МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів
Кафедра математичного моделювання та штучного інтелекту

Пояснювальна записка до дипломної роботи

(тип кваліфікаційної роботи)

другий (магістерський)
(освітній ступінь)

на тему «Розробка програмного забезпечення для аналізу та візуалізації даних на основі методів машинного навчання»

ХАІ.304.365А.23О.122.6002055 ПЗ

Виконав: здобувач 2 курсу групи № 365а
Галузь знань 12 Інформаційні технології
(код та найменування)
Спеціальність 122 Комп`ютерні науки
(код та найменування)
Освітня програма Інтелектуальні системи та технології
(найменування)
Хлистун О. В.
(прізвище та ініціали здобувачки)
Керівник: Базілевич К. О.
(прізвище та ініціали)
Рецензент: Меняйлов Є. С.
(прізвище та ініціали)

Харків – 2023

Міністерство освіти і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів
(повне найменування)
Кафедра математичного моделювання та штучного інтелекту
(повне найменування)
Рівень вищої освіти другий (магістерський)
Галузь знань 12 Інформаційні технології
(код та найменування)
Спеціальність 122 Комп`ютерні науки
(код та найменування)
Освітня програма Інтелектуальні системи та технології
(найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Чухрай А.Г.
(підпис) (ініціали та прізвище)
«___» _____ 20___ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Хлистуну Олексію Вікторовичу
(прізвище, ім'я та по батькові)

1. Тема кваліфікаційної роботи Розробка програмного забезпечення для аналізу та візулізації даних на основі методів машинного навчання
керівник кваліфікаційної роботи Базілевич Ксенія Олексіївна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом Університету № 1968-уч від « 06 » листопада 2023 року.
2. Термін подання здобувачем кваліфікаційної роботи 31 січня 2024 р.
3. Вихідні дані до роботи дата-сети з інформацією про медичні дослідження
4. Зміст пояснювальної записки (перелік завдань, які потрібно розв'язати)
1) аналітичний огляд та сфери застосування методів машинного навчання;
2) методи машинного навчання для вирішення задачі зниження розмірності та класифікації даних;
3) розробка програмного забезпечення для аналізу та візуалізації;
4) використання програмного забезпечення для аналізу та візуалізації.
5. Перелік графічного матеріалу тема дипломної роботи, мета дослідження, об`єкт та предмет дослідження, актуальність теми дослідження, алгоритм зменшення розмірності даних, алгоритм методу класифікатора дерева рішень, алгоритм методу класифікатора опорних векторів, проектування програмного продукту, програмна реалізація, висновки.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина	Базілевич К.О., доцент		
Технологічна частина	Базілевич К.О., доцент		

Нормоконтроль _____ В. О. Халтурін «____» _____ 20__ р.
(підпис) (ініціали та прізвище)

7. Дата видачі завдання «____» _____ 20___ р.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області		
2	Обґрунтування вибору математичної моделі		
3	Опис математичної моделі		
4	Розробка алгоритмічної моделі та методики		
5	Тестування програмного продукту		
6	Розрахунок основних результатів		
7	Оформлення розрахунково-пояснювальної записки		
8	Передзахист дипломної роботи магістра		
9	Виправлення помилок та корегування зауважень		
10	Захист дипломної роботи магістра		

Здобувач _____ О. В. Хлистун
(підпис) (ініціали та прізвище)

Керівник кваліфікаційної роботи

_____ К. О. Базілевич
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота за темою «Розробка програмного забезпечення для аналізу та візуалізації даних на основі методів машинного навчання» містить 82 сторінок текстового документа, 37 ілюстрацій, 2 таблиці, 32 використаних джерела.

Основна мета роботи: визначення та реалізація методів, які дозволяють провести зменшення розмірності ознак та класифікацію даних у медичних дослідженнях із застосуванням машинного навчання, а також візуалізація отриманих результатів. Для реалізації мети необхідно було вирішити такі задачі: провести аналітичний огляд методів машинного навчання для зниження розмірності та класифікації даних; розробити алгоритмічні моделі методів машинного навчання для вирішення задач зниження розмірності та класифікації даних; розробити програмне забезпечення для аналізу та візуалізації даних на основі методів машинного навчання.

Об'єкт дослідження: процес аналізу даних у медичних дослідженнях.

Предмет дослідження: використання методів машинного навчання для зниження розмірності та класифікації даних.

Ключові слова: ЗМЕНШЕННЯ РОЗМІРНОСТІ, КЛАСИФІКАТОР, МЕТОД ГОЛОВНИХ КОМПОНЕНТ, МЕТОД АПРОКСИМАЦІЇ ТА ПРОЕКЦІЇ РІВНОМІРНОГО РІЗНОМАНІТТЯ, RANDOM FOREST, МЕТОД ОПОРНИХ ВЕКТОРІВ.

ЗМІСТ

ВСТУП

У 2015 році світ почав працювати над новим глобальним порядком денним розвитку, прагнучі досягти до 2030 року нових цілей, викладених у Цілях сталого розвитку. Запропонована ціль щодо зниження дитячої смертності спрямована на те, щоб до 2030 року запобігти смертності новонароджених і дітей віком до 5 років, у випадках, які могли бути діагностовані заздалегідь. При цьому всі країни прагнуть скоротити неонатальну смертність принаймні до 12 смертей на 1000 народжених живими та смертність дітей віком до 5 років – принаймні до 25 смертей на 1000 живонароджених [1].

Найбільш уразливою групою населення в різних країнах є новонароджені діти, до яких належать немовлята або недоношені, народжені раніше 37 тижнів вагітності. Дитячі недуги постійно проявляються серед дітей віком до 5 років у вигляді смерті, інвалідності та погіршення здоров'я, особливо в країнах Африки на південь від Сахари та Південної Азії, де на передчасні пологи припадає понад 60% смертей новонароджених у всьому світі. Через ускладнення передчасних пологів понад 1 000 000 немовлят, як повідомляється, помирають із 15 000 000 новонароджених недоношених, а інші стають інвалідами та хворими після народження. Тільки в Бангладеш понад 26 100 дітей віком до 5 років помирають із 439 000 недоношених на рік через передчасні ускладнення зареєстрованих випадків [2]. Існує дуже нагальна потреба зменшити ризики цієї вразливої групи населення, щоб підвищити виживання та добробут новонароджених, матерів і дітей віком до 5 років [3].

З точки зору медичної діагностики кардіотокограми (КТГ) є простим і доступним варіантом оцінки здоров'я плода, що дозволяє медичним працівникам вжити заходів для запобігання дитячій і материнській смертності. Саме обладнання працює, посилаючи ультразвукові імпульси та

зчитуючи їх відповідь, таким чином це дає інформацію про частоту серцевих скорочень плода, рухи плода, скорочення матки тощо. КТГ широко використовується під час вагітності як метод оцінки стану плода, переважно при вагітності з підвищеним ризиком ускладнень [4].

Оцінка стану плода та подальші заходи для роботи з групою ризику є комплексною та міждисциплінарною задачею. Математичні методи мають широке застосування у медичній діагностиці, особливої ефективності набувають методи машинного навчання. Аналіз даних за допомогою методів машинного навчання дозволяє виявляти ранні ознаки можливих проблем з розвитком плода, що ускладнюють відновлення його здоров'я після народження.

Машинне навчання (ML) – це група методів у сфері штучного інтелекту, набір алгоритмів, які використовуються для створення системи, яка навчається на власному досвіді. Під час навчання обробляються величезні обсяги вхідних даних, щоб знайти в них залежності та шаблони. Однією з головних переваг ML порівняно з традиційними методами аналізу даних є те, що він дозволяє системі самостійно виявляти шаблони в даних і вчитися на основі цих шаблонів. Це означає, що система може розпізнавати складні зв’язки між різними факторами. Результатом цього навчання є модель, яка може передбачити здоров’я плоду на основі діагностичних даних [5].

Зменшення розмірності – це техніка, яка використовується для зменшення кількості функцій у наборі даних, зберігаючи якомога більше важливої інформації. Іншими словами, це процес перетворення високовимірних даних у низьковимірний простір, який все ще зберігає суть вихідних даних [6]. Зважаючи на складність та обсяги медичних даних, зниження розмірності даних допомагає виявити складні кореляції та патерни в даних, які можуть бути корисними для діагностики та лікування різних захворювань.

Метод класифікації – це метод аналізу даних, який дозволяє оцінити ймовірність приналежності екземплярів даних до деякого класу залежно від значень їх атрибутів. Класифікація медичних даних на основі методів машинного навчання допомагає створювати персоналізовані підходи до лікування, враховуючи індивідуальні характеристики та потреби пацієнтів.

Отже, вирішення задачі зниження розмірності даних та класифікація новонароджених за групами ризику в діагностичних цілях є важливим і актуальним.

Основна мета роботи: визначення та реалізація методів, які дозволяють провести зменшення розмірності ознак та класифікацію даних у медичних дослідженнях із застосуванням машинного навчання, а також візуалізація отриманих результатів. Для реалізації мети необхідно було вирішити такі задачі:

1) провести аналітичний огляд методів машинного навчання для зниження розмірності та класифікації даних;

2) розробити алгоритмічні моделі методів машинного навчання для вирішення задач зниження розмірності та класифікації даних;

3) розробити програмне забезпечення для аналізу та візуалізації даних на основі методів машинного навчання.

Об'єкт дослідження: процес аналізу даних.

Предмет дослідження: використання методів машинного навчання для зниження розмірності та класифікації даних.

Ключові слова: ЗМЕНШЕННЯ РОЗМІРНОСТІ, КЛАСИФІКАТОР, МЕТОД ГОЛОВНИХ КОМПОНЕНТ, МЕТОД АПРОКСИМАЦІЇ ТА ПРОЕКЦІЇ РІВНОМІРНОГО РІЗНОМАНІТТЯ, МЕТОД ВИПАДКОВИЙ ЛІС, МЕТОД ОПОРНИХ ВЕКТОРІВ.

1 АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ МАШИННОГО НАВЧАННЯ

1.1 Місце машинного навчання у Data Science

Наука про дані (Data science) – це сукупність фундаментальних принципів, які підтримують і спрямовують принципове вилучення інформації та знань з даних [7]. Вона перетинається з такими областями як машинне навчання і наука про мислення, а також з технологіями для роботи з великими даними. Результатом роботи Data science є проаналізовані дані і знаходження правильного підходу для подальшої обробки, сортування, вибірки, пошуку даних.

Штучний інтелект (Artificial intelligence, AI) – це різні технологічні та наукові рішення і методи, які допомагають зробити програми подібні до інтелекту людини. Штучний інтелект включає в себе безліч інструментів, алгоритмів і систем (рис. 1.1), серед яких також всі складові Data science і машинне навчання.

Рисунок 1.1 – Складові штучного інтелекту

Машинне навчання (Machine learning) – це загальний термін, який позначає широкий спектр алгоритмів, які виконують інтелектуальні прогнози на основі набору даних. Ці набори даних часто великі, можливо, складаються

з мільйонів унікальних точок даних [8]. Тобто, машина може знайти закономірність в складних і багатопараметричних завданнях (які мозок людини не може вирішити), таким чином знаходячи більш точні відповіді. Як результат – вірне прогнозування.

Поняття «машинне навчання» та «штучний інтелект» часто використовуються в одному контексті, іноді як взаємозамінні, однак вони мають різне значення. Різниця полягає в тому, що машинне навчання завжди має на увазі використання штучного інтелекту, однак штучний інтелект не завжди має на увазі машинне навчання. Ці інструменти багато в чому перетинаються, але все ж вони різні і кожен зі своїми завданнями (рис. 1.2).

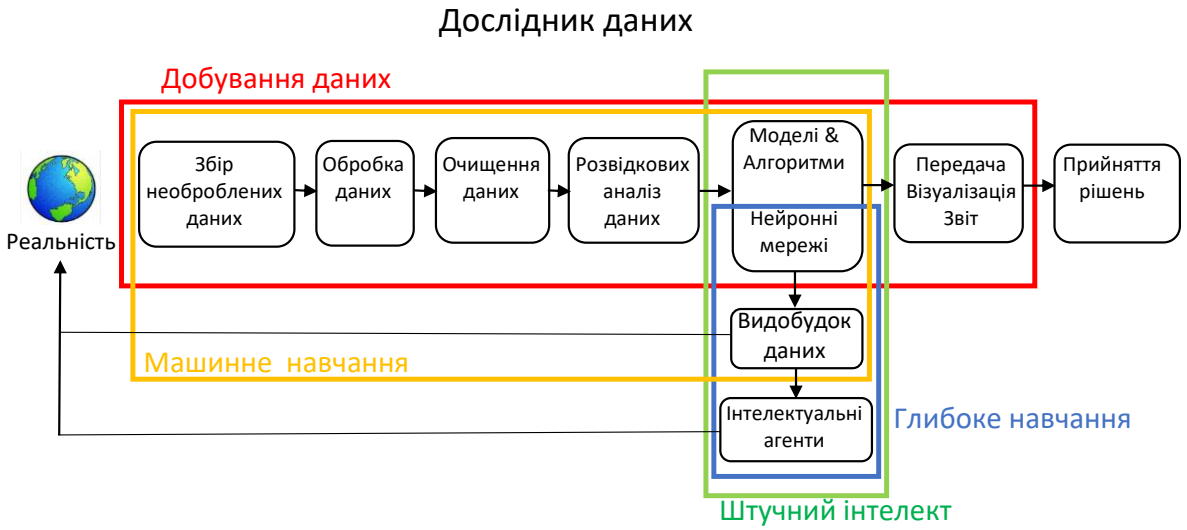


Рисунок 1.2 – Складові Data Science

Мета машинного навчання – частково або навіть повністю автоматизувати рішення різних складних аналітичних задач. Тому машинне навчання покликане давати максимально точні прогнози на підставі вхідних даних, щоб людина могла приймати правильні рішення в своїй роботі. В результаті навчання машина може передбачати результат, запам'ятовувати його, при необхідності відтворювати, вибирати кращий з декількох варіантів.

1.2 Вирішувані задачі та сфера застосування методів машинного навчання

Для кращого розуміння важливості методів машинного навчання та їх сфери застосування, були розглянуті деякі з найпоширеніших задач і галузей:

1.2.1 Регресія. Моделі регресійного аналізу зазвичай використовуються для демонстрації або прогнозування взаємозв'язку між процесом і його можливими впливами. Важливо зауважити, що виявлення кореляції не завжди вказує на причинно-наслідковий зв'язок. Навіть у випадку прямої залежності в простій лінійній регресії, яка чітко відображає відносини між даними, може відсутність конкретної відповіді щодо причинно-наслідкових зв'язків. З цього приводу регресійний аналіз не використовується для тлумачення причинно-наслідкових відносин між змінними. Однак цей аналіз може вказати на те, як і наскільки сильно змінні пов'язані між собою, в той час як визначення причин і наслідків вимагає більш глибоких досліджень, використовуючи інші алгоритми та методи.

1.2.2 Класифікація. Класифікація представляє собою процес створення моделі, яка дозволяє розподілити набір даних на різні категоріальні класи. У цьому процесі дані відносяться до різних класів залежно від певних параметрів, поданих на вході, та подальше передбачення міток для цих даних. Вихідною змінною у цьому випадку є категорія, тобто алгоритми класифікації дозволяють розділити об'єкти відповідно до попередньо визначених класів. На сьогодні алгоритми класифікації застосовуються у різноманітних завданнях, таких як визначення мови, фільтрація спаму, виявлення шахрайства (наприклад, коли зловмисник використовує вкрадені кошти для оплати послуг), пошук схожих документів та інші.

1.2.3 Кластеризація. Кластеризація є методом навчання, який застосовується для аналізу даних у різних областях. Алгоритми кластеризації

виявляють свою корисність, коли мета включає отримання детальної інформації про дані, особливо в ситуаціях, де класи заздалегідь не визначені, і потрібно сформувати кластери на основі схожості елементів. Застосування алгоритмів кластеризації розповсюджене в задачах, таких як сегментація ринку, аналіз нових даних та стиснення зображень. Алгоритми кластеризації визначають схожі ознаки між об'єктами і групують їх у кластери. Кількість кластерів, аналогічно класам, може визначатися дослідником або автоматично машиною. Дослідник може також вказати ознаки, за якими слід розділити вибірку, або це може бути здійснено автоматично машинним алгоритмом.

1.2.4 Прогнозування. Прогностичних моделей існує безліч, і всі вони вирішують завдання передбачення часових рядів – знаходження майбутніх значень залежно від часу. Прогнозування використовують у медичній діагностиці, кредитному скорінгу, передбаченні попиту, під час прийняття рішень на фінансових ринках.

1.2.5 Зменшення розмірності. Зменшення розмірності є процесом, в якому більша кількість ознак зводиться до меншої для полегшення їх подальшого використання. У великих наборах даних машина визначає закономірності, вибираючи лише ті ознаки, які мають значення. Цей процес подібний до стиснення файлу, що передбачає вилучення зайвої інформації, що не має значення. Зменшення розмірності спрощує структуру даних, зменшує їхню складність і дозволяє зберегти лише суттєву інформацію. Цей підхід широко використовується у побудові рекомендаційних систем, пошуку схожих документів або для візуалізації даних.

1.2.6 Виявлення аномалій. Аномалії – це експериментальні дані, які суттєво відрізняються від інших (рис. 1.3). Вони виникають як наслідок збою вимірювальних приладів, описок операторів, помилкової конвертації даних тощо. Наприклад, оператор поставив кому не в тому розряді десяткового числа або вказав ціну в доларах, замість гривень.

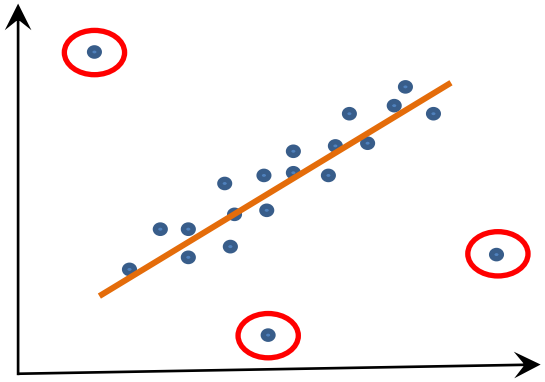


Рис. 1.3 – Виявлення відхилень від стандартних значень

Мета цього типу – виявити аномальні відхилення від стандартних випадків.

1.2.7 Пошук правил. Завдання пошуку правил шукає закономірності в потоці даних. На ці питання може відповісти машинне навчання, коли алгоритм проаналізує інших користувачів, їхній досвід та поведінку.

1.3 Машинне навчання в медицині

Машинне навчання відіграє ключову роль у розвитку медичних технологій, забезпечуючи прогнозування захворювань, оптимізацію планування лікування, створення персоналізованих терапій та багато іншого. У медичних діагностичних системах машинне навчання може використовуватися для аналізу обширних обсягів медичних даних, таких як медичні зображення, генетичні дані, результати лабораторних тестів та історії пацієнтів. Це дозволяє виявляти шаблони та тенденції, що сприяють удосконаленню діагностики та прогнозуванню захворювань.

Один із прикладів використання машинного навчання у медицині – це аналіз медичних зображень, таких як рентгенівські або МРТ-знімки, для виявлення патологій, таких як онкологія чи серцеві захворювання. Ці

алгоритми можуть розпізнавати шаблони в зображеннях, непомітні для людського ока, що дозволяє раннє виявлення та лікування.

Машинне навчання також може служити для прогнозування ходу хвороби та реакції на лікування. Наприклад, алгоритми машинного навчання можуть аналізувати генетичні дані та інформацію про спосіб життя, передбачаючи ризик розвитку конкретних захворювань, таких як 2-типовий діабет чи серцеві захворювання. Це полегшує розробку індивідуальних планів лікування та стратегій профілактики для лікарів.

Загалом, машинне навчання грає важливу роль у сучасній медицині, допомагаючи лікарям та дослідникам краще розуміти хвороби, виявляти їх на ранніх етапах, розробляти ефективні стратегії лікування та підвищувати загальний рівень догляду за пацієнтами. Це відкриває нові перспективи для персоналізованої медицини, де лікування адаптується до індивідуальних потреб кожного пацієнта на підставі їх унікального генетичного профілю, медичної історії та інших важливих факторів.

Однак, не дивлячись на великий потенціал машинного навчання в медицині, існують і значні виклики. Це включає в себе необхідність великих наборів даних для навчання ефективних моделей, проблеми з приватністю та безпекою даних, а також необхідність інтерпретації та перевірки результатів моделей машинного навчання. Незважаючи на ці виклики, переваги машинного навчання в медицині роблять його важливим інструментом для покращення догляду за пацієнтами та просування медичних досліджень [9].

1.4 Методи машинного навчання для вирішення задачі класифікації даних

Методи та алгоритми (рис. 1.4) машинного навчання мають широку популярність і стали важливим інструментом для вирішення різноманітних

завдань у різних галузях. Їх успіх пояснюється здатністю виявляти складні залежності в наборах даних та автоматично адаптувати моделі до конкретних умов.

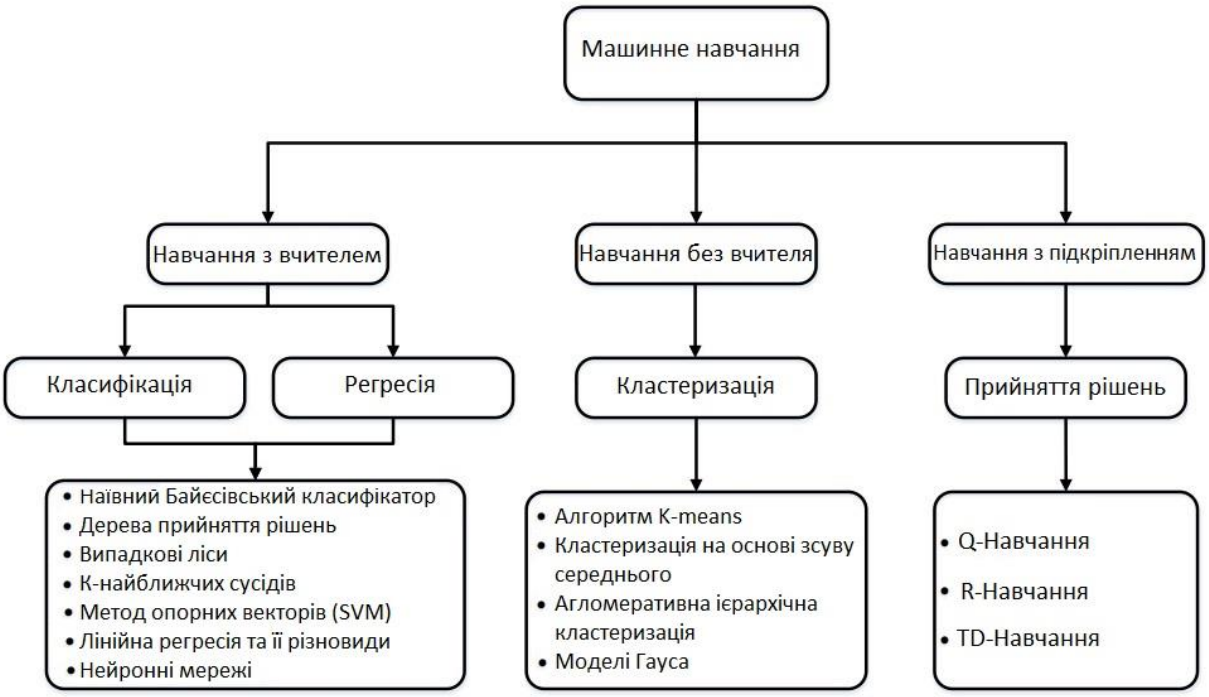


Рисунок 1.4 – Застосування методів і алгоритмів машинного навчання

Для кращого розуміння класифікації машинного навчання, були розглянуті деякі з найпоширеніших методів – випадковий ліс, метод опорних векторів.

1.4.1 Випадковий ліс. Випадковий ліс – алгоритм машинного навчання. Він може бути використаний як для задач класифікації, так і для задач регресії. Він базується на концепції ансамблевого навчання, яке являє собою процес об’єднання кількох класифікаторів для вирішення складної проблеми та покращення продуктивності моделі [10].

У 1995 році Тін Кам Хо вперше висунув загальний метод випадкових лісів рішень, вказавши, що точність лісу дерев, розбитих похилими гіперплощинами, може зростати без перенавчання, якщо ліси обмежуються

для чутливості лише до обраних розмірів функцій. Його висновок, що інші методи розбиття можуть вести схожим чином, якщо вони випадково стають нечутливими до певних розмірів ознак, суперечить ідеї про те, що складність класифікатора може зрости лише до певного рівня точності перед перенавчанням. Стійкість методу лісу до перенавчання пояснюється теорією стохастичного розрізнення Клейнберга. Початковий розвиток ідеї випадкових лісів Бреймана знаходився під впливом роботою Аміта та Джемана, які пропонували випадковий вибір підмножини рішень під час розбиття вузла при вирощуванні одного дерева. Також важливий внесок у метод полягає у випадковому виборі підпростору у навчальних даних перед підгонкою кожного дерева або вузла. Нарешті, концепцію рандомізованої оптимізації вузла, де рішення обирається рандомізовано, а не детерміновано, вперше висунув Томас Г. Дітеріх. Випадкові ліси часто використовуються як моделі «чорної скриньки» в бізнесі, оскільки вони генерують обґрунтовані прогнози для широкого діапазону даних, вимагаючи невеликої конфігурації.

Метод випадкових лісів працює за рахунок невеликого збільшення зміщення та деякої втрати інтерпретації, але загалом значно підвищує продуктивність кінцевої моделі. Ліси схожі на об’єднання зусиль алгоритму дерева рішень. Спільна робота багатьох дерев покращує продуктивність одного випадкового дерева. Хоча ліси не дуже схожі, вони дають ефект k-кратної перехресної перевірки.

У випадкових лісах кожне дерево в ансамблі будується із вибірки, намальованої із заміною (тобто за допомогою bootstrap) з навчального набору. Схему роботи можна побачити на рисунку 1.5.

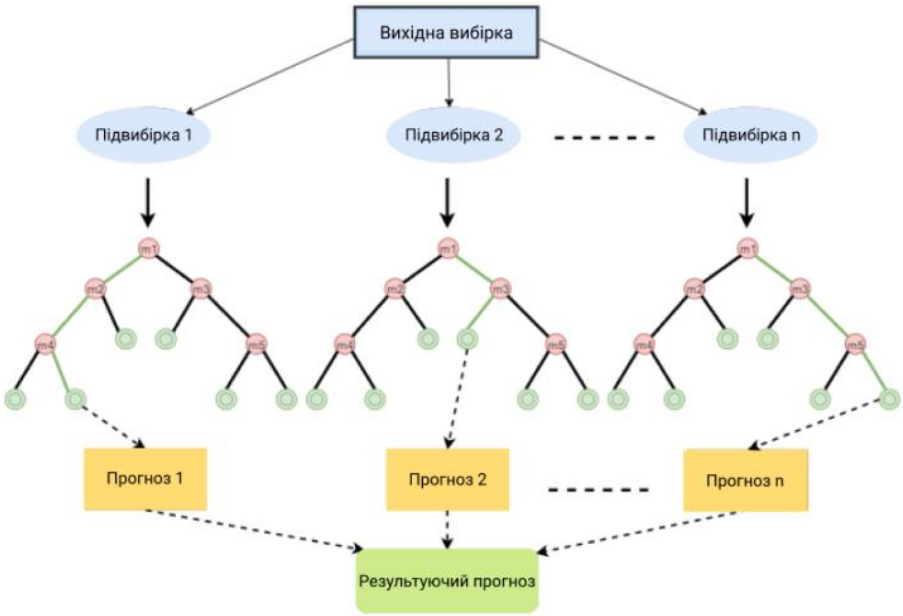


Рисунок 1.5 – Схема роботи методу випадковий ліс

Переваги:

– здатен вирішувати як класифікаційні, так і регресійні завдання;

– ефективно обробляє великі набори даних з високими розмірами;

– підвищує точність моделі та запобігає проблемам надмірної комплектації.

Недоліки:

– обчислювальна витратність;

– важкість інтерпретацій;

– велика кількість дерев може привести до перенавчання.

1.4.2 Метод опорних векторів. Підтримка векторної машини (SVM) – це алгоритм машинного навчання з контрольованим навчанням, який можна використовувати як для класифікації, так і для регресійних завдань. Однак в основному він використовується в задачах класифікації, таких як класифікація тексту [11].

Метою алгоритму SVM є створення найкращої лінії або межі рішення, яка може розділити n-вимірний простір на класи, щоб ми могли легко

поставити нову точку даних у правильну категорію в майбутньому. Ця межа найкращого рішення називається гіперплощиною.

SVM вибирає крайні точки/вектори, які допомагають у створенні гіперплощини. Ці крайні випадки називаються векторами підтримки, і тому алгоритм називається машиною підтримки вектора. На рисунку 1.6 є дві різні категорії, які класифікуються за кордоном прийняття рішення або гіперплощиною.

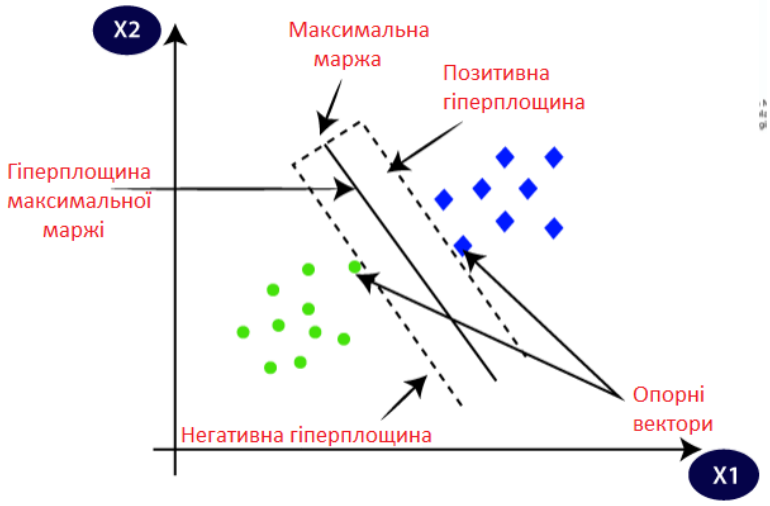


Рисунок 1.6 – Діаграма SVM

Алгоритм SVM може бути використаний для виявлення облич, класифікації зображень, категоризації тексту тощо.

SVM може бути двох типів. Лінійний SVM використовується для лінійно роздільних даних, що означає, що якщо набір даних можна класифікувати у два класи за допомогою однієї прямої лінії, то такі дані називаються лінійно розділимими даними, а класифікатор використовується як лінійний SVM – класифікатор. Нелінійний SVM використовується для нелінійно розділених даних, що означає, що якщо набір даних не можна класифікувати за допомогою прямої лінії, то такі дані називаються

нелінійними даними, а класифікатор, що використовується, називається не лінійним класифікатором SVM.

Гіперплощина SVM є оптимальною межею прийняття рішень для відокремлення класів у n-вимірному просторі. Її розміри залежать від об'єктів у наборі даних, наприклад, при двох об'єктах гіперплощина буде прямою, а при трьох ознаках – двовимірною площиною. Методом SVM створюється гіперплощина з максимальним запасом, тобто максимальною відстанню між точками даних. Опорні вектори, що розташовані найближче до гіперплощини, визначають її положення і підтримують, названі векторами підтримки.

Переваги класичного методу SVM:

– ефективність в високорозмірних просторах;
– ефективність при невеликій вибірці даних;
– універсальність;
– ефективність у випадку лінійно нероздільних даних;
– відсутність локальних мінімумів;

Недоліки класичного методу SVM:

– високі вимоги до обичслювальних ресурсів;
– неефективність при великій кількості прикладів;
– чутливість до величини та розподілу ознак;
– ефективність у випадку лінійно нероздільних даних;
– вибір правильного ядра може бути нетривіальним завданням.

1.5 Методи зменшення розмірності

Останнім часом у світі з’являється величезна кількість даних у різних сферах застосування, що призводить до експоненціального зростання складності, неоднорідності, розмірності та розміру даних [12]. У цю еру інформаційних комунікацій та технологій (ІКТ) такі галузі, як освіта,

медицина, інтернет, соціальні мережі та бізнес, переповнені величезною кількістю даних. Існує постійний розвиток даних у різних формах, таких як цифрові зображення, відео, текстові та мовні сигнали.

Існуючі класичні статистичні методології, на які покладалися, були розроблені в епоху, коли збір даних був непростим, як зараз, і обсяг наборів даних був набагато меншим. Таким чином, існує проблема аналізу цих великих і складних наборів даних, які вимагають більш складного статистичного та обчислювального способу аналізу таких даних. Як наслідок, сфера машинного навчання швидко розвивалася, щоб допомогти вирішити цю проблему. Використовуються штучний інтелект і автоматичне навчання даних. Основна увага приділяється комп'ютерним програмам для доступу до даних, і використання їх для самостійного навчання.

У сфері машинного навчання загально вважається, що зі збільшенням кількості функцій поліпшується прогностична здатність, однак це не завжди вірно. Перевищення певного порогу кількості функцій може призвести до зменшення продуктивності алгоритму машинного навчання (рис. 1.7).

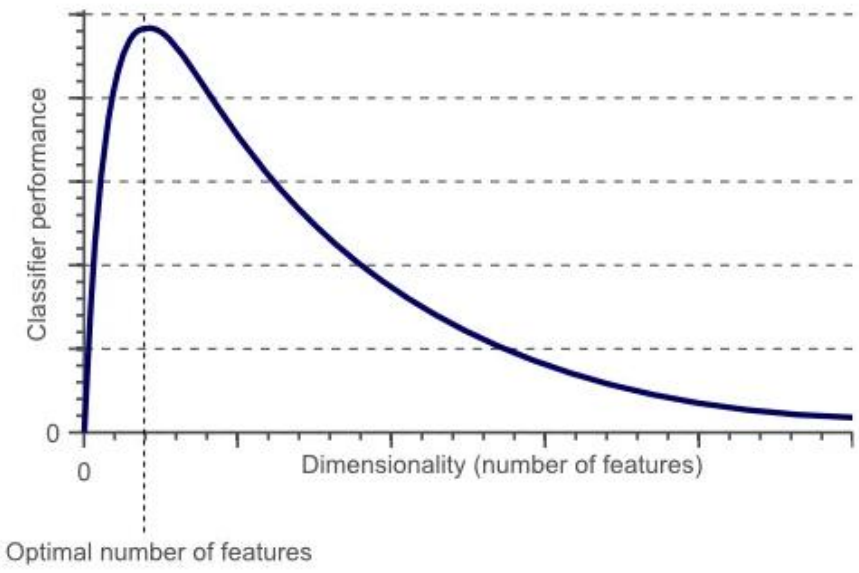


Рисунок 1.7 – Тенденція до зниження

При збільшенні кількості вимірів кожна функція отримує ширший діапазон можливих значень, що вимагає збільшення обсягу даних, щоб врахувати різноманіття комбінацій значень. Зі збільшенням розмірів кількість даних, необхідних для точного прогнозу, зростає логарифмічно. У реальних наборах даних, де може бути тисячі функцій, необхідність у великій кількості даних для врахування зростаючого розміру може виявитися великою. Це значно збільшує час навчання. З іншого боку, якщо розміри зростають, а кількість зразків залишається невеликою, алгоритми машинного навчання можуть стати схильними до перенавчання, що призводить до зниження їх продуктивності.

У моделюванні машинного навчання висока розмірність даних може викликати проблеми з точністю класифікації, розпізнавання образів і візуалізації. Одна з таких ідей, яка описує поведінку даних у просторі великої розмірності та проблеми, які виникають під час аналізу даних великої розмірності, називається прокляттям розмірності, і може призвести до перенавчання [13]. Це все не тільки тягне багато проблем, а також ускладнює пошук оптимального рішення. Але є можливість звести складну задачу до оптимальної, якщо зменшити кількість функцій за допомогою методів пониження розмірності.

Зменшення розмірності – це термінологія, яка використовується, коли дані з величезними розмірами зменшуються до менших розмірів, але гарантують, що вони стисло передають схожу інформацію. Методи зменшення розмірності зазвичай використовуються для вирішення проблем машинного навчання на етапі попередньої обробки, щоб отримати кращі характеристики для завдання класифікації або регресії. За останні кілька років алгоритми зменшення розмірності викликали великий інтерес. Перед застосуванням моделей ML методи зменшення розмірності забезпечують надійний і також ефективний спосіб зменшити кількість розмірів. Деякі

методи можуть підходити для певного типу даних, але не підходити для інших типів даних.

Окрім видалення шумів та зайвої інформації, а також прискорення процесу навчання, методи зменшення розмірності виявляються дуже корисними для візуалізації даних. Зведення простору ознак до двох або трьох вимірювань дозволяє представити багатовимірний навчальний набір на графіку, що сприяє візуальному виявленню важливих ідей та шаблонів, таких як кластери (рис 1.8).

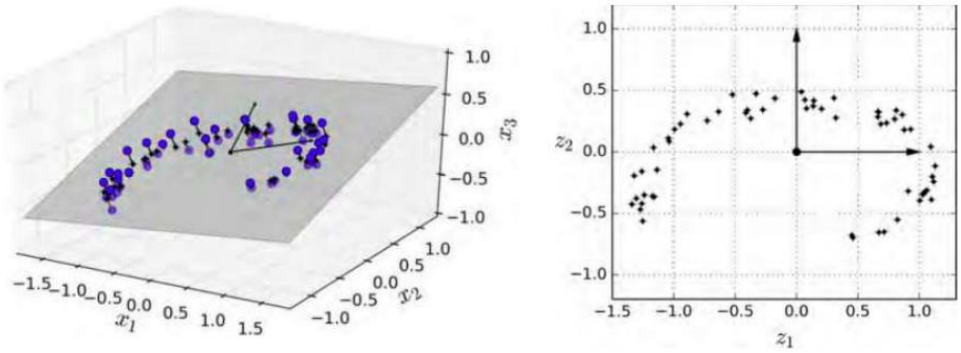


Рисунок 1.8 – Проеціювання простору 3d на 2d

Методи зменшення розмірності, такі як «PCA», «t-SNE» , «LLE», «Isomap», «UMAP» тощо, є важливими для візуалізації та розуміння відстані та схожості між точками у багатовимірних даних. Ці методи дозволяють ефективно перетворювати багатовимірні дані в простори меншої розмірності, зберігаючи ключові характеристики та структурну інформацію. Візуалізація отриманих результатів полегшує інтуїтивне сприйняття взаємозв'язків у даних, що стає корисним для аналізу та прийняття рішень. Зменшення розмірності широко використовується в інтелектуальному аналізі даних, машинному навчанні та обробці зображень. Наприклад, в обробці зображень це полегшує аналіз високорозмірних зображень, а в машинному навчанні допомагає в розумінні та аналізі розподілу даних для вибору відповідних

моделей та алгоритмів. Застосування рекомендацій в алгоритмах допомагає вирішувати завдання персоналізованого рекомендування, використовуючи відповідні алгоритми для зменшення розмірності даних та оптимізації вибору.

Найбільш поширені приклади та добре відомі методи зменшення розмірності: метод головних компонентів (PCA), апроксимація та проекція однорідного різноманіття (UMAP).

1.5.1 Метод головних компонентів. Метод аналізу основних компонент (PCA) – це ефективний спосіб зменшення розмірності великих наборів даних, шляхом перетворення великого набору змінних у менший, який все ще зберігає значну частину інформації [14]. Аналіз головних компонент є одним з найпопулярніших методів зниження розмірності даних. Зменшення кількості змінних у наборі даних часто супроводжується деякою втратою точності, але важливою властивістю є можливість отримання менших, більш простих даних для дослідження та візуалізації. Аналіз даних стає легшим та швидшим для алгоритмів машинного навчання, оскільки зменшені набори даних дозволяють відокремити ключові характеристики.

В основі алгоритму PCA лежить вибір гіперплощини, яка найкращим чином відображає розподіл даних. Для цього обирається вісь, на якій проекції точок дають максимальну дисперсію, що показує їхню велику відстань одна від одної. Важливою метою є вибір вісі, яка зберігає найбільше дисперсії, тобто максимізує кількість збереженої інформації. Цей вибір також може обґрунтовуватися мінімізацією середньоквадратичної відстані між вихідним набором даних та його проекцією на обрану вісь [14].

PCA визначає вісь, на яку припадає найбільша кількість дисперсії в навчальному наборі. Також PCA визначає другу вісь, ортогональну до першої, на яку припадає найбільша кількість дисперсії, що залишилася. На наведеному нижче графіку можна побачити, що червоний вектор дає напрямок

максимальної дисперсії. Зелені точки – вихідні точки даних, а червоні – проекції цих точок на вісь (рис. 1.9).

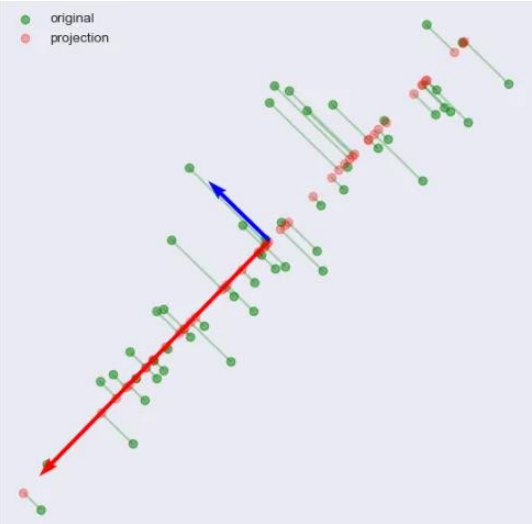


Рисунок 1.9 – Напрямок максимальної дисперсії

1.5.2 Апроксимація та проекція рівномірного разноманіття. Апроксимація та проекція рівномірного разноманіття (Uniform Manifold Approximation and Projection, UMAP) – нова техніка багатоманітного навчання для зменшення розмірності. UMAP побудовано на основі теоретичної основи, заснованої на рімановій геометрії та алгебраїчній топології. Результатом є практичний масштабований алгоритм, який застосовується до реальних даних.

Алгоритм UMAP конкурує з t-SNE за якістю візуалізації та, можливо, зберігає більшу частину глобальної структури з чудовою продуктивністю під час виконання. Крім того, UMAP не має обчислювальних обмежень щодо розмірності вбудовування, що робить його життєздатним як техніку зменшення розмірності загального призначення для машинного навчання [15]. У найпростішому сенсі UMAP створює багатовимірне графічне представлення даних, а потім оптимізує низьковимірний графік, щоб він був максимально подібним за структурою. Хоча математика, яку UMAP використовує для

побудови багатовимірного графіка, є передовою, інтуїція за нею надзвичайно проста.

Щоб побудувати початковий високовимірний граф, UMAP створює так званий «нечіткий симпліціальний комплекс». Насправді це представлення зваженого графіка, де ваги ребер представляють ймовірність того, що дві точки з'єднані. Щоб визначити зв'язок, UMAP розширює радіус назовні від кожної точки, з'єднуючи точки, коли ці радіуси перекриваються. Вибір цього радіуса має вирішальне значення – занадто малий вибір призведе до малих, ізольованих кластерів, тоді як занадто великий вибір об'єднає все разом. UMAP долає цю проблему, вибираючи радіус локально на основі відстані до *n*-го найближчого сусіда кожної точки. Тоді UMAP робить графік «нечітким», зменшуючи ймовірність з'єднання зі збільшенням радіуса. Нарешті, передбачаючи, що кожна точка має бути з'єднана принаймні зі своїм найближчим сусідом, UMAP гарантує, що локальна структура збережеться в балансі з глобальною структурою.

Алгоритм ґрунтується на трьох припущеннях про дані:

1. Дані поступово розподілені на рімановому багатовиді;

2. Риманова метрика локально стала, або може бути зведена;

3. Багатовид локально складний.

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

При зниженні вимірності даних, будується зважений граф, з'єднуючи ребрами тільки ті об'єкти, які знаходяться поруч, і є найближчими сусідами. Множина ребер графу – це нечітка множина з функцією приналежності, вона визначається як ймовірність існування ребра між двома вершинами [15]. Далі алгоритм створює новий граф в низьковимірному просторі, й топологічно

наближає його, до початкового. Після того як високовимірний граф побудовано, UMAP оптимізує макет низьковимірного аналога, щоб бути максимально схожим. Цей тип техніки називається нелінійним, оскільки він може вивчати складні «вигнуті» форми (рис. 1.10).

Це краще видно у випадку 3D-2D. Зведений вектор також називають вкладенням першого.

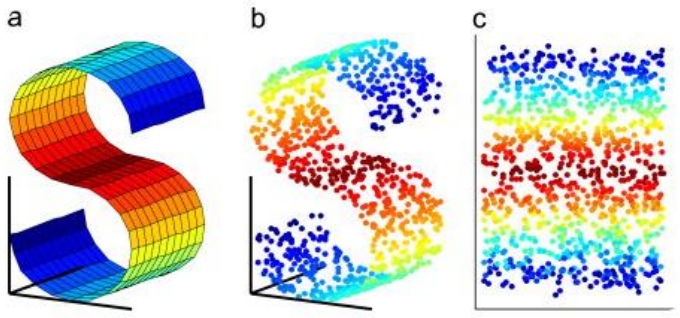


Рисунок 1.10 – Нелінійна техніка побудування графа

Ключ до ефективного використання UMAP полягає в розумінні побудови початкового високовимірного графа. Хоча ідеї, що лежать в основі процесу, дуже інтуїтивно зрозумілі, алгоритм спирається на деякі передові математики, щоб дати сильні теоретичні гарантії щодо того, наскільки добре цей графік насправді представляє дані.

Висновки до розділу 1

В цьому розділі було визначено поняття машинного навчання, розглянуті основні задачі, сфери застосування та методи класифікації даних. Визначено поняття зменшення розмірності та класифікації. Розглянутий лінейний метод головних компонент, нелінейний метод апроксимації та проекції однорідного різноманіття, метод випадковий ліс, метод опорних векторів.

2 АЛГОРИТМІЧНІ МОДЕЛІ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ВИРІШЕННЯ ЗАДАЧ ЗНИЖЕННЯ РОЗМІРНОСТІ ТА КЛАСИФІКАЦІЇ ДАНИХ

2.1 Алгоритмічна модель для зниження розмірності даних на основі методу головних компонент

Метою методу головних компонент (МГК) є виділення потрібної інформації з набору даних для вирішення конкретної задачі. Ця інформація може бути як міститись в самому наборі даних, так і бути відсутня. У випадку, коли дані є надмірними та можуть містити шум, як розрізняти, що є шумом, а що є інформацією, залежить від поставленої задачі та використаних методів. На практиці немає такого поняття, як зайві дані, і чим більше даних, тим краще.

Шум та надмірність в даних виявляються через кореляційні зв'язки між змінними. Для забезпечення адекватності реального процесу замінюють множину ознак на меншу кількість величин (компонентів), які є некорельованими і зберігають всю інформацію про процес, при цьому не впливаючи на точність результатів аналізу. Усі змінні при цьому враховуються, а незначна частина даних відокремлюється та перетворюється в шум [16].

Суть методу полягає у перетворенні вихідного набору даних на простір меншої розмірності таким чином, щоб виконувались такі умови:

– мінімізація суми квадратів відстаней від точок даних до їх проекцій на площину перших головних компонентів, що означає, що екран розташований максимально близько до хмари точок;

– мінімізація суми спотворень квадратів відстаней між усіма парами точок у хмарі даних після їх проектування на площину;

– мінімізація суми спотворень квадратів відстаней між усіма точками даних та їх «центром тяжіння».

На рисунку 2.1 вектори *v1* та *v2* є головними компонентами (ортогональні проекції), на які відображається вхідний набір даних.

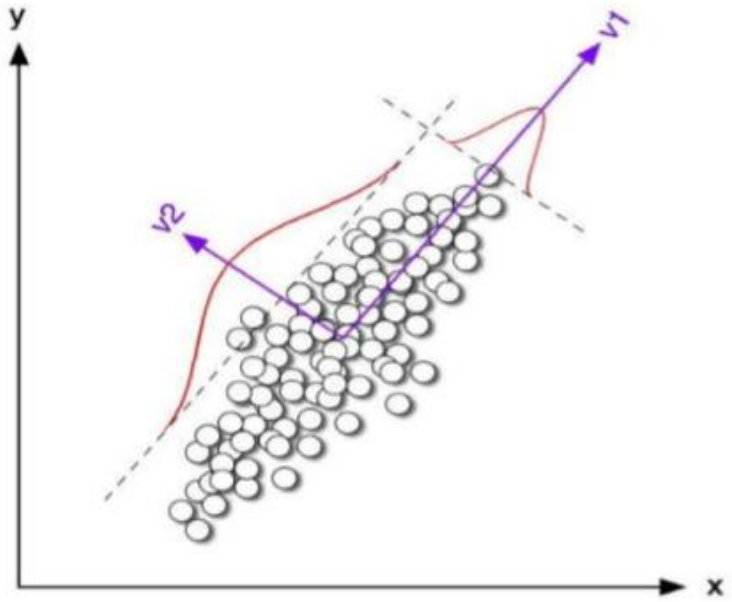


Рисунок 2.1 – Візуалізація методу

Основні компоненти визначаються шляхом обчислення власних векторів та власних значень коваріаційної матриці, яка формується на основі вхідної матриці даних. Власні вектори з найбільшими власними значеннями відображають напрямки максимальної дисперсії, тоді як власний вектор з найменшим власним значенням вказує напрямок найменшої дисперсії.

Метод прагне виділити осі, де максимальна кількість інформації, та перейти до цих осей від вихідної системи координат. Хоча певна частина інформації може бути втрачена, розмірність зменшується. Основний математичний підхід для визначення основних осей полягає в знаходженні власних чисел і власних векторів коваріаційної матриці.

Сума власних чисел дорівнює загальній кількості змінних, а їхній детермінант – детермінанту кореляційної матриці. Кожне власне число представляє дисперсію вздовж відповідної осі, при чому перше власне число є найбільшим, а наступні зменшуються в порядку важливості.

Доля дисперсії, яка припадає на кожну компоненту, легко єхвизначається, розділивши власне число на загальну кількість змінних.

Коефіцієнти навантаження для основних компонент визначаються розподілом коефіцієнтів власних векторів на квадратний корінь відповідних власних чисел.

Розв'язування задач методом головних компонент зводиться до поетапного перетворення матриці вихідних даних X – прямокутної таблиці чисел розмірністю m рядків і n стовпців:

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix}.$$

В більшості випадків стовпці – це змінні (ознаки). Вони нумеруються індексом j $(j = \overline{1,n})$, а рядки – їх реалізації. Вони нумеруються індексом i $(i = \overline{1,m})$.

На рисунку 2.2 відображена геометрична суть методу головних компонент для випадку, коли є тільки дві змінні x_1 і x_2 (рисунок 2.2). Такі дані легко зобразити на площині.

Наведені дані відображені на діаграмі розсіювання (рисунок 2.2, частина а). Пряма проведена через точки діаграми так, що вздовж неї відбувається максимальна зміна даних. Ця пряма відома як перша головна компонента (PC1). Далі відображається проекція всіх вихідних точок на цю пряму (це показано для трьох точок). Якщо припустити, що всі

експериментальні точки насправді мали б лежати на цій новій осі, тоді всі відхилення від нової осі можна вважати шумом.

Щоб визначити, чи це дійсно є шумом чи все ще є важливою частиною даних, необхідно знайти ось максимальних змін в них. Для цього обчислюється перпендикуляр ортогональної лінії до PC1 (рисунок 2.2, частина б). Ця лінія є другою головною компонентою (PC2). І так треба продовжувати, доки шум не стане дійсно випадковим хаотичним набором величин. У випадку багатовимірних даних, процес виділення головних компонент відбувається аналогічно.

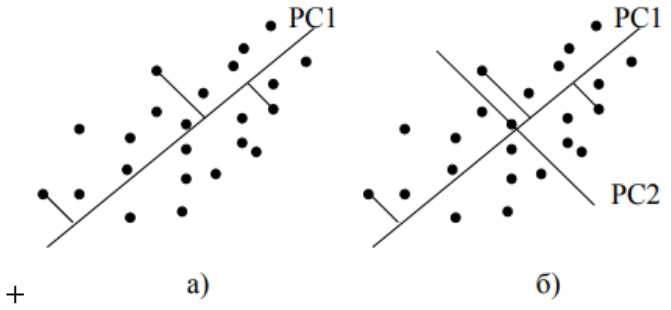


Рисунок 2.2 – Геометрична суть методу головних компонент

Після використання методу головних компонент для матриці *X*, її розмірність значно зменшується. Початкова матриця *X* замінюється двома новими матрицями *T* і *P* (рисунок 2.3), розмірність яких, позначена як *A*, є менше, ніж кількість змінних (стовпців) *n* у вихідній матриці *X*. При правильній декомпозиції, коли розмірність *A* вибрана правильно, матриця *T* зберігає в собі стільки ж інформації, скільки було на початку у матриці *X*. При цьому матриця *T* є меншою і, отже, простішою, ніж *X*.

В МГК використовуються нові змінні $t_a = p_{a1}x_1 + \ldots + p_{an}x_n \ \left(a = \overline{1,A}\right)$, що є лінійною комбінацією вихідних змінних $x_j \ (j = \overline{1,n})$.

Матриця *X* розкладається на добуток двох матриць *T* і *P*:

$$X = TP^1 + E = \sum_{a=1}^{A} t_a p_a^t + E, \tag{2.1}$$

де T – матриця ваг, її розмірність – (mxn), P – матриця навантажень, її розмірність – (nxA), E – матриця залишків, її розмірність – (mxn).

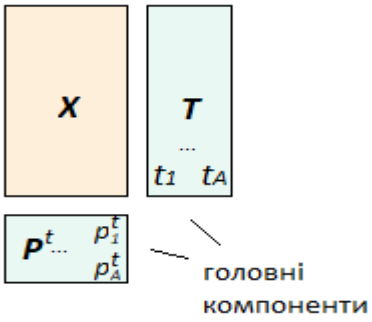


Рисунок 2.3 – Декомпозиція матриці X

Змінні t_a називаються головними компонентами, число A – числом головних компонент.

На рисунку 2.4 приведено блок-схему алгоритмічної моделі.

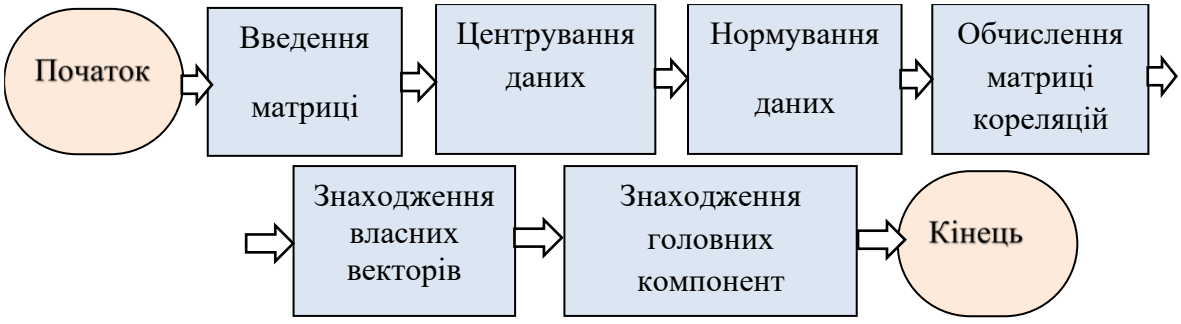


Рисунок 2.4 – Схема алгоритму методу головних компонент

Кроки реалізації методу:

Крок 1: Центрування і нормування вихідних даних за формулою:

$$\frac{x_{ij} - \overline{x_j}}{\sigma_j} \qquad (2.2)$$

де j – номер вихідної змінної, i – номер реалізації j-ої змінної.

Цей крок є підготовчим і обов'язковим в тому випадку, коли різні одиниці вимірювання, або великий розкид значень у вихідних даних.

Крок 2: Обчислення матриці коваріацій (S) або кореляцій (R).

Крок 3: Знаходження власних чисел $\lambda_1, \lambda_1, ..., \lambda_p$ матриці S (або R) з характеристичного рівняння

$$\det(S - \lambda E) = 0 \text{ або } \det(R - \lambda E) = 0, \qquad (2.3)$$

де E – одинична матриця (квадратна матриця на діагоналі якої стоять одиниці, решта елементів – нулі).

Крок 4: Знаходження для кожного власного числа λ_j власного вектора. Власний вектор – це розв'язок системи рівнянь:

$$(S - \lambda E) \cdot \vec{w} = 0, \text{ або } (R - \lambda E) \cdot \vec{w} = 0, \qquad (2.4)$$

де $\vec{w}$ – власний вектор.

Крок 5: Знаходження лінійних комбінацій для головних компонент y_j:

$$y_j = w_{1j} x_1 + w_{2j} x_2 + ... + w_{pj} x_p \qquad (2.5)$$

Метод головних компонент має кілька переваг і деякі недоліки. По-перше, головні компоненти дозволяють візуалізувати складний набір даних і

виділити найбільш інформативні змінні. По-друге, головні компоненти є некорельованими змінними, що робить метод ефективним для боротьби з мультиколінеарністю. По-третє, використання головних компонент дозволяє виявити особливі спостереження. По-четверте, цей метод часто застосовується для вирішення проблеми великої кількості змінних [17].

Негативною стороною методу головних компонент є складність математичного апарату, яка обумовлена вимогами до розуміння теорії ймовірностей, математичної статистики та лінійної алгебри. Недостатнє розуміння математичних аспектів може призвести до некоректних висновків.

2.2 Алгоритмічна модель для зниження розмірності даних на основі методу апроксимації та проекції рівномірного різноманіття

Метод апроксимації та проекції рівномірного різноманіття (UMAP) представляє собою метод зменшення розмірності, який застосовується для візуалізації та нелінійного зменшення розмірності, аналогічно t-SNE [15]. Цей алгоритм ґрунтується на трьох ключових припущеннях щодо даних:

1. Поступове розподілення даних на рімановому багатовиді;
2. Локальна сталість риманової метрики або її можливість зведення;
3. Локальна складність багатовиду.

Виходячи з цих припущень, UMAP моделює багатовид з нечіткою топологічною структурою. Основною метою алгоритму є знаходження низьковимірної проекції даних, яка максимально точно відображає нечітку топологічну структуру, відповідну вихідним даним.

На рисунку 2.5 зображена блок-схема алгоритмічної моделі.

При зниженні вимірності даних формується зважений граф, в якому ребрами з'єднуються лише ті об'єкти, які розташовані поруч і є найближчими сусідами. Множина ребер графу представляє собою нечітку множину з

функцією приналежності, що визначає ймовірність існування ребра між двома вершинами [15].

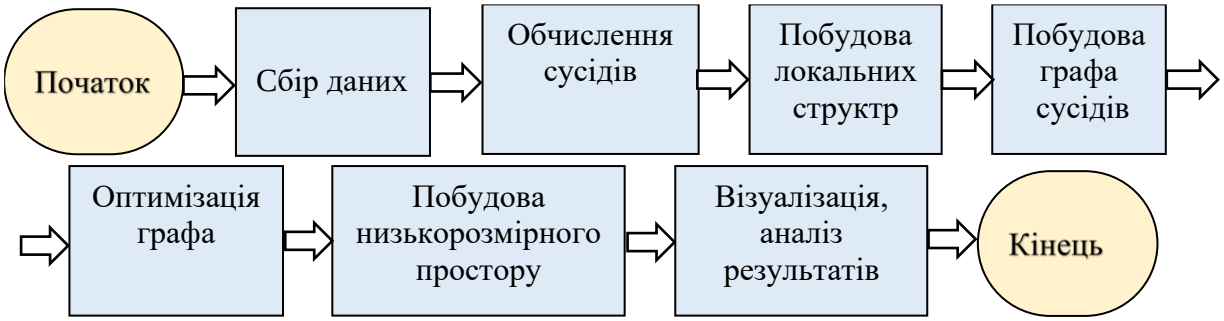


Рисунок 2.5 – Блок-схема алгоритму методу UMAP

Далі алгоритм створює новий граф у низьковимірному просторі і топологічно наближає його до вихідного. На етапі початку алгоритму введені вихідні дані, і за використанням певної метрики будується множина найближчих сусідів. Множина вхідних даних з n об’єктів $X = \{x_1, x_2, ..., x_n\}$ із метрікою $d: X \times X \to R \geq 0$. Враховуючи вхідний гіперпараметр k для кожного x_i ми обчислюємо множину $\{x_{i_1}, ..., x_{i_k}\}$ з k найближчих сусідів x_i за метрикою d. Після цього для кожного об’єкту x_i, вираховується відстань до найближчого сусіда:

$$p_i = \min\{d(x_i, x_{i_j}) | 1 \leq j \leq k, d(x_i, x_{i_j}) > 0\} \tag{2.6}$$

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

$$\sum_{j=1}^{k} \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - p_i}{\sigma_i}\right) = \log_2 k \tag{2.7}$$

Далі здійснюється побудова сполучного дерева. Для цього необхідно визначити зважений орієнтований граф $G^{-} = (V, E, w)$. Вершини V графа G^{-} утворюють множину X. Тоді ми можемо сформувати безліч спрямованих ребер $E = \left\{ \left(x_i, x_{i_j} \right) | 1 \le j \le k, 1 \le i \le N \right\}$, і знайти вагу ребра:

$$v_{j|i} = \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i}{\sigma_i} \right) \tag{2.8}$$

Зазначено, що між вершинами може існувати кілька ребер із різною вагою. Вага ребра інтерпретується як ймовірність існування даного ребра, спрямованого від одного об'єкта до іншого. Виходячи з цього, ребра між двома вершинами об'єднуються в одне з вагою, що дорівнює ймовірності існування хоча б одного ребра:

$$w(x_i, x_j) = w(x_i \to x_j) + w(x_j \to x_i) - w(x_i \to x_j) \cdot w(x_j \to x_i) \tag{2.9}$$

Таким чином, алгоритм отримує зважений не орієнтований граф. Основне завдання пошуку низькорозмірного представлення це обчислення нечіткого топологічного уявлення. У UMAP сила притягнення між двома вершинами i та j з оординатами y_i та y_j відповідно визначається за формулою:

$$\frac{-2ab \left\| y_i - y_j \right\|_2^{2(b-1)}}{1 + \left\| y_i - y_j \right\|_2^2} w((x_i, x_j))(y_i - y_j) \tag{2.10}$$

Сили відштовхування розраховуються шляхом вибірки через обчислювальні обмеження. Таким чином, щоразу, коли до ребра

прикладається сила притягнення, одна з вершин цього ребра відштовхується вибіркою інших вершин. Сила відштовхування визначається виразом:

$$\frac{2b}{\left(\xi+\left\|y_i-y_j\right\|_2^2\right)\left(1+a\left\|y_i-y_j\right\|_2^{2b}\right)}(1-w((x_i,x_j)))(y_i-y_j) \qquad (2.11)$$

Мета UMAP – розмістити точки y_i так, щоб зважений граф, індукований цими точками, найближче наближався до графа *G*, де вимірюється різниця між зваженими графами за сумарною крос-ентропією за всіма ймовірностями існування ребра.

Завдання вирішується шляхом мінімізації функції вартості C_{UMAP}, яка також є перехресною ентропією [15]:

$$C_{UMAP}=\sum_{i\neq j}\log\left(\frac{v_{ij}}{w_{ij}}\right)+(1-v_{ij})\log\left(\frac{1-v_{ij}}{1-w_{ij}}\right) \qquad (2.12)$$

Отримана група ребер визначає нове розташування об'єктів і, відповідно, їхнє низькорвимірне відображення у вихідному просторі. Цей метод ефективно застосовується до різноманітних типів даних.

В інструкціях до бібліотеки цього методу вказано різноманітні завдання, включаючи розпізнавання зображень, кластеризацію та класифікацію зірок. Результати UMAP демонструють високу якість, подібну до алгоритму t-SNE, вважаного одним з найкращих, проте UMAP працює значно швидше.

2.3 Алгоритмічна модель для класифікатора випадковий ліс

Випадковий ліс (Random forest) – це широко використовуваний алгоритм машинного навчання, зареєстрований під торговою маркою Лео

Бреймана та Адель Катлер, який поєднує вихідні дані кількох дерев рішень для досягнення єдиного результату. Простота використання та гнучкість сприяли його прийняттю, оскільки він вирішує як проблеми класифікації, так і регресії [18].

На рисунку 2.6 зображена блок-схема алгоритмічної моделі.

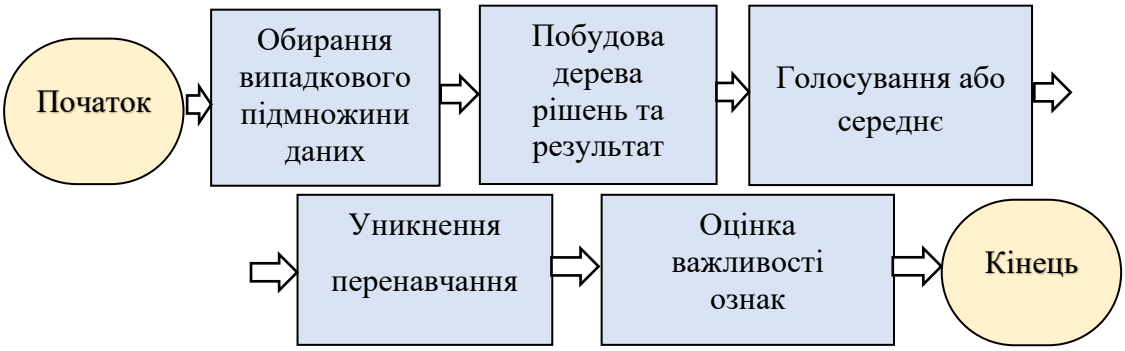


Рисунок 2.6 – Блок-схема алгоритму методу Random forest

Кроки реалізації методу:

Крок 1: Для кожного дерева випадковим чином вибирається підмножина даних з відновленням. Це означає, що кожен об'єкт може вибиратися декілька разів, а деякі об'єкти можуть бути пропущені.

Загальним критерієм розщеплення в задачах класифікації є ентропія, яка є практичним застосуванням теореми про вихідне кодування Шеннона, яка визначає нижню межу довжини бітового представлення випадкової величини. На кожному внутрішньому вузлі дерева рішень ентропія задається формулою:

$$E = -\sum_{i=1}^{c} p_i \times \log(p_i) \tag{2.13}$$

де c – число унікальних класів, а p_i є апріорною ймовірністю кожного даного класу. Це значення максимізується для отримання найбільшої кількості інформації при кожному розбиття дерева рішень.

Крок 2: Для кожного вибраного підмножини даних будується дерево рішень. Кожне дерево вирішує задачу класифікації для об'єктів, використовуючи рекурсивні рішення на основі властивостей даних.

Крок 3: Після побудови всіх дерев результати об'єднуються. Для задач класифікації може використовуватися голосування більшості, де кожне дерево «голосує» за свій клас. Для задач регресії може бути обчислено середнє значення прогнозів дерев.

Крок 4: Випадковий ліс має механізми для уникнення перенавчання. Це може включати випадкове вибору підмножини ознак для кожного вузла дерева і випадкове обмеження глибини дерев.

Крок 5: Випадковий ліс може надати оцінки важливості ознак, що допомагає визначити, які ознаки мають більший вплив на модель.

Таке поєднання декількох моделей називається Ансамбль. Ансамбль використовує два методи:

Пакування (Bagging): Створення іншої навчальної підмножини зі зразками навчальних даних із заміною називається пакуванням (рис. 2.7). Остаточний результат ґрунтується на голосуванні більшістю голосів.

Підсилення (Boosting): Об'єднання слабких учнів у сильних учнів шляхом створення послідовних моделей, таких щоб остаточна модель мала найвищу точність, називається бустінгом (рис. 2.7). Приклад: ADA BOOST, XG BOOST.

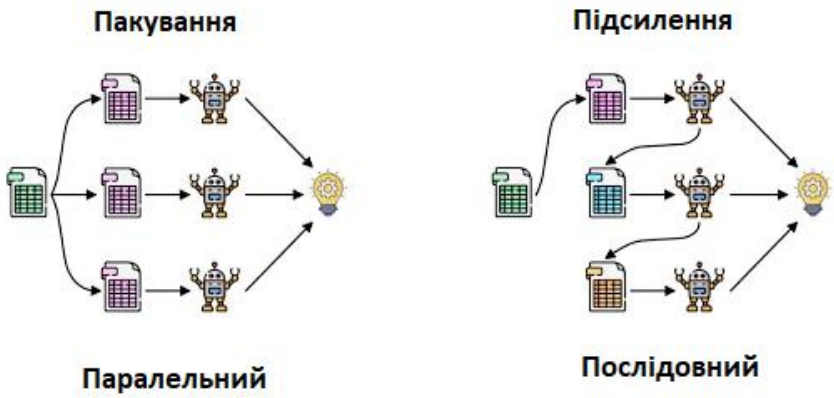


Рисунок 2.7 – Пакування та підсилення

«Bagging» також відомий як «Bootstrap Aggregation», який використовується випадковим лісом. Процес починається з будь-яких вихідних випадкових даних. Після впорядкування він організовується в зразки, відомі як «Bootstrap Sample». Цей процес відомий як початкове завантаження. Крім того, моделі тренуються окремо, що дає різні результати, відомі як агрегація. На останньому кроці всі результати об'єднуються, а отриманий результат базується на голосуванні більшістю голосів. Цей крок відомий як пакування і виконується за допомогою класифікатора ансамблю (рис. 2.8).

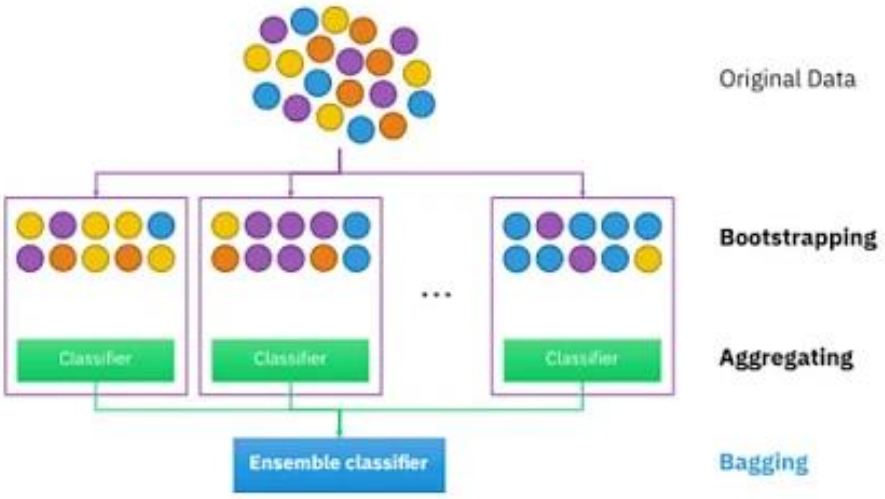


Рисунок 2.8 – Пакування за допомогою класифікатора ансамблю

Існує ряд ключових переваг і проблем, які представляє алгоритм випадкового лісу, коли використовується для задач класифікації або регресії [19].

Основні переваги:

– зниження ризику перенавчання: дерева рішень ризикують бути «перенавчаними», оскільки вони, як правило, щільно вписуються в усі вибірки в навчальні дані. Однак, коли у випадковому лісі є велика кількість дерев рішень, класифікатор не буде перевантажувати модель, оскільки усереднення некорельованих дерев знижує загальну дисперсію та похибку прогнозування;

– забезпечує гнучкість: оскільки random forest може обробляти як регресійні, так і класифікаційні завдання з високим ступенем точності, він є популярним методом серед фахівців з обробки даних. «Мішки» ознак також роблять випадковий класифікатор лісів ефективним інструментом для оцінки відсутніх значень, оскільки він зберігає точність, коли частина даних відсутня;

– легко визначити важливість ознаки: Random forest дозволяє легко оцінити важливість змінної або внесок у модель. Є кілька способів оцінити важливість функції. Коефіцієнт Джині та середнє зменшення домішок (MDI) зазвичай використовуються для вимірювання того, наскільки точність моделі зменшується, коли задана змінна виключається.

Основні недоліки:

– трудомісткий процес: оскільки алгоритми випадкових лісів можуть обробляти великі набори даних, вони можуть надавати більш точні прогнози, але можуть повільно обробляти дані, оскільки вони обчислюють дані для кожного окремого дерева рішень;

– потрібно більше ресурсів: оскільки випадкові ліси обробляють більші набори даних, їм знадобиться більше ресурсів для зберігання цих даних.

2.4 Алгоритмічна модель для класифікатора опорних векторів

Класифікатор опорних векторів (Support Vector Machine, SVM) – це алгоритм машинного навчання, який використовується для задач класифікації і регресії. Основна ідея полягає в тому, щоб знайти гіперплощину в просторі високої розмірності, яка найкращим чином розділяє об'єкти різних класів.

SVM можна використовувати для різних завдань, таких як класифікація тексту, класифікація зображень, виявлення спаму, ідентифікація рукописного тексту, аналіз експресії генів, виявлення облич і виявлення аномалій. SVM

адаптивні та ефективні в різних програмах, оскільки вони можуть керувати високорозмірними даними та нелінійними зв'язками.

Гіперплощини – це межі рішень, які допомагають класифікувати точки даних. Точки даних, що потрапляють по обидва боки гіперплощини, можна віднести до різних класів.

Також розмірність гіперплощини залежить від кількості об'єктів. Якщо число вхідних ознак дорівнює 2, то гіперплощина – це просто лінія. Якщо число вхідних ознак дорівнює 3, то гіперплощина стає двовимірною площиною.

На рисунку 2.9 зображена блок-схема алгоритмічної моделі.

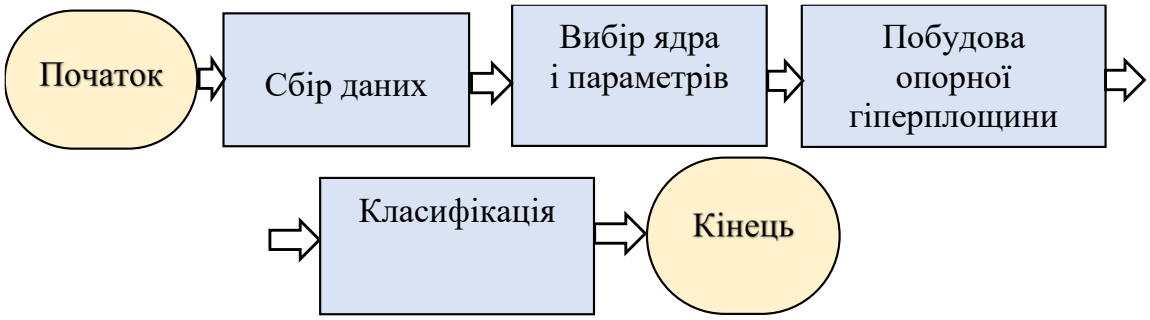


Рисунок 2.9 – Блок-схема алгоритму методу SVM

Кроки реалізації методу:

Крок 1: Сбір даних для тренування, що містять приклади об'єктів і відповідні класи.

Крок 2: Ядро визначає спосіб відображення даних в простір вищої розмірності, де легше виконати розділення.

Крок 3: Знайти гіперплощину, яка найкращим чином розділяє дані класів. Ця гіперплощина розташована так, щоб максимізувати відстань між найближчими точками об'єктів обох класів, які називаються опорними векторами.

Крок 4: Нові об'єкти класифікуються на основі того, по яку сторону від гіперплощини вони знаходяться.

Метод опорних векторів поділяється на лінійний та нелінійний метод опорних векторів.

2.4.1 Лінійний метод опорних векторів. SVM [20] – це алгоритм навчання з учителем. Головна мета SVM як класифікатора - знайти рівняння роздільної гіперплощини:

$$w_1 x_1 + w_2 x_2 + ... + w_n x_n + w_0 = 0 \quad (2.14)$$

у просторі, яка розділила б два класи якимось оптимальним чином. У методі опорних векторів два класи завжди називаються +1 і -1 (а не 1 і 0), а параметр зсуву завжди явно позначається буквою b (а не включається у вектор *w* як константний доданок).

Завдяки цьому математичні викладення стають набагато ясніше. Загальний вид [21] перетворення *F* об'єкта *x* у мітку класу *Y*:

$$F(x) = sign(w^T x - b). \quad (2.15)$$

Значення –1 позначає один клас, а +1 — іншої.

Ми позначили:

$$w = (w_1, w_2, ..., w_n), b = -w_0. \quad (2.16)$$

Після налаштування ваг алгоритму *w* і *b* під час навчання, всі об'єкти, що потрапляють на одну сторону від побудованої гіперплощини, будуть

передбачатися як екземпляри першого класу, тоді як об'єкти, що потрапляють на іншу сторону, визначатимуться як екземпляри другого класу.

Функція *sign()* використовує лінійну комбінацію ознак об'єкта з вагами алгоритму, що підтверджує лінійний характер SVM. Розділяючу гіперплощину можна побудувати різними способами, але в SVM [21] ваги *w* і *b* налаштовуються так, щоб об'єкти класів знаходились якнайдалі від роздільної гіперплощини. Іншими словами, алгоритм максимізує відступ (margin) між гіперплощиною та об'єктами класів, які знаходяться найближче до неї. Ці об'єкти відомі як опорні вектори (рисунок 2.10).

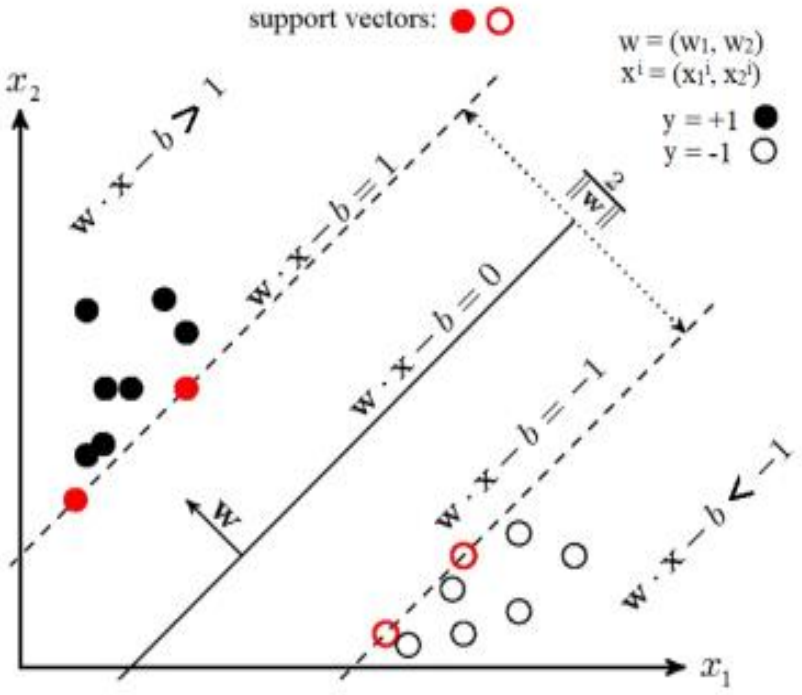


Рисунок 2.10 – Візуалізація методу опорних векторів

2.4.1 Нелінійний метод опорних векторів. Розглянуті літературні літературні джерела належали до лінійно розділимих задач. На практиці така форма категоризації трапляється, але вона є лише винятком, оскільки більшість задач категоризації не дозволяє лінійного розподілу об'єктів.

Для того щоб знову зробити задачу лінійно розділимою і мати можливість застосовувати лагранжіани, потрібно розташувати дані в просторі

більш високого порядку, де можливий лінійний розподіл об'єктів гіперплощиною. Цей процес відомий як спрямлення простору або мапінг даних (рис. 2.11).

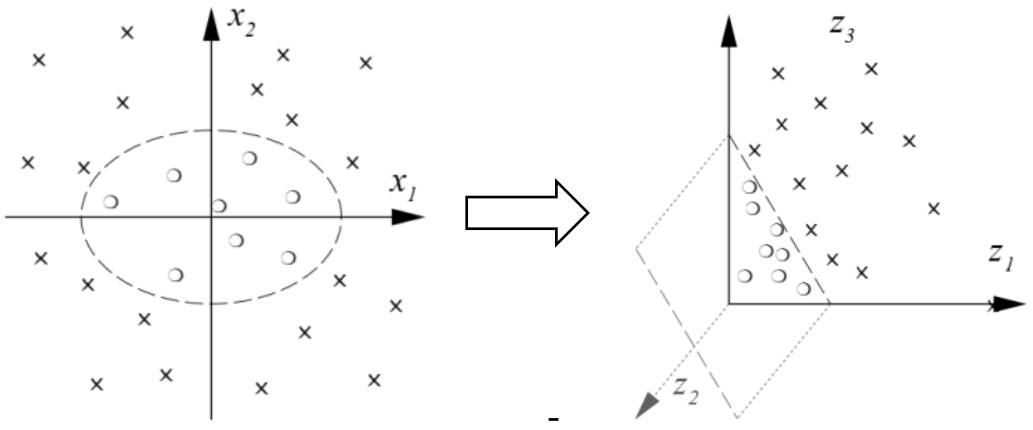


Рисунок 2.11 – Мапінг даних

Приклад мапінгу даних Мапінг даних позначається Φ і виконує відображення даних із вихідного гіперпростору R^d до евклідового простору N

$$\Phi = R^d \rightarrow N. \tag{2.17}$$

Отже, для того, щоб задача стала лінійно розділимою, до кожної з точок треба застосувати перетворення (2.13) і надалі оперувати не з самими точками, а з відповідними їм $\Phi(x)$. Логіка такого розподілення добре ілюструється на рис. 2.12.

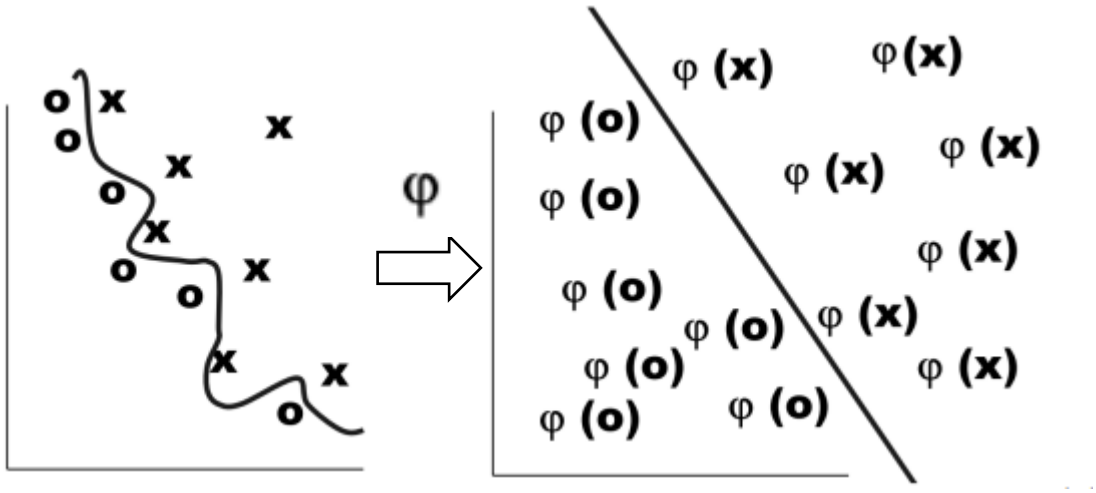


Рисунок 2.12 – Ілюстрація до застосування *Φ(x)* до вихідних даних

У розглянутих лагранжіанах часто використовуються скалярні добутки точок. Для спрощення оптимізаційної задачі було винайдено метод, відомий як «ядерний трюк» (kernel trick), який полягає в переході від скалярних добутків до так званих функцій ядра. При цьому кожен скалярний добуток замінюється нелінійною функцією ядра (скалярним добутком у просторі вищої розмірності). Функцію ядра можна виразити наступним чином [23 – 25]:

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j) \tag{2.18}$$

Використання функції ядра взагалі дозволяє уникнути необхідності прямого використання *Φ(x)*. Користувач методу може навіть не знати, яку саме функцію ядра використовується.

На етапі навчання SVM використовується для обчислення значення *f(x)* за вихідними даними навчання з використанням формули :

$$f(x) = \sum_{i=1}^{n} a_i \gamma_i \Phi(x_i) \cdot \Phi(x) + b = \sum_{i=1}^{n} a_i \gamma_i K(x_i, x) + b \tag{2.19}$$

Найпростіше реалізувати нелінійний метод SVM на базі L_D, або так званого lagrangian trick із застосуванням kernel trick. При цьому максимізується рівняння Лагранжа (2.20). Функція ядра при цьому може бути представлена рівнянням (2.21).

$$L_d = \sum_i a_i - \frac{1}{2} \sum_{i,j}^{n} a_i \cdot a_j \cdot \gamma_i \cdot \gamma_j \cdot x_i \cdot x_j \tag{2.20}$$

$$K(x_i, x_j) = (x_i \cdot x_j + 1)^p, \tag{2.21}$$

де p – деякий параметр, що підлягає налаштуванню користувачем.

Метод опорних векторів зводить процес навчання класифікатора до оптимізаційної задачі, яку вирішується евристичними алгоритмами. Для створення нелінійних класифікаторів використовується розширення простору та функції ядер. Метод SVM в тестах демонструє перевагу над іншими методами за швидкістю та точністю категоризації. Залежно від обраного ядра, метод може емулювати роботу інших математичних методів. Хоча SVM може функціонувати як нейронна мережа, такий підхід обмежує його застосування, оскільки метод виявляє значну перевагу за можливостями.

Висновки до розділу 2

В цьому розділі було розглянуто методи машинного навчання для вирішення задач зниження розмірності та класифікації даних.

Метод головних компонент використовується для зменшення розмірності даних, виявлення кореляцій та виокремлення основних характеристик набору даних. Метод зменшує розмірність даних, зберігаючи при цьому більшість їх варіативності та дозволяє виявляти ключові залежності та взаємозв'язки у наборі даних. Проте лінійний PCA не завжди ефективний

для нелінійних зв'язків у даних. Реалізовується шляхом визначення головних компонент та зменшення розмірів даних за рахунок цих компонент.

Нелінійний метод апроксимації та проекції однорідного різноманіття (UMAP) використовується для візуалізації та зменшення розмірності даних, зокрема медичних зображень та молекулярних даних.

Метод дозволяє ефективно розгортати многовимірні дані на двовимірну площину для візуалізації та здатний працювати з нелінійними структурами даних, проте потребує досить багато ресурсів для обробки великих обсягів даних. UMAP використовується для згортання многовимірних даних на площину та їх візуалізації шляхом збереження топологічних властивостей даних.

Метод випадкових лісів (Random forest) використовується для класифікації та регресії, роботи з великим обсягом даних, та підвищення точності моделі. Цей метод менше схильний до перенавчання порівняно з іншими методами та здатний обробляти великі обсяги даних з великою кількістю ознак, проте метод може бути вимогливий до ресурсів під час «навчання» моделі. Метод базується на ансамблі дерев прийняття рішень, що дозволяє підвищити точність та уникнути перенавчання.

Метод опорних векторів (SVM) використовується для класифікації, регресії та виявлення аномалій. SVM використовується для знаходження оптимальної гіперплощини, що розділяє категорії в наборі даних, або для використання ядер для розмежування класів даних, коли вони не є лінійно роздільними в просторі ознак.

Для практичної реалізації методів було розроблено їх алгоритмічні моделі.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ТА ВІЗУАЛІЗАЦІЇ ДАНИХ НА ОСНОВІ МЕТОДІВ МАШИННОГО НАВЧАННЯ

3.1 Вхідні дані

В якості вхідних даних є дата-сет з інформацією про набори даних [26], які складаються з декількох змінних медичних предикторів та однієї цільової змінної – результат.

Таблиця 3.1 – *Перелік кардіоктографічних даних з «fetal_health.csv»*

Name	**Type**	**Range**
baseline value	float	106-160
Accelerations	float	0-0.019
fetal_movement	float	0-0.481
uterine_contractions	float	0-0.015
light_decelerations	float	0-0.015
severe_decelerations	float	0-0.001
prolongued_decelerations	float	0-0.005
abnormal_short_term_variability	float	12-87
mean_value_of_short_term_variability	float	0.2-7.0
percentage_of_time_with_abnormal_long_term_variability	float	0-91
mean_value_of_long_term_variability	float	0-50.7
histogram_width	float	3-180
histogram_min	float	50-159
histogram_max	float	122-238
histogram_number_of_peaks	float	0-18
histogram_number_of_zeroes	float	0-10
histogram_mode	float	60-87
histogram_mean	float	73-182
histogram_median	float	77-186
histogram_variance	float	0-269
histogram_tendency	float	-1-1
fetal_health	float	1-3

3.2 Вихідні дані

Таблиця 3.2 – *Вихідні дані*

Model	Accuracy	Precision	Recall	F1_Score
Random Forest	float	float	float	float
SVM	float	float	float	float

В розділі було розглянуто вхідні та вихідні дані для алгоритмічних моделей методів класифікації та методів оцінки інформативності показників.

3.3 Проектування архітектури системи

Проектування архітектури системи охоплює процес створення високорівневого плану структури та організації компонентів системи. У цьому процесі ключові архітектурні рішення визначаються разом з вибором технологій, установленням взаємозв'язків між компонентами та формулюванням правил для функціонування системи та її взаємодії з іншими системами чи користувачами, що представляється у візуальній формі. Дві широко використовувані методології для моделювання архітектури систем – це IDEF0 і UML [27-29]:

1. IDEF0 (Integration DEFinition for Function Modeling) – це методологія моделювання функцій, розроблена в рамках програми ICAM (Integrated Computer Aided Manufacturing) від Управління з питань логістики США. IDEF0 [27, 29] використовується для моделювання різних процесів. У цьому підході кожна функція або процес розглядається як блок, який приймає вхідні дані, обробляє їх та видає вихідні дані.

Основна ідея IDEF0 полягає в тому, щоб створити ієрархічні структури функцій для аналізу та документування взаємодії між компонентами системи. За допомогою блок-схем, стрілок та текстового опису, IDEF0 дозволяє

представити складні процеси та їхні взаємозв'язки відповідно до визначених стандартів. Цей метод є ефективним інструментом для аналізу та проектування бізнес-процесів.

2. UML (Unified Modeling Language) – це мова моделювання, що використовується для аналізу, проектування і реалізації систем баз даних і програмного забезпечення [28]. UML є стандартом для моделювання об'єктно-орієнтованих систем і використовується для візуалізації, проектування та документування архітектури програмних систем. UML надає різні види діаграм, які включають діаграми класів, діаграми прецендентів, діаграми активності, діаграми послідовності.

UML дозволяє інженерам з різних областей використовувати спільні терміни та засоби візуалізації для ефективного спілкування та роботи над проектами, спрощуючи розуміння архітектури та функціональності систем.

Для розроблення й оцінювання мультимедіа-інтерфейсів застосовуються ергономічні принципи, описані в ISO 9341-10. Ці сім принципів важливі для розроблення і оцінювання інтерактивних додатків:

1. Придатність для вирішення завдання
2. Наявність контекстної допомоги.
3. Керованість.
4. Узгодженість із очікуваннями користувача.
5. Стійкість до помилок.
6. Можливість індивідуалізації.
7. Зручність навчання.

Додатково до загальних принципів стандарт ISO 14915 визначає специфічні принципи, які необхідно брати до уваги під час проєктування мультимедійних проєктів:

– придатність для мети комунікації;
– зручність сприйняття і розуміння;

– зручність вивчення;

– привабливість.

На рис. 3.1 зображена контекстна діаграма програмного забезпечення для аналізу та візуалізації даних на основі методів машинного навчання в методології IDEF0.

Рисунок 3.1 – Контекстна системи в методології IDEF0

Декомпозиція системи в IDEF0 – це процес розбиття більш складних функцій на менш складні підфункції. Кожен вузол на контекстній діаграмі можна розкрити (або декомпозувати) на низькому рівні деталізації [29]. Декомпозиція починається з ідентифікації ключових функцій системи, які потрібно вивчити. Кожна з цих функцій представляється блоком, що може бути подальшим чином розглянутий в деталях. Ці блоки стають основними компонентами системної моделі. Після визначення блоків системи, моделюються взаємозв'язки між ними. Це включає в себе встановлення потоків даних, що входять та виходять з кожного блоку, а також інших факторів, які визначають взаємодію між блоками. Деталізація кожного блоку

включає в себе створення дочірніх блоків, розширення функціональності та уточнення взаємозв'язків. Цей процес дозволяє ретельніше досліджувати конкретні аспекти системи. Далі відбувається створення ієрархії блоків. Ця ієрархія полегшує розуміння та аналіз системи на різних рівнях абстракції.

На рис. 3.2 зображена декомпозиція системи в методології IDEF0.

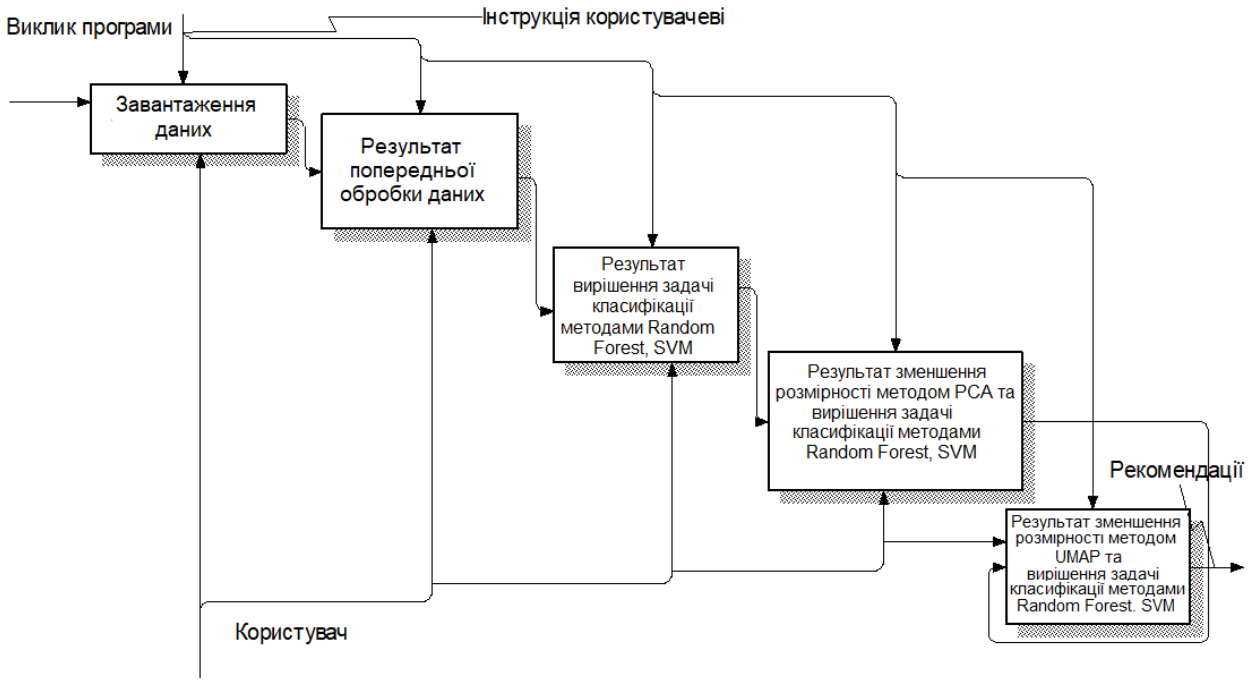


Рисунок 3.2 – Декомпозиція системи в методології IDEF0

Діаграма варіантів використання – це тип діаграми в UML, що відображає взаємодію користувачів (або «акторів») з системою. Варіант використання – це функція або дія, яку система виконує, що дає цінність користувачу [28].

На діаграмі варіантів використання актори зображуються як палички людей, а варіанти використання – як еліпси. Лінії між акторами та варіантами використання вказують на можливі взаємодії.

Діаграми варіантів використання дуже ефективні для визначення функцій системи та взаємодій між різними користувачами і системою.

На рис. 3.3 зображена діаграма варіантів використання UML.

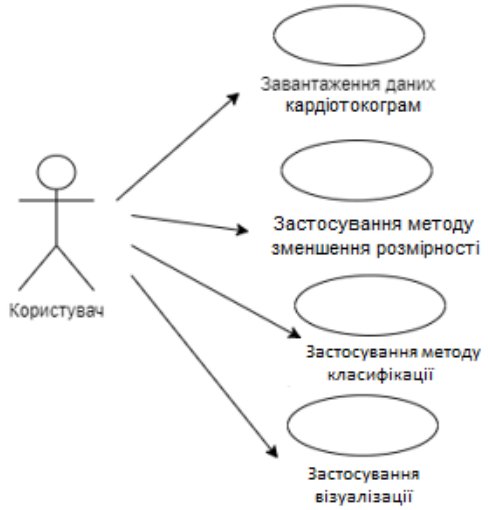


Рисунок 3.3 – Діаграма варіантів використання

На рис. 3.4 - 3.5 зображена діаграма функціональних блоків.

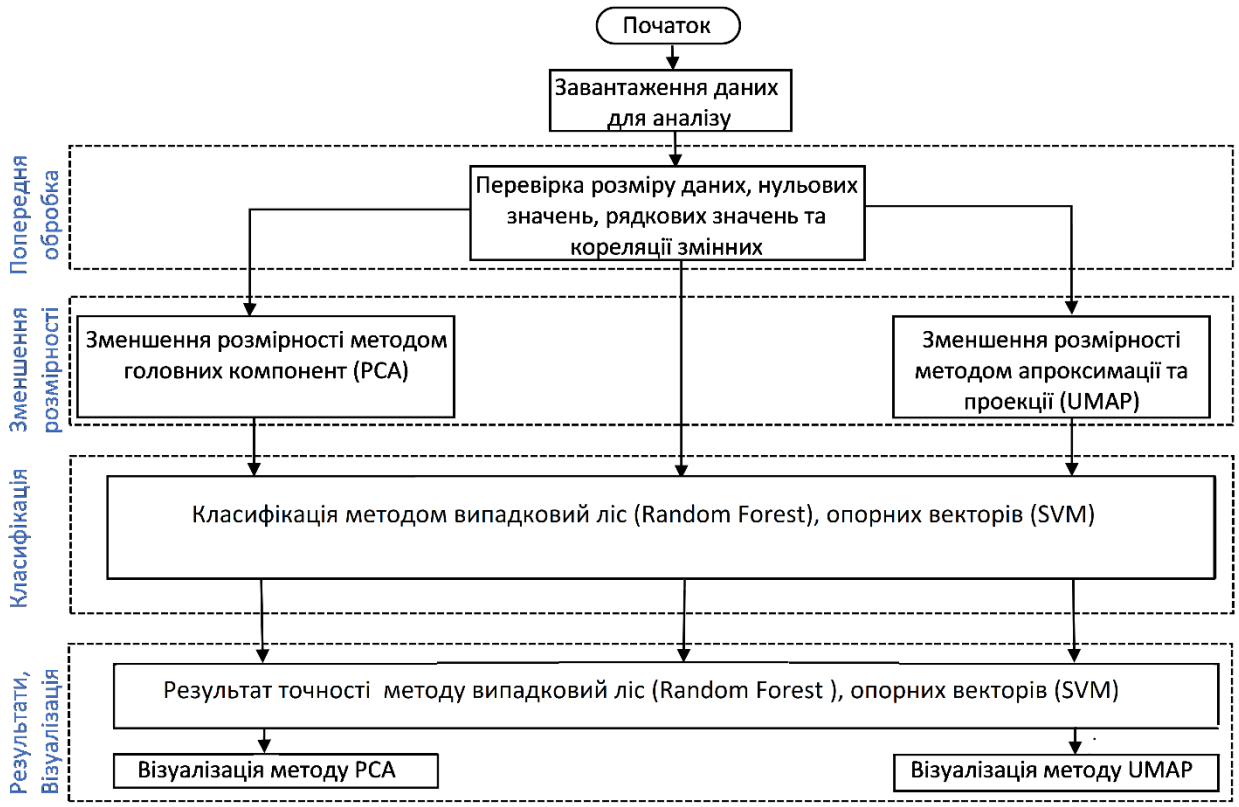


Рисунок 3.4 – Діаграма функціональних блоків

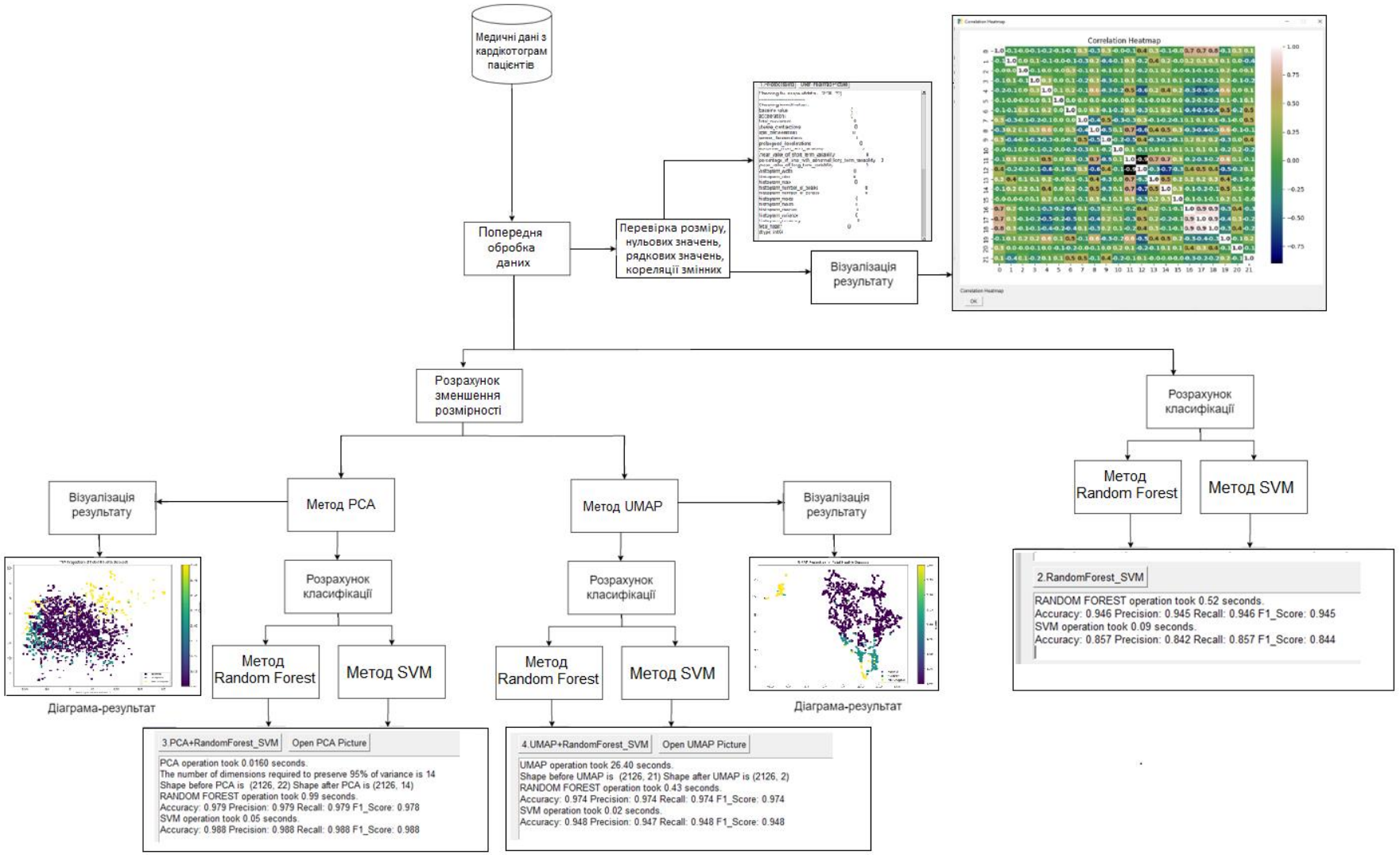


Рисунок 3.5 – Діаграма функціональних блоків

Так, на першому етапі відбувається завантаження медичних даних з кардіокотограм.

До медичних даних належать:

− «baseline value» − значення сердцевих скорочень FHR (ударів на хвилину);

− «accelerations» − кількість прискорень за секунду;

− «fetal_movement» − кількість рухів плода за секунду;

− «uterine_contractions» − кількість скорочень матки за секунду;

− «light_decelerations» − кількість уповільнень за секунду;

− «severe_decelerations» − кількість сильних уповільнень за секунду;

− «prolongued_decelerations» − кількість тривалих уповільнень за секунду;

− «abnormal_short_term_variability» − відсоток часу з аномальною короткостроковою мінливістю;

− «mean_value_of_short_term_variability» − середнє значення короткострокової мінливості;

− «percentage_of_time_with_abnormal_long_term_variability» − відсоток часу з ненормальною довгостроковою мінливістю;

− «mean_value_of_long_term_variability» − середнє значення довгострокової мінливості;

− «histogram_width» − ширина гістограми FHR;

− «histogram_min» − мінімальна (низька частота) гістограма FHR;

− «histogram_max» − максимальна (висока частота) гістограма FHR;

− «histogram_number_of_peaks» − кількість піків гістограми;

− «histogram_number_of_zeroes» − кількість нулів гістограми;

− «histogram_mode» − режим гістограми;

− «histogram_mean» − середнє значення гістограми;

− «histogram_median» − медіана гістограми;

− «histogram_variance» − дисперсія гістограми;

− «histogram_tendency» − тенденція гістограми.

Ціль «fetal_health» з тегами 1 (нормальний), 2 (підозрюваний) і 3 (патологічний)

Це функціональний блок «Оцінка інформативності».

На другому етапі спочатку відбувається розрахунок зменшення розмірності за допомогою методів головних компонент (PCA) та UMAP, а потім класифікація за допомогою методів випадковий ліс (Random Forest) та опорних векторів.

Для візуалізації даних було використано бібліотеку matplotlib.pyplot для інформаційної статистичної графіки.

На третьому етапі відбувається розрахунок точності машинного навчання використовуючи методи класифікації такі як випадковий ліс (Random Fores) та метод опорних векторів (SVM).

При цьому можливим є використання як окремих блоків для зменшення розмірності та класифікації, так і спільне використання двох блоків для скорочення розмірності даних та подальшого аналізу.

Висновки до розділу 3

В розділі було описано вхідні та вихідні дані до програмного продукту. Методологія IDEF0 використовується для створення ієрархічних структур функцій з метою аналізу та документування взаємодії між компонентами системи. З використанням блок-схем, стрілок та текстового опису IDEF0 дозволяє зобразити складні процеси та їхні взаємозв'язки відповідно до визначених стандартів. Цей підхід є ефективним інструментом для аналізу та проектування бізнес-процесів.

Декомпозиція системи в IDEF0 використовується для розбиття більш складних функцій на менш складні підфункції. Зокрема, кожен вузол на контекстній діаграмі може бути декомпозований для детальнішого вивчення. У процесі декомпозиції встановлюються взаємозв'язки між блоками, включаючи потоки даних та інші фактори, що визначають взаємодію між ними. Деталізація кожного блоку включає створення дочірніх блоків, розширення функціональності та уточнення взаємозв'язків, що дозволяє більш ретельно вивчати конкретні аспекти системи.

UML використовується для аналізу, проектування і реалізації систем баз даних і програмного забезпечення. UML надає можливість фахівцям з різних галузей використовувати загальноприйняті терміни та інструменти візуалізації для ефективного взаємодії та спільної роботи над проектами. Це спрощує розуміння архітектури та функціональності системи, сприяючи більш ефективному обміну інформацією та спільній роботі у процесі розробки.

Діаграми варіантів використання дуже ефективні для визначення функцій системи та взаємодій між різними користувачами і системою.

Стандарти, зокрема ISO 9341-10 та ISO 14915, використовуються для розроблення та оцінювання мультимедіа-інтерфейсів з використанням ергономічних принципів. Використання цих стандартів допомагає забезпечити ефективну та ергономічну взаємодію користувачів з мультимедійними інтерфейсами.

4 ВИКОРИСТАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ТА ВІЗУАЛІЗАЦІЇ ДАНИХ НА ОСНОВІ МЕТОДІВ МАШИННОГО НАВЧАННЯ

4.1 Опис середовища програмування

Система програмування PyCharm – це спеціальне інтегроване середовище розробки Python (IDE), що надає широкий спектр основних інструментів для розробників Python, тісно інтегрованих для створення зручного середовища для продуктивної розробки Python, веб-застосунків і науки про дані [30]. Він забезпечує аналіз коду, графічний налагоджувач, інтегрований модульний тестувальник, інтеграцію з системами контролю версій і підтримує веб-розробку за допомогою Django. PyCharm розроблений чеською компанією JetBrains. Він кросплатформний, працює на Microsoft Windows, macOS і Linux. PyCharm має професійну версію, випущену за власною ліцензією, і версію Community Edition.

Середовище програмування Python – це інтерпретоване, об'єктно-орієнтоване, високорівневе програмування мова з динамічною семантикою. Його високорівневі вбудовані дані структури, поєднані з динамічною типізацією та динамічною прив'язкою, роблять його дуже привабливий для швидкої розробки додатків, а також для використання як скриптової або клейової мова для з'єднання існуючих компонентів разом.

Простий, легкий у вивченні синтаксис Python підкреслює читабельність і, отже, знижує витрати на обслуговування програми. Python підтримує модулі та пакети, що заохочує програму модульність і повторне використання коду [31].

Відносно до інших мов програмування, Python вирізняється такими перевагами [32]:

1) Простий синтаксис Python робить його особливо привабливим для новачків, дозволяючи їм швидко освоювати мову програмування. Завдяки його синтаксису можна виражати програмні ідеї коротше, ніж у багатьох інших мовах, таких як Java або C++.

2) Python має обширну стандартну бібліотеку, що охоплює багато загальних завдань програмування, надаючи розробникам готові інструменти для вирішення різноманітних задач.

3) Вбудована підтримка тестування у Python робить процес перевірки правильності коду більш зручним, завдяки наявності вбудованих модулів, спеціально призначених для цієї мети.

4) Існування великої та активної спільноти розробників відзначає Python, що сприяє постійному розвитку та підтримці мови. Ця спільнота дозволяє легко знаходити відповіді на питання, вивчати нові бібліотеки та інструменти, спрощуючи процес програмування.

Для створення інформаційної системи використовувалися різноманітні алгоритми та методи, і Python став ідеальним вибором для цих завдань. Його розширена бібліотека включає в себе обширний набір алгоритмів машинного навчання, що сприяло ефективному розв'язанню завдань розробки інформаційної системи.

Для візуалізації даних було використано бібліотеку matplotlib.pyplot, яка є потужним інструментом для візуалізації в середовищі Python. Ця бібліотека надає широкий спектр можливостей для створення графіків, діаграм, інтерактивних візуалізацій та багато іншого, що сприяло зрозумінню та аналізу результатів роботи системи.

4.2 Інструкція користувача

У системі програмування PyCharm потрібно додати програмний код з додатку «А» та інсталювати пакети, виконати запуск «Run» (рис. 4.1).

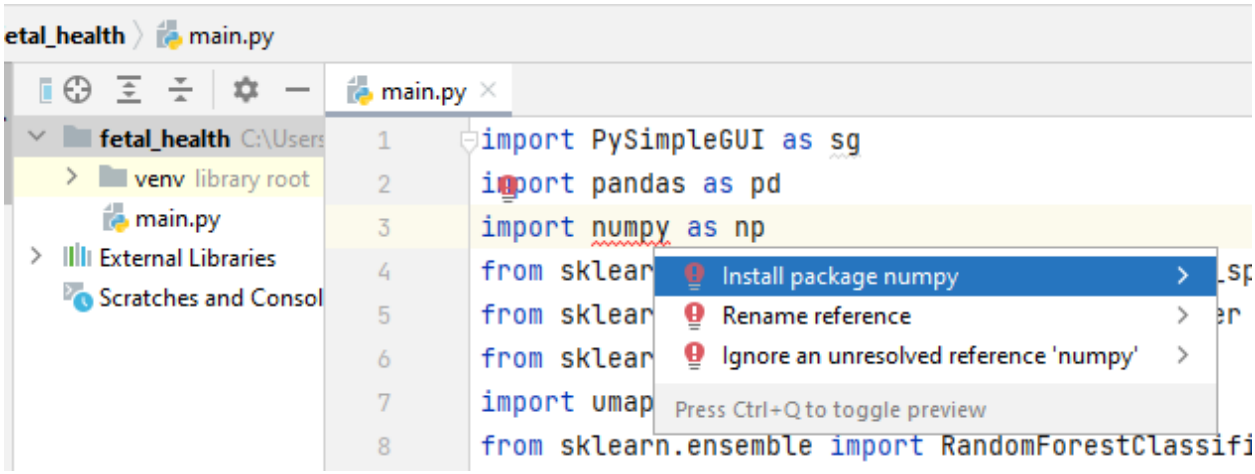


Рисунок 4.1 – PyCharm вікно для запуску програми

У цьому дослідженні порівнюється точність методів машинного навчання до та після використання методів зменшення розмірності на комп'ютері CPU: 8 cores AMD Ryzen 7 5800H - 3.2 Hz; GPU: NVIDIA GeForce RTX 3060 - 6Gb; RAM: 16Gb.

4.3 Опис програмного продукту

Програмний продукт, який розраховує точність методів машинного навчання до та після зменшення розмірності, базуючись на даних кардіокотограм плода вагітних.

На початку потрібно натиснути кнопку «0. Browse fetal_health.csv» та додати набір даних «fetal_health.csv» (рис. 4.2).

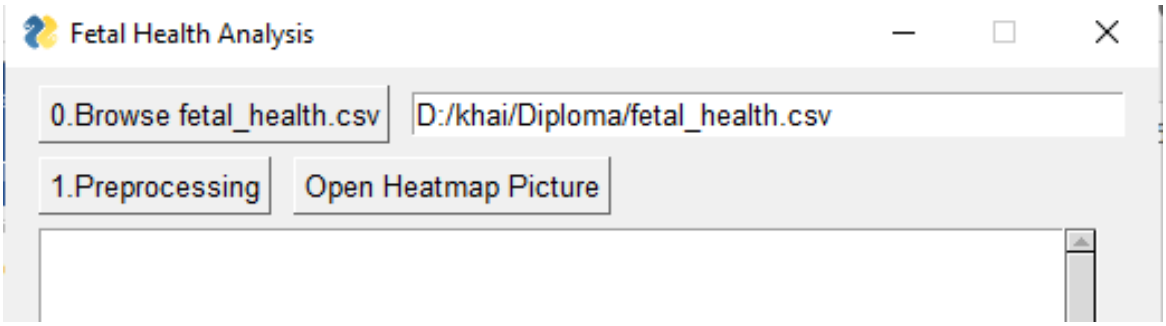


Рисунок 4.2 – Завантаження датасету «fetal_health.csv»

Якщо користувач завантажить пустий файл на формі буде відображено інформування про помилку. Далі при натисканні кнопки на «1. Preprocessing» відбувається попередня обробка даних (рис. 4.3-4.4).

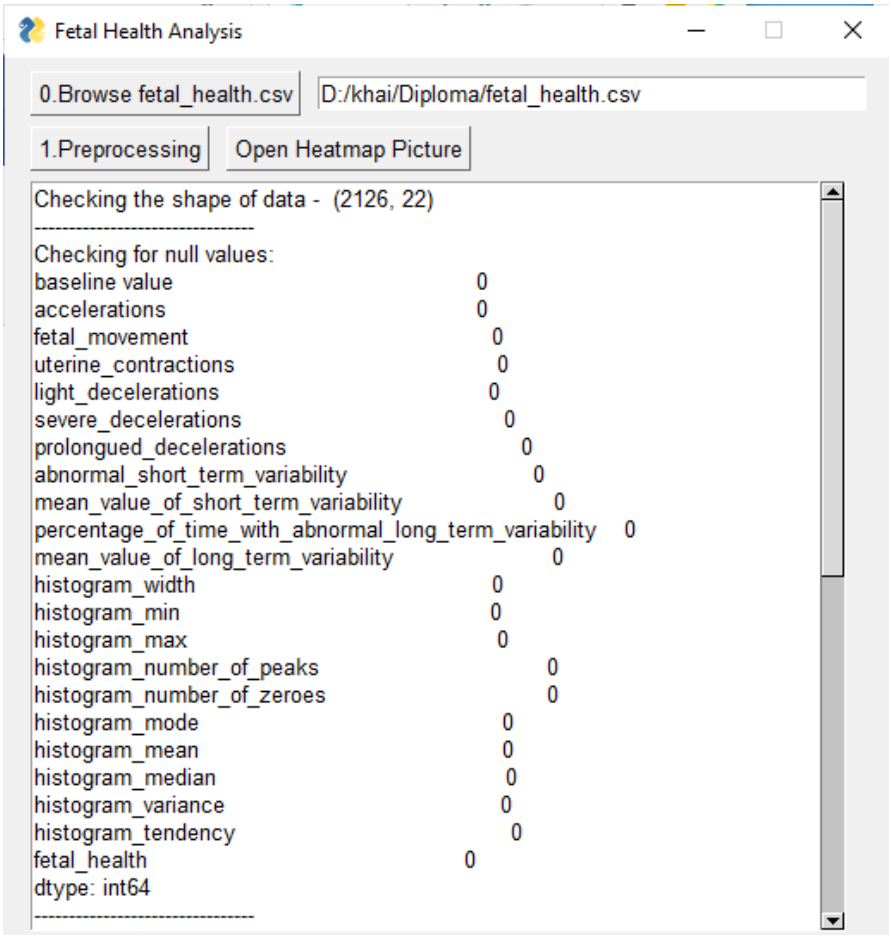


Рисунок 4.3 – Вихідні дані попередньої обробки даних

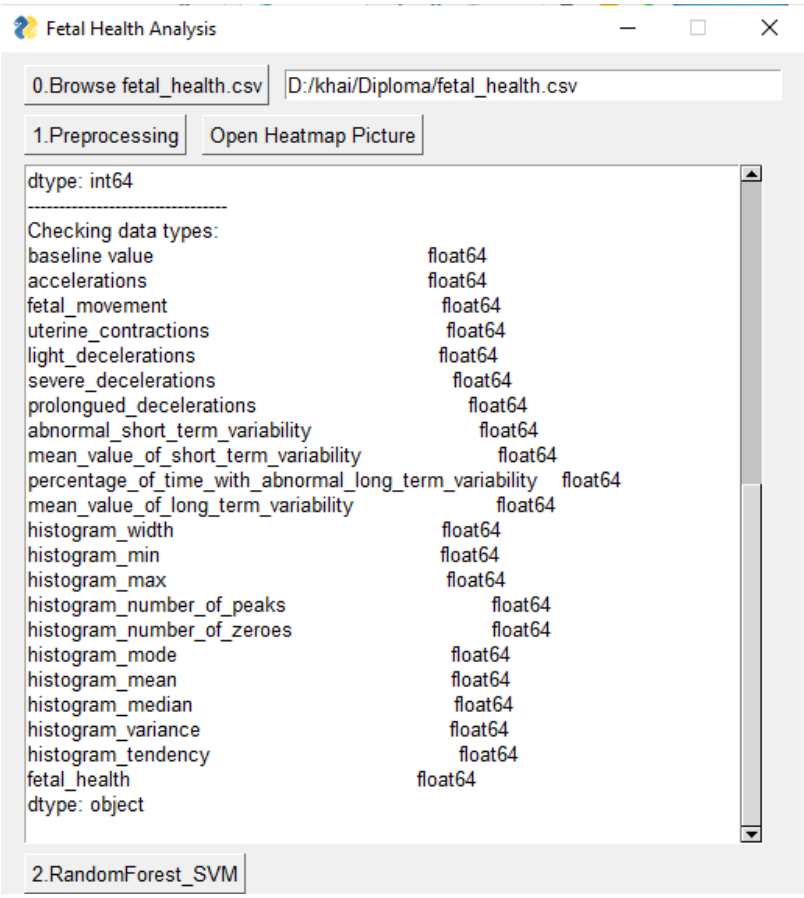


Рисунок 4.4 – Вихідні дані попередньої обробки даних

Якщо натиснути кнопку «Open HeatMap Picture» − можна побачити матрицю кореляції (рис. 4.5).

Примітка: 0 – «baseline value», 1 − «accelerations», 2 − «fetal_movement», 3 - «uterine_contractions», 4 – «light_decelerations», 5 – «severe_decelerations», 6 – «prolongued_decelerations», 7 – «abnormal_short_term_variability», 8 – «mean_value_of_short_term_variability», 9 – «percentage_of_time_with_abnormal_long_term_variability», 10 – «percentage_of_time_with_abnormal_long_term_variability», 11– «mean_value_of_long_term_variability», 12 – «histogram_width», 13 – «histogram_min», 14 – «histogram_max», 15 – «histogram_number_of_peaks», 16 – «histogram_number_of_zeroes», 17 – «histogram_mode», 18 – «histogram_mean», 19 – «histogram_median»,

20 – «histogram_variance», 21– «histogram_tendency».

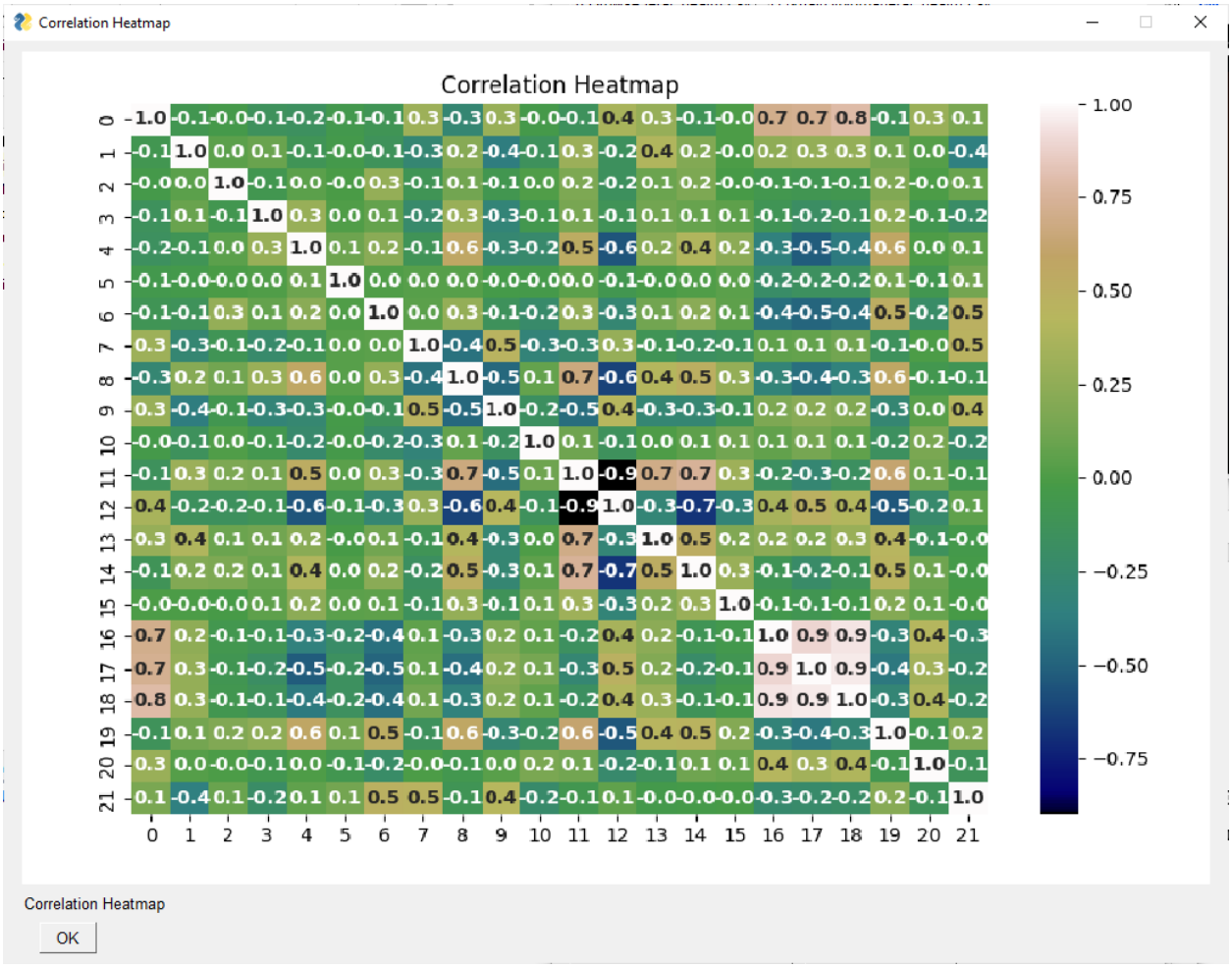


Рисунок 4.5 – Матриця кореляції

З попередньої обробки даних можна зробити висновки:

1) Розмір датасету 2126 x 22;

2) У датасеті немає нульових значень, тому дані чисті;

3) Усі значення в наборі даних є цілими числами або числами з плаваючою точкою. У наборі даних немає рядкових значень;

4) Змінні «histogram_median», «histogram_mode» та «histogram_mean» сильно корелюють, що показує, що значення всіх трьох змінних у визначенні цільового стовпця «fetal_health» (здоров'я плода) однакове. Крім того, дві змінні «histogram_width» and «histogram_min» сильно

негативно корелюють. Але оскільки всі функції є надзвичайно важливими, жодна з них не буде видалена з набору даних.

Далі при натисканні кнопки «2. RandomForest_SVM» відбувається розрахунок показників машинного навчання методів випадковий ліс та опорних векторів до зменшення розмірності (рис. 4.6).

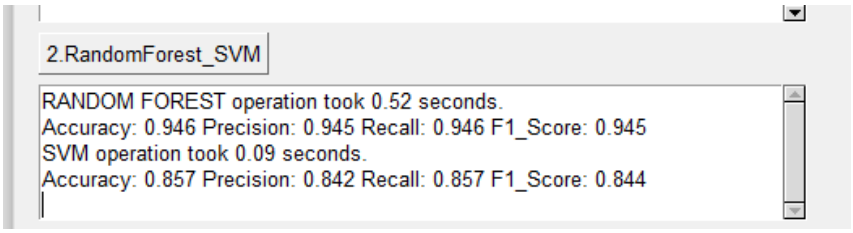


Рисунок 4.6 – Розрахунок до зменшення розмірності

З розрахунків можно зробити висновок що метод випаковий ліс виконувався довше у 5.78 разів, але показав точність на 8,9% краще ніж у методу опорних векторів.

Далі при натисканні кнопки «3. PCA+RandomForest_SVM» відбувається розрахунок показників машинного навчання методів випадковий та опорних векторів після зменшення розмірності методом головних компонент (рис. 4.7).

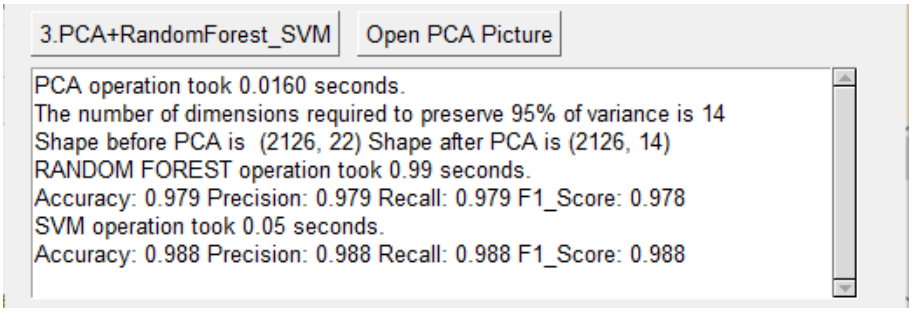


Рисунок 4.7 – Розрахунок після зменшення розмірності методом головних компонет

Якщо натиснути кнопку «Open PCA Picture», то можна побачити візуалізацію методу головних компонент (рис. 4.8).

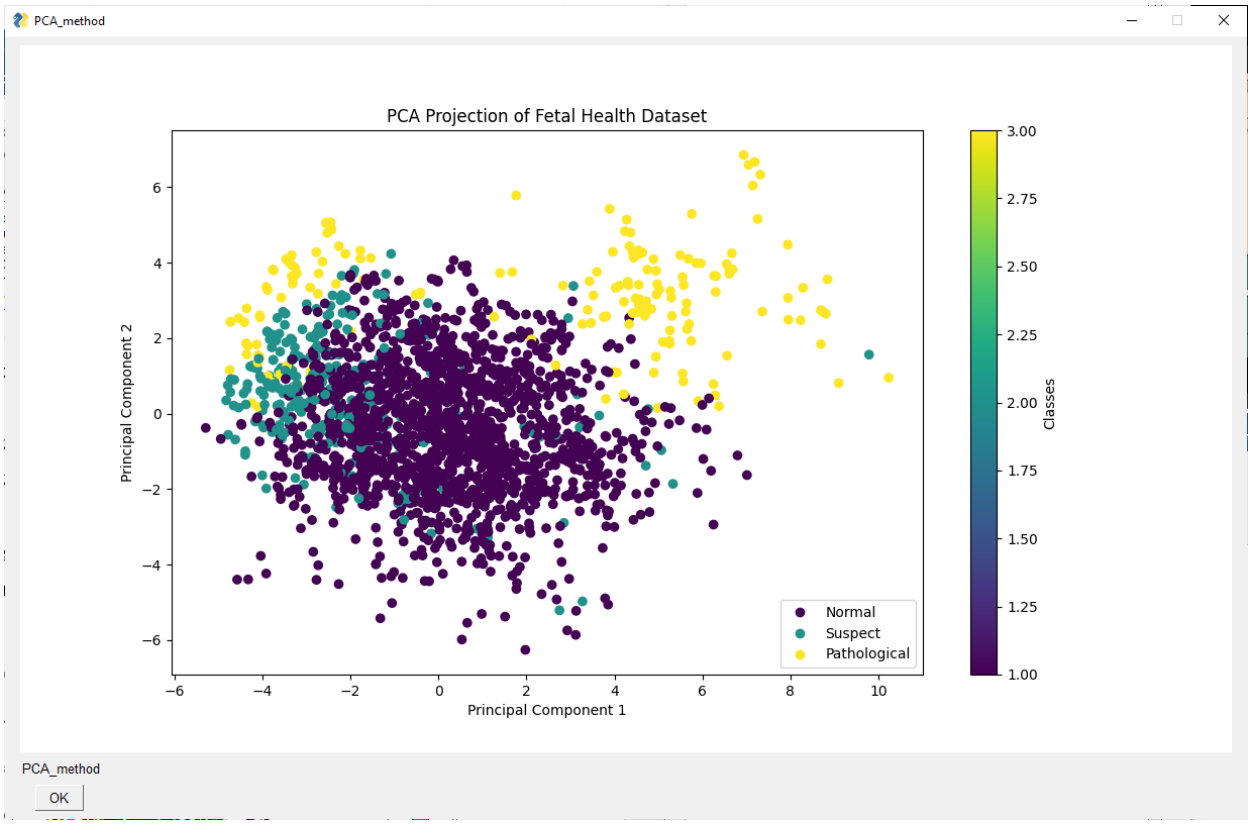


Рисунок 4.8 – Вихідні дані методів випадковий ліс та опорних векторів після зменшення розмірності методом головних компонент

Набір даних, використаний у цьому дослідженні, було скорочено до 15 компонентів із 23 компонентів за допомогою аналізу головних компонентів за 0,01 секунди.

На метод випадковий ліс було витрачено 0,55 секунди, що у двічі довше ніж у попередньому результаті. Метод опорних векторів у двічі швидше – 0,02 секунди.

Точність було підвищено: випадковий ліс – з 0,946 до 0,979; опорних векторів – з 0,857 до 0,988.

Метод зменшення розмірності головних компонент підвищив точність методу опорних векторів більше ніж у методі випадковий ліс.

Далі при натисканні кнопки «4. UMAP+RandomForest_SVM» відбувається розрахунок показників машинного навчання методів випадковий

та опорних векторів після зменшення розмірності методом UMAP (рис. 4.6 та 4.7).

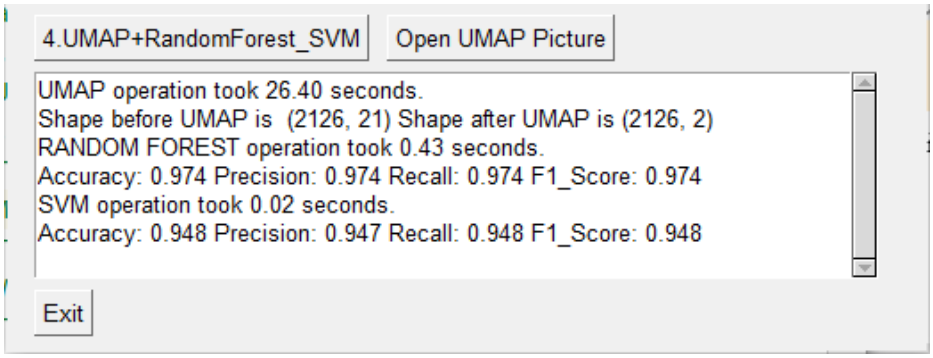


Рисунок 4.9 – Вихідні дані методів випадковий ліс та опорних векторів після зменшення розмірності методом UMAP

Далі якщо натиснути кнопку «Open UMAP Picture», то можна побачити візуалізацію методу UMAP (рис. 4.10).

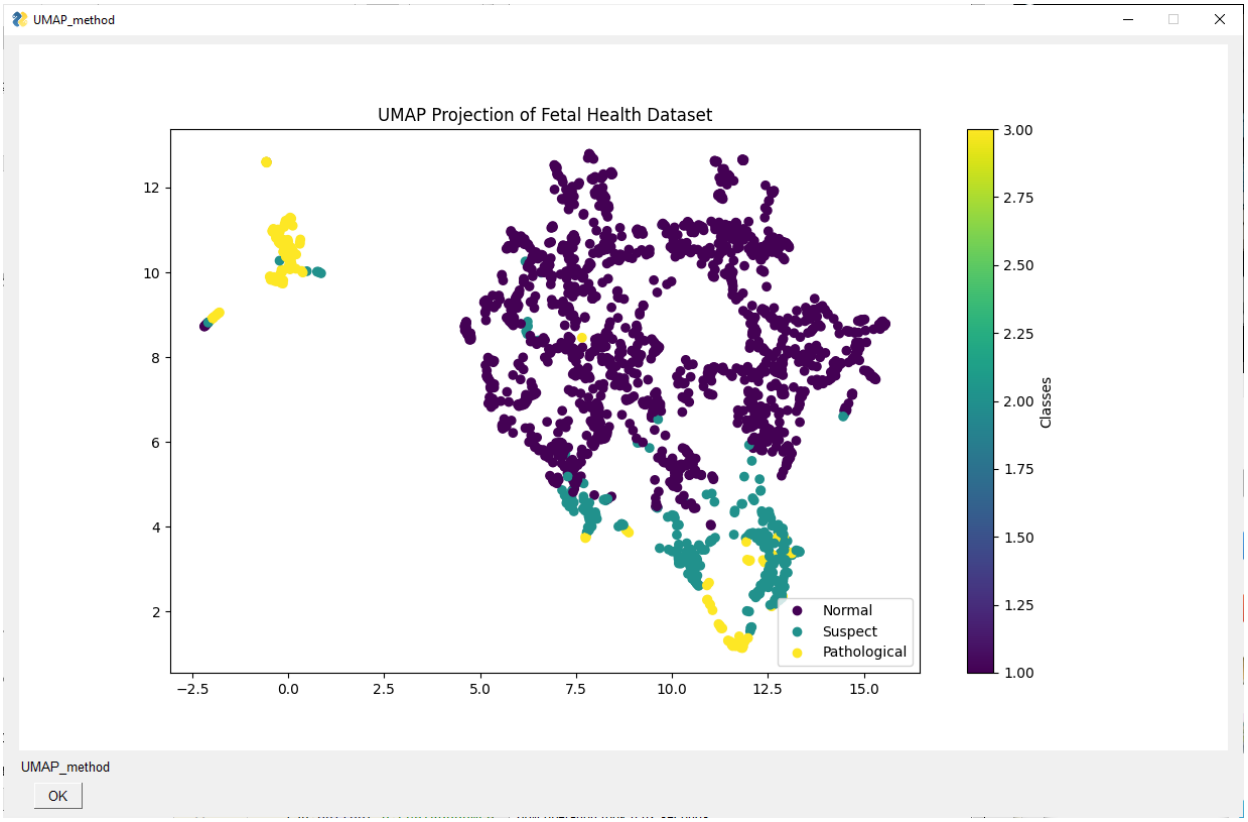


Рисунок 4.7 – Візуалізація методу UMAP

Набір даних було скорочено до 3 компонентів із 23 компонентів за допомогою методу UMAP за 26,4 секунд.

На метод випадковий ліс було витрачено 0,43 секунди, що приблизно так само як без зменшення розмірності. Метод опорних векторів у четверо швидше – 0,02 секунди.

Точність було підвищено : випадковий ліс – з 0,946 до 0,974; опорних векторів – з 0,857 до 0,948.

По закінченню розрахунків потрібно завершити програму, натиснувши кнопку «Exit».

Висновки до розділу 4

У цьому розділі була розроблена інструкція для користувача, наведена інформація про датасет, з’ясована найкраща комбінація методів зменшення розмірності та класифікації за допомогою середовища Python.

Середовище програмування Python вирізняється як ефективний інструмент для розробки програм, завдяки своєму інтерпретованому, об'єктно-орієнтованому та високорівневому характеру. Простий синтаксис сприяє легкому використанню мови, а обширна стандартна бібліотека робить його потужним засобом для вирішення різноманітних завдань програмування. Додатково, переваги Python включають вбудовану підтримку тестування, наявність широкої спільноти розробників та розширену бібліотеку для машинного навчання.

У конкретному випадку створення інформаційної системи використовувало різноманітні алгоритми та методи, з використанням бібліотеки matplotlib.pyplot для візуалізації даних, що допомогло ефективно розв'язати завдання розробки системи та забезпечити зрозумілість та аналіз результатів роботи.

Попередня обробка даних використовується для підготовки та оптимізації набору даних перед їх використанням у моделюванні чи аналізі. З попередньої обробки даних можна зробити кілька важливих висновків. По-перше, розмір датасету становить 2126 рядків на 22 стовпці, вказуючи на достатню обсяжність інформації. По-друге, виявлено відсутність нульових значень, що свідчить про чистоту даних. По-третє, усі значення в датасеті є числовими, більше того, відсутні рядкові значення. Найважливіше, аналіз кореляції вказує на сильну взаємозалежність між змінними «histogram_median», «histogram_mode» та «histogram_mean», що може вказувати на їхню схожість у визначенні цільового стовпця «fetal_health». Також виявлено сильну негативну кореляцію між змінними «histogram_width» та «histogram_min». Зазначимо, що всі ці функції є важливими, тому жодна з них не буде видалена з набору даних, і це може вказувати на їхню значущість у визначенні статусу здоров’я плода.

Були розраховані показники машинного навчання методів випадковий ліс (Random Forest), опорних векторів (SVM). Далі було задіяне зменшення розмірності методами головних компонент (PCA) і апроксимації та проекції рівномірного різноманіття (UMAP), а після знову розраховані показники машинного навчання.

Найкраще покращення точності машинного навчання отримав метод головних компонент та метод опорних векторів: на метод головних компонент було витрачено 0,01 секунди, а метод опорних векторів показав точність 98,8%.

ВИСНОВКИ

У даній роботі були розглянуті методи машинного навчання, зокрема методи зменшення розмірності та класифікації даних у медичних дослідженнях.

В ході ознайомлення з теорією з даної проблеми були сформульовані основні поняття, алгоритми і характеристики задачі зменшення розмірності та класифікації даних.

Було розроблено алгоритмічну модель лінійного методу головних компонент (PCA), алгоритмічну модель нелінейного методу апроксимації та проекції однорідного різноманіття (UMAP), алгоритмічну модель методу випадковий ліс (Random forest), алгоритмічну модель метод опорних векторів (SVM).

Було спроектовано та розроблено спеціальний програмний продукт для вирішення завдань дослідження та візуалізації отриманих даних. Під час проектування використовувались методологія IDEF та графічна мова UML. Розробка інтерфейсів виконувалась з використанням стандартів ISO 9341-10 та ISO 14915.

Основні результати роботи програмного продукту:

− попередня обробка даних показала розмір датасету 212x22, відсутність нульових значень та аналіз кореляції;

− на першому розрахунку була виконана класифікація методом випадковий ліс (Random Forest) за 0,52 секунди з резульататом точності 94.6% та опорних векторів (SVM) за 0,09 секунди з результатом точності 85,7%;

− на другому розрахунку було проведено зменшення розмірності методом головних компоненент (PCA) за 0,016 секунди, розмір датасету зменшився з 2126x22 до 2126x14 залишаючи 95% дисперсії;

− метод класифікації Random Forest виконався за 0,99 секунди з результатом точності 97,9%, а метод SVM виконався за 0,05 секунди з результатом точності 98,8%;

− на третьому розрахунку була проведено зменшення розмірності методом апроксимації та проекції рівномірного різноманіття (UMAP) за 26,4 секунди, розмір датасету зменшився з 2126x22 до 2126x2;

− метод класифікації Random Forest виконався за 0,43 секунди з результатом точності 97,9%, а метод SVM виконався за 0,02 секунди з результатом точності 94,8%.

Таким чином після використання методу PCA можна підвищити точність методу SVM з 85,7% до 98,8% з мінімальнім часом використання методів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dutt R, Sahu M. Temporal Analysis of Infant and Child Health Indicators from Health Management and Information System of a Vulnerable District of India: Tracking the Road toward the Sustainable Development Goal-3. Indian J Community Med (2019) 397-398. doi: 10.4103/ijcm.IJCM_5_19. [Стаття]

2. M. A. Rafi, M. M. Z. Miah, M. A. Wadood and M. G. Hossain, "Risk factors and etiology of neonatal sepsis after hospital delivery: A case-control study in a tertiary care hospital of Rajshahi Bangladesh", PLoS One, vol. 15, no. 11, pp. e0242275, 2020. [Журнал]

3. Khlystun O., Bazilevych K. «Comparative analysis of the machine learning dimensionality reduction methods on the example of fetal health determination», 3rd International Workshop of IT-Professionals on Artificial Intelligence ProfIT AI 2023, 20-22 листопада 2023, м. Ватерлоо, Канада.

4. Yunida H. Saving of Maternal and Infant Lives with Sustainable Midwifery Services. Int J Community Based Nurs Midwifery (2022) 313-314. doi: 10.30476/IJCBNM.2022.95877.2092. [Стаття]

5. Хлистун О., Базілевіч К. «Визначення рівня стресу за допомогою методів машинного навчання», V Міжнародна науково-практична конференція IT-професіоналів і аналітиків комп'ютерних систем «ProfIT Conference», 2023. [Стаття]

6. A. Uberoi: Introduction to Dimensionality Reduction, 2023. URL: https://www.geeksforgeeks.org/dimensionality-reduction/

7. Foster P, Tom F. Data Science and its Relationship to Big Data and Data-Driven Decision Making (2013). doi: 10.1089/big.2013.1508. [Стаття]

8. James N, Hsien H, Matthew B. Machine learning: applications of artificial intelligence to imaging and diagnosis (2019) 111–118. doi: 10.1007/s12551-018-0449-9. [Стаття]

9. Chao Deng, M. Zu Guo. A new co-training-style random forest for computer aided diagnosis. URL: https://kd.nsfc.gov.cn/paper Download/1000001632658.pdf.

10. Yanli L, Yourong W. New Machine Learning Algorithm: Random Forest. (2012) 246-252. doi:10.1007/978-3-642-34062-8_32. [Стаття]

11. Sunil R. Learn How to Use Support Vector Machines (SVM) for Data Science (2023). URL: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/.

12. Shamsolmoali, P., Kumar Jain, D., Zareapoor, M., Yang, J. and Afshar Alam, M. High-Dimensional Multimedia Classification Using Deep CNN and Extended Residual Units. Multimedia Tools and Applications, (2019), 23867-23882 doi: 10.1007/s11042-018-6146-7. [Стаття]

13. Naveen V. The Curse of Dimensionality: Inside Out (2018) doi:10.13140/RG.2.2.29631.36006. [Стаття]

14. Goodfellow I. Deep Learning (2018) 652 с. [Книга]

15. McInnes, Leland & Healy, John. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction (2018). doi: 10.21105/joss.00861. [Стаття]

16. Haghighat M., Zonouz S., Abdel-Mottaleb M. CloudID. Trustworthy Cloud-based and Cross-Enterprise Biometric Identification. місце видання невідоме : Expert Systems with Applications, 2015. [Журнал]

17. Гойко О.В. Аналіз сучасного програмного забезпечення для статистичного оброблення й аналізу біомедичних досліджень. місце видання невідоме : Медична інформатика та інженерія., 2012. [Книга]

18. Sruthi R. Understand Random Forest Algorithms With Examples (2023) URL: https://www.simplilearn.com/tutorials/machine-learning-tutorial/random-ке

19. What is random forest? URL: https://www.ibm.com/topics/random-forest.

20. Воронцов К. В., «Лекції методу опорних векторів». 2007. – 16 с. [Журнал]

21. Дадиверін В. В., Фещенко І.О., Потапова К.Р., Тарасенко-Клятченко О.В. «система розпізнавання обличчя з використанням поєднання методів Haar та SVM». – Науковий журнал «Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки», Херсон, 2022. – 12 с. [Журнал]

22. Kai-Bo D., S. Sathiya K. «Which Is the Best Multiclass SVM Method? An Empirical Study» – Multiple Classifier Systems, LNCS. pp.278–285. [Журнал]

23.Smola A. J., Schölkopf B. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. Algorithmica, 22:211 — 231, 1998. [Журнал]

24. Nello Cristianini, John Shawe-Taylor. An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press, 2000. — 204 p. [Журнал]

25. John Shawe-Taylor, Nello Cristianini. Kernel Methods for Pattern Analysis. Cambridge University Press, 2004. — 462 p. [Журнал]

26. Статистичні дані, набір кардіокотограм плода: Fetal Health Classification. https://www.kaggle.com/datasets/andrewmvd/fetal-health-classification

27. Малкович О. (2013). Методологія функціонального моделювання IDEF0. Львів: Афіша. [Книга]

28. Рамбо Дж. (2014). UML 2.5: Основи. Видавництво МСЕ. [Книга]

29. Радклифф Д., Вега-Медіна А., Галлардо В. (2012). Моделювання та аналіз бізнес-процесів з використанням IDEF0. Видавництво Спрінгер. [Книга]

30. Quick start guide. URL: https://www.jetbrains.com/help/pycharm/quick-start-guide.html.

31. What is Python? Executive Summary. URL: https://www.python.org/doc/essays/blurb/.

32. Phython: плюси і мінуси мови, які завдання вирішує и чи варто вивчати. URL: https://avada-media.ua/ua/services/python-plyusy-i-minusy-yazyka-kakiye-zadachi-reshayet-i-stoit-li-izuchat/.

ДОДАТОК А

```
import os
import PySimpleGUI as sg
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import umap
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score
import time
import seaborn as sns
import matplotlib.pyplot as plt

sg.theme('Default1')
layout = [
    [sg.FileBrowse('0.Browse fetal_health.csv', target="-PATH-",
file_types=[("csv", '*.csv')]), sg.I(key="-PATH-")],
    [sg.Button('1.Preprocessing'), sg.Button('Open Heatmap Picture')],
    [sg.Output(size=(65, 27), key='-OUTPUT-')],
    [sg.Button('2.RandomForest_SVM')],
    [sg.Output(size=(65, 5), key='-OUTPUT-RF_SVM-')],
    [sg.Button('3.PCA+RandomForest_SVM'), sg.Button('Open PCA Picture')],
    [sg.Output(size=(65, 8), key='-OUTPUT-PCA-RF_SVM-')],
    [sg.Button('4.UMAP+RandomForest_SVM'), sg.Button('Open UMAP Picture')],
    [sg.Output(size=(65, 7), key='-OUTPUT-UMAP-RF_SVM-')],
    [sg.Button('Exit')]
]

window = sg.Window('Fetal Health Analysis', layout)

while True:
    event, values = window.read()

    if event in (None, 'Exit'):
        break

    elif event == '1.Preprocessing':
        try:
            csv_adress = values["-PATH-"]
            fetal_data = pd.read_csv(csv_adress)
            y = fetal_data.fetal_health
            x = fetal_data.drop(["fetal_health"], axis=1)
            window['-OUTPUT-'].print("Checking the shape of data - ",
fetal_data.shape)
            window['-OUTPUT-'].print("---------------------------------")
            window['-OUTPUT-'].print("Checking for null values:")
            window['-OUTPUT-'].print(fetal_data.isnull().sum())
            window['-OUTPUT-'].print("---------------------------------")
            window['-OUTPUT-'].print("Checking data types:")
            window['-OUTPUT-'].print(fetal_data.dtypes)
            # HeatMap
            corr = fetal_data.corr()
            plt.figure(figsize=(13, 5))
            plt.title('Correlation Heatmap')
```

```
            sns_plot = sns.heatmap(corr, annot=True,
annot_kws={"fontsize":10, "fontweight":"bold"},
fmt=".1f", cmap='gist_earth',
xticklabals=range(len(fetal_data.columns)),
yticklabals=range(len(fetal_data.columns)))
            fig = sns_plot.get_figure()
            fig.savefig('HeatMap.png')
        except:
            sg.popup_auto_close("Oops! Browse 'fetal_health.csv' first!")

    elif event == 'Open Heatmap Picture':
        if os.path.isfile("HeatMap.png"):
            sg.popup_ok('Correlation Heatmap', image="HeatMap.png")
        else:
            sg.popup_auto_close("Click the button 1.Preprocessing first!")

    elif event == '2.RandomForest_SVM':
        try:
            csv_adress = values["-PATH-"]
            fetal_data = pd.read_csv(csv_adress)
            y = fetal_data.fetal_health
            x = fetal_data.drop(["fetal_health"], axis=1)
            # SPLITTING THE DATA
            x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42)
            # RANDOM FOREST CLASSIFIER
            model_rf = RandomForestClassifier()
            start_time_rf = time.time()
            model_rf.fit(x_train, y_train)
            end_time_rf = time.time()
            elapsed_time_rf = end_time_rf - start_time_rf
            window['-OUTPUT-RF_SVM-'].print(f"RANDOM FOREST operation took
{elapsed_time_rf:.2f} seconds.")
            # Predict on test data
            y_pred_rf = model_rf.predict(x_test)
            # Calculate evaluation metrics
            accuracy_rf = accuracy_score(y_test, y_pred_rf)
            precision_rf = precision_score(y_test, y_pred_rf,
average='weighted')
            recall_rf = recall_score(y_test, y_pred_rf, average='weighted')
            f1_rf = f1_score(y_test, y_pred_rf, average='weighted')
            # Print results to the second output section
            window['-OUTPUT-RF_SVM-'].print(f"Accuracy: {accuracy_rf:.3f}
Precision: {precision_rf:.3f} "
                                            f"Recall: {recall_rf:.3f} F1_Score:
{f1_rf:.3f}")
            # SVM
            model_svm = SVC()
            start_time_svm = time.time()
            model_svm.fit(x_train, y_train)
            end_time_svm = time.time()
            elapsed_time_svm = end_time_svm - start_time_svm
            window['-OUTPUT-RF_SVM-'].print(f"SVM operation took
{elapsed_time_svm:.2f} seconds.")
            # Predict on test data
            y_pred_svm = model_svm.predict(x_test)
            # Calculate evaluation metrics
            accuracy_svm = accuracy_score(y_test, y_pred_svm)
            precision_svm = precision_score(y_test, y_pred_svm,
average='weighted')
```

```
            recall_svm = recall_score(y_test, y_pred_svm,
average='weighted')
            f1_svm = f1_score(y_test, y_pred_svm, average='weighted')
            # Print results to the second output section
            window['-OUTPUT-RF_SVM-'].print(f"Accuracy: {accuracy_svm:.3f}
Precision: {precision_svm:.3f} "
                                            f"Recall: {recall_svm:.3f} F1_Score:
{f1_svm:.3f}")
        except:
            sg.popup_auto_close("Oops! Browse 'fetal_health.csv' first!")

    elif event == '3.PCA+RandomForest_SVM':
        try:
            csv_adress = values["-PATH-"]
            fetal_data = pd.read_csv(csv_adress)
            y = fetal_data.fetal_health
            x = fetal_data.drop(["fetal_health"], axis=1)
            # 1) Right number of dimensions
            scaler = StandardScaler()  # Standardize the Data
            scaled_fetal_data = scaler.fit_transform(fetal_data)
            pca = PCA()  # method PCA
            start_time_pca = time.time()  # Record the start time PCA
            pca.fit(scaled_fetal_data)  # Train
            end_time_pca = time.time()  # Record the end time PCA
            elapsed_time_pca = end_time_pca - start_time_pca  # Calculate
PCA time
            window['-OUTPUT-PCA-RF_SVM-'].print(f"PCA operation took
{elapsed_time_pca:.4f} seconds.")
            cumsum = np.cumsum(pca.explained_variance_ratio_)
            dim = np.argmax(cumsum >= 0.95) + 1
            window['-OUTPUT-PCA-RF_SVM-'].print('The number of dimensions
required to preserve 95% of variance is', dim)
            # 2) Use method PCA
            scaler.fit(fetal_data)
            scaled_data = scaler.transform(fetal_data)
            pca = PCA(n_components=14)
            pca.fit(scaled_data)
            x_pca = pca.transform(scaled_data)  # Data after PCA
            window['-OUTPUT-PCA-RF_SVM-'].print('Shape before PCA is ',
scaled_data.shape, 'Shape after PCA is', x_pca.shape)
            # SPLITTING THE DATA
            x_train_pca, x_test_pca, y_train, y_test =
train_test_split(x_pca, y, test_size=0.2, random_state=42)
            # 3) RANDOM FOREST CLASSIFIER
            pca_model_rf = RandomForestClassifier()
            start_time_rf = time.time()  # Record the start time RANDOM
FOREST
            pca_model_rf.fit(x_train_pca, y_train)  # Train the DT model on
the train data
            end_time_rf = time.time()  # Record the end time RANDOM FOREST
            elapsed_time_rf = end_time_rf - start_time_rf  # Calculate
RANDOM FOREST
            window['-OUTPUT-PCA-RF_SVM-'].print(f"RANDOM FOREST operation
took {elapsed_time_rf:.2f} seconds.")
            # Predict on test data
            y_pred_rf = pca_model_rf.predict(x_test_pca)
            # Calculate evaluation metrics
            accuracy_rf = accuracy_score(y_test, y_pred_rf)
            precision_rf = precision_score(y_test, y_pred_rf,
average='weighted')
```

```
            recall_rf = recall_score(y_test, y_pred_rf, average='weighted')
            f1_rf = f1_score(y_test, y_pred_rf, average='weighted')
            window['-OUTPUT-PCA-RF_SVM-'].print(f"Accuracy:
{accuracy_rf:.3f} Precision: {precision_rf:.3f} "
                                                f"Recall: {recall_rf:.3f} F1_Score:
{f1_rf:.3f}")
            # 4) SUPPORT VECTOR MACHINES (SVM)
            model_svm = SVC()
            start_time_svm = time.time()  # Record the start time SVM
            model_svm.fit(x_train_pca, y_train)  # Train SVC on train data
            end_time_svm = time.time()  # Record the end time SVM
            elapsed_time_svm = end_time_svm - start_time_svm  # Calculate
SVM
            window['-OUTPUT-PCA-RF_SVM-'].print(f"SVM operation took
{elapsed_time_svm:.2f} seconds.")
            # Predict on the test data
            y_pred_svm = model_svm.predict(x_test_pca)
            # Calculate evaluation metrics
            accuracy_svm = accuracy_score(y_test, y_pred_svm)
            precision_svm = precision_score(y_test, y_pred_svm,
average='weighted')
            recall_svm = recall_score(y_test, y_pred_svm,
average='weighted')
            f1_svm = f1_score(y_test, y_pred_svm, average='weighted')
            window['-OUTPUT-PCA-RF_SVM-'].print(f"Accuracy:
{accuracy_svm:.3f} Precision: {precision_svm:.3f} "
                                                f"Recall: {recall_svm:.3f} F1_Score:
{f1_svm:.3f}")
            # Visual PCA
            labels_str = ["Normal", "Suspect", "Pathological"]
            plt.figure(figsize=(12, 7))
            scatter = plt.scatter(x_pca[:, 0], x_pca[:, 1], c=y,
label=labels_str, cmap='viridis')
            plt.xlabel('Principal Component 1')
            plt.ylabel('Principal Component 2')
            plt.legend(handles=scatter.legend_elements()[0],
labels=labels_str, loc="lower right")
            plt.colorbar(label='Classes')
            plt.title('PCA Projection of Fetal Health Dataset')
            plt.savefig('PCA.png')
        except:
            sg.popup_auto_close("Oops! Browse 'fetal_health.csv' first!")

    elif event == 'Open PCA Picture':
        if os.path.isfile("PCA.png"):
            sg.popup_ok('PCA_method', image="PCA.png")
        else:
            sg.popup_auto_close("Click '3.PCA+RandomForest_SVM' first ")

    elif event == '4.UMAP+RandomForest_SVM':
        try:
            csv_adress = values["-PATH-"]
            fetal_data = pd.read_csv(csv_adress)
            y = fetal_data.fetal_health
            x = fetal_data.drop(["fetal_health"], axis=1)
            sg.popup_auto_close('I am working...It takes more time than
PCA...')
            # 2) Use method UMAP
            scaler = StandardScaler()  # Standardize the Data
            scaled_fetal_data = scaler.fit_transform(fetal_data)
```

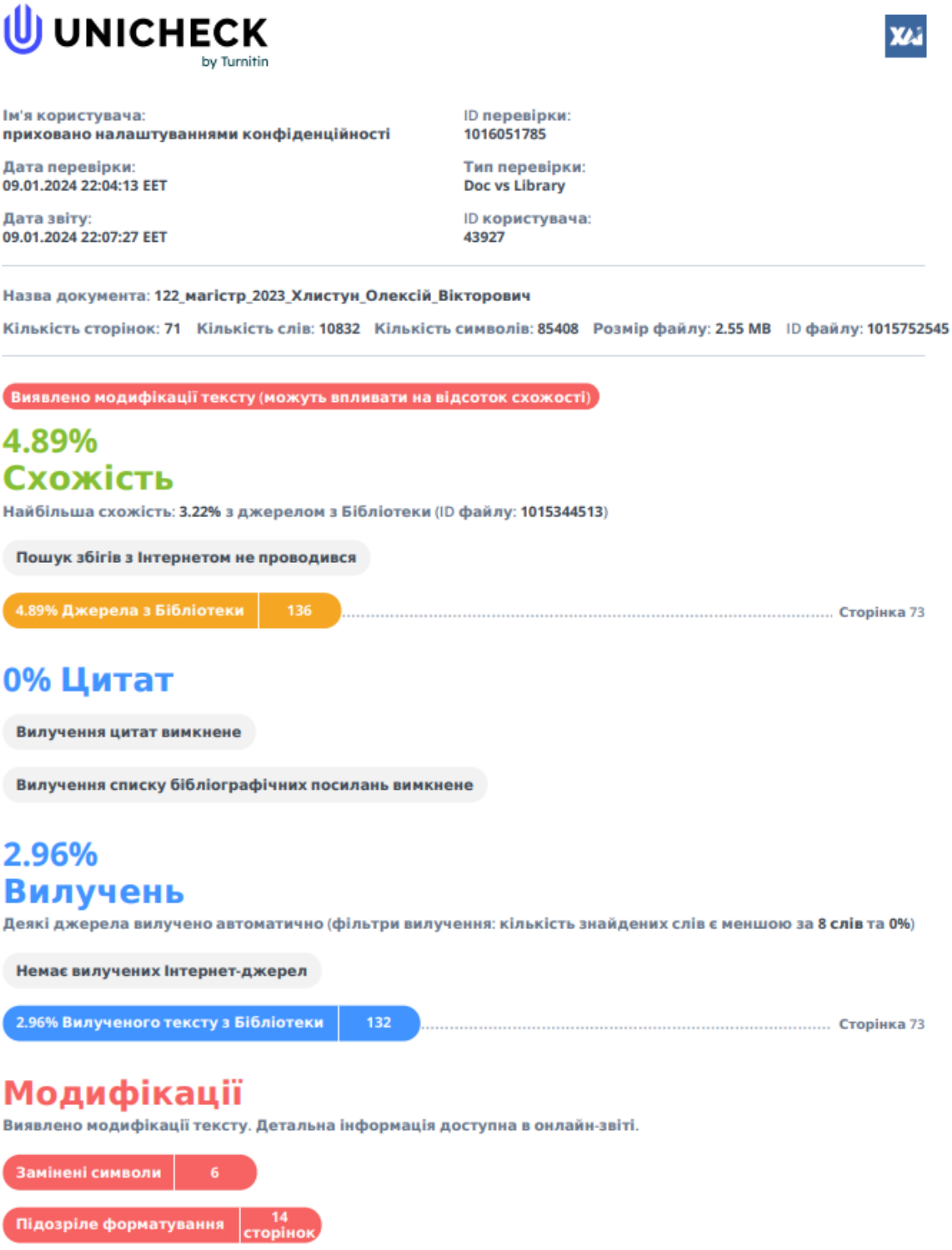
```
            umap = umap.UMAP()
            start_time_umap = time.time()  # Record the start time UMAP
            x_umap = umap.fit_transform(scaled_fetal_data)  # Data after
UMAP
            end_time_umap = time.time()  # Record the end time UMAP
            elapsed_time_umap = end_time_umap - start_time_umap  # Calculate
UMAP time
            window['-OUTPUT-UMAP-RF_SVM-'].print(f"UMAP operation took
{elapsed_time_umap:.2f} seconds.")
            window['-OUTPUT-UMAP-RF_SVM-'].print('Shape before UMAP is ',
x.shape, 'Shape after UMAP is', x_umap.shape)
            # SPLITTING THE DATA
            x_train_umap, x_test_umap, y_train, y_test =
train_test_split(x_umap, y, test_size=0.2, random_state=42)
            # 3) RANDOM FOREST CLASSIFIER
            model_rf = RandomForestClassifier()
            # Train the DT model on the train data
            start_time_rf = time.time()  # Record the start time RANDOM
FOREST
            model_rf.fit(x_train_umap, y_train)
            end_time_rf = time.time()  # Record the end time RANDOM FOREST
            elapsed_time_rf = end_time_rf - start_time_rf  # Calculate
RANDOM FOREST
            window['-OUTPUT-UMAP-RF_SVM-'].print(f"RANDOM FOREST operation
took {elapsed_time_rf:.2f} seconds.")
            # Predict on test data
            y_pred_rf = model_rf.predict(x_test_umap)
            # Calculate evaluation metrics
            accuracy_rf = accuracy_score(y_test, y_pred_rf)
            precision_rf = precision_score(y_test, y_pred_rf,
average='weighted')
            recall_rf = recall_score(y_test, y_pred_rf, average='weighted')
            f1_rf = f1_score(y_test, y_pred_rf, average='weighted')
            window['-OUTPUT-UMAP-RF_SVM-'].print(f"Accuracy:
{accuracy_rf:.3f} Precision: {precision_rf:.3f} "
                                                 f"Recall: {recall_rf:.3f}
F1_Score: {f1_rf:.3f}")
            # 4) SUPPORT VECTOR MACHINES (SVM)
            model_svm = SVC()
            # Train SVC on train data
            start_time_svm = time.time()  # Record the start time SVM
            model_svm.fit(x_train_umap, y_train)
            end_time_svm = time.time()  # Record the end time SVM
            elapsed_time_svm = end_time_svm - start_time_svm  # Calculate
SVM
            window['-OUTPUT-UMAP-RF_SVM-'].print(f"SVM operation took
{elapsed_time_svm:.2f} seconds.")
            # Predict on the test data
            y_pred_svm = model_svm.predict(x_test_umap)
            # Calculate evaluation metrics
            accuracy_svm = accuracy_score(y_test, y_pred_svm)
            precision_svm = precision_score(y_test, y_pred_svm,
average='weighted')
            recall_svm = recall_score(y_test, y_pred_svm,
average='weighted')
            f1_svm = f1_score(y_test, y_pred_svm, average='weighted')
            window['-OUTPUT-UMAP-RF_SVM-'].print(f"Accuracy:
{accuracy_svm:.3f} Precision: {precision_svm:.3f} "
                                             f"Recall: {recall_svm:.3f} F1_Score:
{f1_svm:.3f}")
```

```
            # Visual UMAP
            labels_str = ["Normal", "Suspect", "Pathological"]
            plt.figure(figsize=(12, 7))
            scatter = plt.scatter(x_umap[:, 0], x_umap[:, 1], c=y,
label=labels_str, cmap='viridis')
            plt.legend(handles=scatter.legend_elements()[0],
labels=labels_str, loc="lower right")
            plt.colorbar(label='Classes')
            plt.title('UMAP Projection of Fetal Health Dataset')
            plt.savefig('UMAP.png')
        except:
            sg.popup_auto_close("Oops! Browse 'fetal_health.csv' first!")

    elif event == 'Open UMAP Picture':
        if os.path.isfile("UMAP.png"):
            sg.popup_ok('UMAP_method', image="UMAP.png")
        else:
            sg.popup_auto_close("Click the button '4.UMAP+RandomForest_SVM'
first ")

window.close()
```

ДОДАТОК Б



UNICHECK
by Turnitin

Ім'я користувача:
приховано налаштуваннями конфіденційності

ID перевірки:
1016051785

Дата перевірки:
09.01.2024 22:04:13 EET

Тип перевірки:
Doc vs Library

Дата звіту:
09.01.2024 22:07:27 EET

ID користувача:
43927

Назва документа: 122_магістр_2023_Хлистун_Олексій_Вікторович

Кількість сторінок: 71 Кількість слів: 10832 Кількість символів: 85408 Розмір файлу: 2.55 MB ID файлу: 1015752545

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

4.89%
Схожість

Найбільша схожість: 3.22% з джерелом з Бібліотеки (ID файлу: 1015344513)

Пошук збігів з Інтернетом не проводився

4.89% Джерела з Бібліотеки 136 ..... Сторінка 73

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

2.96%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за 8 слів та 0%)

Немає вилучених Інтернет-джерел

2.96% Вилученого тексту з Бібліотеки 132 ..... Сторінка 73

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 6

Підозріле форматування 14 сторінок