	34	0.19333333333333333	0.0	
30	28	4	5	1250	11	0.14288888888888893	0.0	
31	29	4	8	2000	21	0.1551309523809524	0.0	

Рис 3.2. Файл із передбаченнями отриманий в результаті прогнозного аналізу

Вищезгаданий веб-сервіс являє собою ASP.NET Core MVC додаток. Саме використовуючи цей веб-сервіс користувач матиме змогу взаємодіяти із всією системою. Отже цей додаток являє собою і користувацький інтерфейс, і механізм обробки запитів та зв'язку із зовнішнім API машинного навчання та базою даних. Всю ідею взаємодії системи можна описати наступною схемою.

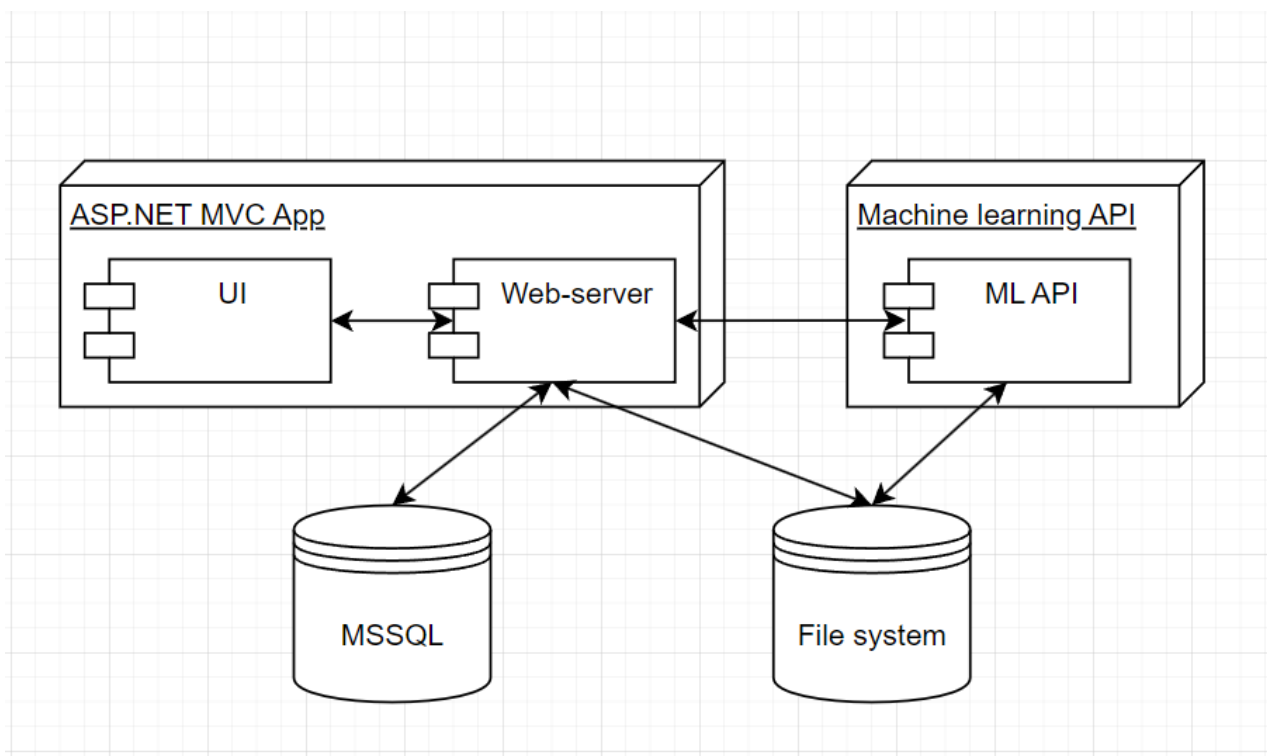


Рис 3.3. Схема взаємодії системи

3.2 Структура інтерфейсу та порядок роботи з програмним продуктом

Основною ідеєю додатку є його використання звичайними медичними працівниками, які не є фахівцями в машинному навчанні. Тому додаток виконує ряд вимог таких як: інтуїтивно зрозумілий і неперевантажений інтерфейс, наявність лише необхідних інструментів для простого прогнозування завчасно відомої області.

Користувачу надана можливість створити на навчити декілька моделей. Адже набори даних можуть відрізнятися не тільки за своєю структурою, а й безпосередньо даними. Тому, наприклад, для кожного центру крові є

можливість створити та навчити власну модель, яка найкраще працює саме з даними відповідного центру крові.

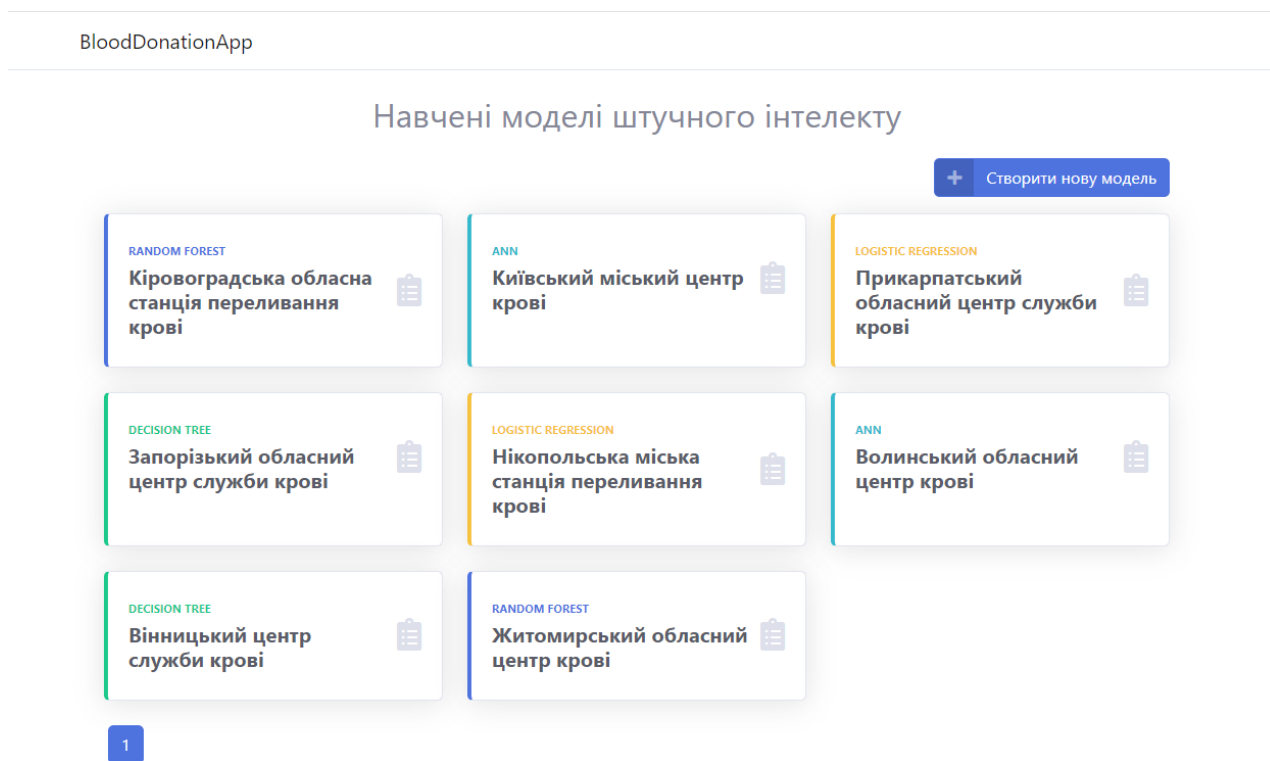


Рис 3.4. Сторінка переліку усіх створених моделей

Як ми можемо бачити, на цій сторінці наявна кнопка створення нової моделі. Воно перенаправляє користувача на сторінку створення моделі. Вона являє собою форму, котра вимагає від користувача назвати модель, завантажити файл з набором даних, обрати цільову колонку та метод машинного навчання. Для зручності цільова колонка обирається після завантаження файлу, щоб користувач міг обрати тільки ту колонку, котра наявна у файлі.

Файл даних навчання

Choose File train_data.csv ⓘ

Цільова колонка

recency

recency

frequency

quantity

time

donation

✓ Створити ➔ Назад

Рис 3.5. Вибір цільової колонки

Створення та навчення нової моделі штучного інтелекту

Назва моделі

Житомирський центр крові

Файл даних навчання

Choose File train_data.csv ⓘ

Цільова колонка

donation

Тип моделі

Random Forest ⓘ

✓ Створити ➔ Назад

Рис 3.6. Форма створення нової моделі

Звичайно наявна валідація, щоб користувач не міг відправити запит, повністю його не сформувавши всією необхідною інформацією. Це також дозволяє користувачу розуміти, які поля є обов'язковими, а які ні.

Створення та навчення нової моделі штучного інтелекту

Назва моделі

The Назва моделі field is required.

Файл даних навчання

Choose File train_data.csv i

Цільова колонка

Тип моделі

i

✓ Створити → Назад

Рис 3.9. Валідація форми

Так як додаток зберігає можливість обрати набір файлів довільної структури та метод машинного навчання із переліку, для користувачів наведені рекомендації щодо найбільш оптимальної структури та методу навчання, адже система передбачає, що нею користуватимуться не фахівці штучного інтелекту.

Створення та навчення нової моделі штучного інтелекту

Назва моделі

Житомирський центр крові

Файл даних навчання

Choose File train_data.csv

Цільова колонка

donation

Тип моделі

Random Forest

✓ Створити → Назад

Для найбільш ефективного навчання файл даних повинен містити колонки, такі як, місяці з моменту першої та останньої здачі крові, загальна кількість здач кров, кількість зібраної крові

Створення та навчення нової моделі штучного інтелекту

Назва моделі

Житомирський центр крові

Файл даних навчання

Choose File train_data.csv

Цільова колонка

donation

Тип моделі

Random Forest

Найбільш ефективними типами моделі класифікації є Random Forest та Decision Tree

✓ Створити → Назад

Рис 3.8. Рекомендації для користувача щодо навчання моделі

Після навчання моделі користувач потрапляє на сторінку самої моделі. На ній ми можемо бачити назву моделі та метрики, отримані під час навчання та тренування моделі. Попри те, що користувачі системи, котрі не є фахівцями

штучного інтелекту, вони використовуючи базові метрики матимуть уявлення про ефективність та правильність навчання моделі. Адже на цій сторінку можна переглянути інформацію, про важливість і перелік характеристик даних, точність, чутливість та влучність. Користувачам не потрібно знати і розуміти природу цих метрик, лише знати що їх значення повинно бути високим. Це засвідчуватиме про правильність обраного набору даних, цільової колонки тощо. Це є необхідністю, адже система не може знати чи правильний файл чи колонку обрав користувач.

Житомирський обласний центр крові

Зробити передбачення

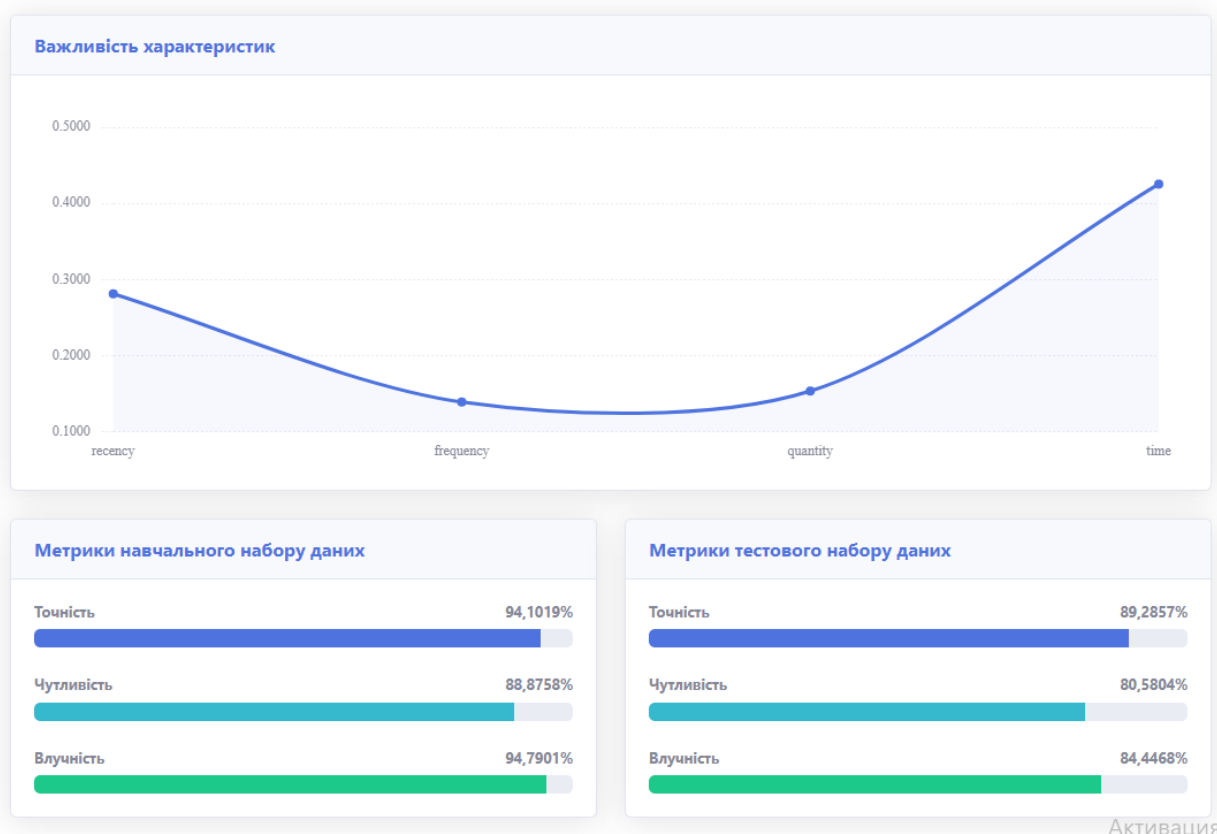


Рис 3.9. Сторінка моделі штучного інтелекту

Не всі моделі підтримують повний набір метрик, через особливості методу класифікації. Тому користувач також бачить повідомлення, якщо певні метрики не підтримуються тією чи іншою моделлю. Важливо також правильно сформулювати повідомлення, щоб користувач розумів, що це особливості моделі, а не якась помилка системи чи його самого при створенні моделі.

Волинський обласний центр крові

Зробити передбачення

Важливість характеристик не підтримується методом типом класифікації цієї моделі...

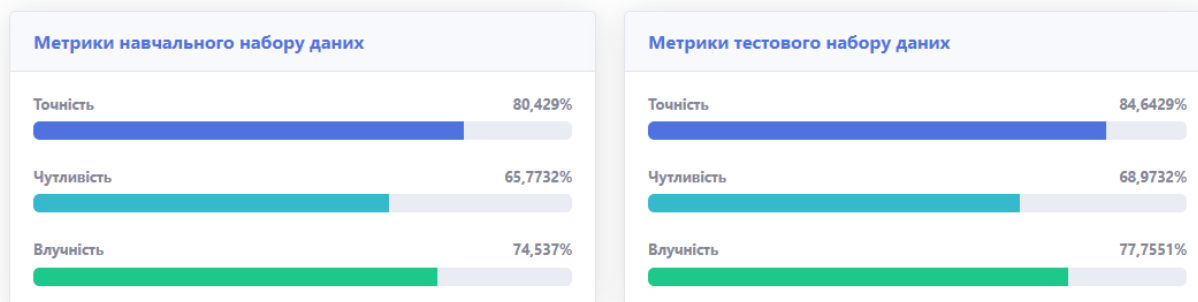


Рис 3.10. Сторінка моделі, котра не підтримує визначення важливості характеристик

У верхній частині сторінки ми бачимо кнопку завдяки якій ми можемо зробити передбачення. При натисненні на неї перед нами з'являється форма, в котру потрібно лише завантажити файл, на основі якого потрібно зробити передбачення.

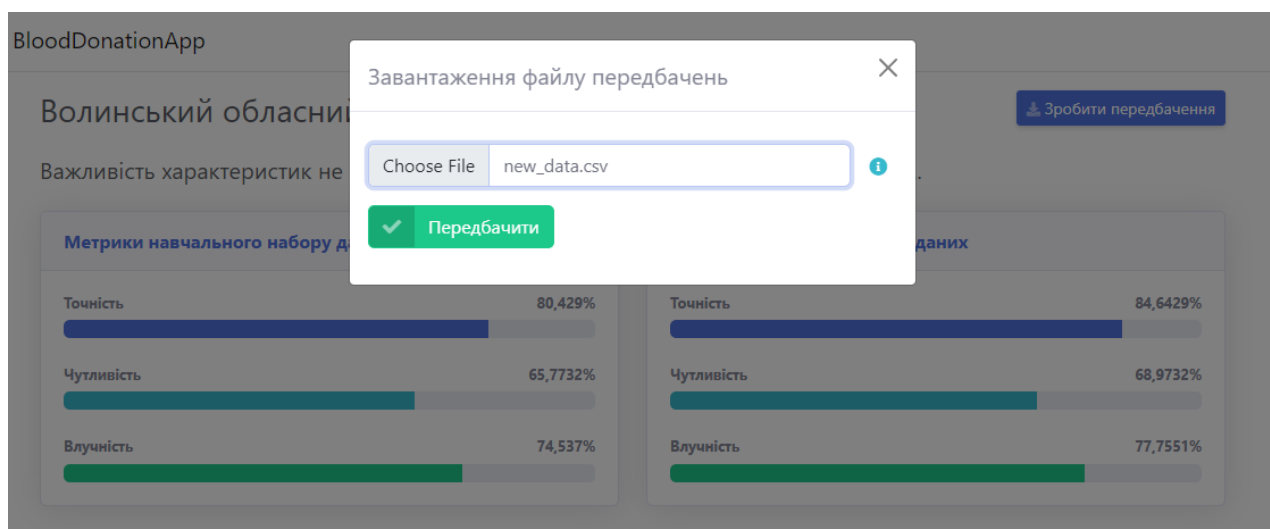


Рис 3.11. Форма завантаження файлу для передбачень

В цій формі також наявні рекомендації щодо правильного використання моделі, аби користувач розумів обмеження роботи системи.

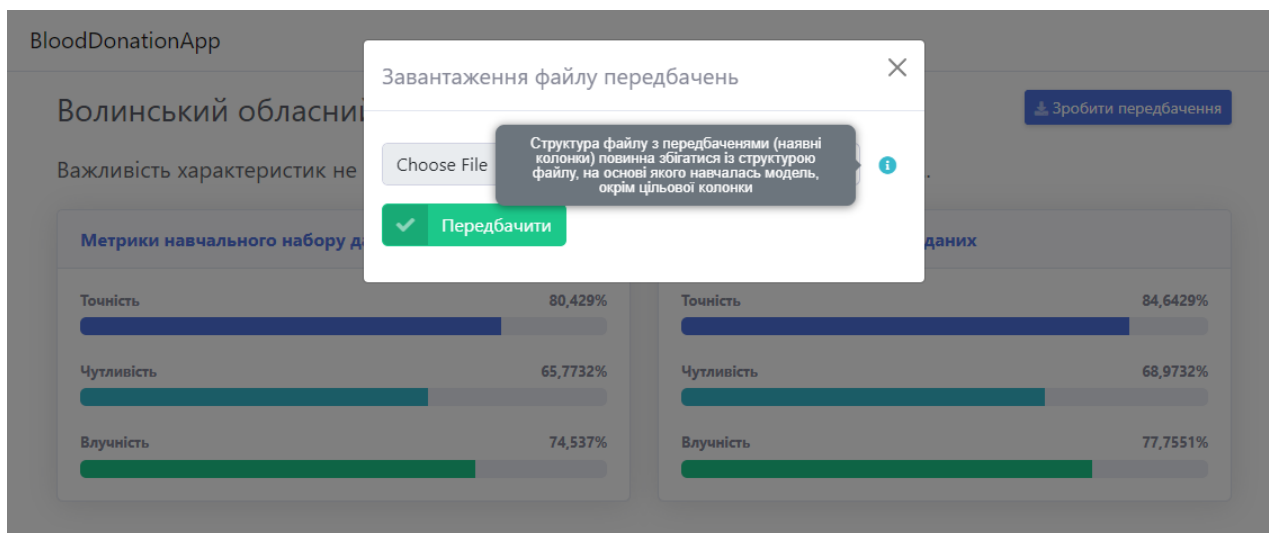


Рис 3.12. Рекомендації системи

Після цього файл буде відправлено на сервер для формування передбачень. Після завершення на комп'ютер користувача буде завантажено новий файл, створений на основі завантаженого, в якому будуть сформовані передбачення та їх ймовірності на основі прогнозного аналізу.

Висновки до третього розділу

В даному розділі була розглянута інтеграція проекту машинного навчання у веб-сервіс. Було наведено взаємодію компонентів системи та їх будову, особливості роботи навчання моделі та формування відповідних передбачень.

Продемонстровано структуру інтерфейсу та порядок роботи із додатком. Згадані важливі особливості концепції додатку та наведена їх реалізація.

ВИСНОВКИ

Під час кваліфікаційної роботи було визначено проблему передбачення потенційних донорів крові. Було проведено попередній аналіз найбільш ефективних методів класифікації для вирішення поставленої задачі та аналогів програмного забезпечення із реалізацією виявлених алгоритмів.

Проведено дослідження щодо ефективності методів машинного навчання на попередньо визначеному наборі даних. Виявлено найбільш оптимальну структуру набору даних, проаналізована важливість кожної характеристики. Досліджені різні методи і принципи попередньої обробки даних.

Практично реалізовані обрані методи машинного навчання та протестовані на визначеному наборі даних. Експериментальним чином виявлені найбільш оптимальні параметри для алгоритмів навчання.

Також було проведено дослідження найважливіших метрик та їх вплив на потенційну оцінку ефективності моделі, після чого проведена порівняльна характеристики різних методів класифікації.

На основі отриманих результатів обрані найбільш ефективні моделі класифікації, реалізації яких були додані до розробленої програмної системи. Досліджені особливості та актуальність такого програмного продукту, додані рекомендації для користувачів для отримання найкращих результатів. Розроблений додаток було протестовано.

Загалом проведене дослідження та виконану роботу можна вважати вдалими, адже додаток вимагає від користувача мінімальну кількість навичок та знань машинного навчання, а створені на основі досліджених алгоритмів моделі з підібраними параметрами надають чудові результати прогнозування потенційних донорів крові.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hong-yun Wu, Zheng-gang Li, Xin-kai Sun, Wei-min Bai, An-di Wang, Yu-chi Ma, Ren-hua Diao, Eng-yong Fan, Fang Zhao, Yun-qi Liu, Yi-zhou Hong, Ming-hua Guo, Hui Xue. Predicting willingness to donate blood based on machine learning: two blood donor recruitments during COVID-19 outbreaks. 2022. <https://doi.org/10.1038/s41598-022-21215-2>
2. Marco Bani, Barbara Giussani. Gender differences in giving blood: a review of the literature. 2010. <https://doi.org/10.2450/2010.0156-09>
3. Why is my test accuracy higher than train accuracy? SKLEARN [Електронний ресурс] – Режим доступу до ресурсу: <https://datascience.stackexchange.com/questions/107834/why-is-my-test-accuracy-higher-than-train-accuracy-sklearn>
4. ML | Underfitting and Overfitting [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/>
5. SWASTIK SATPATHY. SMOTE for Imbalanced Classification with Python/ 2023.
6. Aayush Bajaj. Performance Metrics in Machine Learning. 2023.
7. Destin Gong. Top 6 Machine Learning Algorithms for Classification. 2022.
8. Shashikala B M., M. P. Pushpalatha. Machine Learning Approaches for Potential Blood Donors Prediction. 2019. Pp. 483-491. https://doi.org/10.1007/978-981-13-5802-9_44
9. Deepti Bahel, Prerana Ghosh, Arundhyoti Sarkar, Matthew A. Lanham. PREDICTING BLOOD DONATIONS USING MACHINE LEARNING TECHNIQUES. 2019.
10. Susanne Suessner, Norbert Niklas, Ulrich Bodenhofer, Jens Meier. Machine learning-based prediction of fainting during blood donations using donor properties and weather data as features. 2022. <https://doi.org/10.1186/s12911-022-01971-x>

- 11.Смутних Іван Олександрович, Єфремов Юрій Миколайович. ДОЦІЛЬНІСТЬ ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В СУЧАСНІЙ МЕДИЦИНІ. 2024. Рр. 114-115.
- 12.Niklas Kühl, Max Schemmer, Marc Goutier, Gerhard Satzger. Artificial intelligence and machine learning. 2022. Рр. 2235-2244.
- 13.Youdi Gong, Guangzhen Liu, Yunzhi Xue, Rui Li, Lingzhong Meng. A survey on dataset quality in machine learning. 2023. <https://doi.org/10.1016/j.infsof.2023.107268>
- 14.Daniel Berrar. Cross-Validation. 2018. <https://doi.org/10.1016/B978-0-12-809633-8.20349-X>
- 15.Max A Little, Gael Varoquaux, Sohrab Saeb, Luca Lonini, Arun Jayaraman, David C Mohr, Konrad P Kording. Using and understanding cross-validation strategies. Perspectives on Saeb et al. 2017. <https://doi.org/10.1093/gigascience/gix020>

ДОДАТКИ

ДОДАТОК А

Таблиця метрик алгоритмів класифікації

Name	CV	Train			Test			Training matrix	Testing matrix	Optimal options
		Accuracy	Recall	Precision	Accuracy	Recall	Precision			
Logistic Regression	+	0.7751	0.5423	0.7057	0.77	0.5711	0.7143	[[[0.98 0.019] [0.9 0.1]]]	[[[0.97 0.03] [0.83 0.17]]]	C': 10, 'penalty': 'l2', 'solver': 'liblinear'
SVM	+	0.7651	0.5	nan	0.75	0.5	nan	[[[1. 0.] [1. 0.]]]	[[[1. 0.] [1. 0.]]]	C': 10, 'kernel': 'linear'
ANN	+	0.8027	0.6676	0.7436	0.8133	0.6779	0.716	[[[0.93 0.067] [0.6 0.4]]]	[[[0.91 0.089] [0.56 0.44]]]	'activation': 'tanh', 'hidden_layer_sizes': 500, 'max_iter': 1000
Decision Tree	+	0.9339	0.8693	0.9454	0.87	0.8022	0.8377	[[[0.99 0.0087] [0.25 0.75]]]	[[[0.94 0.062] [0.33 0.67]]]	max_depth: 15, 'max_features': None, 'min_samples_split': 2
Random Forest	+	0.9348	0.8871	0.9272	0.8867	0.8222	0.8642	[[[0.98 0.023] [0.2 0.8]]]	[[[0.95 0.049] [0.31 0.69]]]	max_depth: None, 'max_features': 'log2', 'min_samples_split': 2, 'n_estimators': 200
Logistic Regression	-	0.7651	0.5504	0.6842	0.7967	0.5508	0.6896	[[[0.97 0.03] [0.87 0.13]]]	[[[0.97 0.03] [0.87 0.13]]]	C': 10, 'penalty': 'l2', 'solver': 'liblinear'
SVM	-	0.7818	0.5516	0.7509	0.7767	0.5756	0.7512	[[[0.99 0.01] [0.88 0.12]]]	[[[0.98 0.022] [0.83 0.17]]]	C': 10
ANN	-	0.8043	0.6502	0.7394	0.8067	0.6756	0.7582	[[[0.94 0.06] [0.64 0.36]]]]	[[[0.94 0.06] [0.59 0.41]]]	"activation": 'tanh', 'hidden_layer_sizes': 500, 'max_iter': 1000
Decision Tree	-	0.9306	0.8622	0.9431	0.87	0.8022	0.8377	[[[0.99 0.0087] [0.27 0.73]]]	[[[0.94 0.062] [0.33 0.67]]]	max_depth=15
Random Forest	-	0.9348	0.8846	0.9297	0.8933	0.8356	0.8707	[[[0.98 0.0201] [0.21 0.79]]]	[[[0.95 0.049] [0.28 0.72]]]]	max_depth=15

ДОДАТОК Б

Код навчання моделі та підбору оптимальних параметрів

```
def train_and_save_model(self, train_size, test_size):
    self.raw_data = pd.read_csv(self.train_data_path, encoding='latin-1',
sep=';')
    data_helper = DataHelper()
    self.raw_data = data_helper.scale_data(self.raw_data)
    X_train, X_test, y_train, y_test = self.split_and_shuffle_data(train_size,
test_size)

    param_grid = {
        'n_estimators': [100, 200, 300],
        'min_samples_split': [2, 5, 10],
        'max_features': [None, 'sqrt', 'log2'],
        'max_depth': [None, 5, 10, 15]
    }

    self.model = RandomForestClassifier()
    grid_search = GridSearchCV(self.model, param_grid=param_grid, n_jobs=5,
scoring='roc_auc', cv=5, verbose=3)
    grid_search.fit(X_train, y_train)
    best_params = grid_search.best_params_
    self.model = RandomForestClassifier(**best_params)
    self.model.fit(X_train, y_train)

    model_path = self.save_model()

    train_metrics = self.calculate_metrics(X_train, y_train)
    test_metrics = self.calculate_metrics(X_test, y_test)
    features_importance = self.calculate_features_importance()

    return {
        'trainMetrics': train_metrics,
        'testMetrics': test_metrics,
        'featuresImportance': features_importance,
        'filePath': model_path
    }
```

ДОДАТОК В

Код формування передбачення на основі завантаженої моделі

```
def predict_and_save_result(self):
    self.model = joblib.load(self.model_path)

    new_data = pd.read_csv(self.new_data_path, encoding='latin-1', sep=';')
    data_helper = DataHelper()
    self.original_data = new_data
    new_data = data_helper.scale_data(new_data)

    X = new_data.values

    self.pred_final = self.model.predict(X)
    self.pred_prob_final = self.model.predict_proba(X)

    datahelper = DataHelper()
    output = datahelper.process_prediction(self.original_data,
self.pred_final, self.pred_prob_final)
    guid = uuid.uuid4().hex
    directory_path = os.path.join(constants.SHARED_PREDICTIONS_FOLDER, guid)
    os.mkdir(directory_path)

    filePath = os.path.join(directory_path, 'prediction.csv')
    output.to_csv(filePath, encoding='latin-1', sep=';')

    return {
        'filePath': os.path.join(guid, 'prediction.csv'),
    }
```