

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 – Комп'ютерні науки,
освітня програма «Інформаційна аналітика та впливи»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему:

“ Прогнозування серцево-судинних хвороб методами машинного навчання ”

Студента 2-го курсу групи ІАВ-21

Лавриновича Владислава Валерійовича
(прізвище, ім'я, по батькові)

Науковий керівник:

доктор технічних наук, доцент
(науковий ступінь, вчене звання)

Хлевна Юлія Леонідівна
(прізвище, ім'я, по батькові)

(підпис студента)

(дата)

(підпис)

Попередній захист:

(Висновок: «До захисту в Екзаменаційній комісії»)

Завідувач кафедри
технологій управління

(підпис)

(прізвище, ініціали)

(дата)

Київ – 2023

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА
ШЕВЧЕНКА**
Факультет інформаційних технологій

Кафедра технологій управління
Освітньо-кваліфікаційний рівень Магістр
Спеціальність 122 - Комп'ютерні науки
Освітня програма Інформаційна аналітика та впливи

ЗАТВЕРДЖУЮ
Завідувач кафедри
професор Морозов В.В.

“__” _____ 20__р

З А В Д А Н Н Я
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студент Лавринович Владислав Валерійович

Група ІАВ-21

1. Тема кваліфікаційної роботи: «Прогнозування серцево-судинних хвороб методами машинного навчання»

Затверджена наказом по від «08» грудня 2022р. №5.

2. Строк подання студентом готової роботи – “17” травня 2023р.

3. Цільова установка та вихідні дані до роботи: дослідження особливостей використання методів машинного навчання для прогнозування серцево-судинних хвороб, розробка та оцінка моделей машинного навчання, створення інтерфейсу та впровадження найрезультативніших моделей

4. Зміст роботи: аналіз предметної області та обґрунтування доцільності застосування методів машинного навчання для передбачення серцево-судинних хвороб, огляд попередніх застосувань та досліджень, аналіз методів та алгоритмів, аналіз даних, будування моделей, оцінка моделей, створення інформаційної системи та впровадження моделей.

5. Перелік графічного матеріалу (слайдів) : приклад даних з датасету, кореляційна теплова карта даних, 3д графік розподілення даних за результуючою ознакою, графік розподілення даних за віком, засоби реалізації моделей, оптимальний графік навчання нейронної мережі, точність застосованих алгоритмів, результати критичного тесту, інтерфейс розробленого додатку, приклад концепту для стенду медичного закладу.

6. Календарний план виконання роботи:

№ п/п	Назва частин роботи	%	Виконання роботи	
			За планом	Фактично
1.	Вибір теми дипломної роботи	3	27.11.22	01.10.22
2.	Протокол кафедри ТУ про затвердження тем дипломних робіт та призначення наукових керівників	2	28.11.22	28.11.22
3.	Формування переліку нормативних матеріалів, літератури з проблематики дипломної роботи	10	08.01.23	08.01.23
4.	Складання розгорнутого плану кваліфікаційної роботи	5	17.01.23	17.01.23
5.	Ознайомлення наукового керівника з розгорнутим планом кваліфікаційної роботи. Внесення змін.	5	20.01.23	20.01.23
6.	Підготовка розділу 1 «Аналіз теоретико-методологічних основ застосування алгоритмів машинного навчання в прогнозуванні серцево-судинних захворювань»	10	14.02.23	14.02.23
7.	Підготовка розділу 2 «Алгоритми машинного навчання для прогнозування серцево-судинних захворювань»	14	10.03.23	10.03.23
8.	Підготовка розділу 3 «Побудова та оцінка моделей машинного навчання»	14	03.04.23	03.04.23
9.	Підготовка розділу 4 «Застосування побудованих моделей для прогнозування серцево-судинних захворювань в інформаційній системі»	13	17.04.23	17.04.23
10.	Оформлення кваліфікаційної роботи. Підготовка висновків і пропозицій	15	01.05.23	01.05.23
11.	Передача кваліфікаційної роботи науковому керівникові	2	04.05.23	04.05.23
12.	Передача кваліфікаційної роботи рецензенту для рецензування	2	08.05.23	08.05.23
13.	Попередній захист кваліфікаційної роботи	5	17.05.23	17.05.23

Дата видачі завдання «08» грудня 2022р.

Керівник роботи доктор технічних наук, доцент Хлевна Юлія Леонідівна
(посада, прізвище, ім'я, по батькові)

(підпис)

Завдання прийняв до виконання студент групи ІАВ-21

Лавринович Владислав Валерійович

(прізвище, ім'я, по батькові)

ЗМІСТ

АНОТАЦІЯ.....	7
ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ.....	9
ВСТУП.....	10
РОЗДІЛ 1. АНАЛІЗ ТЕОРЕТИКО-МЕТОДОЛОГІЧНИХ ОСНОВ ЗАСТОСУВАННЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ В ПРОГНОЗУВАННІ СЕРЦЕВО-СУДИННИХ ЗАХВОРЮВАНЬ.....	12
1.1 Аналіз об'єкту дослідження.....	12
1.2 Аналіз застосування інтелектуального аналізу даних в медицині	14
1.3 Аналіз необхідності застосування алгоритмів машинного навчання в при прогнозуванні серцево-судинних захворювань	19
1.4 Аналіз попереднього використання моделей машинного навчання при прогнозуванні серцево-судинних захворювань	19
1.5 Постановка задачі.....	34
Висновок до розділу 1.....	34
РОЗДІЛ 2. АЛГОРИТМИ МАШИННОГО НАВЧАННЯ ДЛЯ ПРОГНОЗУВАННЯ СЕРЦЕВО-СУДИННИХ ЗАХВОРЮВАНЬ.....	36
2.1 Метод опорних векторів (SVM)	36
2.2 Наївний Байєсівський алгоритм	38
2.3 Дерево рішень.....	39
2.4 Random Forest	44
2.5 XGBoost.....	47
2.6 Глибокі нейронні мережі.....	49
2.7 Основні бібліотеки та інструменти	56
2.8 Алгоритм вирішення задачі	57
Висновок до розділу 2.....	58
РОЗДІЛ 3. ПОБУДОВА ТА ОЦІНКА МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ	59
3.1 Аналіз даних для моделей.....	59
3.2 Побудова моделей	66
3.3 Отримання прогнозів	70

3.4 Оцінка якості моделей	73
Висновок до розділу 3.....	76
РОЗДІЛ 4. ЗАСТОСУВАННЯ ПОБУДОВАНИХ МОДЕЛЕЙ ДЛЯ ПРОГНОЗУВАННЯ СЕРЦЕВО-СУДИННИХ ЗАХВОРЮВАНЬ В ІНФОРМАЦІЙНІЙ СИСТЕМІ	77
4.1 Аналіз технологічних інструментів.....	77
4.2 Створення та розгортання інформаційної системи	82
4.3 Перспективи використання застосунку	88
ВИСНОВКИ	91
СПИСОК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ.....	94
ДОДАТКИ.....	101
ДОДАТОК А.....	102
ДОДАТОК Б	105
ДОДАТОК В	110

АНОТАЦІЯ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 - Комп'ютерні науки,
освітня програма "Інформаційна аналітика та впливи"

Дипломна робота магістра Лавриновича Владислава Валерійовича.

Тема роботи – «Прогнозування серцево-судинних хвороб методами машинного навчання».

Мета дипломної роботи магістра – аналіз, розробка та впровадження моделей прогнозування серцево-судинних захворювань у пацієнтів методами машинного навчання.

Об'єкт дослідження – процеси прогнозування серцево-судинних захворювань за допомогою методів машинного навчання.

Предмет дослідження – моделі, методи, технології машинного навчання для прогнозування серцево-судинних захворювань.

Наукова новизна роботи – розроблено методику застосування алгоритмів машинного навчання для передбачення серцево-судинних хвороб. Отримані в результаті роботи моделі з досить високою точністю прогнозування впроваджено в інформаційну систему, що дозволяє пацієнтам та лікарям проводити ранню діагностику та попередити серцево-судинні захворювання.

У роботі досліджуються існуючі підходи до використання алгоритмів машинного навчання для прогнозування серцево-судинних захворювань. Розробляються та впроваджуються моделі машинного навчання для прогнозування серцево-судинних захворювань, а також проводиться обґрунтування доцільності та необхідності впровадження запропонованої інформаційної системи. Наводяться рекомендації щодо практичного використання та подальшого розвитку системи.

Дипломна робота складається зі вступу, основної частини, яка включає чотири розділи, висновків та списку використаних джерел. Всього налічує 90 сторінок та перелік посилань з 46 джерел на 7 сторінках.

Ключові слова: машинне навчання, серцево-судинні хвороби, прогнозування, інформаційна система, нейронна мережа.

ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ

ІМТ – індекс маси тіла

ІХС – ішемічна хвороба серця

ССЗ – серцево-судинні захворювання

ANN – Artificial Neural Network

DNN – Deep Neural Network

CNN – Convolutional Neural Network

RF – Random Forest

DT – Decision Tree

GB – Gradient Boost

NB – Naïve Bayes

CAD – Coronary Artery Disease

SVM – Support Vector Machine

KNN – K-Nearest Neighbours

ML – Machine Learning

КТ – комп'ютерна томографія

МРТ – магнітно-резонансна терапія

ВСТУП

В наш час, з стрімким розвитком інформаційних технологій, все більша увага прикута до застосування інформаційних технологій та інтелектуального аналізу даних в передбаченні та лікуванні різних хвороб.

Індустрія охорони здоров'я є одним із найбільш значущих бенефіціарів Data Science. Завдяки аналітикам даних медична діагностика стає ефективнішою та доступнішою, лікування персоналізованим, а медичні дослідження більше орієнтованими на дані.

Медицина може використовувати алгоритми аналізу даних для запобігання поширеним захворюванням за допомогою передбачення захворювань на основі різних метрик. Прогнозні моделі використовують наявні дані, аналізують їх та інтерпретують, щоб встановити кореляції та забезпечити точні прогнози. Наука про дані також допомагає зрозуміти стан здоров'я людини та надає допомогу, щоб запобігти майбутнім ризикам.

Серцево-судинні захворювання (ССЗ) зараз є найбільшою проблемою в медичній галузі. Це одні з найбільш смертельних і хронічних захворювань, які призводять до найбільшої кількості смертей. За останніми статистичними даними Всесвітньої організації охорони здоров'я (ВООЗ) [1], щорічно від серцево-судинних захворювань помирає 20,5 мільйона людей, тобто приблизно 31,5% усіх смертей у світі. Також підраховано, що кількість щорічних смертей зросте до 24,2 мільйонів до 2030 року. Близько 85% смертей від серцево-судинних захворювань пов'язані з серцевими нападами та інсультами [2].

Застосування аналізу даних в цій області дозволить вчасно діагностувати багато серцевих захворювань та підвищити рівень обізнаності серед громадян. В даній роботі ми будемо розглядати атеросклероз як вид серцево-судинних захворювань та проведемо дослідження над даними, які складаються з клінічних показників хворих та здорових людей. Розроблені моделі та

інформаційна система можуть використовуватись в медичних закладах під час проведення різних аналізів а також в медичних лабораторіях.

У відповідності із поставленою метою сформовано такі завдання дослідження: аналіз датасету та моделей для прогнозування серцево-судинних захворювань, розробка моделей для передбачення серцевих хвороб з метою їх раннього попередження. Тренування тестування та порівняння результатів роботи алгоритмів машинного навчання, визначення найефективніших алгоритмів. Впровадження натренованих моделей для використання пацієнтами під час діагностики серцево-судинних захворювань.

Об’єкт дослідження – процеси прогнозування серцево-судинних захворювань технологіями Data Science.

Предмет дослідження – моделі, методи, технології (методології) Data Science прогнозування серцево-судинних захворювань.

Апробація результатів роботи. Автор виступав доповідачем на VIII International conference “Information Technology and Implementation” (IT&I-2021). За результатами конференції стаття опублікована та індексована в Scopus.

За результатами магістерської роботи, здобувачем підготовлено роботу на Всеукраїнський конкурс студентських наукових робіт зі спеціальності 122 Комп’ютерні науки, яка зайняла перше місце (I тур).

РОЗДІЛ 1. АНАЛІЗ ТЕОРЕТИКО-МЕТОДОЛОГІЧНИХ ОСНОВ ЗАСТОСУВАННЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ В ПРОГНОЗУВАННІ СЕРЦЕВО-СУДИННИХ ЗАХВОРЮВАНЬ

1.1 Аналіз об'єкту дослідження

Діагностика є важливою частиною циклу догляду за пацієнтом, оскільки вона визначає характер лікування, яке буде надано. Діагностика є першим кроком для правильного дослідження будь-якої хвороби, а також може визначити характер лікування, яке має бути надано. Навіть у епоху пікових технологій методика діагностики 21 століття далека від досконалості.

Помилки, спричинені лише в клінічній практиці в США, були причиною від 40 000 до 80 000 смертей у 2018 році. Наука про дані, та аналітики з методами глибокого навчання можуть поставити більш точний діагноз. Завдяки певним функціям аналіз даних може дозволити виявити ранні ознаки та дати підказки щодо профілактики завчасно.

Інструменти Data Science та Big Data показали позитивні результати в прогнозуванні, профілактиці та розробці планів лікування. Мобільні пристрої, розумні пристрої, датчики разом із Data Science можуть стати засобами попередження серцевого нападу. Наука про дані та хмарні обчислення зменшили відстань між лікарнями та світовими експертами з серцевих захворювань. Ці технології можуть бути використані для розробки системи охорони здоров'я на основі даних, де медсестри можуть допомогти надати першу допомогу, поки не будуть доступні лікарі-експерти. Наука про дані може зробити революцію в лікуванні серця і підвищити обізнаність [35] щодо нездорового способу життя який може призвести до серцевих захворювань.

Проблема серцевих захворювань полягає в тому, що перша допомога потрібна під час інфаркту та інсульту. У більшості випадків, з якими тикаються в усьому світі смерть від серцевого нападу настає через відсутність надання

першої допомоги [3]. Оскільки спосіб життя людини в усьому світі був змінений, що є основною основою для різних серцевих ускладнень, є очевидні дослідження, проведені для завчасного прогнозування захворювань серця, які показали точність майже 90% і вище.

Проблема в тому, що лише прогноз не може повністю виключити хворобу з організму. Її можна виключити трьома важливими основними факторами

1. Медицина
2. Запобіжні заходи
3. Зміна способу життя, пропонуючи фізичну активність, враховуючи різні особливості пацієнтів.

Методи, які в даний час використовуються для прогнозування та діагностики захворювань серця, створені на основі аналізу історії хвороби пацієнта, симптомів та протоколів оглядів лікарів. У більшості випадків медичним експертам важко точно передбачити захворювання серця у пацієнта, у більшості випадків вони можуть передбачити захворювання з точністю до 67%.

В даний час діагностика будь-якого захворювання проводиться за схожими симптомами [4], які спостерігаються у пацієнтів з раніше діагностованими захворюваннями.

Отже, медична галузь потребує автоматизованої інтелектуальної системи для точного прогнозування захворювань серця. Це може бути досягнуто шляхом використання величезної кількості даних про пацієнтів, які є в медичній галузі, разом з алгоритмами машинного навчання. Останнім часом дослідницькі групи Data Science приділили велику увагу прогнозуванню захворювань. Це пояснюється швидким розвитком передових комп'ютерних технологій у сфері охорони здоров'я, а також доступністю масивних баз даних охорони здоров'я.

Поєднання глибокого навчання та розумної системи прийняття рішень мають великий потенціал для покращення медичної допомоги в нашому

суспільстві. Дані є найціннішим ресурсом для отримання нових або додаткових знань і збирання важливої інформації. Існує величезна кількість даних (великих даних) різних секторів, таких як наука, технології, сільське господарство, бізнес, освіта та охорона здоров'я.

Це повністю необроблені дані, у структурованій або неструктурованій формі. Наразі в секторі охорони здоров'я інформація, що стосується пацієнтів із медичними висновками, легко доступна в базах даних і з кожним днем швидко зростає. Це сирі дані дуже надлишкові та незбалансовані. Потрібна попередня обробка для вилучення важливих метрик, що скорочують час виконання алгоритмів навчання та покращують класифікацію

Останні досягнення в області обчислювальних можливостей і програмування та можливості машинного навчання покращують ці процеси та відкривають двері для досліджень в секторі охорони здоров'я, особливо щодо раннього прогнозування захворювань, таких як ССЗ [36] та рак, для підвищення рівня виживання.

Машинне навчання використовується в широкому діапазоні застосувань, від виявлення факторів ризику захворювань до розробки передових систем безпеки для автомобілів. Одним із потужних методів машинного навчання для передбачення є класифікація. Класифікація є ефективним методом машинного навчання під наглядом для виявлення захворювання при навчанні з використанням відповідних даних.

1.2 Аналіз застосування інтелектуального аналізу даних в медицині

Машинне навчання пропонує принциповий підхід до розробки складних, автоматичних і об'єктивних алгоритмів для аналізу багатовимірних і мультимодальних біомедичних даних. Моделі машинного навчання нещодавно увійшли в медичну діагностику та продемонстрували величезний потенціал

завдяки підвищенню точності виявлення захворювань [37]. Його використовували, щоб допомогти лікарям приймати обґрунтовані рішення на основі даних пацієнтів [21]. Хоча машинне навчання не може замінити людський фактор догляду за пацієнтами, воно здебільшого зменшило вартість діагностики та оптимізувало діагностичний етап. Машинне навчання використовувалося для діагностики декількох захворювань навіть із більшою точністю, ніж більшість медичних працівників. А розробка моделей глибокого навчання зробила революцію в тому, як лікарі інтерпретують медичні зображення.

Серцево-судинні захворювання - це сімейство захворювань, які вражають серце та кровоносні судини, що оточують серце. Ця хвороба поєднує в собі інші захворювання, такі як «ішемічна хвороба серця (ІХС)», серцеві напади, фібриляція передсердь, аневризма аорти, венозний тромбоз, вроджені вади серця, інфекційний ендокардит, тромбоз глибоких вен. Щороку багато людей помирає від цієї хвороби. За оцінками, близько «17,9 мільйонів людей померли від серцево-судинних захворювань у 2019 році, і 85% відсотків з них були спричинені інфарктом та інсультом. Серцеві напади та інсульти зазвичай є гострими явищами і в основному спричинені блокадою судин [38], яка перешкоджає надходженню крові до серця чи мозку». Набір даних про хвороби серця UCI та інструмент інтелектуального аналізу даних WEKA можна використовувати для отримання корисних даних для навчання моделей машинного навчання. Алгоритми машинного навчання, такі як SVM, наївний класифікатор Байеса, випадковий ліс і логістичні регресії, дають багатообіцяючі результати у виявленні серцево-судинних захворювань використовуючи датасет UCI [22].

Діабет є дуже смертельною хворобою та величезною проблемою для суспільства. Цукровий діабет починає розвиватися, коли людина споживає в своєму раціоні надзвичайно високий рівень цукру. Якщо ми не подбаємо про

цю хворобу на ранньому етапі, вона може перерости в дуже шкідливу хворобу з багатьма ускладненнями. Згідно з нещодавнім дослідженням, діабет може спричинити хворобу Альцгеймера та сліпоту, а деякі люди зазнали проблем з нирками через високий рівень цукру в крові

Досі не знайдено ліків від цієї хвороби, хоча є способи контролювати її. Нещодавно алгоритми машинного навчання показали величезний потенціал для раннього виявлення діабету за допомогою зібраних від пацієнтів даних. А. Mujumdar та V. Vaidehi [23] у своєму дослідженні реалізували різні алгоритми машинного навчання для прогнозування діабету. Ці алгоритми включають SVM, Random Forest, Decision Tree, класифікатор Extra Tree Classifier, алгоритм Ada Boost, персептрон, алгоритм лінійного дискримінантного аналізу, логістичну регресію, KNN, наївний метод Байєса за Гауссом, алгоритм пакетування, Gradient Boost. Відповідно до їхньої оцінки, вони отримали найкращу продуктивність для логістичної регресії, яка мала точність 96%, а після використання конвеєрної обробки вона була оптимізована до 97,5%. Після конвеєрної обробки AdaBoost Classifier мав найвищу точність 98,8%.

Медична візуалізація в радіології використовується медичними працівниками для діагностики захворювань шляхом спостереження за тим, що відбувається всередині тіла, не вимагаючи хірургічного втручання чи інших інвазивних процедур. Це дозволяє медичним експертам оцінити ситуацію за зображеннями та призначити необхідні процедури для вирішення проблеми. Візуалізація зменшує час і ресурси, витрачені на проведення непотрібних тестів, і допомагає лікарям прийняти обґрунтоване рішення щодо пацієнта. Деякі з найпоширеніших типів медичних зображень, які використовуються для діагностики пацієнтів, включають КТ (комп'ютерну томографію), МРТ (магнітно-резонансну томографію), ультразвук, рентгенівські та ядерно-медичні зображення (включно з позитронно-емісійною томографією (ПЕТ)). Хоча медичні зображення існують уже давно, моделі потребують тривалого навчання,

щоб їх правильно інтерпретувати. Навіть маючи належний рівень навчання, клініцисти схильні робити помилки. Зазвичай для остаточного тлумачення потрібен консенсус між різними експертами в галузі охорони здоров'я. Зазвичай це призводить до розбіжностей і знижує точність діагнозу. У результаті дослідники працювали над тим, щоб використати деякі з найкращих корисних аспектів машинного навчання, щоб пришвидшити процес і підвищити точність виявлення. Алгоритми поглибленого навчання виявилися корисними для вилучення ознак із зображень, які потім використовуються для класифікації, виявлення та сегментації зображень. Нещодавно запроваджені DNN, особливо CNN, покращила ефективність класифікації на основі зображень у різних медичних додатках, включаючи діагностику туберкульозу, діабету та раку [24].

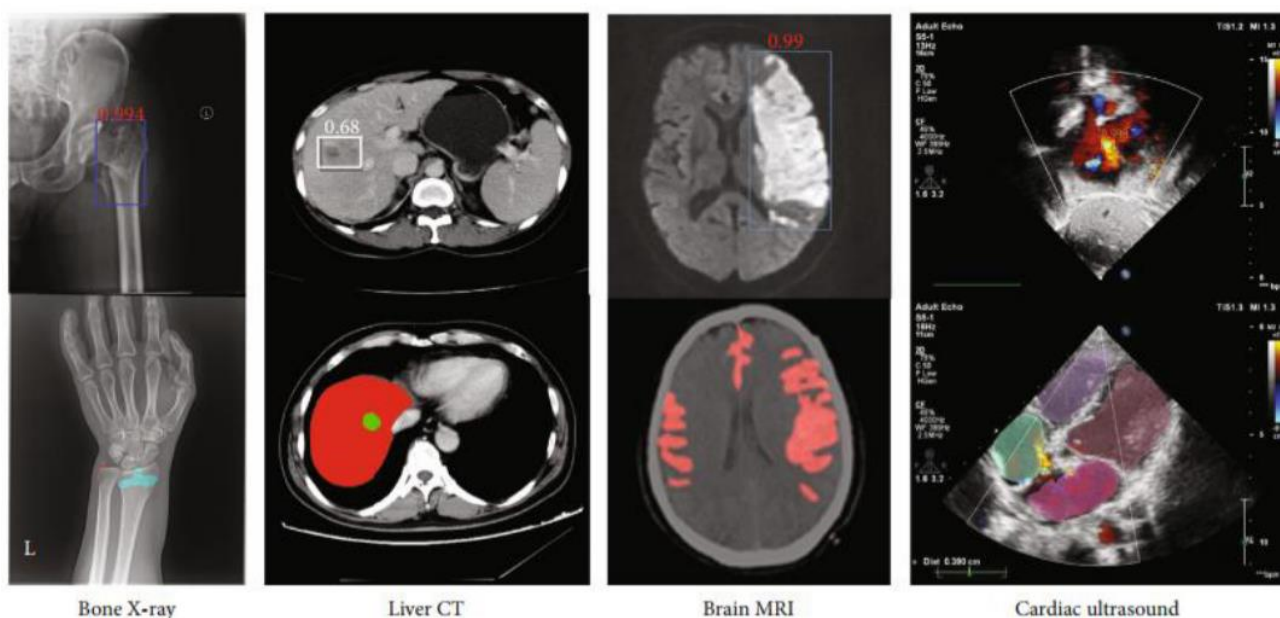


Рисунок 1.1 – Приклади медичних зображень та застосування методів машинного навчання

Рак — це загальний термін для сотень різних захворювань. Існують різні типи раку, але всі раки починаються, коли рівень аномальних клітин перевищує певну межу. Протягом останніх десяти-двадцяти років дослідники працювали над різними видами раку, щоразу надаючи нам нове й точне рішення. Згідно з дослідженнями, рак легенів, рак шкіри, рак крові, рак молочної залози та рак

шлунка є найпоширенішими видами раку. Як наслідок, різні методи машинного навчання та інтелектуального аналізу даних використовуються для прогнозування раку [39-40], деякі з яких є майже абсолютно точними. За останні 5 років було розроблено кілька оптимізованих моделей раку молочної залози з використанням добре відомого алгоритму та методів прогнозування [25]. Існує безліч статей і оглядів про різні типи досліджень і прогнози щодо лікування раку, типів раку та підтипів раку.

Моделі ML можуть отримати хороші результати на навчальному наборі, але при застосуванні до реального світу точність значно знижується. Причина полягає в тому, що навчальний набір є ретельно відібраними та повністю маркованими зразками. Навпаки, зразки реального світу мають різні фонові шуми, а деякі винятки не з'являються в навчальному наборі, що призводить до зниження точності.

Рішення полягає в тому, щоб збільшити розмір вибірки навчального набору та підвищити здатність протистояти шуму. Однак збільшення розміру вибірки означає велику кількість маркування вручну. Великі медичні дані, особливо дані зображень, є надзвичайно трудомісткими, що обмежує поточну ефективність ML.

Людям відносно легко розпізнати зображення [26-27] але машинам це досить складно ідентифікувати деформовані зображення. Коли до системи додається людський шум, першою проблемою є зниження швидкості розпізнавання системи; але є й серйозніші проблеми. Дослідження виявили, що під час розпізнавання зображень контрзаходи, сформовані шляхом навмисного додавання тонких перешкод, призведуть до того, що система видасть помилку. Наприклад, на зображенні літака, який змінює невелику кількість пікселів для створення конкурентної вибірки, люди не знайдуть на ньому зміни та все одно розпізнають його як літак. Тим не менш, модель може розпізнати це як інші абсолютно не пов'язані об'єкти.

1.3 Аналіз необхідності застосування алгоритмів машинного навчання в при прогнозуванні серцево-судинних захворювань

Діагностика та прогноз серцево-судинних захворювань є найважливішими медичними завданнями правильна класифікація, яка допомагає кардіологам надати пацієнту належне лікування. Кількість застосувань машинного навчання в медичній галузі зростає, оскільки вони можуть розпізнавати шаблони з даних і допомагати лікарям та пацієнтам автоматизовуючи медичні процеси.

Використання машинного навчання для класифікації серцево-судинних захворювань може допомогти лікарям зменшити шанс неправильного діагностування [41]. Діагностика та лікування серцево-судинних захворювань дуже складні, особливо в країнах, що розвиваються, через відсутність діагностичних пристроїв і брак лікарів та інших ресурсів, що впливають на правильне прогнозування та лікування серцево-судинних пацієнтів. У зв'язку з цим останнім часом комп'ютерні технології та методи машинного навчання [32-33] використовуються для розробки програмного забезпечення, щоб допомогти лікарям прийняти рішення про захворювання серця на ранніх стадіях. Виявлення хвороби на ранніх стадіях і прогнозування ймовірності того, що людина ризикує захворіти серцево-судинними хворобами, може знизити рівень смертності.

Проблема ефективного використання величезних обсягів даних стає головною проблемою для сектору охорони здоров'я. Інтелектуальний аналіз даних забезпечує методологію та технологію для перетворення великих даних у корисну інформацію для прийняття рішень.

1.4 Аналіз попереднього використання моделей машинного навчання при прогнозуванні серцево-судинних захворювань

Широкий діапазон методологій і технік у Data Mining (DM) є однією з найбільших переваг, які можна застосувати до різних проблем науки про

здоров'я. Дослідники застосували різні методи DM, такі як аналіз правил асоціацій, кластеризація, класифікація, щоб покращити діагностику захворювань з хорошою точністю та низькою ймовірністю помилок. Існуюча література [42] вказує, що DM за допомогою класифікації є ефективнішим у прогнозуванні серцевих захворювань порівняно з кластеризацією, правилом асоціації та регресією.

Протягом останніх років було проведено кілька досліджень різних підходів аналізу даних до цього питання, але незважаючи на це, проблема залишається дуже складною і сьогодні. Інтелектуальний аналіз даних і машинне навчання — це найсучасніша технологія, яка дозволяє нам виявляти зв'язки між атрибутами великомасштабних даних і моделями тренування, щоб робити прогнози більш точними. Машинне навчання вже знайшло своє застосування в різних сферах медицини [5, 6] для прогнозування захворювань і виявилось досить ефективним залежно від завдань, алгоритмів та даних.

Варто зазначити, що проведено чимало досліджень, які проводили класифікацію для прогнозування діагностики захворювань серця з використанням різноманітних моделей і методів [8, 9, 10, 11]. Тим не менш, більшість із них були обмежені в даних. Такий підхід може не лише звужити поле дослідження, а й призвести до низки неточних висновків, спричинених відсутністю даних, необхідних для навчання моделі.

Для нашої моделі ми будемо використовувати реальний набір даних, наданий Національним інститутом серцево-судинної хірургії імені Амосова, який допоможе нам говорити не лише про теоретичні результати, а й знайти різні підходи до реальних випадків. На відміну від інших досліджень, ми не будемо обмежуватися прогнозуванням серцево-судинних захворювань, а спробуємо передбачити атеросклероз, який є передумовою серцево-судинних захворювань і недостатньо вивчений через описані вище причини. Таким

чином, дослідження представляє як науковий, так і практичний інтерес для аудиторії і може висвітлити важливі проблеми медицини.

На основі даних про кров'яний тиск, ліпіди плазми за допомогою фізичного тесту, для визначення ішемічної хвороби серця (ІХС) у пацієнтів та осіб без ІХС у популяції Південного Китаю було застосовано Support Vector Machine (SVM) [12] для подальшої профілактики. і лікування захворювання.

По-перше, SVM-класифікатор був побудований з використанням радіальної базисної функції ядра, лінійної функції ядра та функції поліноміального ядра відповідно. По-друге, коефіцієнт похибки SVM C і параметр ядра σ були оптимізовані за допомогою оптимізації роя частинок (PSO), а потім використані для діагностики та прогнозування ІХС.

У порівнянні з результатами штучної нейронної мережі з моделлю зворотного поширення (BP), лінійним дискримінантним аналізом, методом логістичної регресії та неоптимізованим SVM, загальні результати розрахунку продемонстрували, що ефективність класифікації оптимізованої моделі RBF-SVM може бути кращою в порівнянні до іншого алгоритму класифікатора з вищими показниками точності, чутливості та специфічності, які склали 94,51%, 92,31% та 96,67% відповідно. Таким чином, можна зробити висновок, що SVM може бути використаний як дійсний метод для допомоги в діагностиці ІХС.

Інші підходи описують ефективність системи підтримки прийняття рішень для прогнозування ризику серцевої недостатності. Ця система заснована на штучній нейронній мережі (ANN) і Fuzzy-АНР. Досягнуті результати показали, що запропонований метод може досягти середньої точності прогнозування 91,10% порівняно зі звичайним методом штучної нейронної мережі.

Зовсім недавно, у 2018 році, автори обговорювали розробку та впровадження експертної системи для серцево-судинних захворювань. Автори розробили цю систему з використанням Fuzzy-АНР і Fuzzy Inference System

(FIS) [34]. Результати розробленого підходу показали ймовірність розвитку серцевих захворювань. Згідно з експериментальними результатами, ця система показала, що методи штучного інтелекту і машинного навчання в медицині досягли хороших результатів.

У роботі [12] автори використали три класифікатори: наївний байєсівський алгоритм, C4.5 та метод опорних векторів (SVM). Ці методи були застосовані до бази даних Z-Alizadeh Sani для діагностики CAD для виявлення стенозу трьох коронарних артерій, тобто лівого окружного згину (LCX), лівої передньої низхідної (LAD) та правої коронарної артерії (RCA). Результат цього дослідження досяг 96,40% як найвищої точності для виявлення CAD.

У 2019 році автори запропонували комбінацію моделі nu-Support Vector Classification (NEnu SVM), яка поєднує різні методи машинного навчання та методи комбінованого навчання для прогнозування CAD, використовуючи два набори даних: Cleveland і Z Alizadeh Sani. Запропонована модель досягла найвищої точності 94,66% з Z-Alizadeh Sani і 98,60% з наборами даних Cleveland CAD для прогнозування CAD.

Також нещодавно, дослідники розробили новий метод під назвою Hybrid Feature Selection (2HFS) з використанням класифікаторів Gaussian Naive Bayes (GNB) [13], Random Forest (RF) [14, 15, 16], Decision Tree (DT) і XGBoost [17, 18]. У цьому дослідженні автори використовували базу даних Nasarian CAD. Вони також протестували цей підхід з базами даних Угорщини, Long Beach та Z-Alizadeh Sani, щоб досягти показників точності 83,94%, 81,58% і 92,58% відповідно.

Aadar Pandita, Sarita Yadav зробили дослідження, під назвою “Prediction of Heart Disease using Machine Learning Algorithms”. В цьому дослідженні вони використовували UCI датасет, та кілька алгоритмів машинного навчання таких як Naive Bayes, K-Nearest Neighbor, Decision Tree і Random Forest, які

взаємодіють, щоб знайти найточнішу модель. Етап попередньої обробки даних вони поділили на чотири підетапи.

Очищення: дані, які вони хочуть обробити, не будуть чистими, тобто вони можуть містити шум або значення, яких не вистачає під час обробки. Вони не можуть отримати хороші результати, тому для отримання хороших і ідеальних результатів їм потрібно все це усунути. Процес усунення всього цього є очищення даних. Вони заповняють відсутні значення та можуть усунути шум за допомогою деяких методів, як-от заповнення найбільш поширеним значенням у пропущеному місці.

Трансформація: це передбачає зміну формату даних одної форми в іншу, що робить їх найбільш зрозумілими за допомогою нормалізації, згладжування та узагальнення, агрегації

Інтеграція: Дані, які їм не потрібно обробляти, можуть походити не з одного джерела, іноді вони можуть надходити з різних джерел, вони їх інтегрують. Під час обробки це може бути проблемою, тому інтеграція є одним із важливих етапів попередньої обробки. Тут розглядаються різні питання для інтеграції.

Зменшення: коли вони працюють над даними, іноді розмірність може бути високою, а дані складними і важко зрозуміти, тому, щоб зробити їх зрозумілими системі, вони зменшують їх до необхідного формату, щоб досягти гарних результатів.

В загальному запропонована дослідниками система зображена на Рис. 1.1.

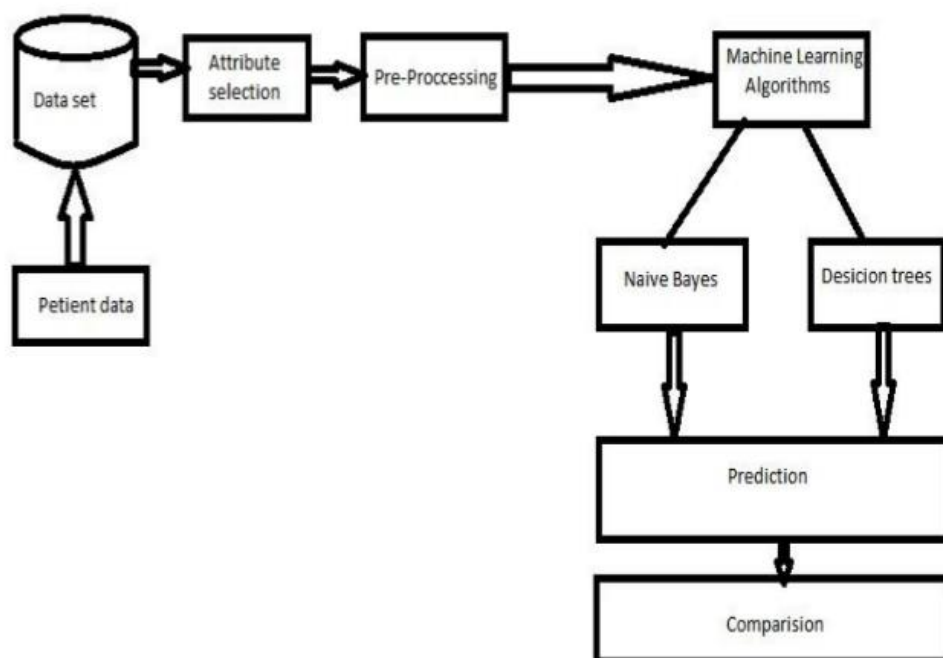


Рис. 1.1. Запропонована система для вирішення задачі

Після етапів відбору атрибутів та попередньої обробки, дослідники виділили наступні атрибути датасету, на основі яких в подальшому побудували моделі, наведені в Рис. 1.2.

Attribute	Values and meaning
Age1	Age in year
Gender1	Value 1 and 0 for male and female
Cp1	Pain in chest Yes/No
Trest bps1	blood pressure during resting
Chol1	Cholesterol of serum in mg/dl
Fbs1	blood sugar during fasting
Restecg1	Resting electrocardiographic results
Oldpeak1	ST depression induced by exercise relative to rest
Slope1	The slope of peak exercise of ST segment. Value 1: upsloping Value 2: flat Value 3: down sloping
Ca1	Number of major vessels (0-3) colored by flourosopy
Thal1	3 = normal; 6 = fixed defect
Num1	Diagonal of heart disease Value 0: No Risk; Value 1: Low Risk; Value 2: Risk; Value 3: High Risk; Value 4: Higher Risk

Рис. 1.2. Вихідні параметри датасету після попередньої обробки

За результатами роботи автори виявили, що поодинокі алгоритми дають неточні результати, тому найкращим рішенням є комбінація алгоритмів k-means, ID3 або k-means і Naïve Bayes.

Науковці Saleh Mahdi Muhammed, Ghassan Abdul-Majeed, Mahmoud Shuker Mahmoud зробили дослідження під назвою «Prediction oh heart diseases by using Supervised Machine Learning». У цьому дослідженні вони обговорюють кілька методологій, які мають передбачити наслідки серцевих захворювань за допомогою методів машинного навчання. Були застосовані наступні алгоритми: ANN, Naive Bayes, DT, Random Forest, і Gradient Boosting. Однак точність, отримана в кожному дослідженні, наразі не вважається задовільною, оскільки певні алгоритми працюють краще, ніж інші.

У цьому дослідженні було визначено п'ять алгоритмів, які показали задовільну точність через 10-fold кросс-валідацію, демонструючи потенціал для використання в прогнозуванні. Таким чином, мета дослідження полягала в тому, щоб виявити класифікатори, які можуть ефективно передбачати результати серцево-судинних захворювань і можуть мати практичне значення. За результатами дослідження автори запропонували наступну методологію, зображену на рис. 1.3.

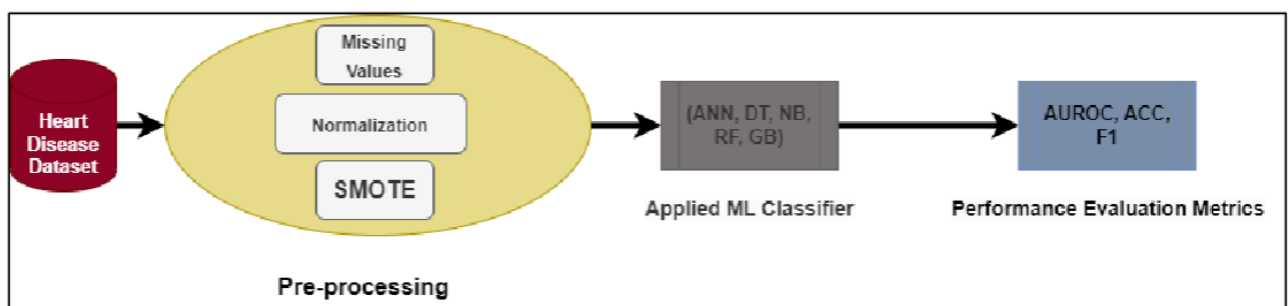


Рис. 1.3. Методологія запропонована дослідниками

Confusion matrix, створена для кожного алгоритму, включала чотири параметри: істинно позитивний (TP), істинно негативний (TN), хибно позитивний (FP) і хибно негативний (FN). Для оцінки алгоритмів використовувалися чутливість, специфічність і точність. Чутливість відноситься

до частки фактичних позитивних результатів, які були правильно визначені класифікатором, тоді як специфічність вказує на здатність класифікатора правильно ідентифікувати негативні результати.

Точність — це відсоток правильно класифікованих екземплярів класифікатором, і це середня наближеність до цілі. Для порівняння продуктивності різних алгоритмів використовувалися різні статистичні вимірювання, такі як точність, відкликання та F-міра.

Точність вимірює відсоток очікуваних позитивних результатів, які є фактичними позитивними, запам'ятовування представляє частку позитивних результатів, які правильно класифіковані, а F-вимірювання врівноважує точність і запам'ятовування для класифікатора. AUROC — це метрика продуктивності для розрізнення, яка оцінює здатність моделі розрізняти хворобливі випадки та здорові.

Щоб оцінити модель у цьому дослідженні, як для навчання, так і для тестування використовувався підхід K-fold cross-validation. Ця техніка передбачає поділ набору даних на K груп, або «згортки», де K означає кількість груп. K-fold cross-validation — це метод оцінки машинного навчання, у якому модель навчається на (K-1) групах, тоді як решта групи використовується для оцінки навченої моделі.

У цьому методі модель навчається K разів, при цьому кожне згортання використовується для оцінки моделі. У цьому дослідженні була використана техніка 10-кратної перехресної перевірки, щоб уникнути переобладнання в прогнозній моделі.

Для визначення найефективнішого алгоритму класифікації до набору даних було застосовано п'ять різних алгоритмів, а саме ANN, RF, DT, GB і NB, а їх точність та інші статистичні змінні порівнювали за допомогою техніки 10-кратної перехресної перевірки. Алгоритми були оцінені на основі показників продуктивності. Щоб визначити чутливість, специфічність і точність

результатів кожного алгоритму, була створена матриця помилок. Для обчислення цих показників були використані відповідні рівняння (1,2,3,4,5):

$$Accuracy = (TP+TN)/(TP+FP+TN+FN) \quad (1)$$

$$Precision = TP/(TP+FP) \quad (2)$$

$$Recall = TP/(TP+FN) \quad (3)$$

$$F1=2*(precision*recall)/(precision + recall) \quad (4)$$

$$AUROC: TP/(TP+FN) \quad (5)$$

Де кожен наступних параметрів охарактеризовує кількість передбачень того чи іншого типу: істинно позитивний (TP), істинно негативний (TN), хибно позитивний (FP) і хибно негативний (FN)

Продуктивність кожного алгоритму оцінювалася за допомогою різних показників перехресної перевірки, і було визначено найефективніший алгоритм. Весь процес показаний на рис. 1, а рис. 4 відображає результати продуктивності використаних класифікаторів.

Використані показники оцінки включають AUROC, точність, F1, точність і запам'ятовування, які були вибрані для забезпечення всебічної оцінки продуктивності кожного класифікатора та забезпечення справедливого порівняння між ними.

Результати дослідження показують, що ANN забезпечує максимальну точність, чутливість і специфічність, а потім йдуть Gradient Boosting, Decision Tree, Random Forest, і Naive Bayes. Це дослідження виявило кілька класифікаторів ML з потенціалом для точного виявлення захворювань серця, що може бути цінним для лікарів, які прагнуть передбачити виникнення захворювань серця у своїх пацієнтів. Однак слід зазначити, що набір даних, використаний у цьому дослідженні (Cleveland), містив обмежені дані про захворювання серця, і для створення більш надійної моделі прогнозування потрібні додаткові дані та аналіз.

Evaluation metrics of five supervised ML classifiers					
ML classifiers	AUC	Accuracy	F1	Precision	Recall
ANN	0.991574	0.941379	0.941149	0.941452	0.941379
Decision Tree	0.882650	0.743103	0.738877	0.737667	0.743103
Gradient Boosting	0.966494	0.85	0.847661	0.850678	0.85
Random Forest	0.930146	0.787931	0.783868	0.787009	0.787931
Naive Bayes	0.897837	0.746551	0.745020	0.746611	0.746551

Рис. 1.4. Результати досліджуваних алгоритмів

Незважаючи на це обмеження, ми очікуємо, що майбутні дослідження покращать наше розуміння переваг і обмежень цього підходу, а використання алгоритмів машинного навчання для аналізу додаткових даних призведе до високоточних прогнозів серцево-судинних захворювань і пов'язаних із ними станів.

Md. Julker Nayeem, Sohel Rana, і Md. Rabiul Islam у своєму дослідженні під назвою «Prediction of Heart Disease Using MachineLearning Algorithms» використали декілька алгоритмів машинного навчання і досягли точності в 95%.

У їх дослідницькій роботі були використані різні типи алгоритмів керованого машинного навчання, щоб передбачити наявність захворювань серця у пацієнтів. Вони також зосередилися на ефективному способі покращення продуктивності своїх прикладних класифікаторів. Для обробки нульових значень у їхньому наборі даних була використана техніка середнього значення. Параметри, які не є необхідними, видалялися за допомогою техніки вибору параметрів з посиленням інформації. Щоб обчислити точність передбачення, вони застосовували К-найближчих сусідів (KNN), Naive Bayes і Random Forest до свого набору даних про хвороби серця. Вони розраховували точність, прецизійність, запам'ятовування, F1-оцінку та ROC, щоб порівняти

ефективність своїх моделей класифікації. За допомогою техніки обробки нульових значень та вибору ознак з посиленням інформації вони змогли підвищити точність своїх моделей прогнозування. Зокрема, Random Forest дав найкращу точність класифікації - 95,63%, з точністю, запам'ятовуванням, оцінкою F1 та ROC 0,93, 0,92, 0,92 та 0,9 відповідно. Вони використовували методологію, зображену на рис. 1.5.

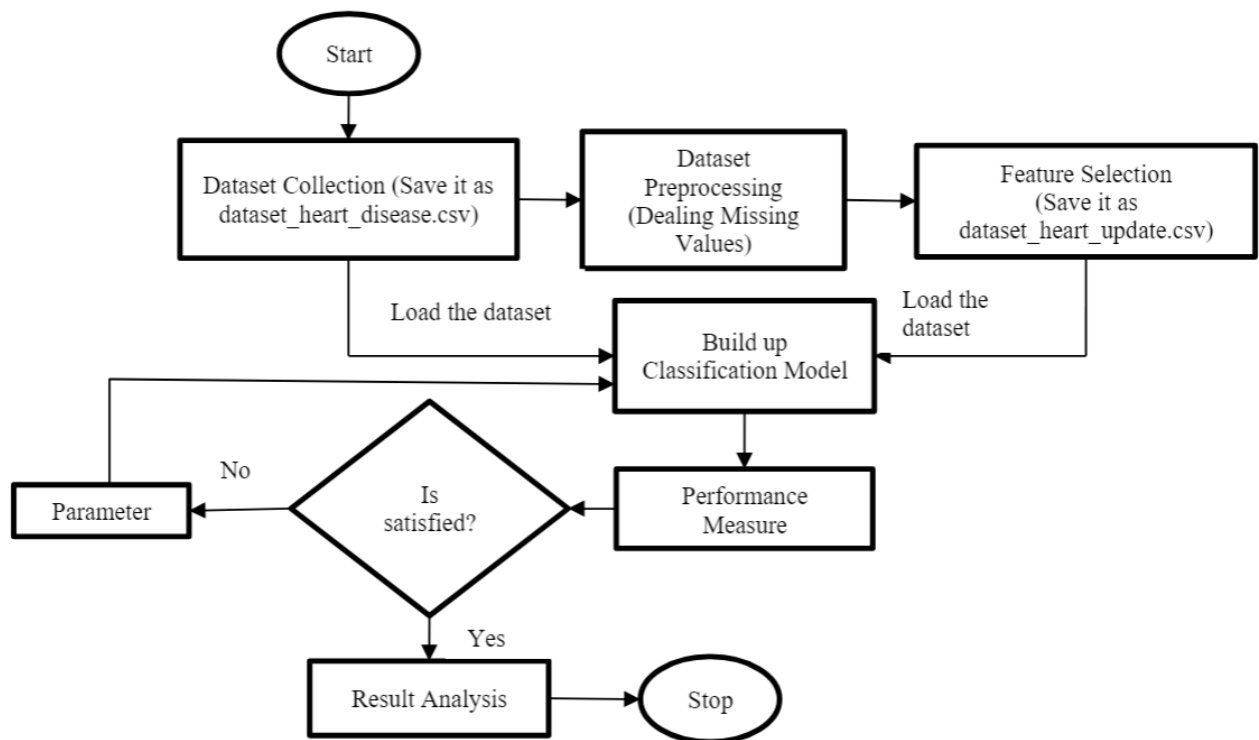


Рис. 1.5. Алгоритм проведення дослідження.

В загальному вони виділили наступні сім кроків виконання роботи:

Крок 1. Спочатку необхідно створити файл з назвою набору даних серцевих захворювань після збору набору даних із репозиторію машинного навчання UCI. Формат файлу - CSV (значення, розділені комами).

Крок 2: Наступним кроком є перевірка наявності нульових значень у кожному стовпці.

Крок 3: Якщо виявляються нульові значення, вони обробляються за допомогою техніки середньої вказівки.

Крок 4: Потім створюється новий файл CSV з назвою "heart update" після перевірки висококорельованих функцій за допомогою підходу до вибору функції отримання інформації для виявлення висококорельованих функцій.

Крок 5: Для визначення наявності серцевих захворювань завантажується набір даних "heart.csv" (містить усі атрибути та нульові значення) і класифікується за допомогою трьох класифікаторів.

Крок 6: Для визначення наявності серцевих захворювань завантажується набір даних "heart update.csv" (який містить лише сильно корельовані ознаки та виключає спостереження з нульовими значеннями) і класифікується за допомогою трьох класифікаторів.

Крок 7: Насамкінець, продуктивність моделі класифікації, отриманої на кроках 5 і 6, порівнюється за допомогою наборів даних "heart.csv" та "heart update.csv" відповідно.

Крок 8: Для порівняння точності моделі класифікації з результатами попередніх досліджень.

За результатами досліджень отримали результати, відображені на рис. 1.6.

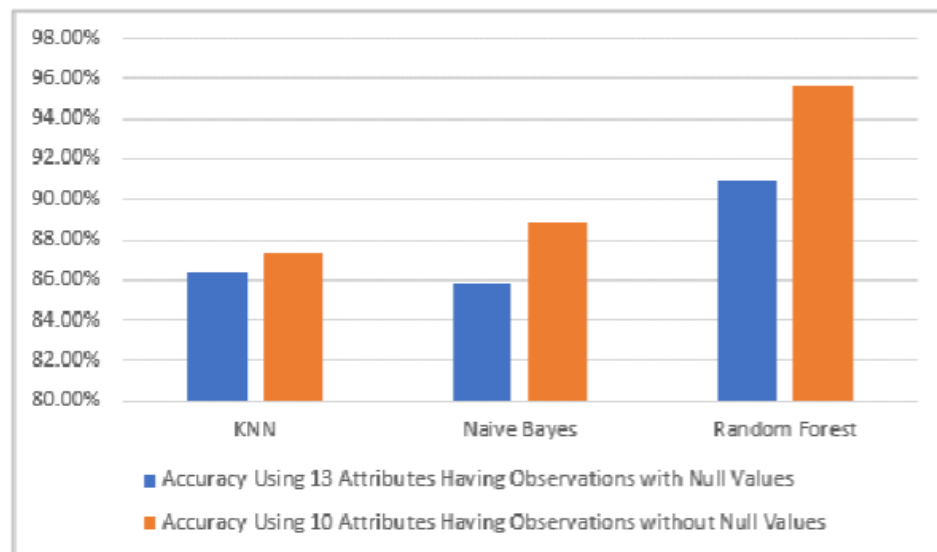


Рис. 1.6. Результати проведеного дослідження.

Arooj S, Rehman Su, Imran A, Almuhaimeed A, Alzahrani AK, Alzahrani A. у науковій публікації «A Deep Convolutional Neural Network for the Early Detection of Heart Disease» випробували використання згорткових нейронних мереж для передбачення серцево-судинних хвороб. Завдяки різноманітним технологіям і методам, розробленим для виявлення серцевих захворювань, використання класифікації зображень може ще більше покращити результати. Класифікація зображень є важливою проблемою в наш час. Це одна з найпростіших робіт з ідентифікації візерунків і комп'ютерного зору, яка стосується призначення однієї або кількох міток зображенням. Ідентифікація шаблонів із зображень стала легшою завдяки використанню машинного навчання, а глибоке навчання зробило його більш точним, ніж традиційні методи класифікації зображень. Це дослідження спрямоване на використання підходу глибокого навчання з використанням класифікації зображень для виявлення захворювань серця.

Глибока згорткова нейронна мережа (DCNN) наразі є найпопулярнішим методом класифікації для розпізнавання зображень. Запропонована модель оцінюється на загальнодоступному наборі даних про захворювання серця UCI, що включає 1050 пацієнтів і 14 атрибутів. Зібравши набір ознак, які можна отримати безпосередньо з набору даних про хвороби серця, автори вважають цей вектор ознак вхідним для DCNN, щоб визначити, чи належить примірник до класу здорових чи серцевих захворювань. Для оцінки ефективності запропонованого методу використовувалися різні показники ефективності, запам'ятовування та показник F1, і модель досягла точності перевірки 91,7%. Результати експерименту свідчать про ефективність запропонованого підходу в реальних умовах.

Архітектура запропонованої моделі DCNN представляє собою мережу прямого зв'язку з послідовною моделлю, в якій кожен рівень з'єднаний з одним входом і одним виходом. Атрибут класифікації захворювань серця є бінарним

атрибутом, який класифікується як «1» для пацієнтів з захворюванням серця та «0» для пацієнтів без захворювання серця. Модель мала 2 згорткових шари, за якими слідували 8 щільних шарів. 14 вибраних атрибутів були об'єднані в повністю пов'язаний щільний шар. Загалом для побудови каркасу CNN було використано 8 повністю з'єднаних щільних шарів. Перші чотири шари містили 128 нейронів, наступні - 64, а останній - 1 нейрон. Перед нелінійним перетворенням ці шари нормалізують змінні. Експоненціальна лінійна одиниця (ELU) використовувалася як функція активації, за винятком останнього шару. Сигмоїдна функція була використана як функція активації в останньому шарі моделі CNN. Оптимізатор Nadam використовувався зі швидкістю навчання 0,001. Для функції втрат було встановлено двійкову крос-ентропію. Під час фази навчання кількість епох було встановлено на 100 для кращої класифікації. Рівень випадання становив 3%, щоб уникнути перенавчання. Графіки тренування моделі та її точність зображені на рис. 1.7.

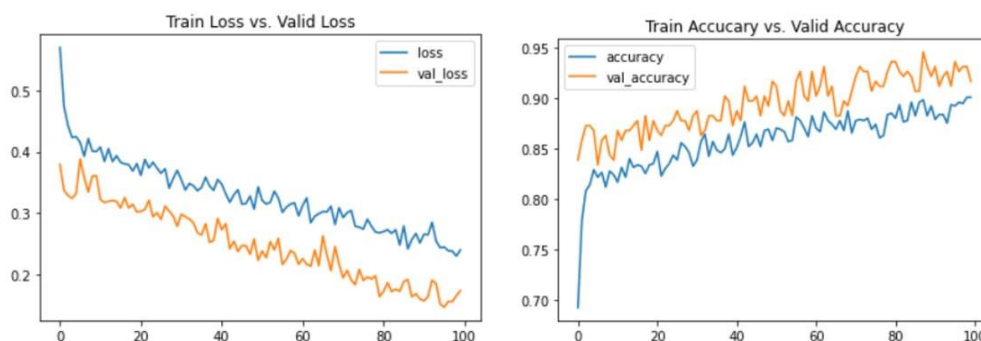


Рис. 1.7. Графіки навчання нейронної мережі.

Попередньо оброблені дані були використані з алгоритмом CNN для прогнозування захворювань серця на Google Collab. Запропоновану систему оцінювали щодо таких показників ефективності, як точність, точність, запам'ятовування та оцінка F1, і вона досягла 91,71%, 88,88%, 82,75% та 85,70% відповідно.

З проаналізованих досліджень ми можемо визначити, що найчастіше дослідники використовуються наступні алгоритми машинного навчання: Naïve Bayes, Decision Trees, Random Forest, Gradient Boosting та нейронні мережі. Всі проаналізовані роботи використовували доволі застрілий датасет UCI та показали непогані показники точності. В подальшій роботі ми спробуємо застосувати ті ж самі алгоритми на іншому датасеті та побудувати свою нейронну мережу. Таким чином ми спробуємо покращити точність роботи алгоритмів для передбачення серцево-судинних хвороб.

Syed Nawaz Pasha, Dadi Ramesh, Sallauddin Mohmmad, A. Harshavardhan і Shabana написали наукову статтю під назвою «Cardiovascular disease prediction using deep learning techniques». В цій статті вони використовували датасет з Kaggle. Вони застосували 3 алгоритми машинного навчання, а саме SVM, KNN та Decision Tree, а також побудували нейронну мережу. За результатами порівняння точностей алгоритмів, зображених на рис. 1.8 можна сміливо стверджувати що нейронна мережа має найбільшу точність з усіх представлених алгоритмів в більшості випадків. При цьому SVM проявляє кращу точність на менших датасетах.

Algorithm	Accuracy
SVM	81.97
KNN	67.2
Decision Tree	81.97
ANN(binary model)	85.24

Рис. 1.8. Порівняння точності використаних моделей.

1.5 Постановка задачі

Отже, з наведених вище фактів можна зрозуміти, що більшість відомих сьогодні алгоритмів машинного навчання вже були застосовані для передбачення різних серцево-судинних захворювань [46]. Тим не менш не всі з них досягли високих показників точності передбачень. Таким чином залишається широке поле для експериментів зі збільшення точності передбачень, а також комбінування існуючих та розробки власних алгоритмів та моделей для прогнозування серцево-судинних хвороб.

В цій роботі нам необхідно буде проаналізувати наявні дані та оцінити можливість та перспективи використання тих чи інших алгоритмів машинного навчання. У вище згаданих дослідженнях, моделі переважно використовувались для передбачення захворювань коронарної артерії (ішемічної хвороби серця). В даному дослідженні ми будемо ставити за мету передбачення атеросклерозу, який є мало дослідженим на даний час через брак даних.

Нам необхідно буде проаналізувати доступні моделі, для кожної моделі підготувати дані для тренування і тестування, провести тренування та оцінити точність моделей. Після цього нам необхідно буде проаналізувати характеристики кожної моделі окремо, а також зробити порівняльний аналіз моделей та обрати найефективнішу, яку і можна буде впроваджувати для використання та практичного застосування в медицині.

Висновок до розділу 1

У цьому розділі проаналізували сферу застосування методології Data Science на прикладі нашої роботи. Ми визначили актуальність дослідження, методологію реалізації проекту, та провели аналіз використання вже існуючих моделей. Також ми визначили основні задачі дослідження, та алгоритм виконання відповідно до методології. Також важливим кроком було визначення цілей та вимог до проекту для порівняння відповідності результату до вимог. В

наступних розділах ми будемо реалізувати проект відповідно до наведених в цьому розділі задач, цілей та методологій.

РОЗДІЛ 2. АЛГОРИТМИ МАШИННОГО НАВЧАННЯ ДЛЯ ПРОГНОЗУВАННЯ СЕРЦЕВО-СУДИННИХ ЗАХВОРЮВАНЬ

2.1 Метод опорних векторів (SVM)

Існує ланцюжок подій, які призвели до винаходу методу опорних векторів, які сягають середини 20 століття. У 1950 році Ароншайн публікує «Теорію відтворення ядер». У 1957 році Френк Розенблатт прийняв цю ідею і винайшов персептрон, простий лінійний класифікатор. Через 6 років приходять Вапнік і Лернер і оголошують «Узагальнений портретний алгоритм» (1963). Це було справжнім натхненням для роботи Бозера, Гуйона та Вапніка 1992 року на конференції COLT, в якій і був представлений метод опорних векторів. У 1998 році Шоу, Тейлор та ін. зробили значний внесок в узагальнення жорсткого поля Support Vector Machines. Потім Шоу, Тейлор, Крістіаніні дали статистичні межі узагальненню методів опорних векторів із м'якими відступами у 2000 році.

Мета машинного алгоритму опорних векторів — знайти гіперплощину в N -вимірному просторі (N — кількість ознак), яка чітко класифікує точки даних. Щоб розділити два класи точок даних, можна вибрати багато можливих гіперплощин. Наша мета — знайти площину, яка має максимальний запас, тобто максимальну відстань між точками даних обох класів. Збільшення межі відстані надає деяке посилення, щоб майбутні точки даних можна було класифікувати з більшою впевненістю. Загальна схематична візуалізація алгоритму наведена на рисунку 2.1.

Як видно з рисунка — оптимальна гіперплощина — це та, яка має найбільшу відстань між обома ознаками класів. В нашому випадку у нас бінарна класифікація, тому ця ілюстрація також ілюструє необхідний нам результат.

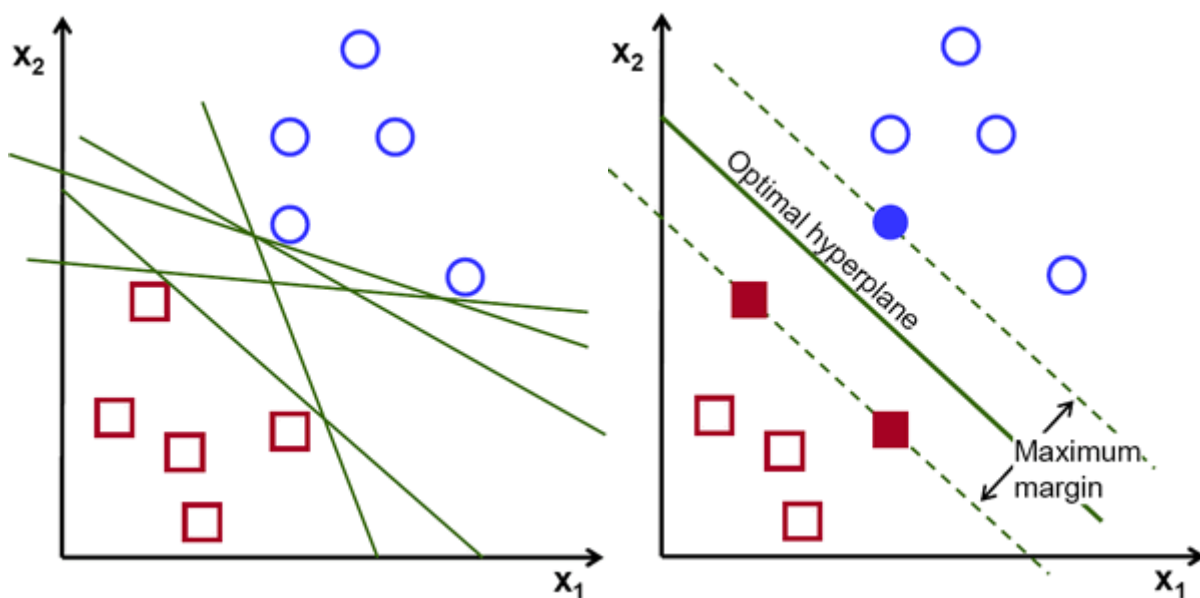


Рисунок 2.1 – Схема роботи алгоритму опорних векторів

Гіперплощини — це межі рішень, які допомагають класифікувати точки даних. Точки даних, що падають по обидва боки від гіперплощини, можна віднести до різних класів. Крім того, розмір гіперплощини залежить від кількості ознак. Якщо кількість вхідних об'єктів дорівнює 2, то гіперплощина — це просто лінія. Якщо кількість вхідних об'єктів дорівнює 3, то гіперплощина стає двовимірною площиною. Важко уявити, коли кількість функцій перевищує 3.

Опорні вектори — це точки даних, які знаходяться ближче до гіперплощини і впливають на положення та орієнтацію гіперплощини. Використовуючи ці опорні вектори, ми максимізуємо запас класифікатора. Видалення опорних векторів змінить положення гіперплощини. Це характеристики, які допомагають нам будувати нашу модель SVM. Лінійний класифікатор методу опорних векторів описується формулою (6).

$$f(\vec{x}) = \text{sign}(\vec{w}\vec{x} + b) \quad (6)$$

Де $f(\vec{x})$ – функція класифікатора для поділяючої гіперплощини, b – параметр зсуву (точка перетинання з віссю x), \vec{w} – вектор нормалі до поділяючої гіперплощини, \vec{x} – гіперплощина. Для задачі нелінійної класифікації використовується формула (7):

$$f(\vec{x}) = \text{sign} \left(\sum_i \alpha_i y_i K(\vec{x}_i, \vec{x}_j) + b \right) \quad (7)$$

Де $f(\vec{x})$ – функція класифікатора для поділяючої гіперплощини, b – параметр зсуву (точка перетинання з віссю x), \vec{w} – вектор нормалі до поділяючої гіперплощини, \vec{x} – гіперплощина, y_i – мітка класу, α_i – відповідний зв'язаний множник Лагранжа, K - функція ядра, що відповідає скалярному добутку ознак в розширеному просторі.

2.2 Наївний Байєсівський алгоритм

Найпростіші рішення зазвичай є найпотужнішими, і Наївний Байєсівський алгоритм є хорошим прикладом цього. Незважаючи на досягнення в області машинного навчання за останні роки, застосування алгоритму виявилось не тільки простим, але й швидким, точним і надійним. Він успішно використовується для багатьох цілей, але він особливо добре працює з проблемами обробки природної мови (NLP).

Наївний Байєсівський алгоритм — це імовірнісний алгоритм машинного навчання, заснований на теоремі Байєса, який використовується в широкому спектрі завдань класифікації. Наївний класифікатор Байєса — це імовірнісний класифікатор, заснований на теоремі Байєса, яка передбачає, що кожна ознака робить незалежний і рівний внесок у цільовий клас. Класифікатор NB передбачає, що кожна ознака незалежна і не взаємодіє один з одним, так що кожна ознака незалежно і однаково впливає на ймовірність належності вибірки

до певного класу. Класифікатор NB простий у реалізації та швидкий у обчислювальному відношенні та добре працює на великих наборах даних, які мають високу розмірність. Класифікатор NB сприятливий для додатків у реальному часі та не чутливий до шуму. Класифікатор NB обробляє навчальний набір даних для обчислення ймовірностей класів $P(y_i)$ та умовних ймовірностей, які визначають частоту кожного значення ознаки для заданого значення класу, поділену на частоту екземплярів із цим значенням класу. NB класифікатор найкраще працює, коли корельовані ознаки видаляються, оскільки корельовані ознаки будуть голосувати двічі в моделі, що призведе до надмірного підкреслення важливості корельованих ознак. Формули (8, 9) виражають ймовірність для деякого класу.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \quad (8)$$

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c) \quad (9)$$

Де $P(c|x)$ - апостеріорна ймовірність класу, c - клас, x - атрибути, $P(c)$ - попередня ймовірність класу, $P(x|c)$ - ймовірність предиктора даного класу, $P(x)$ - попередня ймовірність предиктора. Наївний алгоритм Байєса відомий своєю високою продуктивністю на великомасштабованих даних і іноді перевершує навіть більш складні методи.

2.3 Дерево рішень

У світі машинного навчання розробники можуть легко створювати незалежне середовище для проектів. Щоб досягти надійних результатів, потрібно лише кілька кліків, щоб встановити та підігнати моделі. Проте багато алгоритмів може бути досить важко зрозуміти, не кажучи вже про пояснення. Дерева рішень, незважаючи на погану продуктивність у своїй базовій формі,

легко зрозуміти, і в складі комплексних алгоритмів (Random Forest, XGBoost) досягають чудових результатів.

Коротка історія дерев рішень:

1963: Департамент статистики Університету Вісконсін-Медісон пише, що перше дерево регресії було винайдено в 1963 році (проект AID, Морган і Сонквіст). Він мав міру домішки і рекурсивно розбивав дані на дві підмножини.

1966: Інститут обчислювальної техніки Познанського технологічного університету стверджує, що одна з перших публікацій про дерева рішень була в 1966 році (Хант). У психології для моделювання концепції навчання людини використовувалися методи дерева рішень. Досліджуючи людський розум, дослідники виявили, що алгоритм корисний для програмування.

1972: Перше дерево класифікації з'явилося в проекті THAID (за Месенджером і Менделлом). Він працював за допомогою розділення даних, щоб максимізувати суму випадків у модальній категорії. Прогнозований клас був режимом.

1974: професори статистики Лео Брейман і Чарльз Стоун з Берклі, а також Джером Фрідман і Річард Олшен зі Стенфорда почали розробку алгоритму дерева класифікації та регресії (CART).

1977: Брейман, Стоун, Фрідман та Олшен винайшли першу версію CART.

1984: Офіційне видання з програмним забезпеченням CART. Це була революція у світі алгоритмів. Навіть сьогодні CART є одним з найбільш використовуваних методів аналізу даних. Основні оновлення включають усічення непотрібних дерев, тунелювання та вибір найкращої версії дерева. CART став світовим стандартом для дерев рішень, і його розвиток продовжував прогресувати.

1986: Джон Росс Квінлан запропонував нову концепцію: дерева з кількома відповідями. Важлива примітка: CART та всі інші алгоритми мають лише дві відповіді на кожне запитання (так звані двійкові дерева). Квінлан винайшов ID3

(Ітеративний дихотомайзер 3), використовуючи критерій домішки, який називається коефіцієнтом посилення. Він може охоплювати інші питання та запропонувати деякі результуючі вузли. ID3 не був ідеальним, тому його автор продовжував оновлювати структуру алгоритму. Так народився C4.5. C4.5 усунув недоліки свого попередника, зайнявши 1-е місце в 10 найкращих алгоритмах інтелекту даних (Springer LNCS, 2008). На той момент алгоритм існував вже 15 років.

Ідея досить проста і нагадує людський розум. Якби ми спробували розділити дані на частини, наші перші кроки були б засновані на запитаннях. Крок за кроком дані будуть розділятися на унікальні частини, і, нарешті, ми розділяли б вибірки. Це суть всієї концепції.

- Щоб краще пояснити це поняття, ми скористаємося популярною термінологією:
- Вузол: кожен об'єкт в дереві. Вузли містять підмножини даних, і за винятком листових вузлів питання розбиває підмножину.
- Батьківський вузол: Питання, яке розбиває дані.
- Дочірній вузол: результуючий вузол. Він також може бути батьком для своїх дітей.
- Листковий вузол: Останній вузол без додаткових запитань. Лише частина даних, що представляють відповіді на попередні запитання.
- Відділення: Унікальний рядок запитань з відповідями, які надходять до листового вузла.
- Корінь: верхній вузол.

Можна використовувати дерева рішень з регресією або класифікацією. Методики дещо відрізняються, тому ми розглянемо обидві, починаючи з класифікації. Як правило, для класифікації модель слід вивчати на навчальних даних із заздалегідь визначеним набором міток. Він передбачав би мітку (тобто клас) для нових зразків. Отже, ми маємо набір даних з різними атрибутами

(функціями). Кожен зразок має свою комбінацію значення ознак. Як дерево рішень робить припущення щодо поділу даних?

Щоб відповісти на це питання, ми повинні спочатку зрозуміти мету та її результат. Мета полягає в тому, щоб розділити вибірки з більш схожою структурою, і в результаті отримують чистіші підмножини. Чистота в цьому випадку відповідає якості розщеплення. Для цього існує кілька типових критеріїв:

- Індекс Gini: вимірювання, яке бере суму відношень для кожного екземпляра класу до цілих екземплярів вузла. Після цього від нього віднімається 1. «Ідеальний» вузол мав би домішку 0.
- Інформаційна ентропія: на основі концепції ентропії. Коротше кажучи, чим рівномірніше розташовані точки в певному просторі, тим більша ентропія.

Відмінності між індексом Gini та інформаційною ентропією не є величезними. Індекс Gini робить більш чисте поділ, але ентропія, як правило, створює більш збалансовані підмножини. Інформаційна ентропія дорожча з точки зору обчислень.

Математичний опис ентропії для одного атрибуту наведений у формулі (10):

$$E(S) = \sum_{i=1} -p_i \log_2(p_i) \quad (10)$$

Де p_i – ймовірність певної події, S – відповідно сама подія або метрика. Ентропія для декількох ознак наведена на формулі (11):

$$E(T, X) = \sum_{c \in X} P(c) E(c) \quad (11)$$

Де T, X – певні метрики, c – значення метрики X , $P(c)$ – вірогідність значення метрики, $E(c)$ – ентропія метрики.

Таким чином, в основному, алгоритм шукає між усіма функціями на найбільшого розділювача домішок, де батьківський вузол матиме більшу кількість домішок, ніж його дочірні.

Ефективність розщеплення вимірюється його забрудненістю та кількістю даних у кожному дочірньому вузлі, прагнучи мінімізувати поділ розмірів між дочірніми вузлами. Ілюстрація роботи дерева рішень наведена на рисунку 2.2.

Survival of passengers on the Titanic

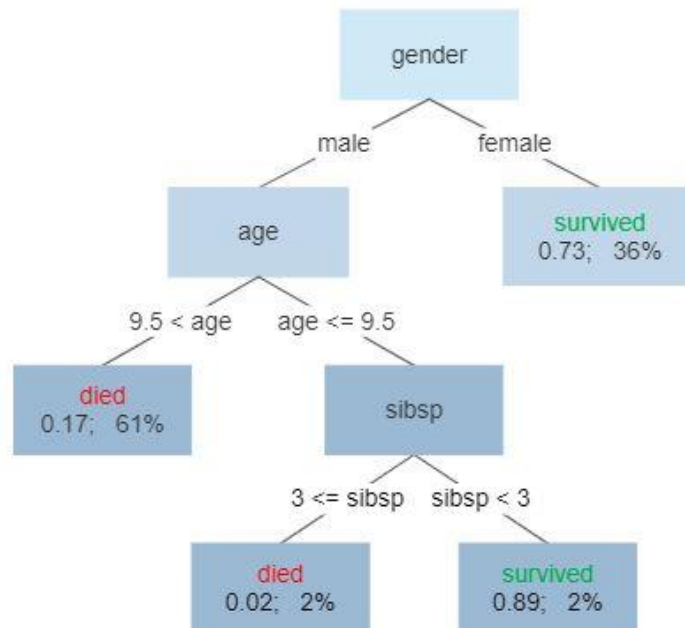


Рисунок 2.2 – Схема роботи алгоритму дерева рішень

Як видно з рисунку – класи визначаються на основі дерева умовних операторів, які будуються на початкових ознаках. Цей алгоритм широко використовується для цілей класифікації та регресії і являє собою деревоподібний класифікатор.

Листки представляють цільові класи, кожен вузол - тестовий набір для певного атрибута даних, а ребра є результатом тестового випадку.

Дерева рішень утворюють вкладені оператори if-else, і чим глибше дерево, тим краще модель. Коротка ілюстрація компонентів алгоритму показана на малюнку 2.3.

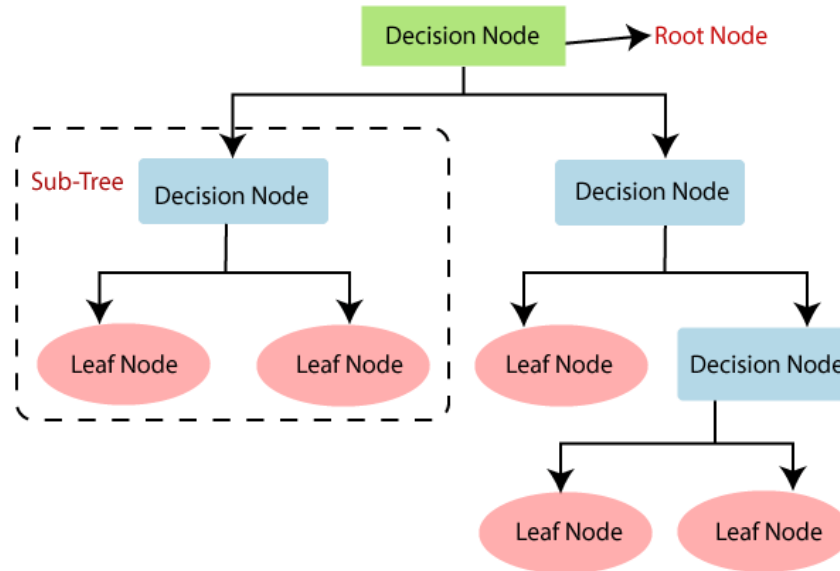


Рисунок 2.3 – Компоненти дерева рішень

2.4 Random Forest

У 2000 році Лео Брейман з Університету Берклі зазначив, що дерева рішень такі ж, як ядра з істинними полями. Своє відкриття він опублікував у цій феноменальній дослідницькій роботі в 2001 році. Після відкриття Брейманом і Катлером Random Forest було зроблено чимало покращень. Лін і Чон: внесок адаптивного найближчого сусіда до Random Forest Це означало, що Random Forest можуть бути адаптивними оцінками ядра. Дослідницька робота тут. Девіс і Гахрамані прийшли і пропонують, що ядро Random Forest може перевершити інші методи ядра. Ерван Скорнет був першим, хто винайшов KeRF і вказав на зв'язок між KeRF і Random Forest. Арло і Генуер продемонстрували, що

упередження нескінченного лісу зменшується швидше, ніж окремого дерева, і що нескінченні ліси мають значно кращий рівень ризику, ніж окремі дерева.

Random Forest — це алгоритм навчання під наглядом. «Ліс», який він будує, являє собою комбінацію дерев рішень, зазвичай навчених методом «мішкування». Загальна ідея методу мішків полягає в тому, що комбінація моделей навчання підвищує загальний результат.

Однією з великих переваг Random Forest є те, що його можна використовувати як для задач класифікації, так і для задач регресії, які формують більшість сучасних систем машинного навчання. Random Forest має майже ті самі гіперпараметри, що й дерево рішень.

Random Forest додає моделі додаткову випадковість під час зростання дерев. Замість того, щоб шукати найважливішу характеристику під час розбиття вузла, він шукає найкращу характеристику серед випадкової підмножини ознак. Це призводить до широкого розмаїття, що зазвичай призводить до кращої моделі.

Отже, у Random Forest алгоритмом розбиття вузла до уваги враховується лише випадкова підмножина ознак. Ви навіть можете зробити дерева більш випадковими, додатково використовуючи випадкові пороги для кожної функції, а не пошук найкращих можливих порогових значень (як це робить звичайне дерево рішень). Схема роботи алгоритму наведена на рисунку 2.4.

Однією з найбільших переваг Random Forest є його універсальність. Його можна використовувати як для завдань регресії, так і для завдань класифікації, а також легко побачити відносну важливість, яку він надає вхідним функціям.

Random Forest також є дуже зручним алгоритмом, оскільки гіперпараметри за замовчуванням, які він використовує, часто дають хороший результат прогнозу. Зрозуміти гіперпараметри досить просто, і їх також не так багато.

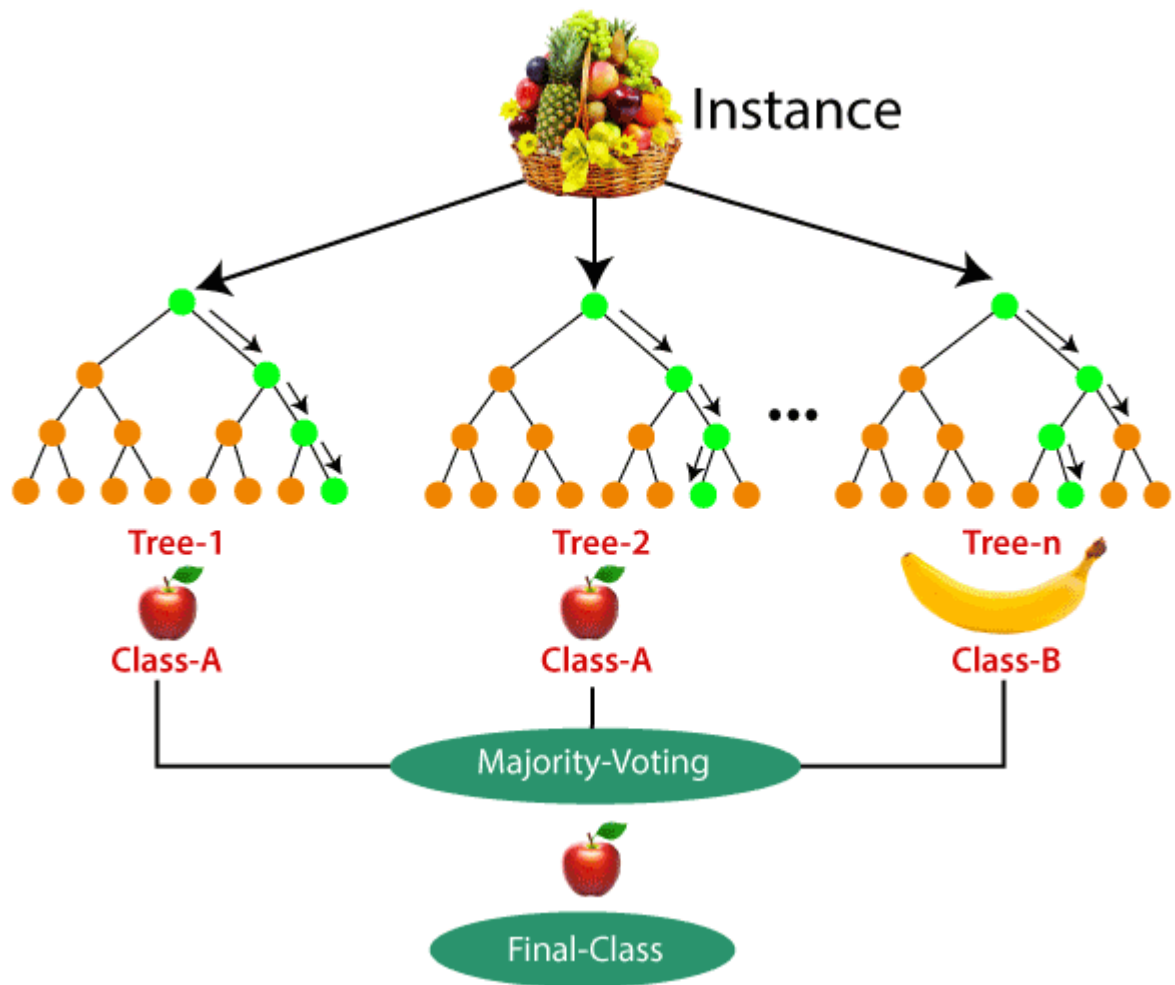


Рисунок 2.4 – Схема роботи Random Forest

Однією з найбільших проблем машинного навчання є перенавчання, але в більшості випадків цього не станеться завдяки класифікатору Random Forest. Якщо в лісі достатньо дерев, класифікатор не переповнить модель.

Основним обмеженням Random Forest є те, що велика кількість дерев може зробити алгоритм занадто повільним і неефективним для передбачення в реальному часі. Загалом, ці алгоритми швидко навчаються, але досить повільно створюють передбачення після навчання. Для більш точного прогнозу потрібно більше дерев, що призводить до повільнішої моделі. У більшості реальних додатків алгоритм Random Forest досить швидкий, але, безсумнівно, можуть виникнути ситуації, коли продуктивність під час виконання є важливою, а інші

підходи були б кращими. Для знаходження ваги кожного дерева використовується наступна формула (12)

$$ni_j = w_j C_j - w_{\text{лівий}(j)} C_{\text{лівий}(j)} - w_{\text{правий}(j)} C_{\text{правий}(j)} \quad (12)$$

Де ni_j – пріоритет дерева j , w_j – зважена кількість екземплярів класу які доходять до цього вузла (дерева), C_j – значення індексу Gini, $\text{лівий}(j)$ – лівий дочірній вузол (дерево). Рівняння для визначення індексу Gini наведене на формулі (13):

$$C_i = 1 - \sum_{j=1} p_j^2 \quad (13)$$

Де p_j – ймовірність належності до деякого класу j . Звичайно, Random Forest є інструментом прогнозного моделювання, а не інструментом опису, тобто, якщо ви шукаєте опис взаємозв'язків у ваших даних, інші підходи були б кращими.

2.5 XGBoost

Спочатку XGBoost розпочався як дослідницький проект Tianqi Chen'а як частина групи Distributed (Deep) Machine Learning Community (DMLC). Спочатку він починався як термінальна програма, яку можна було налаштувати за допомогою файлу конфігурації libsvm. Він став добре відомим у колах змагань з машинного навчання після його використання в переможному рішенні Higgs Machine Learning Challenge. Незабаром після цього були створені пакети Python і R, і XGBoost тепер має реалізації пакетів для Java, Scala, Julia, Perl та інших мов. Це принесло бібліотеку більше розробників і сприяло її популярності серед спільноти Kaggle, де вона використовувалася для великої кількості конкурсів.

XGBoost працює як метод Ньютона-Рафсона у функціональному просторі, на відміну від градієнтного прискорення, яке працює як градієнтний спуск у функціональному просторі, у функції втрат використовується наближення Тейлора другого порядку для встановлення зв'язку з методом Ньютона-Рафсона.

При використанні градієнтного підвищення для регресії слабкі підмоделі є деревами регресії, і кожне дерево регресії відображає точку вхідних даних на один зі своїх листків, що містить безперервну оцінку. XGBoost мінімізує регуляризовану (L1 і L2) цільову функцію, яка поєднує опуклу функцію втрат (на основі різниці між прогнозованими та цільовими результатами) та штрафний термін за складність моделі (іншими словами, функції дерева регресії).

Навчання відбувається ітераційно, додаючи нові дерева, які передбачають залишки або помилки попередніх дерев, які потім об'єднуються з попередніми деревами, щоб зробити остаточний прогноз. Це називається посиленням градієнта, оскільки він використовує алгоритм градієнтного спуску, щоб мінімізувати втрати при додаванні нових моделей. Коротку ілюстрацію того, як працює підсилення градієнтного дерева наведено на рисунку 2.5.

Переваги XGBoost – це Висока точність моделей, що досягається за рахунок використання різних технік оптимізації, регуляризації та побудови ансамблю моделей, гнучкість та налаштовуваність алгоритму, яка дозволяє легко підлаштовувати модель під конкретну задачу, вбудована підтримка багатокласової класифікації, що дозволяє ефективно вирішувати задачі з більш ніж двома класами, обробка відсутніх даних та підтримка роботи з різними типами даних, що забезпечує ефективну обробку даних різної якості та формату, можливість масштабування та розподіленої обробки на багатьох комп'ютерах, що дозволяє ефективно працювати з великими об'ємами даних.

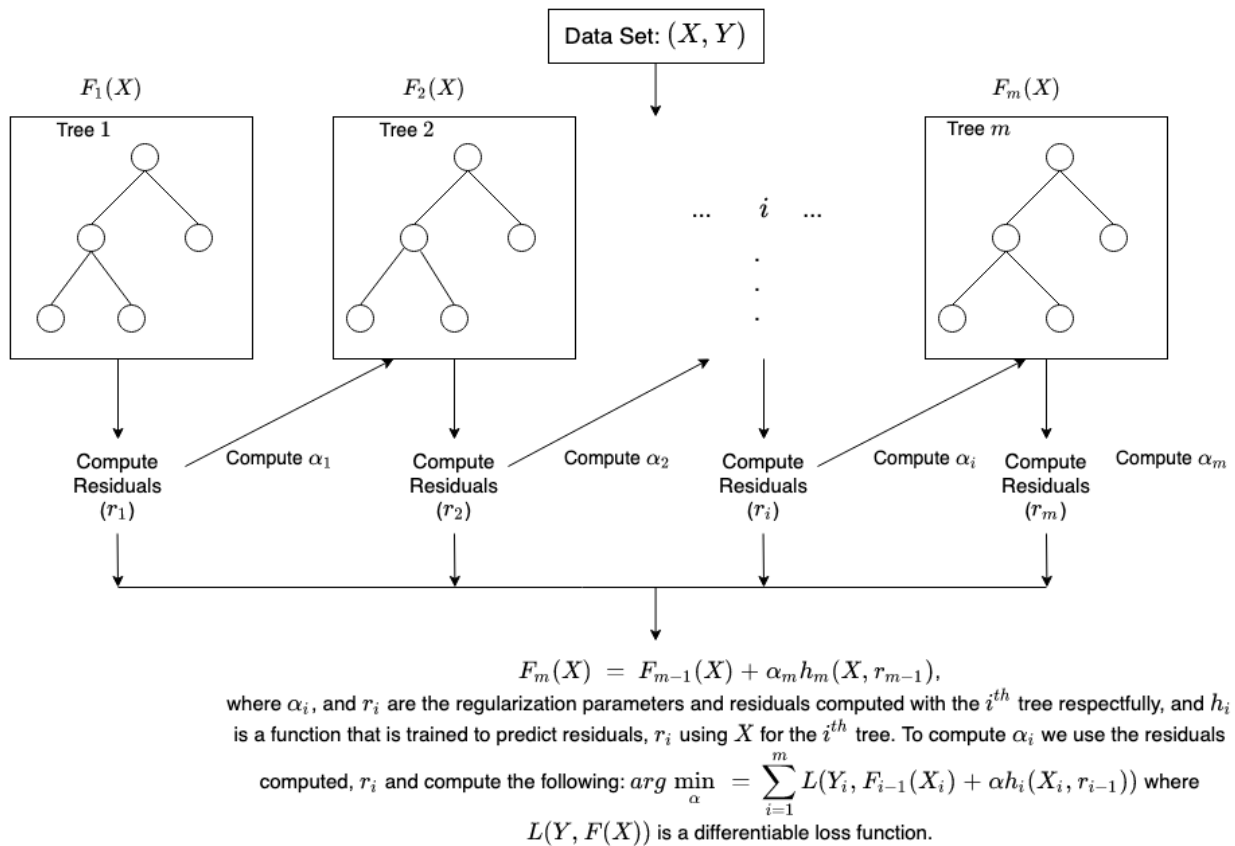


Рисунок 2.5 – Ілюстрація роботи XGBoost

2.6 Глибокі нейронні мережі

Ідея нейронних мереж виникла як модель того, як функціонують нейрони в мозку, названа «коннекціонізмом» і використовувала підключені схеми для імітації розумної поведінки. У 1943 році її зобразили за допомогою простої електричної схеми нейрофізіолог Уоррен МакКалох і математик Волтер Піттс. Дональд Хебб продовжив цю ідею у своїй книзі «Організація поведінки» (1949), запропонувавши, що нейронні шляхи посилюються під час кожного наступного використання, особливо між нейронами, які мають тенденцію спрацьовувати в той же час, таким чином починаючи довгий шлях до кількісної оцінки складних процесів.

Дві основні концепції, які є попередниками нейронних мереж

- «Порогова логіка» — перетворення безперервного введення в дискретний вихід
- «Hebbian Learning» — модель навчання, заснована на нейропластичності, запропонована Дональдом Хеббом у його книзі «Організація поведінки», часто підсумовується фразою: «Клітини, які запускаються разом, з'єднуються».

обидва запропоновані в 1940-х роках. У 1950-х роках, коли дослідники почали намагатися перевести ці мережі на обчислювальні системи, перша мережа Hebbian була успішно реалізована в Массачусетському технологічному інституті в 1954 році.

Приблизно в цей час Френк Розенблат, психолог з Корнелла, працював над розумінням порівняно простіших систем прийняття рішень, наявних в очі мухи, які лежать в основі та визначають її реакцію втечі. Намагаючись зрозуміти і кількісно оцінити цей процес, він запропонував ідею персептрона в 1958 році, назвавши його персептроном Mark I. Це була система з простим співвідношенням вхідних і вихідних даних, змодельована на нейроні Маккалоха-Піттса, запропонована в 1943 році нейробіологом Уорреном С. Маккалоком і логіком Уолтером Піттсом для пояснення складних процесів прийняття рішень у мозку за допомогою лінійного порогу. Нейрон Маккалоха-Піттса приймає вхідні дані, приймає зважену суму і повертає «0», якщо результат нижче порогового значення, і «1» в іншому випадку (рисунок 2.6).

Зворотне розповсюдження разом із градієнтним спуском утворює основу й потужність нейронних мереж. У той час як градієнтний спуск постійно оновлюється та переміщує ваги та зміщення до мінімуму функції вартості, зворотне поширення оцінює градієнт вартості за значенням ваги та зміщення, величина та напрямок яких використовуються градієнтним спуском для оцінки розміру та напрямку поправок до ваг і параметрів зміщення.

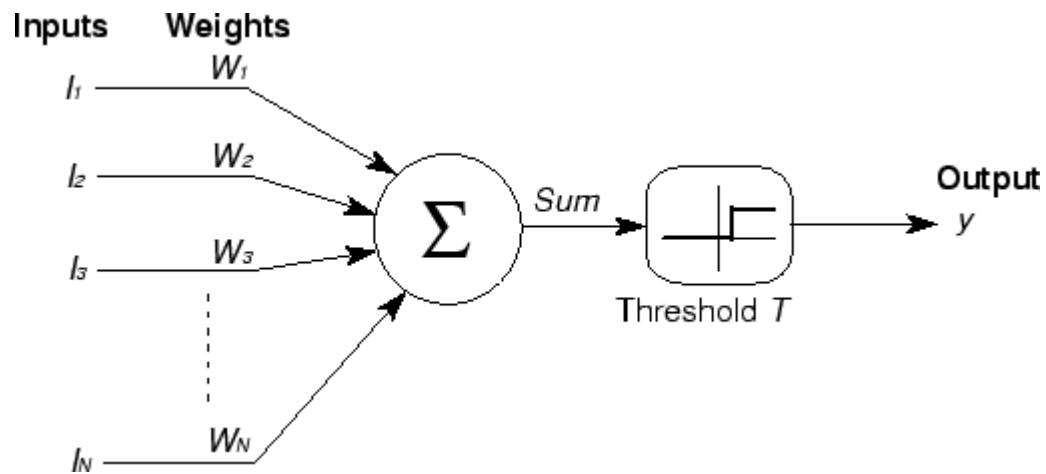


Рисунок 2.6 – Ілюстрація нейрона Маккалоха-Піттса

Нейронна мережа складається з вертикально розміщених компонентів, які називаються шарами. Кожна пунктирна лінія на рисунку 2.7 представляє шар. У нейронних мережах є три типи шарів

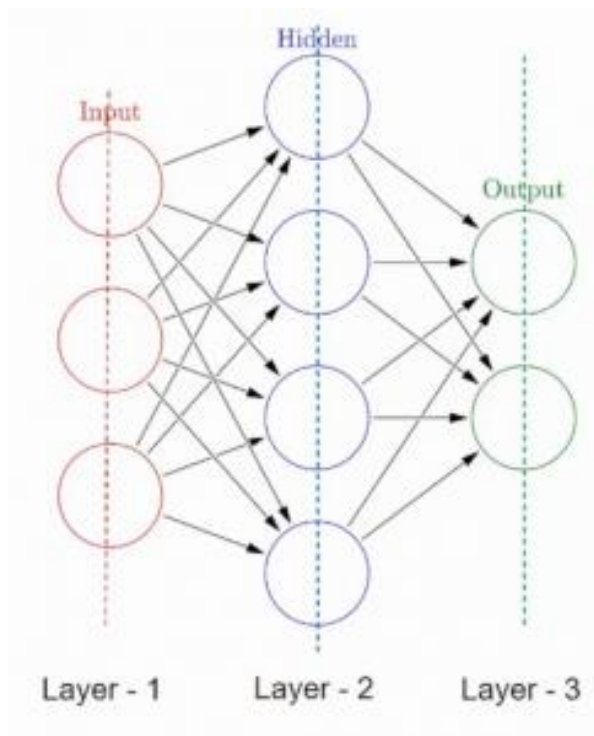


Рисунок 2.7 Архітектура нейронної мережі

Перший – це вхідний шар. Цей рівень прийме дані та передасть їх решті мережі. Другий тип шару називається прихованим шаром. Приховані шари для нейронної мережі – це один або кілька шарів. У наведеному вище випадку число дорівнює 1. Приховані шари — це ті, які насправді відповідають за чудову продуктивність і складність нейронних мереж. Вони виконують кілька функцій одночасно, наприклад, перетворення даних, автоматичне створення функцій тощо. Останній тип шару – вихідний шар. Вихідний шар містить результат або вихід мережі. Необроблені зображення передаються на вхідний шар, а ми отримуємо результат у вихідному шарі. Наприклад – номер або назву класу до якого належить об’єкт в бінарній класифікації, в нашому випадку присутність або відсутність атеросклерозу у пацієнтів.

Шар складається з невеликих окремих одиниць, які називаються нейронами. Нейрон в нейронній мережі можна краще зрозуміти за допомогою біологічних нейронів. Штучний нейрон схожий на біологічний нейрон. Він отримує вхідні дані від інших нейронів, виконує певну обробку та виробляє вихід. Математичний опис нейрона наведений на формулі (14):

$$z = \beta + w_1x_1 + w_2x_2 + \dots + w_nx_n \quad (14)$$

Де z – значення нейрона, β – зміщення, w_n – вага, або бета-коефіцієнт, x_n – незалежна змінна, або ввід.

Кожному зі своїх вхідних з’єднань вузол призначає значення, відоме як «вага». Коли мережа активна, вузол отримує інший елемент даних — інше значення — над кожним своїм з’єднанням і множить його на відповідну вагу. Потім отримані результати додаються разом, у результаті чого виходить єдине число. Якщо це число нижче порогового значення, вузол не передає дані наступному шару. Якщо число перевищує порогове значення, вузол

«спрацьовує», що в сучасних нейронних мережах зазвичай означає надсилання числа — суми зважених вхідних даних — по всіх своїх вихідних з'єднаннях.

Коли нейронну мережу навчають, усі її ваги та пороги спочатку встановлюються на випадкові значення. Дані навчання надходять до нижнього шару — вхідного — і проходять через наступні шари, множачись і складаючись разом, доки не потрапляють, трансформовані, на вихідний шар. Під час тренування ваги та пороги постійно коригуються, доки дані навчання з однаковими мітками постійно дають подібні результати.

Функція активації в нейронній мережі визначає, як зважена сума вхідних даних перетворюється на вихід з вузла або вузлів у шарі мережі. Іноді функцію активації називають «функцією передачі». Якщо діапазон виходу функції активації обмежений, це можна назвати «функцією здавлення». Багато функцій активації є нелінійними, і їх можна назвати «нелінійністю» в шарі або конструкції мережі.

Вибір функції активації має великий вплив на можливості та продуктивність нейронної мережі, і різні функції активації можуть використовуватися в різних частинах моделі. Технічно функція активації використовується в межах або після внутрішньої обробки кожного вузла в мережі, хоча мережі призначені для використання однієї і тієї ж функції активації для всіх вузлів у шарі. Мережа може мати три типи шарів: вхідні шари, які отримують необроблені вхідні дані з домену, приховані шари, які беруть вхідні дані з іншого шару і передають вихід на інший шар, і вихідні шари, які роблять прогноз.

Усі приховані шари зазвичай використовують ту саму функцію активації. Вихідний рівень зазвичай використовує функцію активації, відмінну від прихованих шарів, і залежить від типу передбачення, якого вимагає модель.

Функції активації також зазвичай диференційовані, що означає, що похідну першого порядку можна обчислити для заданого вхідного значення. Це

необхідно з огляду на те, що нейронні мережі зазвичай навчаються за допомогою алгоритму зворотного поширення помилки, який вимагає похідної від помилки передбачення для оновлення ваг моделі.

Існує багато різних типів функцій активації, які використовуються в нейронних мережах, хоча, можливо, лише невелика кількість функцій використовується на практиці для прихованих і вихідних шарів.

Функція випрямленої лінійної активації, або функція активації ReLU, є, мабуть, найпоширенішою функцією, яка використовується для прихованих шарів.

Вона є поширеною, оскільки проста у реалізації та ефективна у подоланні обмежень інших раніше популярних функцій активації, таких як Sigmoid і Tanh. Зокрема, вона менш чутлива до зникаючих градієнтів, які заважають навчанню глибоких моделей, хоча і може страждати від інших проблем, наприклад, насичених або «мертвих» одиниць.

Функція ReLU розраховується наступним чином: $\max(0.0, x)$. Це означає, що якщо вхідне значення (x) є негативним, то повертається значення 0.0, інакше повертається дане значення.

Сигмовидна функція активації також називається логістичною функцією. Це та сама функція, що використовується в алгоритмі класифікації логістичної регресії. Функція приймає будь-яке реальне значення в якості вхідних даних і виводить значення в діапазоні від 0 до 1. Чим більше вхідний сигнал (більш позитивний), тим ближче вихідне значення буде до 1,0, тоді як чим менше вхід (більш негативний), тим ближче вихід буде до 0,0. Сигмовидна функція активації розраховується наступним чином: $1,0 / (1,0 + e^{-x})$, де e — математична константа, яка є основою натурального логарифма.

Функція активації гіперболічного тангенса також називається просто функцією Tanh (також «tanh» і «TanH»). Вона дуже схожа на функцію активації сигмоїда і навіть має таку ж S-подібну форму. Функція приймає будь-яке

реальне значення в якості вхідних даних і виводить значення в діапазоні від -1 до 1. Чим більше вхідний сигнал (більш позитивний), тим ближче вихідне значення буде до 1,0, тоді як чим менше вхід (більш негативний), тим ближче вихід буде до -1,0.

Функція активації Tanh розраховується наступним чином:

$(e^x - e^{-x}) / (e^x + e^{-x})$, де e — математична константа, яка є основою натурального логарифма.

Нейронна мережа майже завжди матиме однакову функцію активації у всіх прихованих шарах. Дуже незвичайно змінювати функцію активації за допомогою мережевої моделі. Традиційно сигмовидна функція активації була функцією активації за замовчуванням у 1990-х роках. Можливо, з середини до кінця 1990-х до 2010-х років функція Tanh була функцією активації за замовчуванням для прихованих шарів.

Як сигмовидна, так і Tanh функції можуть зробити модель більш сприйнятливою до проблем під час навчання через так звану проблему зникаючих градієнтів. Функція активації, що використовується в прихованих шарах, зазвичай вибирається на основі типу архітектури нейронної мережі.

Сучасні моделі нейронних мереж із загальною архітектурою, такими як MLP і CNN, будуть використовувати функцію активації ReLU або розширення.

Рекурентні мережі досі зазвичай використовують функції активації Tanh або сигмовидної форми, або навіть обидві. Наприклад, LSTM зазвичай використовує сигмоїдну активацію для повторних з'єднань і активацію Tanh для виведення.

Вихідний шар — це шар моделі нейронної мережі, який безпосередньо виводить прогноз. Є три функції активації, які ви можна розглянути для використання на вихідному рівні.

Лінійна функція активації також називається «ідентичність» (помножена на 1,0) або «без активації». Це пов'язано з тим, що функція лінійної активації

жодним чином не змінює зважену суму вхідних даних і натомість повертає значення безпосередньо.

Сигмовидна функція логістичної активації була описана вище.

Функція softmax виводить вектор значень, які досягають суми 1,0, які можна інтерпретувати як ймовірності приналежності до класу. Це пов'язано з функцією argmax, яка виводить 0 для всіх параметрів і 1 для вибраного параметра. Softmax — це «м'яка» версія argmax, яка дозволяє отримати схожий на ймовірність вихід функції «переможець отримує все». Таким чином, вхід до функції є вектором дійсних значень, а вихідним є вектор такої ж довжини зі значеннями, які сумують до 1,0, як ймовірності.

Функція softmax розраховується наступним чином: $e^x / \sum(e^x)$

Де x — вектор виходів, а e — математична константа, яка є основою натурального логарифма.

Нейронна мережа використовувалась для передбачення серцевл-судинних хвороб і досягла високої точності [19], але ми спробуємо розробити власну модель і підвищити точність.

2.7 Основні бібліотеки та інструменти

Для цього дослідження ми використовували мову програмування Python версії 3.8 та її екосистему. На сьогоднішній день Python є найпопулярнішою мовою програмування для аналізу даних і машинного навчання, і пропонує безліч бібліотек і рішень для вирішення таких проблем. Python надає багато утиліт, які скорочують час розробки та забезпечують високоефективні результати. Для цього дослідження ми використали наступний набір бібліотек Python:

- pandas - бібліотека, яка надає функціональні можливості для створення та роботи з наборами даних;

- `pumpru` - дозволяє виконувати складні обчислення на високопродуктивних багатовимірних масивах та оперувати ними;
- `matplotlib` - пропонує програмний інтерфейс різної візуалізації даних;
- `seaborn` - бібліотека візуалізації даних на основі `matplotlib`, яка надає високорівневий інтерфейс і безліч пресетів для малювання більш зручних графіків, а також різноманітних діаграм, теплових карт, кольорових тем тощо;
- `scikit-learn` - пропонує різні готові до використання алгоритми машинного навчання без нагляду та контролю, побудовані на основі `pumpru`, `pandas` та `plotlib`;
- `xgboost` - оптимізована розподілена бібліотека, що забезпечує реалізації алгоритму підвищення градієнта;
- `keras` - високорівневий API нейронної мережі, що забезпечує функціональність для розробки та оцінки моделей глибокого навчання;
- `tensorflow` - платформа з відкритим вихідним кодом, яка надає бекенд-движок для `keras`
- `ann_visualizer` - бібліотека візуалізації, яка використовується для роботи з `keras`, використовує бібліотеку `graphviz` для створення графа нейронної мережі;
- `graphviz` - програмне забезпечення для візуалізації графіків з відкритим вихідним кодом, яке надає функціональні можливості для представлення структурної інформації;

2.8 Алгоритм вирішення задачі

Для початку нам необхідно ознайомитись з датасетом. Для цього з використанням `pandas` та бібліотек для малювання графіків ми зробимо візуалізації для кращого розуміння наших даних.

Після цього ми якщо необхідно, попередньо обробимо дані, вилучимо прогалини та замінимо назви класів чисельними значеннями. Далі розіб'ємо датасет на тренувальну та тестувальну частини, проведемо навчання і порівняємо результати за допомогою матриці невідповідностей. Зробимо висновки та проаналізуємо можливі шляхи впровадження найефективнішої моделі, а також застосуємо найефективнішу модель у веб-застосунку. Методологія проведення дослідження зображена на рис. 2.8.



Рисунок 2.8. Методика проведення дослідження

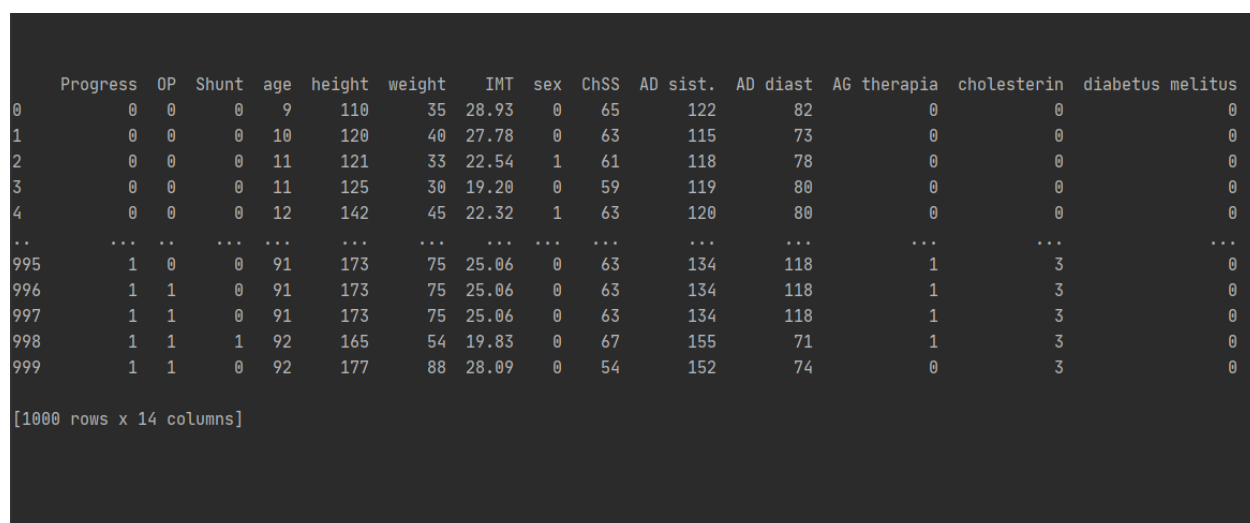
Висновок до розділу 2

У цьому розділі ми визначили основні алгоритми машинного навчання, які будемо застосовувати в даній роботі. Ми провели дослідження історії становлення цих алгоритмів, визначили математичне підґрунтя та алгоритми їх роботи. Також ми обрали набір необхідних бібліотек та програмних інструментів для реалізації програмного продукту та навели опис основних інструментів роботи. Після цього ми розробили визначили алгоритм технічної реалізації рішення. В наступних розділах ми реалізуємо рішення та проведемо аналіз отриманих результатів.

РОЗДІЛ 3. ПОБУДОВА ТА ОЦІНКА МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

3.1 Аналіз даних для моделей

Набір даних, використаний для цього дослідження, надано Національним інститутом серцево-судинної хірургії імені Амосова. Набір даних містить 14 стовпців і 1000 записів пацієнтів. Більшість останніх досліджень серцево-судинних захворювань посилаються на набір даних UCI, який датується 1988 роком. Наявність такого нового та точного набору даних дає унікальну можливість для прогнозування атеросклерозу на основі вже існуючих методів та застосування нових, що відкриває нові двері для застосування алгоритми машинного навчання в медицині. Зразок цього набору даних показано на малюнку 3.1.



	Progress	OP	Shunt	age	height	weight	IMT	sex	ChSS	AD sist.	AD diast	AG therapia	cholesterin	diabetes melitus
0	0	0	0	9	110	35	28.93	0	65	122	82	0	0	0
1	0	0	0	10	120	40	27.78	0	63	115	73	0	0	0
2	0	0	0	11	121	33	22.54	1	61	118	78	0	0	0
3	0	0	0	11	125	30	19.20	0	59	119	80	0	0	0
4	0	0	0	12	142	45	22.32	1	63	120	80	0	0	0
...
995	1	0	0	91	173	75	25.06	0	63	134	118	1	3	0
996	1	1	0	91	173	75	25.06	0	63	134	118	1	3	0
997	1	1	0	91	173	75	25.06	0	63	134	118	1	3	0
998	1	1	1	92	165	54	19.83	0	67	155	71	1	3	0
999	1	1	0	92	177	88	28.09	0	54	152	74	0	3	0

[1000 rows x 14 columns]

Рисунок 3.1 Приклад даних для аналізу

Набір даних не містить порожніх комірок, усі атрибути заповнені та мають нормальний розподіл. Набір даних включає наступні атрибути:

- Progress - відображення наявності (1) або відсутності (0) атеросклерозу.

- OP- хірургічна інтрузія, 1 - присутній, 0 - відсутній;
- Shunt - серцевий шунт, 1 - присутній, 0 - відсутній
- age – вік, роки;
- height - висота в см;
- weight - вага в кілограмах;
- IMT - IMT, індекс маси тіла;
- sex - 0 - чоловіча, 1 - жіноча;
- ChSS - ЧСС, ударів за одну хвилину;
- AD sist. - систолічний кров'яний тиск;
- AD diast - діастолічний артеріальний тиск
- AG therapia– антигіпертензивна терапія, 1 – є, 0 – відсутня
- cholesterin – рівень загального холестерину;
- diabetes melitus - цукровий діабет, 1 - є, 0 - відсутній;

Атрибут Progress є цільовим значенням цього дослідження, і прогнози алгоритмів будуть порівнюватися з ним. Набір даних містить дані 591 пацієнта з наявним атеросклерозом і 409 рядків даних здорових людей. Також важливо перевірити розподіл за статтю в наборі даних, який представлений на малюнку 3.2.

Як бачимо на малюнку 3.2, набір даних містить трохи більше записів про жінок, ніж про чоловіків. Також середній вік здорових людей становить 30,96, а середній вік хворих на атеросклероз – 54,78, і ця тенденція справедлива як для чоловіків, так і для жінок. Враховуючи цю інформацію, ми можемо зробити висновок, що атеросклероз частіше зустрічається у літніх людей, і щоб довести це, ми перевіримо віковий розподіл пацієнтів у нашому наборі даних.

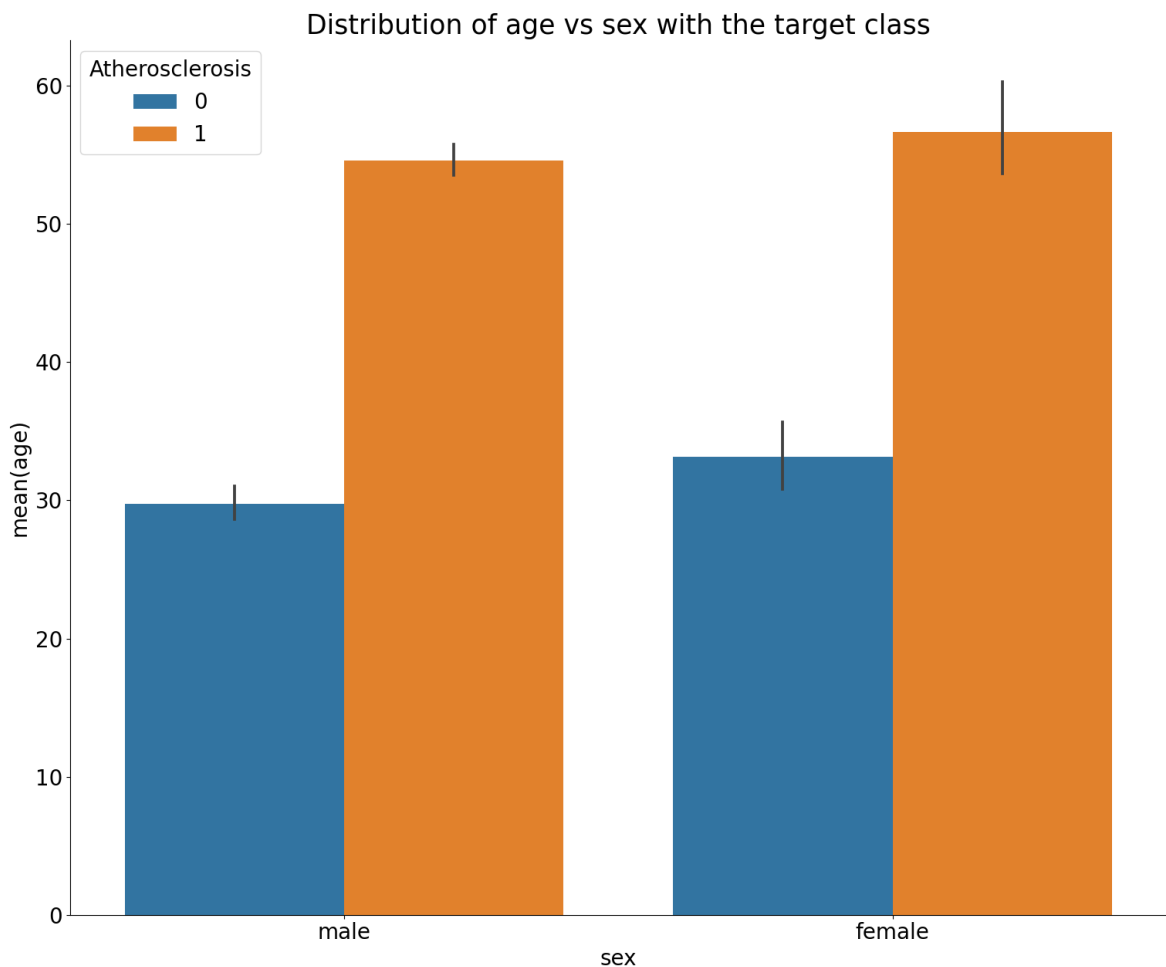


Рисунок 3.2 Розподіл даних за статтю, середнім віком та наявністю атеросклерозу.

Як бачимо на малюнку 3.3, атеросклероз переважно не зустрічається у молодих людей, а навпаки, діагностується у більшості людей середнього та старшого віку. Цей висновок також підтверджується результатами нещодавнього дослідження [20], в якому стверджується, що атеросклероз швидко розвивається у віці від 40 до 50 років. На малюнку показано великий розрив у частоті захворювань у віці від 37 до 38 років, а потім і тенденцію захворювання. спостерігається збільшення, що відображає загальну статистику захворювань серця та вплив факторів ризику.

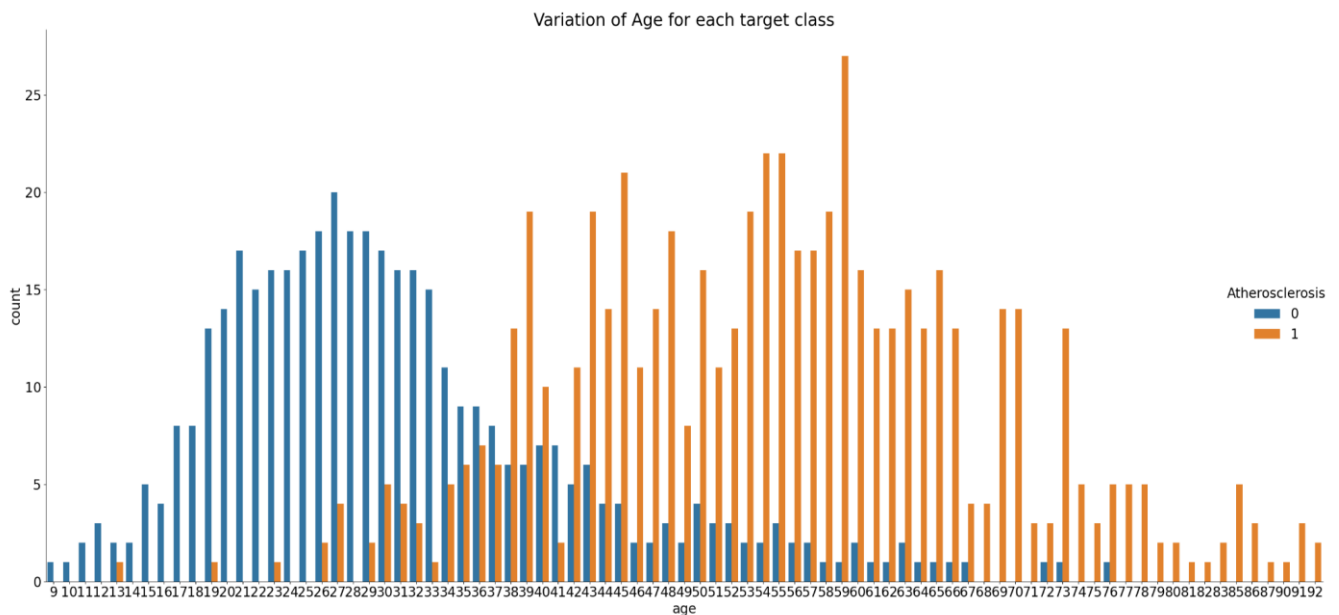


Рисунок 3.3 Розподіл захворювання за віком

Також кількість здорових пацієнтів після 30 років постійно зменшується, а це означає, що у людей часто діагностують захворювання, коли вже пізно. Як було зазначено у вступі, атеросклероз зазвичай є довгостроково прогресуючим захворюванням, однак іноді він може прогресувати більш агресивно.

Захворювання зазвичай проявляється на пізніх стадіях, коли артерії звужуються або закупорюються, що супроводжується болем у грудях або іншими проявами. Рання діагностика атеросклерозу може допомогти уникнути низки серцевих захворювань і, як наслідок, врятувати багато життів.

Важливо визначити відносини між атрибутами та їх вплив на цільове значення, перш ніж ми почнемо застосовувати алгоритми машинного навчання. Для цього ми обчислимо кореляцію між усіма атрибутами та побудуємо теплову карту, яка відображена на малюнку 3.4.

На основі теплової карти ми можемо визначити кореляцію між атрибутами нашого набору даних.

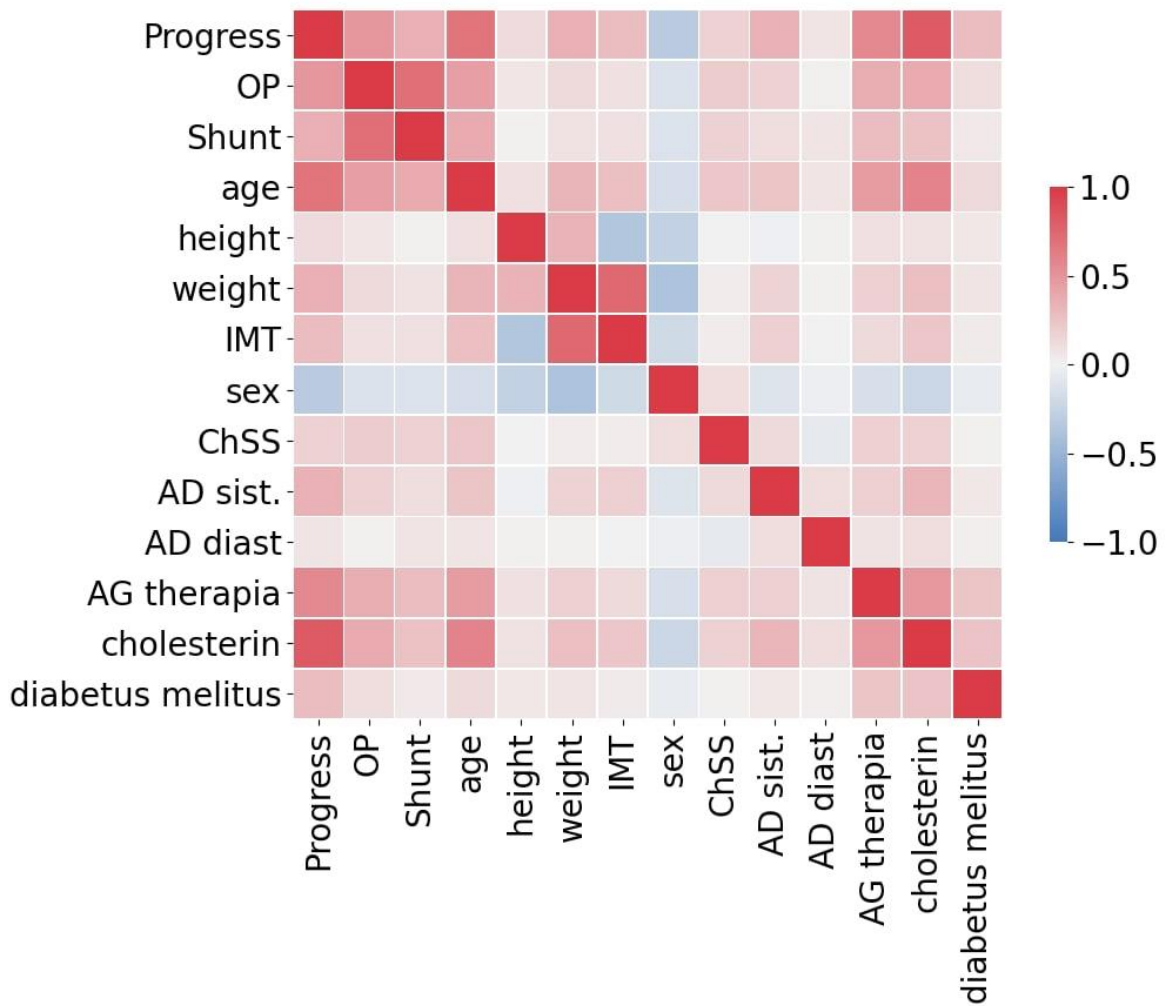


Рисунок 3.4 Теплова карта кореляції атрибутів

Перше, на що ми повинні подивитися, це перший рядок діаграми, який представляє кореляцію кожного окремого атрибута з нашими цільовими даними. Наступне – визначити, які атрибути впливають на ціль через інші атрибути опосередковано. Таким чином, cholesterin, age і AG therapia мають сильну позитивну кореляцію з цільовим атрибутом OP, Shunt, weight та AD sist. мають помірні позитивні відношення, сильних непрямих кореляційних ознак не виявлено.

Тепер давайте візьмемо найбільш корельовані параметри та відобразимо розподіл даних у цій тривимірній площині. Оскільки AG thrapia має двійкові

значення, ми замінимо її вагою, яка має помірну кореляцію, але більш різноманітний розподіл значень. Візуалізація даних для такої площини показана на малюнку 3.5.

Weight, Age and Cholesterin distribution by target classes

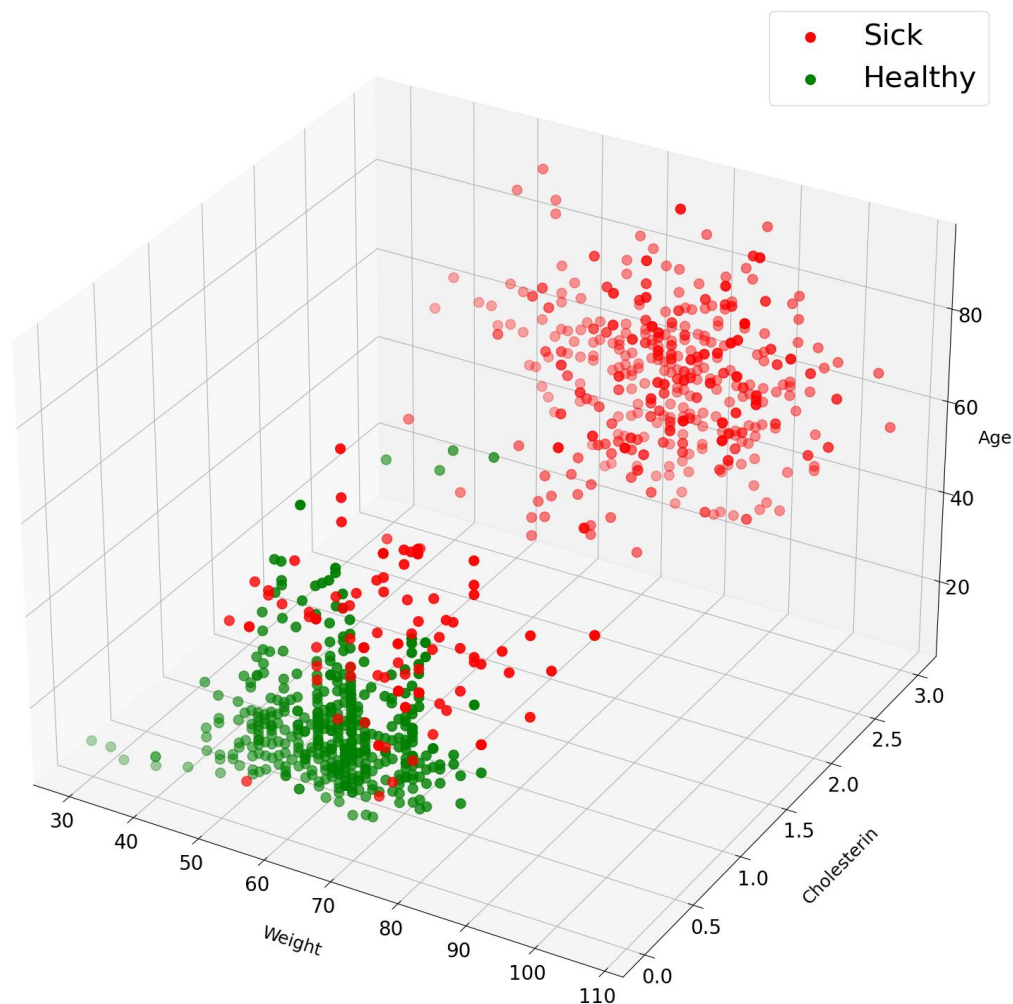


Рисунок 3.5 Розподіл даних у тривимірній площині за найбільш корельованими атрибутами та цільовим класом

На малюнку 3.5 ми бачимо, що більшість пацієнтів з атеросклерозом мають високий рівень холестерину, вони середнього та старшого віку та мають надлишкову вагу, тоді як здорові пацієнти переважно мають вагу до 80 кг, низький рівень холестерину та менше 40.

Цей висновок відображає загальний уявлення про атеросклероз і відповідає факторам ризику: атеросклероз частіше зустрічається у людей похилого віку, людей із зайвою вагою і високим рівнем холестерину. На основі наведеного вище графіка можна побачити, що записи атеросклерозу та не атеросклерозу можна розрізнити за цими трьома параметрами, однак деякі випадки атеросклерозу зустрічаються навіть у людей із нормальною вагою та низьким рівнем холестерину, але вони незначні.

Так як кількість даних в датасеті невелика – всього 1000 записів, для кращої побудови та оцінки моделей ми використали техніку генерування синтетичних даних. Для цього ми скористалися платформою mostly.ai. Це онлайн-інструмент для генерації синтетичних даних. Цей інструмент дозволяє користувачам створювати великі та різноманітні набори даних з високою точністю та якістю, що можуть бути використані для тестування, навчання моделей машинного навчання та інших цілей.

Інтерфейс інструменту простий та інтуїтивно зрозумілий, що дозволяє користувачам швидко та ефективно генерувати нові дані. Інструмент має кілька вбудованих функцій, таких як зміна формату даних, зміна структури, зміна розподілу даних, зміна об'єму даних та інші. Загалом, <https://synthetic.mostly.ai> є потужним інструментом для генерації синтетичних даних, який дозволяє користувачам створювати великі та різноманітні набори даних з високою точністю та якістю, що можуть бути використані для тестування, навчання моделей машинного навчання та інших цілей.

На основі вже наявних даних ми згенерували ще два датасети такого ж розміру і таким чином пропорція синтетичних даних до реальних становить 2 до 1. Платформа генерує синтетичні дані на основі зв'язків і розподілів у вже існуючому датасеті, а також надає візуалізацію, приклад якої можна побачити на рис. 3.6. На рисунку можна побачити розподіл оригінальних та синтетичних даних.

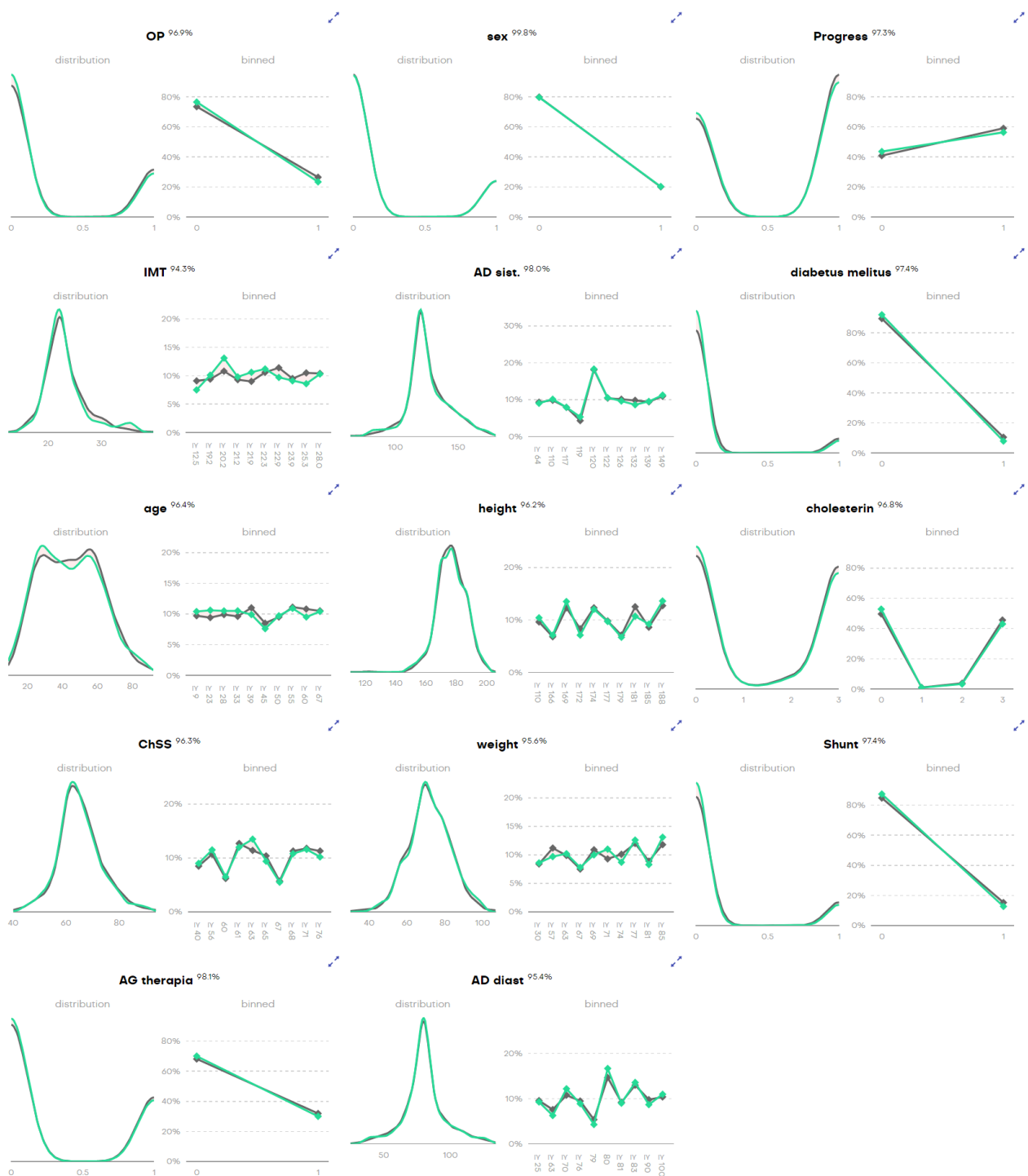


Рисунок 3.6 Візуалізація синтетичних даних.

3.2 Побудова моделей

Для більшості алгоритмів ми використаємо бібліотеку `sci-kit learn` яка містить реалізації багатьох алгоритмів машинного навчання. Для конфігурації

нейронної мережі ми використаємо keras і tensorflow. Реалізація нейронної мережі зображена на рис 3.7.

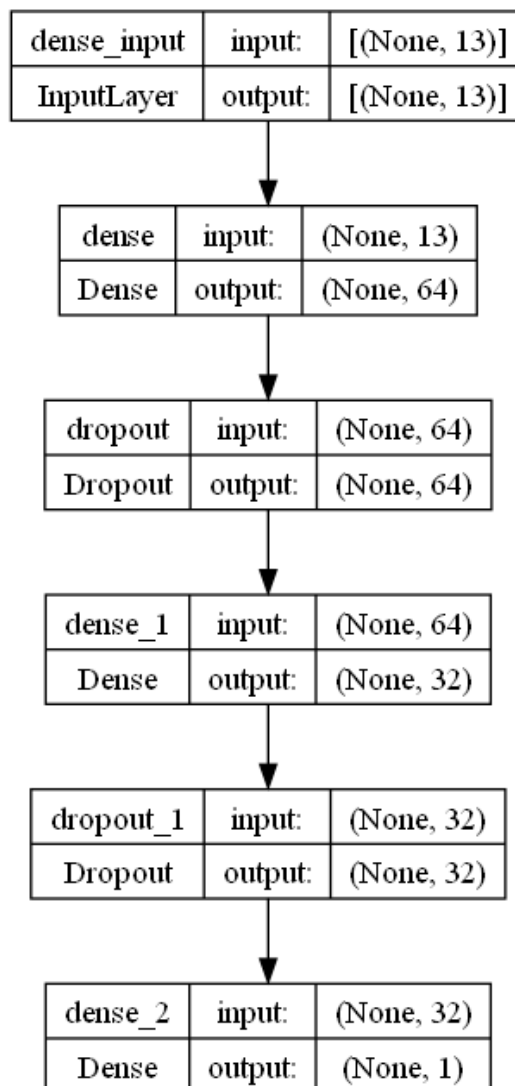


Рисунок 3.7 Архітектура нейронної мережі.

Нейронна мережі має послідовну архітектуру і складається з таких шарів:

1. Вхідний Dense шар з 64 нейронами та функцією активації relu. Цей шар отримує на вхід вектор, що дорівнює кількості стовпців у матриці (в нашому випадку 13).
2. Другий шар (Dense) з 32 нейронами та функцією активації relu. Цей шар також має регуляризацию l2 з параметром 0.01, що допомагає уникнути перенавчання моделі.

3. Третій шар (Dense) має лише один нейрон та функцію активації sigmoid. Цей шар використовується для бінарної класифікації.

Модель містить два шари Dropout, що допомагають уникнути перенавчання моделі шляхом випадкового вимкнення частини нейронів.

У процесі тренування моделі використовується алгоритм оптимізації adam, який дозволяє швидко знаходити локальні мінімуми функції втрат. Як функція втрат використовується бінарна перехресна ентропія (binary_crossentropy), а метрикою є точність класифікації (accuracy).

Для тренування нейронної мережі ми використаємо 500 епох та розмір вибірки для крос-валідації 20%:

Для решти моделей поділимо нашу вибірку 20% і 80% для тестування і тренування відповідно. Решта алгоритмів тренувалися з конфігурацією за замовчуванням. Слід зауважити, що для алгоритму Random Forest кількість дерев була обрана 20.

Ми докладніше зупинимося на нейронній мережі, оскільки вона складніша в конфігурації та налаштуванні. Для нейронної мережі ми використовували набір щільних шарів з випаданням, щоб уникнути перенавчання, і ReLU як функцію активації. Для вихідного шару ми використовували сигмоїдну функцію, бінарну перехресну функцію втрат ентропії, оскільки завданням моделі є бінарна класифікація та оптимізатор Адама. Набір даних був розділений на тестовий і навчальний набори, 20% і 80% відповідно. Навчальний процес складався з 500 епох, щоб досягти кращої точності результату. Візуалізація втрат моделі та підвищення точності моделі показано на малюнку 3.8.

Як бачимо, процес навчання нейронної мережі був збалансований без перенавчання і модель досягла хорошої точності.

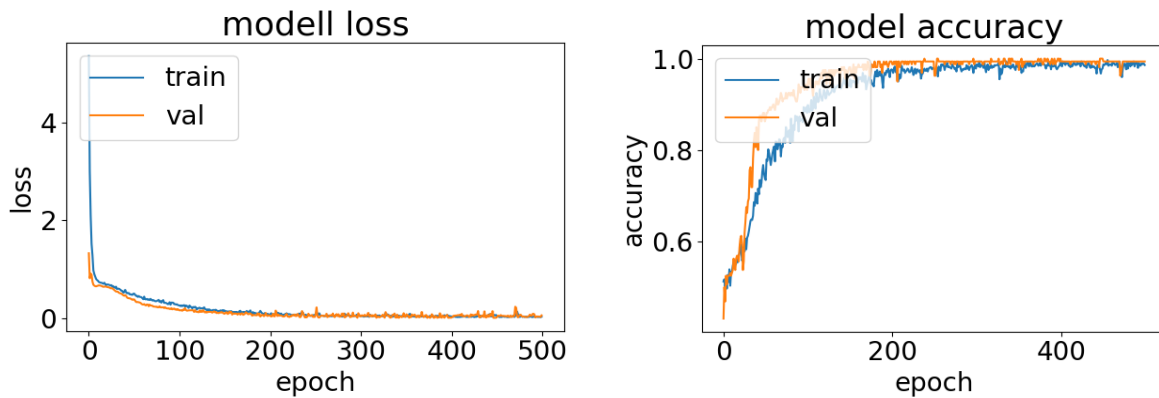


Рисунок 3.8 Навчальний графік нейронної мережі для точності та втрат

Ще одна річ, про яку слід згадати перед порівнянням результатів, - це структура дерева рішень. Класифікатор дерева рішень створює правила на основі параметрів, які дозволяють йому класифікувати дані. Таким чином, ця структура може допомогти зрозуміти, які параметри найбільше впливають на результат класифікації. Структура отриманого дерева рішень відображена на рисунку 3.9.

На основі рисунка 3.9 можна зробити висновок, що найважливішими параметрами для результату класифікації є *cholesterin*, *AD sist.* і *weight*. Так як алгоритм Decision Tree складається аж з двадцяти дерев рішень, візуалізація не має практичного сенсу та корисної інформації.

Алгоритм XGBoost базується на деревах рішень, і він працює шляхом навчання послідовної серії слабких моделей, кожна з яких намагається виправити помилки попередньої моделі. Це досягається шляхом зменшення помилок через вагові коефіцієнти та використання градієнтного спуску.

XGBoost є дуже гнучким та налаштовується на багато параметрів, що дозволяє користувачам налаштовувати алгоритм для різних задач та типів даних. Крім того, XGBoost підтримує паралельну обробку, що дозволяє прискорювати час тренування та покращувати ефективність алгоритму. Так як XGBoost – це складена модель, її візуалізація також є доволі ресурсоємною.

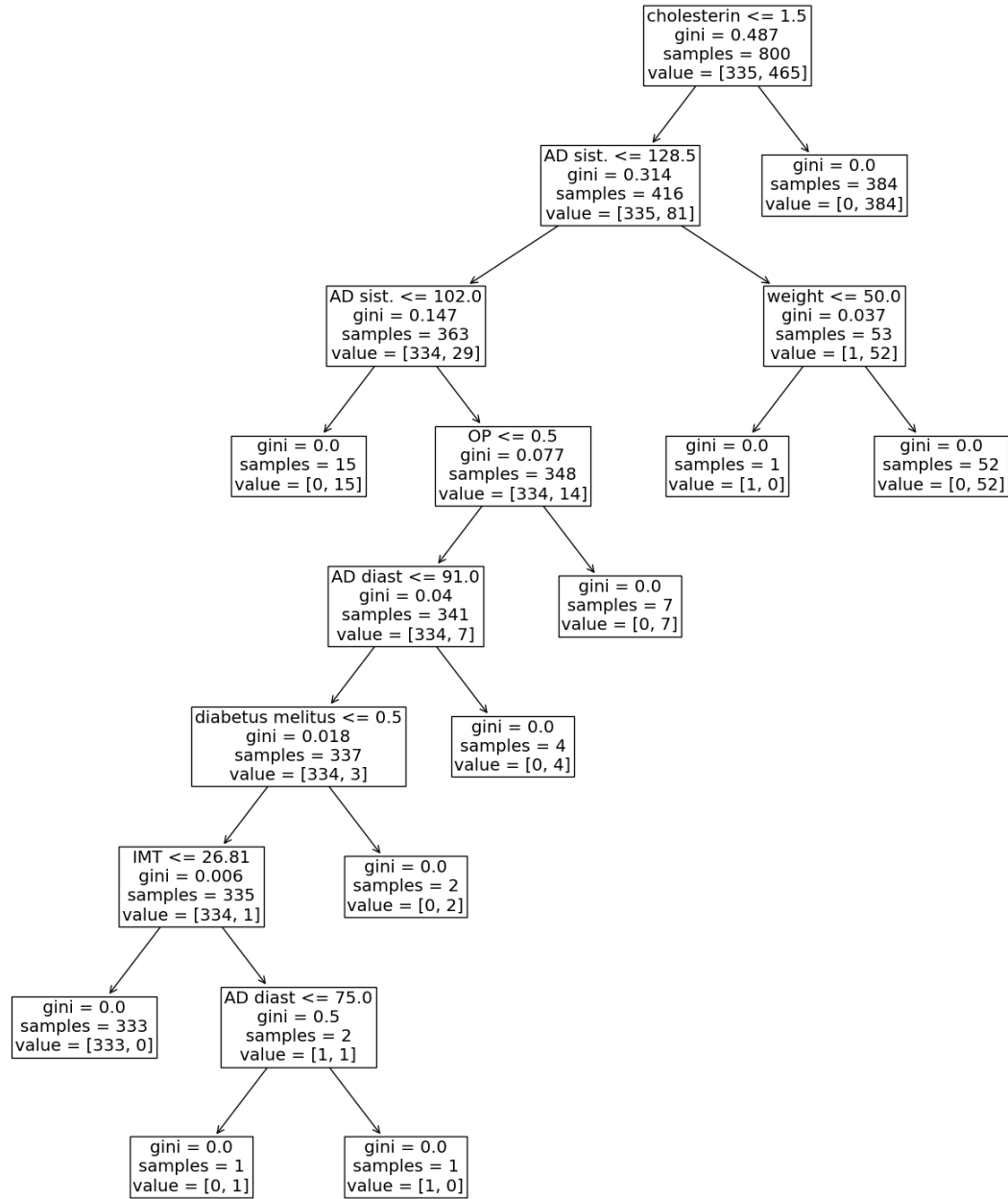


Рисунок 3.9 Структура дерева рішень

3.3 Отримання прогнозів

Тепер ми беремо кожну модель і отримуємо прогнози для 20% тестових даних, які ми відділили на етапі побудови моделі. Далі ми будуємо матрицю невідповідностей порівнюючи прогнози моделі та реальні значення, і

отримуємо відсоток точності моделі для даних тренування і тестування Матриці невідповідностей кожного алгоритму зображені на рис. 3.10.

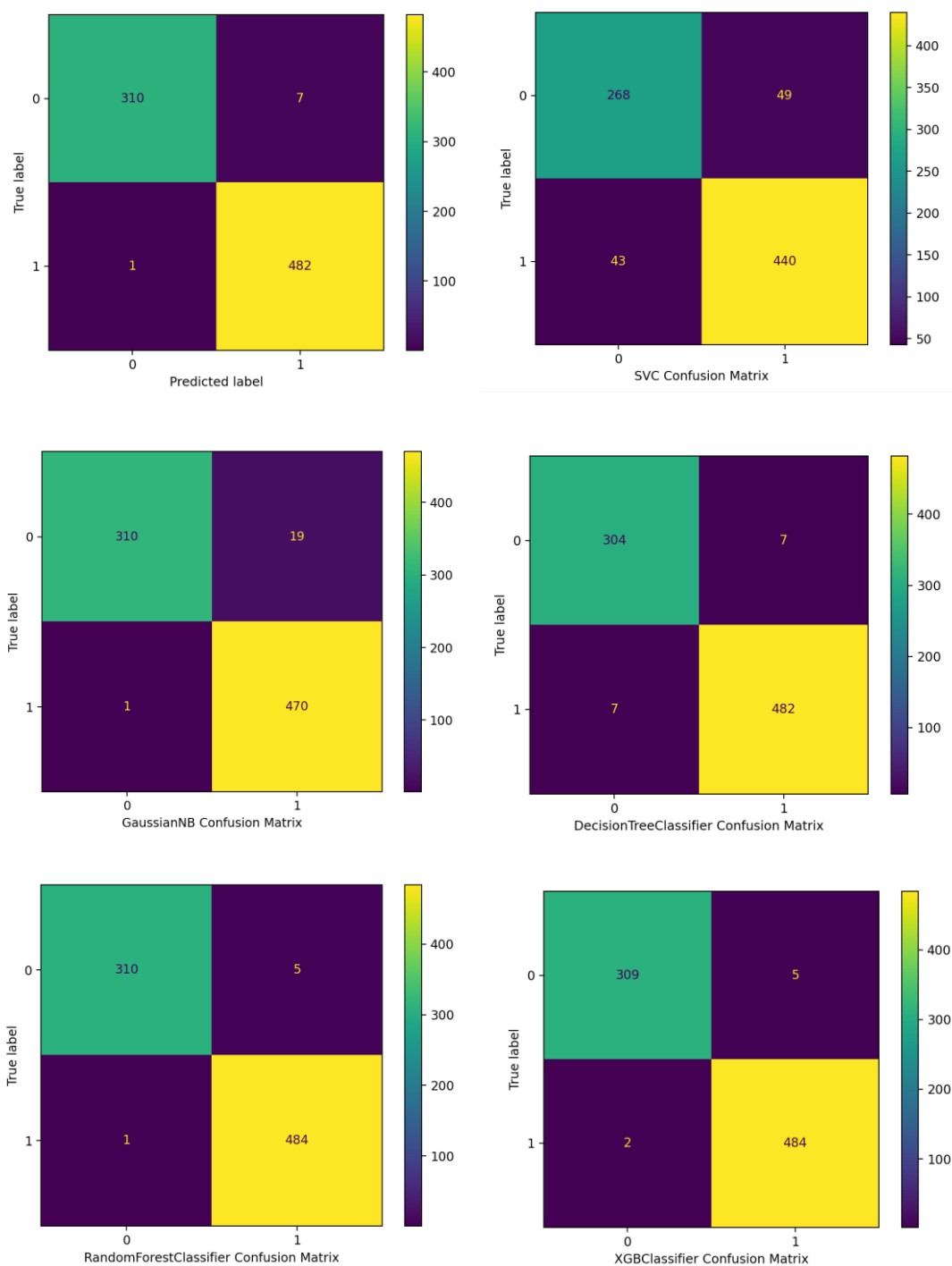


Рисунок 3.10 Матриці відповідностей тестових даних

Як ми можемо побачити з даних матриць відповідностей, найкраще себе проявили XGBoost, Random Forest та нейронна мережа, досягаючи 99.25%,

98.2% та 99% точності на тестових даних відповідно, при цьому точність передбачень нейронної мережі на другому місці.

Тепер ми можемо побудувати ROC-криву. Це графік, який використовується для візуалізації та аналізу результатів класифікації. Вона відображає залежність між чутливістю (true positive rate, TPR) та специфічністю (false positive rate, FPR) класифікатора при зміні порогу рішення.

ROC-крива будується шляхом розрахунку значень TPR та FPR для різних значень порогу рішення. Чим більше площа під кривою ROC (ROC AUC), тим краще якість класифікатора.

ROC-крива є корисним інструментом для оцінки якості класифікатора в тих випадках, коли розмір позитивного та негативного класів не збалансований, або коли класифікаційні помилки різних типів мають різні наслідки. Вона дозволяє зробити збалансовану оцінку ефективності класифікатора, з урахуванням помилок обох типів. ROC-крива зображена на рис. 3.11.

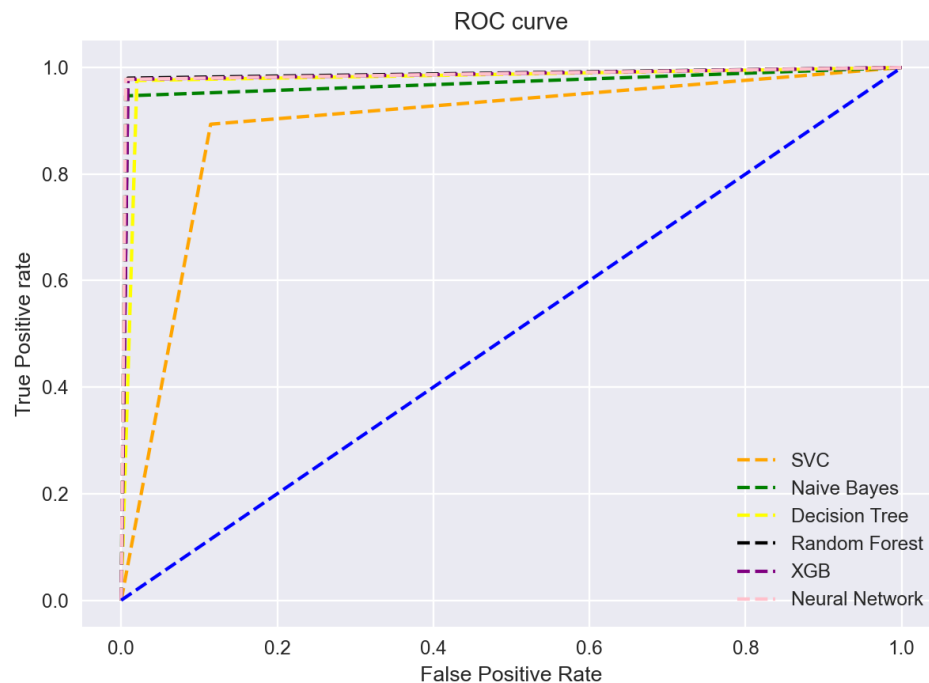


Рисунок 3.11 ROC-крива навчання алгоритмів.

З графіку можна сказати що процес тренування пройшов найкраще для алгоритмів XGBoost та нейронної мережі.

3.4 Оцінка якості моделей

Точність моделей для тестування та тренування наведена в таблиці 3.1

Таблица 3.1 - Точність моделей

Algorithm	Training accuracy	Test accuracy
SVM	0.895	0.885
Naive Bayes	0.975625	0.97
Decision Tree	1.0	0.96375
Random Forest	0.999375	0.98125
XGBoost	1.0	0.98625
Neural network	0.9878125	0.98875

Серед усіх застосовуваних алгоритмів XGBoost і Neural Network показали найкращу продуктивність як для процесу навчання, так і для тестування. Також слід зазначити, що всі застосовані моделі досягли дуже високої точності, що робить їх застосовними для прогнозування атеросклерозу в медичних установах.

Машинне навчання використовує передові алгоритми, які аналізують дані, навчаються на них і використовують ці знання для виявлення значущих шаблонів, що представляють інтерес. Тоді як нейронна мережа складається з

набору алгоритмів, що використовуються в машинному навчанні для моделювання даних з використанням графіків нейронів.

У той час як модель машинного навчання приймає рішення відповідно до того, що вона дізналася з даних, нейронна мережа організовує алгоритми таким чином, що вона може приймати точні рішення самостійно. Таким чином, хоча моделі машинного навчання можуть навчатися на даних, на початкових етапах вони можуть вимагати певного втручання людини.

Нейронні мережі не вимагають втручання людини, оскільки вкладені шари всередині передають дані через ієрархії різних концепцій, що в кінцевому підсумку робить їх здатними навчатися через власні помилки.

Таким чином нейронні мережі є більш сучасними та комплексними системами і дозволяють здійснювати інтелектуальний аналіз даних на складніших системах.

Тепер нам необхідно зробити критичний тест – беремо 3 набори даних. Перший, де чоловік здоровий, молодий, не має ніяких проблем зі здоров'ям. Другий – непевний, де людина здорова, але має певні погані звички, і останній – хвора людина, яка страждає на зайву вагу та має проблеми зі здоров'ям, Рис. 3.12 в такому порядку: OP, Shunt, age, height, weight, IMT, sex, ChSS, AD sist., AD diast, AG therapia, cholesterin, diabetes melitus.

```
=====1 - CRIT TEST")
[1, 0, 55, 175, 100, 24.7, 1, 90, 153, 93, 1, 3, 0]
=====1 - NORM TEST")
[0, 0, 20, 175, 70, 26.68, 1, 65, 120, 80, 0, 0, 0]
=====2 50/50")
[0, 0, 25, 175, 75, 23.42, 1, 60, 120, 80, 0, 2, 0]
```

Рисунок 3.12 Дані для критичного тесту.

Результати критичного тесту наведені в таблиці 3.2. Для тесту обрано три найточніших алгоритми за метриками точності під час тестування та матрицями невідповідності – це Neural Network, XGBoost та Random Forest.

Таблиця 3.2 – Результати критичного тесту

Тип пацієнта\Алгоритм	Neural Network	XGBoost	Random Forest
Здоровий	7.63%	0.0%	15.0%
Непевний	75.58%	4.22%	40.0%
Хворий	100.0%	100.0%	100.0%

Портрети пацієнтів:

Здоровий пацієнт – чоловік 20 років, зріст 175 см, вага 70 кг, пульс 65 уд/хв, тиск 120/80, холестерин завжди в нормі, діабетом не хворіє, хірургічних втручань не було.

Хворий пацієнт – чоловік 55 років, зріст 175 см, вага 100 кг, пульс 90 уд/хв, тиск 153/93, холестерин іноді вище норми, діабетом не хворіє, мав оперативні втручання та антигіпертензивну терапію.

Непевний – чоловік 25 років, зріст 175 см, вага 75 кг, пульс 60 уд/хв, тиск 120/80, холестерин вище в межах норми, діабетом не хворіє, хірургічних втручань не було.

Як бачимо з таблиці, для здорового пацієнта, Neural Network дає вірогідність хвороби 7%, що є задовільним показником, XGBoost дає 0%, а Random Forest 15.0%, що є доволі високим відсотком якщо враховувати характеристики пацієнта.

Для хворого пацієнта показники всіх трьох алгоритмів є доволі однозначними. А ось для непевного випадку показники різняться. Тут нам допоможе аналіз портрету пацієнта. Так як пацієнт зі зростом 175 см вживає 75 кг, то індекс маси тіла становить 24.5, коли 25 – порогове значення зайвої ваги. Крім того пацієнт має завищений показник холестерину, що може свідчити про нездоровий спосіб життя, тому вірогідність наявності ранніх стадій серцево-судинних хвороб є середньою. Якщо поглянемо на передбачення алгоритмів,

XGBoost показує 4%, що не є задовільним показником, і ми можемо стверджувати, що алгоритм перетренований в сторону заниження вірогідності хвороби. Якщо поглянемо на показник Random Forest, то побачимо вірогідність 40%, але при цьому показник для здорового пацієнта був 15%, тому можемо стверджувати що цей алгоритм також є неточним. Але показник нейронної мережі відповідає нашим критеріям, оскільки показник в 75% означає що наявність серцево-судинної хвороби є досить вірогідною.

Висновок до розділу 3

У цьому розділі ми проаналізували наявні дані, а також побудували графіки для візуалізації зв'язків та розподілів параметрів датасету. Для кращої якості моделі ми збільшили датасет за допомогою синтетичних даних, які були побудовані на оригінальному датасеті та переконались, що вони відповідають характеру оригінального датасету.

Після цього ми визначились із технологіями, які будемо використовувати для побудови моделей та створили і натренували 6 моделей машинного навчання на датасеті. Після тренування ми зібрали параметри тренування та переконались у якості тренувального процесу.

З отриманих моделей за допомогою матриць несумісностей ми отримали параметр точності моделей на тренувальному та тестовому датасетах, і визначили найточніші моделі, а саме Neural Network XGBoost Random Forest.

Для того щоб обрати найбільш відповідну нашим вимогам модель ми провели критичний тест і визначили, що нейромережа найточніше відображає сутність та зв'язки предметної області та найкраще підходить для наших потреб. В наступному розділі ми використаємо три найточніші моделі, побудовані в цьому розділі для створення інформаційної системи для взаємодії користувачів (лікарів та пацієнтів) з моделями машинного навчання.

РОЗДІЛ 4. ЗАСТОСУВАННЯ ПОБУДОВАНИХ МОДЕЛЕЙ ДЛЯ ПРОГНОЗУВАННЯ СЕРЦЕВО-СУДИННИХ ЗАХВОРЮВАНЬ В ІНФОРМАЦІЙНІЙ СИСТЕМІ

4.1 Аналіз технологічних інструментів

Вибір технологічного стеку для даного застосунку базується на вимогах до функціональності, ефективності та зручності використання. Для даного застосунку використано такі технології:

Python - мова програмування, що відома своєю простотою та зручністю використання, що дозволяє легко та швидко розробляти програми [29]. Python — це інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Його високорівневі вбудовані структури даних у поєднанні з динамічною типізацією та динамічним зв'язуванням роблять його дуже привабливим для швидкої розробки додатків, а також для використання як мови сценаріїв або з'єднувальної мови для з'єднання існуючих компонентів. Простий, легкий для вивчення синтаксис Python підкреслює читабельність і, отже, знижує вартість обслуговування програми. Python підтримує модулі та пакети, що заохочує модульність програми та повторне використання коду. Інтерпретатор Python і обширна стандартна бібліотека доступні у вихідному або двійковому вигляді безкоштовно для всіх основних платформ і можуть вільно поширюватися.

Часто програмісти обирають Python через підвищену продуктивність, яку він забезпечує. Оскільки етапу компіляції немає, цикл редагування-тестування-налагодження відбувається неймовірно швидко. Налаштовувати програми на Python легко: помилка чи неправильний вхід ніколи не спричинить помилку сегментації. Натомість, коли інтерпретатор виявляє помилку, він викликає виняток. Якщо програма не вловлює виняток, інтерпретатор друкує трасування стека. Налаштовувач рівня вихідного коду дозволяє перевіряти локальні та

глобальні змінні, оцінювати довільні вирази, встановлювати контрольні точки, покроково виконувати код по рядках і так далі. Сам налагоджувач написаний на Python, що свідчить про інтроспективну силу Python. З іншого боку, часто найшвидшим способом налагодження програми є додавання кількох операторів друку до джерела: швидкий цикл редагування-тестування-налагодження робить цей простий підхід дуже ефективним.

Pandas - бібліотека для обробки та аналізу даних, що дозволяє зручно працювати з табличними даними та виконувати їх обробку [28]. Як бібліотека програмного забезпечення з відкритим вихідним кодом, створена на основі Python спеціально для обробки та аналізу даних, Pandas пропонує структуру даних і операції для потужного, гнучкого та легкого у використанні аналізу й обробки даних. Pandas покращує Python, надаючи цій популярній мові програмування можливість працювати з даними, подібними до електронних таблиць, уможливорюючи швидке завантаження, вирівнювання, маніпулювання та об'єднання на додаток до інших ключових функцій. Pandas цінується за високооптимізовану продуктивність, коли внутрішній вихідний код написаний на C або Python.

Назва «Pandas» походить від економетричного терміну «панельні дані», що описує набори даних, які включають спостереження за кілька періодів часу. Бібліотека Pandas була створена як інструмент високого рівня або будівельний блок для виконання дуже практичного аналізу реального світу на Python. У майбутньому її творці мають намір перетворити Pandas на найпотужніший і найгнучкіший інструмент аналізу та обробки даних із відкритим кодом для будь-якої мови програмування.

Pandas, є одним з найпопулярніших і широко використовуваних інструментів для так званої суперечки даних або munging. Це набір концепцій і методологія, яка використовується для переведення даних із непридатних або помилкових форм до рівнів структури та якості, необхідних для сучасної

аналітичної обробки. Pandas відрізняється простотою роботи з форматами структурованих даних, такими як таблиці, матриці та дані часових рядів. Він також добре працює з іншими науковими бібліотеками Python.

Streamlit - фреймворк для створення веб-додатків з мінімальними зусиллями та без необхідності в розробці складних інтерфейсів.

Keras - високорівнева бібліотека для розробки нейронних мереж, що дозволяє легко створювати та тренувати моделі. Keras — це бібліотека компонентів нейронної мережі з відкритим кодом, написана мовою Python. Keras здатний працювати поверх TensorFlow, Theano, PlaidML та інших. Бібліотека була розроблена як модульна та зручна для користувача, однак спочатку вона була частиною дослідницького проекту для відкритої нейроелектронної інтелектуальної операційної системи або ONEIROS. Основним автором Keras є Франсуа Шолле, інженер Google, який також написав Xception, модель глибокої нейронної мережі. Хоча Keras було офіційно запущено, його не було інтегровано в основну бібліотеку Google TensorFlow до 2017 року. Додаткову підтримку також було додано для інтеграції Keras із Microsoft Cognitive Toolkit.

Платформа Keras із відкритим кодом, що складається з бібліотеки часто використовуваних компонентів машинного навчання, включаючи цілі, функції активації та оптимізатори, також пропонує підтримку рекурентних і згорткових нейронних мереж. Крім того, Keras пропонує розробку мобільних платформ для користувачів, які мають намір реалізувати моделі глибокого навчання на смартфонах як iOS, так і Android. Станом на 2018 рік бібліотека використовується на 22% із понад 200 000 користувачів.

XGBoost - бібліотека для роботи з градієнтним бустингом, що дозволяє покращити якість передбачень. XGBoost, що розшифровується як Extreme Gradient Boosting, — це масштабована бібліотека машинного навчання з розподіленим деревом рішень із підсиленням градієнта (GBDT). Він забезпечує

паралельне прискорення дерева та є провідною бібліотекою машинного навчання для проблем регресії, класифікації та ранжування. Для розуміння XGBoost життєво важливо спершу зрозуміти концепції та алгоритми машинного навчання, на яких базується XGBoost: контрольоване машинне навчання, дерева рішень, ансамблеве навчання та посилення градієнта.

Контрольоване машинне навчання використовує алгоритми для навчання моделі пошуку шаблонів у наборі даних із мітками та функціями, а потім використовує навчену модель для прогнозування міток на функціях нового набору даних.

Використання такого технологічного стеку дозволяє легко та швидко розробити застосунок з високою точністю передбачень. Pandas забезпечує зручну обробку та аналіз даних, Streamlit дозволяє створити зручний та легкий у використанні інтерфейс для користувачів, Keras допомагає в створенні та тренуванні нейронних мереж, а XGBoost забезпечує підвищення якості передбачень. Основною метою застосунку є передбачення наявності хвороби на основі введених користувачем даних про своє здоров'я та медичну історію. Так як в даному випадку маємо справу з задачею класифікації, то використання різних моделей машинного навчання є досить обґрунтованим.

Streamlit був вибраний як фреймворк для створення інтерактивного інтерфейсу, що дозволяє користувачеві ввести свої дані та отримати прогноз. Streamlit є досить простим та легким інструментом, який дозволяє швидко розробити та випустити прототип застосунку без великих затрат на розробку інтерфейсу. Streamlit — це безкоштовний фреймворк із відкритим вихідним кодом для швидкого створення та спільного використання чудових веб-додатків для машинного навчання та обробки даних [31]. Це бібліотека на основі Python, спеціально розроблена для інженерів машинного навчання. Науковці даних або інженери з машинного навчання не є веб-розробниками, і вони не зацікавлені витрачати тижні на навчання використовувати ці фреймворки для створення

веб-додатків. Натомість їм потрібен інструмент, який легше вивчати та використовувати, якщо він може відображати дані та збирати необхідні параметри для моделювання. Streamlit дозволяє створити приголомшливий додаток лише за допомогою кількох рядків коду.

Найкраще у Streamlit полягає в тому, що вам навіть не потрібно знати основи веб-розробки, щоб почати чи створити свою першу веб-програму. Одним із важливих аспектів успішної програми є ефективний та інтуїтивно зрозумілий інтерфейс користувача. Багато сучасних додатків, які використовують велику кількість даних, стикаються з проблемою швидкого створення ефективного інтерфейсу користувача без виконання складних кроків. Streamlit — це багатообіцяюча бібліотека Python з відкритим вихідним кодом, яка дозволяє розробникам миттєво створювати привабливі інтерфейси користувача.

Для збереження та завантаження моделей використовується бібліотека Joblib, яка дозволяє зручно зберігати та завантажувати моделі з диску. Joblib — це бібліотека Python для паралельного виконання завдань, що містять інтенсивні обчислення [30]. Він надає набір функцій для паралельного виконання операцій над великими наборами даних і для кешування результатів обчислювально дорогих функцій. Joblib особливо корисний для моделей машинного навчання, оскільки він дозволяє зберегти стан ваших обчислень і відновити роботу пізніше або на іншій машині.

Порівняно з іншими методами зберігання та завантаження моделей машинного навчання використання Joblib має ряд переваг. Оскільки дані зберігаються як рядки байтів, а не як об'єкти, їх можна швидко й легко зберігати в меншому обсязі, ніж традиційно. Крім того, він автоматично виправляє помилки під час читання або запису файлів, що робить його більш надійним, ніж маринування вручну. І останнє, але не менш важливе:

використання Joblib дає змогу зберігати численні ітерації однієї моделі, полегшуючи їх порівняння та визначення найточнішої.

Joblib забезпечує багатопроцесорну роботу на кількох машинах або ядрах на одній машині, що дозволяє програмістам розпаралелювати завдання на багатьох машинах. Це полегшує програмістам використання розподілених обчислювальних ресурсів, таких як кластери або графічні процесори, для прискорення процесу навчання моделі.

Отже, в цілому можна сказати, що технологічний стек, що використовується в даному застосунку, є оптимальним та дозволяє ефективно вирішувати поставлену задачу.

4.2 Створення та розгортання інформаційної системи

Для створення інформаційної системи перш за все після процесу тренування нам необхідно зберегти отримані моделі для подальшого використання. Моделі машинного навчання, створені за допомогою бібліотеки scikit-learn ми можемо портувати в файли з розширенням .pkl за допомогою бібліотеки joblib. Але модель нейронної мережі не підтримує цей формат тому її потрібно серіалізувати засобами keras в файл з розширенням .h5.

Отримавши необхідні моделі, нам потрібно створити репозиторій та зберегти моделі на github. Git - це розподілена система керування версіями, яка дозволяє зберігати та відстежувати зміни в коді проекту та спільно працювати над ним з допомогою різних гілок (branches) та внести зміни, не впливаючи на інші частини коду.

GitHub - це веб-сервіс, який надає хмарне сховище для Git-репозиторіїв. GitHub забезпечує централізований доступ до Git-репозиторіїв, дозволяє розповсюджувати вихідний код проекту, спільно працювати над ним, використовувати різні сервіси для автоматизації процесу розробки, такі як CI/CD, підтримувати відкритий вихідний код та інші корисні функції для


командної розробки. GitHub також надає можливість деплоювати веб-сайти та веб-додатки на хмарному сервісі GitHub Pages.

Для початку необхідно створити новий репозиторій, як показано на рис. 4.1:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

 vlad-lavrynovych ▾

Repository name *

/

Great repository names are short and memorable. Need inspiration? How about [supreme-spoon?](#)

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None ▾

Рисунок 4.1 Створення нового репозиторію.

Після цього нам необхідно клонувати репозиторій за допомогою команди `git clone` в папку на локальному ПК, і перемістити туди файли моделей. Після цього ми повинні створити новий python скрипт який назвемо `app.py` в якому ми пропишемо вигляд та логіку нашої інформаційної системи. Але перед цим нам необхідно розібратися із залежностями, які будемо використовувати.

Для роботи з залежностями, в python використовується файл під назвою requirements.txt. Під час створення віртуальної машини в хмарі streamlit викликає менеджер пакетів pip та передає йому цей файл як аргумент. Використовуючи цей файл менеджер пакетів встановлює на віртуальну машину необхідні залежності. Розроблений у 2008 році pip (акронім «pip Install Packages») сьогодні є стандартним інструментом для безпечного встановлення пакетів Python та їхніх залежностей. Найновіші дистрибутиви Python постачаються з попередньо встановленим pip. Python 2.7.9 і Python 3.4 і новіші версії містять pip за замовчуванням.

Після цього нам необхідно наповнити інструкціями сам скрипт. Логіка наступна: спочатку завантажуюмо всі моделі які є і зберігаємо в словник. Після цього створюємо наповнення для сайту – поля вводу, слайдери та кнопки відповідно до параметрів датасету. Далі ми створюємо кнопку зробити передбачення, а поруч – випадаючий список з назвами моделей, щоб користувач міг самостійно обрати потрібну модель.

Текстові поля нам необхідно перетворити в числові дані, тому попередньо всі категоріальні параметри ми зберігаємо як константи, та використовуємо їх індекси. Поле ІМТ (Індекс Маса Тіла) ми будемо вираховувати автоматично на основі інших параметрів, таких як вік, зріст та вага користувача. Слід зауважити, що порядок полів при використанні даних для отримання передбачення моделі важливий, оскільки модель тренується на датасеті з фіксованим порядком параметрів, які повинні відповідати вхідним даним при отриманні передбачення.

Результат розробки інформаційної системи та готовий сайт можна переглянути на рис. 4.2. Як бачимо з рисунку, інтерфейс доволі простий та зручний для кінцевого користувача, що робить користування застосунком приємним та допомагає збільшити аудиторію клієнтів інформаційної системи.

Прогнозування серцево-судинних хвороб

Стать

Чоловіча

Вік

30

Зріст (см)

181.0

100.0 220.0

Вага (кг)

76.0

0.0 150.0

Пульс (в стані спокою)

70

Систолічний тиск

120

Діастолічний тиск

80

Показник холестерину:

Вище в межах норми - від 4,7 ммоль/л до 5

Ви хворієте на діабет?

Ні

Чи були оперативні втручання стосовно серцево-судинної системи?

Ні

Чи робили Вам шунтування артерій?

Ні

Чи проходили ви антигіпертензивну терапію?

Ні

Передбачити

Нейронна мережа
RandomForestClassifier.pkl
XGBClassifier.pkl
Нейронна мережа

Вірогідність наявності серцево-судинної хвороби становить 34.87%

Рисунок 4.2 Інтерфейс додатку.

Після створення локальної, необхідно задеплоїти додаток в хмару для загального доступу, щоб користувачі мали змогу користуватися ним по всьому світу. Серед найпопулярніший хостингів на даний момент є: Streamlit Sharing - це офіційний хостинг для додатків, розроблених у Streamlit. Він безкоштовний для використання та забезпечує легкий деплоймент додатків на основі Git репозиторіїв.

Heroku - це хостинг-провайдер, який підтримує багато мов програмування, включаючи Python. Він простий у використанні та дозволяє швидко розмістити додаток Streamlit в Інтернеті.

Google Cloud Platform - це інший популярний хостинг-провайдер, який підтримує Python та інші мови програмування. Цей сервіс дозволяє розміщувати додатки Streamlit на серверах Google та керувати ними зі зручного веб-інтерфейсу.

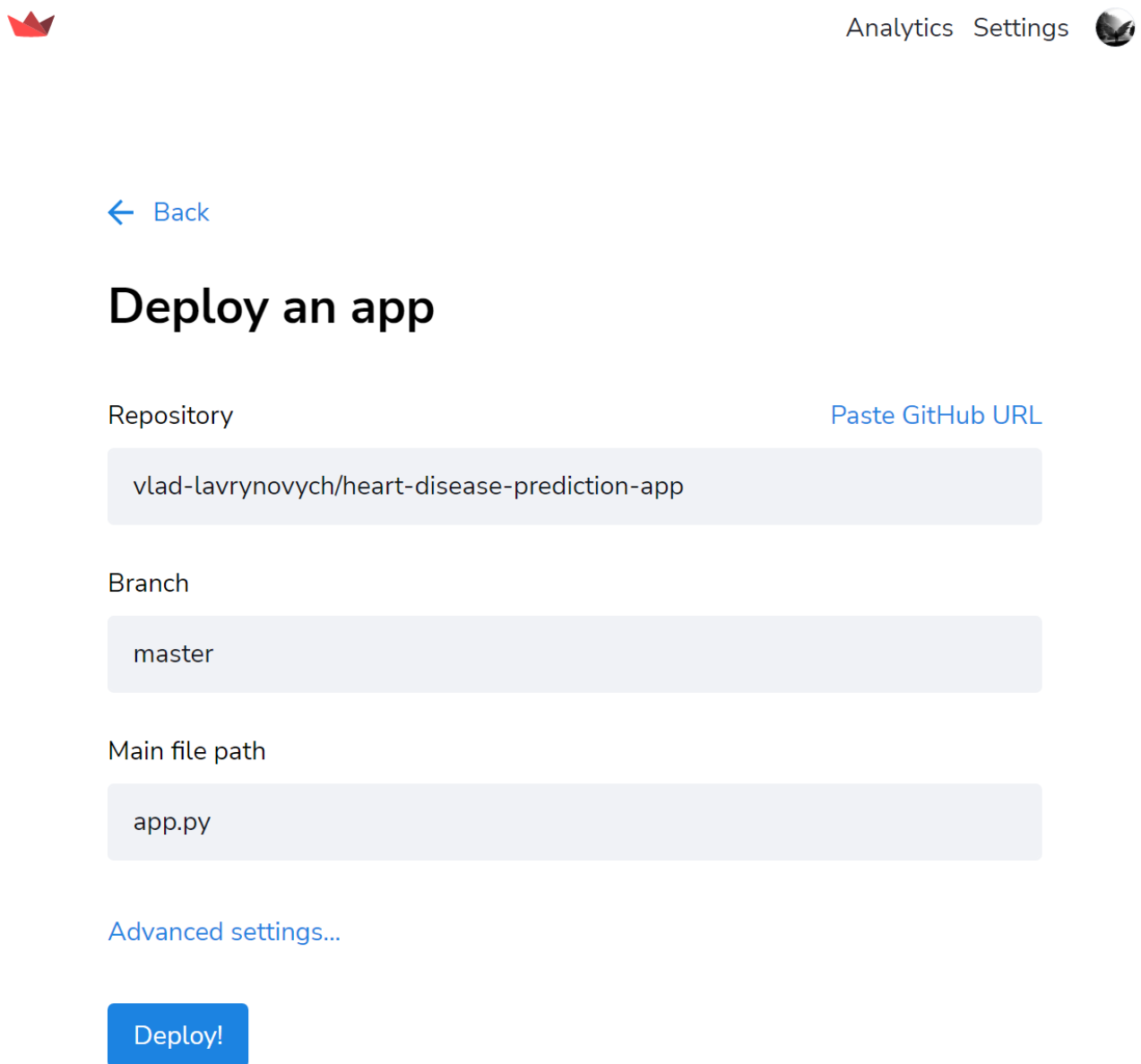
AWS: Amazon Web Services (AWS) - це хостинг-провайдер, який підтримує багато мов програмування, включаючи Python. Він має багато функцій, таких як автоматичне масштабування, що дозволяє розміщувати додатки Streamlit на високопродуктивних серверах. Більшість з цих сервісів платні, але на щастя streamlit надає безкоштовну платформу для хостингу додатків.

Streamlit Hosting - це безкоштовна хмарна платформа, яка дозволяє розгорнути ваші Streamlit-додатки з мінімальними зусиллями. Ви можете використовувати цю платформу для простого та швидкого розгортання ваших додатків у хмарі та доступу до них з будь-якого місця та пристрою. Для розгортання свого додатку на Streamlit Hosting необхідно виконати декілька простих кроків:

- Створити акаунт на Streamlit Cloud.
- Створити новий проект і завантажити в нього ваш додаток.
- Налаштувати налаштування проекту та вибрати параметри розгортання.
- Запустити розгортання вашого додатку на хмарному сервері.

- Крім того, Streamlit Hosting надає інші корисні функції, такі як автоматичні оновлення додатків та налаштування доступу до них для декількох користувачів.

Загалом, Streamlit Hosting є чудовим варіантом для тих, хто шукає простий та швидкий спосіб розгорнути свій додаток в хмарі, забезпечивши його доступність та зручний доступ до нього. Приклад створення та заливки проекту на streamlit hosting з використанням git репозиторію наведений на рис. 4.3



Analytics Settings

← Back

Deploy an app

Repository [Paste GitHub URL](#)

vlad-lavrynovych/heart-disease-prediction-app

Branch

master

Main file path

app.py

[Advanced settings...](#)

Deploy!

Рисунок 4.3 Розгортання застосунку на платформі Streamlit Cloud.

Після розгортання застосунку нам стане доступний інтерфейс адміністратора, де ми зможемо переглянути логи застосунку, а також виконувати певні дії, як перезавантажити додаток, видалити додаток та ін. Слід зауважити, що при оновленні репозиторію на github, streamlit автоматично підтягне нову версію застосунку та перезавантажить додаток. Вигляд логу застосунку під час підтягнення нової версії зображений на рис 4.4.

```
1/1 [=====] - 0s 46ms/step-05-13 18:51:11.568530
[18:52:01] 🦋 Pulling code changes from Github...
[18:52:02] 📁 Processing dependencies...
[18:52:02] 📁 Processed dependencies!
[18:52:05] 🔄 Updated app!
  OP  Shunt  age  height  weight  IMT  sex  ChSS  AD  sist.  AD diast  \
0   0      0   20   189.0   68.0  0.0   0    70    120    80
  AG therapia  cholesterin  diabetes melitus
0              0              0
1/1 [=====] - 0s 51ms/step-05-13 18:52:16.372465
  OP  Shunt  age  height  weight  IMT  sex  ChSS  AD  sist.  AD diast  \
0   0      0   20   189.0   68.0  0.0   0    70    120    80
  AG therapia  cholesterin  diabetes melitus
0              0              0
1/1 [=====] - 0s 45ms/step
  OP  Shunt  age  height  weight  IMT  sex  ChSS  AD  sist.  AD diast  \
0   0      0   20   189.0   78.0  0.0   0    70    120    80
  AG therapia  cholesterin  diabetes melitus
0              0              0
```

Рисунок 4.4 Вкладка з логом застосунку.

4.3 Перспективи використання застосунку

ІІІ зменшує витрати на охорону здоров'я порівняно зі звичайними методами. Раніше було показано, що економія коштів завдяки штучному інтелекту при лікуванні є більш ефективною порівняно з діагностикою. ІІІ скорочує час діагностики та лікування порівняно зі звичайними методами [43]. За короткий час можна досягти високої точності діагностики та лікування. ІІІ допомагає підвищити діагностичну точність, усуваючи упередження та

суб'єктивність. Медична діагностика на основі ШІ зменшує ймовірність неточного обстеження. Завдяки технології штучного інтелекту пацієнти можуть відчувати себе спокійніше під час звернення до лікаря. AI фільтрує значну кількість даних, щоб визначити, яка терапія дасть найкращі результати. Впровадження технології штучного інтелекту в охорону здоров'я може не тільки зменшити витрати, але й допомогти організаціям збільшити рентабельність інвестицій.

Методи DL привернули багато уваги, оскільки вони часто перевершують людей у таких завданнях, як системи рекомендацій, розпізнавання мови та зображень та багато інших [44]. Однак ці програми не є надійними та зрозумілими. Поширене помилкове уявлення про моделі DL полягає в тому, що вони є непрозорими, складними для розуміння чорними ящиками зі складними базовими механізмами. Однак, залежно від програми, помилки систем ШІ можуть бути катастрофічними. У медичній промисловості від цих рішень залежить життя пацієнтів, тоді як помилка системи комп'ютерного зору безпілота приведе до авіакатастрофи.

Машинне навчання може допомогти в серцево-судинних хвороб за допомогою медичних вимірювань та історії хвороб. Застосовуючи свою перевагу в обчислювальній потужності для проведення аналізу даних швидше, ніж це можуть робити медичні працівники самостійно, машинне навчання може завершувати скринінг за менший час. Це може скоротити час очікування направлення для пацієнтів з високим ризиком і зменшити навантаження на клініки, які стикаються з нестачею персоналу або іншими проблемами.

Технології машинного навчання можуть також покращити послідовність і точність діагностики, усунувши ситуації, які сприяють людському фактору. Наприклад, спеціалісти, які проводять діагностику, залежать від таких факторів, як втома, і можуть по-різному інтерпретувати дані.

Машинне навчання також може розширити доступ до медичної допомоги. Деякі регіони та групи населення мають обмежений доступ до медичних працівників. Ця нова технологія може автоматизувати певні завдання, що, у свою чергу, може зменшити клінічне навантаження та дати можливість неспеціалістам виконувати складні завдання, такі як візуалізація та аналіз серця. Це може дозволити медичним працівникам охопити більші верстви населення за допомогою догляду вдома або в невеликих клінічних установах і забезпечити більшій кількості пацієнтів доступ до медичної допомоги.

Звичайно, модель потребує розширення та збільшення датасету для підвищення точності прогнозування серцево-судинних хвороб у пацієнтів [45]. Також необхідно розширити кількість показників щоб враховувати інші характеристики пацієнтів, які можуть включати спосіб життя, кількість та якість їжі та ін. Це може покращити точність прогнозувань та зробити систему більш дружньою та зрозумілою для користувача.

ВИСНОВКИ

У цій роботі було проаналізовано 6 методів машинного навчання для прогнозування атеросклерозу. Ми провели аналіз датасету, навчили та протестували всі алгоритми на основі клінічних даних. Ми досягли багатообіцяючих результатів, після чого було порівняно точність моделей. У більшості країн не вистачає досвіду в галузі серцево-судинної діяльності та є значний відсоток неправильно діагностованих випадків, які можна вирішити шляхом розробки точного та ефективного прогнозування серцево-судинних захворювань на ранніх стадіях шляхом аналітичної підтримки прийняття клінічних рішень за допомогою цифрових записів пацієнтів.

Усі моделі показали надзвичайно високі показники ефективності та показали кращі результати в цьому дослідженні в порівнянні з розглянутими прикладами з набором даних про захворювання CAD. Ми використали матрицю невідповідності для порівняння продуктивності алгоритмів ML для навчальних і тестових наборів та визначили найефективніші алгоритми. Багато дослідників відзначають, що алгоритми ML показують кращу продуктивність для невеликих наборів даних, тоді як нейронні мережі глибокого навчання краще для великих масштабованих даних. Однак при правильному налаштуванні гіперпараметрів та архітектурі можна досягти хороших результатів навіть для невеликих наборів даних, що було доведено в цьому дослідженні.

Також ми проаналізували попередні використання наведених цій роботі алгоритмів та їх особливостей, а також їх ефективність застосування до датасетів меншого розміру. Ми визначили алгоритми підвищення точності моделей в попередніх дослідженнях та підібрали параметри тренування моделей для того, щоб досягти максимальної ефективності для даного датасету.

Ми проаналізували датасет атеросклерозу та визначили його основні характеристики за допомогою низки візуалізацій. Було визначено відношення кількості хворих та здорових записів датасету а також їх порівняння за статтю

для перевірки збалансованості датасету. Також ми створили теплову карту кореляції атрибутів датасету для визначення основних кореляційних зв'язків в датасеті та атрибутів які найбільше впливають на результуюче значення, яке ми і намагаємось передбачити. В цій роботі нашою ціллю було застосування алгоритмів машинного навчання для задачі бінарної класифікації, оскільки наші моделі тренувались на датасеті з цільовими значеннями, які відповідали наявності або відсутності атеросклерозу.

Після визначення важливості атрибутів, ми зробили візуалізацію даних за найбільш корельованими числовими атрибутами для того, щоб визначити розподіл записів із наявним та відсутнім атеросклерозом. Така візуалізація крім того, що показала чітке візуальне розділення між екземплярами датасету різних класів, також підтвердила основні тенденції виникнення та розвитку атеросклерозу, як старший вік, більша вага та високий рівень холестерину в крові. Також ми зробили окрему візуалізацію розподілу даних за віком, визначили наближеність розподілу до нормального та підтвердили попередні висновки, про те, що в середньому вік людей хворих на атеросклероз є більшим.

На основі отриманих моделей ми обрали три найточніші та інтегрували їх в інформаційну систему, а саме веб застосунок для взаємодії з користувачем. Перспективи застосування такого застосунку полягають у використанні в лікарнях для покращення точності діагностування, для поліпшення обізнаності населення про серцево-судинні хвороби та ін.

Враховуючи обмеження дослідження, існує багато широких можливостей для застосування згаданих методів до даних більшого розміру, що, однак, може призвести до більших технічних проблем, таких як складна попередня обробка даних та налаштування алгоритмів. Крім того, високі показники, досягнуті при роботі з цим датасетом можуть також характеризувати вплив зв'язків атрибутів датасету на моделі, і відповідно при роботі з більшими датасетами, необхідно

буде окремо підбирати параметри для тренування моделей, і точність може варіюватися відповідно до збалансованості та ступені сили кореляції.

Також наявність ще одного датасету з такими ж атрибутами дасть змогу перевірити правдивість результатів роботи та передбачення моделей, оскільки ми зможемо порівнювати передбачення моделі з ще не відомими моделі даними іншого датасету з іншими зв'язками та закономірностями, і таким чином незалежно від якості зібраних даних (при незбалансованих даних) давати гарні передбачення.

У цій роботі ми також зробили декілька візуалізацій які відображають результат тренування моделей. Для дерева рішень ми відобразили архітектуру вузлів, і виявили що найбільше на результат впливає рівень холестерину в крові. Наступним по важливості атрибутом був систолічний тиск. Крім того ми здійснили візуалізацію зміни точності та похибки при тренуванні нейронної мережі. Базуючись на цих графіках ми можемо сказати, що процес тренування був рівномірним і збалансованим.

Щодо перспектив дослідження, можна застосувати багато інших архітектур нейронних мереж, а також методи ML для досягнення кращих результатів. Хоча сьогодні існує дуже обмежена кількість наборів даних для аналізу атеросклерозу (що робить цю сферу привабливою для багатьох дослідників), існує багато можливих інтеграцій розглянутих методів разом з комп'ютерним баченням та іншими технологіями, які можуть покращити діагностику та лікування атеросклерозу.

Ми створили декілька моделей які інтегрували у веб-застосунок який може бути використаний у великій кількості медичних закладів, що створить можливості для ранньої діагностики атеросклерозу та дасть великі шанси на попередження більш серйозних захворювань.

СПИСОК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Fact sheets Cardiovascular diseases (CVDs) / World Health Organization, 2021. Режим доступу: [https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-\(cvds\)](https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)).
2. Cardiovascular diseases Overview / World Health Organization, 2021. Режим доступу: <https://www.who.int/health-topics/cardiovascular-diseases>.
3. Rafieian-Kopaei M, Setorki M, Douidi M, Baradaran A, Nasri H. Atherosclerosis: Process, Indicators, Risk Factors and New Hopes. *Int J Prev Med* (2014) 5(8):927–46.
4. Konstantina Kourou, Themis P. Exarchos, Konstantinos P. Exarchos, Michalis V. Karamouzis, Dimitrios I. Fotiadis “Machine learning applications in cancer prognosis and prediction.” *Computational and Structural Biotechnology Journal*, Volume 13, с. 8-17, 2015.
5. D. P. Yadav and S. Rathor, “Bone Fracture Detection and Classification using Deep Learning Approach,” 2020 International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC), 2020, с. 282-285.
6. Verma AK, Pal S, Kumar S. Classification of Skin Disease using Ensemble Data Mining Techniques. *Asian Pac J Cancer Prev*. 2019 Jun 1;20(6):1887-1894.
7. UCI Machine Learning Repository, “Heart disease data set,” 2021, Режим доступу: <http://archive.ics.uci.edu/ml/datasets/heart+disease>.
8. Singh P, Singh S, Pandi-Jain GS. Effective heart disease prediction system using data mining techniques. *Int J Nanomedicine*. 2018 Mar 15;13(T-NANO 2014 Abstracts):121-124.
9. O. Terrada, B. Cherradi, A. Raihani and O. Bouattane, "Atherosclerosis disease prediction using Supervised Machine Learning Techniques," 2020 1st

- International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET), 2020, pp. 1-5.
10. Munger E, Hickey JW, Dey AK, Jafri MS, Kinser JM, Mehta NN. Application of machine learning in understanding atherosclerosis: Emerging insights. *APL Bioeng*. 2021 Feb 16;5(1):011505.
 11. Y.Khlevna, D.Mochalova. Prediction of atherosclerosis disease with artificial neural network. *Sciences of Europe. Technical sciences*. VOL 1, No 50 (2020) c. 53 –58.
 12. Zhu Y, Wu J, Fang Y. [Study on application of SVM in prediction of coronary heart disease]. *Sheng Wu Yi Xue Gong Cheng Xue Za Zhi*. 2013 Dec 30(6):1180-5. Chinese.
 13. 6 Easy Steps to Learn Naive Bayes Algorithm with codes in Python and R / Sunil Ray, 2017. Режим доступа:
<https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained>
 14. Oumaima Terrada, Bouchaib Cherradi, Abdelhadi Raihani, Omar Bouattane, A novel medical diagnosis support system for predicting patients with atherosclerosis diseases, *Informatics in Medicine Unlocked*, Volume 21, 2020.
 15. Qawqzeh, Y.K.; Otoom, M.M.; Al-Fayez, F.; Almarashdeh, I.; Alsmadi, M. and Jaradat, G. A Proposed Decision Tree Classifier for Atherosclerosis Prediction and Classification. *IJCSNS*, 2019,19(12), c.197.
 16. Akella A, Akella S. Machine learning algorithms for predicting coronary artery disease: efforts toward an open source solution. *Future Sci OA*. 7(6):FSO698. March 2021.
 17. Fan, J., Chen, M., Luo, J. et al. The prediction of asymptomatic carotid atherosclerosis with electronic health records: a comparative study of six machine learning models. *BMC Med Inform Decis Mak* 21, c. 115 (2021).
 18. Kartik Budholiya, Shailendra Kumar Shrivastava, Vivek Sharma, An optimized XGBoost based diagnostic system for effective prediction of heart

- disease, Journal of King Saud University - Computer and Information Sciences, 2020.
19. Syed Nawaz Pasha, Dadi Ramesh, Sallauddin Mohmmad, A. Harshavardhan and Shabana / Cardiovascular disease prediction using deep learning techniques, Режим доступа: <https://iopscience.iop.org/article/10.1088/1757-899X/981/2/022006>
 20. Journal Article, Beatriz López-Melgar, Leticia Fernández-Friera, Belén Oliva, José Manuel García-Ruiz, Fátima Sánchez-Cabo, Héctor Bueno, José María Mendiguren, Enrique Lara-Pezzi, Vicente Andrés, Borja Ibáñez, Antonio Fernández-Ortiz, Javier Sanz, Valentín Fuster Short-Term Progression of Multiterritorial Subclinical Atherosclerosis, 2020, Journal of the American College of Cardiology, c. 1617-1627
 21. T. Tassew, X. Nie, A Comprehensive Review of the Application of Machine Learning in Medicine and Health Care, (2022). DOI: 10.36227/techrxiv.21204779.v1
 22. Dinesh, K.G., Arumugaraj, K., Santhosh, K., & Mareeswari, V. (2018). Prediction of Cardiovascular Disease Using Machine Learning Algorithms. 2018 International Conference on Current Trends towards Converging Technologies (ICCTCT), 1-7.
 23. A. Mujumdar and V. Vaidehi, "Diabetes prediction using machine learning algorithms," Procedia Computer Science, vol.165, pp. 292-299, 2019
 24. M. Kim, J. Yun, Y. Cho, K. Shin, R. Jang, H. Bae and N. Kim, "Deep Learning in Medical Imaging," Neurospine, vol.16, no. 4, p. 657, 2019
 25. Y. Xiao, J. Wu, Z. Lin and X. Zhao, "A deep learning-based multi-model ensemble method for cancer prediction" Computer methods and programs in biomedicine, vol. 153, pp.1-9, 2018

26. Rahman Z, Aamir M, Pu YF, Ullah F, Dai Q. A smart system for low-light image enhancement with color constancy and detail manipulation in complex light environments. *Symmetry*. 2018 Dec 5
27. Aamir M, Rehman Z, Pu YF, Ahmed A, Abro WA. Image enhancement in varying light conditions based on wavelet transform. 2019 16th International Computer Conference on Wavelet Active Media Technology and Information Processing 2019 Dec 14 (pp. 317-322). IEEE.
28. X10 What is Pandas? веб-сайт. URL: <https://www.nvidia.com/en-us/glossary/data-science/pandas-python/> (дата звернення: 12.05.2023).
29. What is Python? Executive Summary? веб-сайт. URL: <https://www.python.org/doc/essays/blurb/> (дата звернення: 12.05.2023).
30. How to Save and Load Machine Learning Models in Python Using Joblib Library? веб-сайт. URL: <https://www.analyticsvidhya.com/blog/2023/02/how-to-save-and-load-machine-learning-models-in-python-using-joblib-library/#:~:text=Joblib%20is%20a%20Python%20library,results%20of%20computationally%20expensive%20functions.> (дата звернення: 10.05.2023).
31. Python Tutorial: Streamlit веб-сайт. URL: <https://www.datacamp.com/tutorial/streamlit> (дата звернення: 12.05.2023).
32. K. Butchi Raju, Suresh Dara, Ankit Vidyarthi, V. MNSSVKR Gupta, Baseem Khan, "Smart Heart Disease Prediction System with IoT and Fog Computing Sectors Enabled by Cascaded Deep Learning Model", *Computational Intelligence and Neuroscience*, vol. 2022, Article ID 1070697, 22 pages, 2022. DOI: 10.1155/2022/1070697
33. P. Rani, R. Kumar, N. M. O. S. Ahmed, and A. Jain, "A decision support system for heart disease prediction based upon machine learning," *Journal of Reliable Intelligent Environments*, vol. 7, no. 3, pp. 263-275, 2021, doi: 10.1007/s40860-021-00133-6

34. C. Boukhatem, H. Y. Youssef and A. B. Nassif, "Heart Disease Prediction Using Machine Learning," 2022 Advances in Science and Engineering Technology International Conferences (ASET), Dubai, United Arab Emirates, 2022, pp. 1-6, doi: 10.1109/ASET53988.2022.9734880.
35. R. Katarya and P. Srinivas, "Predicting Heart Disease at Early Stages using Machine Learning: A Survey," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2020, pp. 302-305, doi: 10.1109/ICESC48915.2020.9155586.
36. S. Verma and A. Gupta, "Effective Prediction of Heart Disease Using Data Mining and Machine Learning: A Review," 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), Coimbatore, India, 2021, pp. 249-253, doi: 10.1109/ICAIS50930.2021.9395963.
37. H. Allahabadi et al., "Assessing Trustworthy AI in Times of COVID-19: Deep Learning for Predicting a Multiregional Score Conveying the Degree of Lung Compromise in COVID-19 Patients," in IEEE Transactions on Technology and Society, vol. 3, no. 4, pp. 272-289, Dec. 2022, doi: 10.1109/TTS.2022.3195114.
38. H. Mai, T. T. Pham, D. N. Nguyen and E. Dutkiewicz, "Non-Laboratory-Based Risk Factors for Automated Heart Disease Detection," 2018 12th International Symposium on Medical Information and Communication Technology (ISMICT), Sydney, NSW, Australia, 2018, pp. 1-6, doi: 10.1109/ISMICT.2018.8573706.
39. R. Hajjo, "The Ethical Challenges of Applying Machine Learning and Artificial Intelligence in Cancer Care," 2018 1st International Conference on Cancer Care Informatics (CCI), Amman, Jordan, 2018, pp. 231-231, doi: 10.1109/CANCERCARE.2018.8618186.
40. J. Alam, S. Alam and A. Hossan, "Multi-Stage Lung Cancer Detection and Prediction Using Multi-class SVM Classifie," 2018 International Conference

- on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2), Rajshahi, Bangladesh, 2018, pp. 1-4, doi: 10.1109/IC4ME2.2018.8465593. B. Anishfathima, R. Vikram, S. R. T, M. Sri Vishnu and C. Venumadhav, "A Comparative Analysis on Classification Models to predict Cardio-vascular disease using Machine Learning Algorithms," 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2022, pp. 259-264, doi: 10.1109/ICAIS53314.2022.9741831.
41. B. Anishfathima, R. Vikram, S. R. T, M. Sri Vishnu and C. Venumadhav, "A Comparative Analysis on Classification Models to predict Cardio-vascular disease using Machine Learning Algorithms," 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2022, pp. 259-264, doi: 10.1109/ICAIS53314.2022.9741831.
 42. A. Aldallal and A. A. A. Al-Moosa, "Using Data Mining Techniques to Predict Diabetes and Heart Diseases," 2018 4th International Conference on Frontiers of Signal Processing (ICFSP), Poitiers, France, 2018, pp. 150-154, doi: 10.1109/ICFSP.2018.8552051.
 43. M. Raihan et al., "Smartphone based ischemic heart disease (heart attack) risk prediction using clinical data and data mining approaches, a prototype design," 2016 19th International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 2016, pp. 299-303, doi: 10.1109/ICCITECHN.2016.7860213.
 44. H. Yuliang, Q. Junxuan, W. Haibin and W. Xiubo, "Quantized analysis for heart valve disease based on cardiac sound characteristic waveform method," 2010 2nd International Conference on Signal Processing Systems, Dalian, China, 2010, pp. V1-258-V1-262, doi: 10.1109/ICSPS.2010.5555596.
 45. M. H. Abu Yazid, M. Haikal Satria, S. Talib and N. Azman, "Artificial Neural Network Parameter Tuning Framework For Heart Disease Classification," 2018

5th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), Malang, Indonesia, 2018, pp. 674-679, doi: 10.1109/EECSI.2018.8752821.

46. C. Sowmiya and P. Sumitra, "Analytical study of heart disease diagnosis using classification techniques," 2017 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS), Srivilliputtur, India, 2017, pp. 1-5, doi: 10.1109/ITCOSP.2017.8303115.

ДОДАТКИ

ДОДАТОК А

Програмний код візуалізації даних

```
from warnings import simplefilter

import numpy as np
import pandas as pd
from sklearn import metrics
from sklearn.metrics import roc_curve

pd.set_option('display.width', 330)

np.set_printoptions(linewidth=330)

# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)
# import warnings filter
from warnings import simplefilter

# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)
res = []
df = pd.read_csv(r'data_syn.csv')
print("\n\n\n\n\n\n\n")
print(df)
print(df[df.Progress == 1].age.mean())
print(df[df.Progress == 0].age.mean())
print("\n\n\n\n\n\n\n")
# Progres,OP,Shunt,vik,zrist,vaha,IMT,stat,ChSS,AD sist.,AD diast,AG
therapia,cholesterin,diabetus melitus
### 1 = male, 0 = female
print(df.isnull().sum()) # -- no missing values

df['sex'] = df.sex.map({0: 'male', 1: 'female'})

import matplotlib.pyplot as plt

import seaborn as sns

# distribution of target vs age
```

```

sns.set_context(font_scale=2, rc={"font.size": 45, "axes.titlesize": 55,
"axes.labelsize": 45})

s = sns.catplot(height=13, aspect=2.1, kind='count', data=df, x='age', hue='Progress',
                order=df['age'].sort_values().unique(), legend=True)
import matplotlib.ticker as ticker

s.axes.flat[0].xaxis.set_major_locator(ticker.MultipleLocator(5))
s.axes.flat[0].xaxis.set_major_formatter(ticker.ScalarFormatter())
s.legend.set_title('Atherosclerosis')
plt.title('Variation of Age for each target class')
plt.show()

# barplot of age vs sex with hue = target
g = sns.catplot(height=13, aspect=1.2, kind='bar', data=df, y='age', x='sex',
                hue='Progress', legend_out=False)
g.legend.set_title('Atherosclerosis')
plt.title('Distribution of age vs sex with the target class')
plt.ylabel('mean(age)')

plt.show()

df['sex'] = df['sex'].map({'female': 1, 'male': 0})
corr = df.corr()
mask = np.zeros_like(corr, dtype=bool)
mask[np.triu_indices_from(mask)] = True

cmap = sns.diverging_palette(250, 10, as_cmap=True)
f, ax = plt.subplots(figsize=(11, 9))
# Draw correlation plot with or without duplicates
sns.heatmap(corr, cmap=cmap, vmin=-1, vmax=1,
            square=True,
            linewidth=.5, cbar_kws={"shrink": .5}, ax=ax)
plt.show()

fig = plt.figure(figsize=(20, 16))

ax = fig.add_subplot(111, projection='3d')

h = df[df.Progress == 1]
i = df[df.Progress == 0]

```

```
print(h)

ax.scatter(h.weight, h.cholesterin, h.age, marker="o", c="red", label='Sick', s=100)
ax.scatter(i.weight, i.cholesterin, i.age, marker="o", c="green", label='Healthy',
s=100)

ax.set_title("Weight, Age and Cholesterin distribution by target classes", fontsize=40)

ax.set_xlabel(
    "Weight",
    labelpad=25, fontsize=18)
ax.set_ylabel("Cholesterin", labelpad=30, fontsize=18)
ax.set_zlabel(
    "Age",
    labelpad=10, fontsize=18)

ax.legend(prop={'size': 30})

plt.show()
```


ДОДАТОК Б

Програмний код створення, тренування та збереження моделей

```
import pickle

import pandas as pd
from keras import layers, Sequential
from keras.regularizers import l2
from keras.utils import plot_model
from matplotlib import pyplot as plt
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import Perceptron
from sklearn.metrics import roc_curve, confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier

pd.set_option('display.max_columns', None)
df = pd.concat([pd.read_csv("ateros.csv"), pd.read_csv("data_syn1.csv"),
pd.read_csv("data_syn2.csv")])
print(df)
# label = LabelEncoder()
# for col in df.columns.values.tolist():
#     if pd.api.types.is_string_dtype(df[col].dtype):
#         df[col] = label.fit_transform(df[col])
#         le_name_mapping = dict(zip(label.classes_, label.transform(label.classes_)))
#         print(le_name_mapping)
print(df)
# df = df.sort_values(by=["HeartDisease"])
y = df["Progress"]
X = df.drop("Progress", axis=1)

# smote = SMOTE(random_state=42, k_neighbors=2)
# nearmiss = NearMiss(n_neighbors=2)

# X, y = nearmiss.fit_resample(X, y)
```

```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
print(y_train)
print(y_test)
# from sklearn.preprocessing import StandardScaler
# sc=StandardScaler()
# X_train=sc.fit_transform(X_train)
# X_test=sc.transform(X_test)

# print('Imbalanced: {}'.format(Counter(y_train)))
# print('Balanced: {}'.format(Counter(y_train_smote)))

print(X_train)
print(y_train)

# mean = X_train.mean(axis=0)
# std = X_train.std(axis=0)
# X_train = (X_train - mean) / std

# Define the model architecture
model = Sequential()
model.add(layers.Dense(64, input_shape=(len(X.columns.values),), activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(32, activation='relu', kernel_regularizer=l2(0.01)))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Fit the model
history = model.fit(X_train, y_train, epochs=200, validation_split=0.2)
model.save("my_model.h5")
# workers=10, use_multiprocessing=True)
print(model.summary())

y_pred = [round(float(x)) for x in model.predict(X_test)]
y_pred_train = [round(float(x)) for x in model.predict(X_train)]
fprnn, tprnn, threshnn = roc_curve(y_test, y_pred, pos_label=1)

cm_test = confusion_matrix(y_pred, y_test)
cm_train = confusion_matrix(y_pred_train, y_train)

```

```

plot_model(model, to_file='model_plot.png', show_shapes=True,
show_layer_names=True)

print(cm_test)
disp = ConfusionMatrixDisplay(confusion_matrix=cm_test)
disp.plot()
plt.show()
print(cm_train)
print('Accuracy for training set for Neural network = {}'.format((cm_train[0][0] +
cm_train[1][1]) / len(y_train)))
print('Accuracy for test set for Neural network = {}'.format((cm_test[0][0] +
cm_test[1][1]) / len(y_test)))
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

#

def test(model):
    model.fit(X_train, y_train)

    # Predicting the Test set results
    y_pred = model.predict(X_test)
    cm_test = confusion_matrix(y_pred, y_test)

    y_pred_train = model.predict(X_train)
    cm_train = confusion_matrix(y_pred_train, y_train)

    fpr2, tpr2, thresh2 = roc_curve(y_test, y_pred, pos_label=1)

```

```

print(cm_test)
disp = ConfusionMatrixDisplay(confusion_matrix=cm_test)
disp.plot()
disp.ax_.set(xlabel=type(model).__name__ + ' Confusion Matrix')
plt.show()
print('Accuracy for training set for model { } = {}'.format(type(model).__name__,
                                                             (cm_train[0][0] + cm_train[1][1]) /
len(y_train)))
print('Accuracy for test set for model { } = {}'.format(type(model).__name__,
                                                         (cm_test[0][0] + cm_test[1][1]) / len(y_test)))
with open(type(model).__name__ + '.pkl', 'wb') as files:
    pickle.dump(model, files)
return fpr2, tpr2

```

```

fpr1, tpr1 = test(svm.SVC(probability=True))
fpr2, tpr2 = test(GaussianNB())
fpr3, tpr3 = test(DecisionTreeClassifier())
fpr4, tpr4 = test(RandomForestClassifier(n_estimators=20))
fpr5, tpr5 = test(XGBClassifier(use_label_encoder=False))

```

```

plt.style.use('seaborn')

```

```

random_probs = [0 for i in range(len(y_test))]
p_fpr, p_tpr, _ = roc_curve(y_test, random_probs, pos_label=1)

```

```

# plot roc curves

```

```

plt.plot(fpr1, tpr1, linestyle='--', color='orange', label='SVC')
plt.plot(fpr2, tpr2, linestyle='--', color='green', label='Naive Bayes')
plt.plot(fpr3, tpr3, linestyle='--', color='yellow', label='Decision Tree')
plt.plot(fpr4, tpr4, linestyle='--', color='black', label='Random Forest')
plt.plot(fpr5, tpr5, linestyle='--', color='purple', label='XGB')
plt.plot(fprnn, tprnn, linestyle='--', color='pink', label='Neural Network')
plt.plot(p_fpr, p_tpr, linestyle='--', color='blue')

```

```

# title

```

```

plt.title('ROC curve')

```

```

# x label

```

```

plt.xlabel('False Positive Rate')

```

```

# y label

```

```

plt.ylabel('True Positive rate')

```

```
plt.legend(loc='best')  
plt.savefig('ROC', dpi=300)  
plt.show()
```

```
with open(type(model).__name__ + '.pkl', 'wb') as files:  
    pickle.dump(model, files)
```

ДОДАТОК В

Програмний код застосунку

```
import os

import joblib
import pandas as pd
import streamlit as st
from keras.saving.saving_api import load_model

pd.set_option('display.max_columns', None)
st.title("Прогнозування серцево-судинних хвороб")

models = {}
for file in os.listdir("."):
    if file.endswith(".pkl"):
        models[file] = joblib.load(open(file, "rb"))

models = {**{"Нейронна мережа": load_model("my_model.h5")}, **models}

YES_NO = ['Hi', 'Так']
CHOLEST = ["Норма - завжди в межах норми",
           "Вище в межах норми - від 4,7 ммоль/л до 5",
           "Іноді вище норми",
           "Завжди вище норми"]
SEX = ["Чоловіча", "Жіноча"]

sex = st.selectbox("Стать", options=SEX)
age = st.number_input("Вік", 0, 90, 30)
height = st.slider("Зріст (см)", value=180.0, min_value=100.0, max_value=220.0,
step=1.0, format="%.1f")
weight = st.slider('Вага (кг)', value=75.0, min_value=0.0, max_value=150.0,
step=1.0, format="%.1f")
ChSS = st.number_input("Пульс (в стані спокою)", 0, 120, 70)
ADsist = st.number_input("Систолічний тиск", 90, 160, 120)
ADdiast = st.number_input("Діастолічний тиск", 50, 120, 80)

cholesterin = st.selectbox("Показник холестерину: ",
                           options=CHOLEST)
```

```

diabetus = st.selectbox("Ви хворієте на діабет?", options=YES_NO)

OP = st.selectbox("Чи були оперативні втручання стосовно серцево-судинної системи?", options=YES_NO)

Shunt = st.selectbox("Чи робили Вам шунтування артерій?", options=YES_NO)

AGtherapia = st.selectbox("Чи проходили ви антигіпертензивну терапію?", options=YES_NO)

IMT = round((weight / height ** 2), 2)
data = {
    "OP": [YES_NO.index(OP)],
    "Shunt": [YES_NO.index(Shunt)],
    "age": [age],
    "height": [height],
    "weight": [weight],
    "IMT": [IMT],
    "sex": [SEX.index(sex)],
    "ChSS": [ChSS],
    "AD sist.": [ADsist],
    "AD diast": [ADdiast],
    "AG therapia": [YES_NO.index(AGtherapia)],
    "cholesterin": [CHOLEST.index(cholesterin)],
    "diabetus melitus": [YES_NO.index(diabetus)],
}
input_df = pd.DataFrame(data, index=[0])

st.divider()

col1, col2 = st.columns([1, 1])
with col1:
    submit = st.button("Передбачити")
with col2:
    model = st.selectbox("Оберіть модель: ", options=list(models.keys()))

if submit:
    print(input_df)
    if model == "Нейронна мережа":
        model = models[model]
        prediction = model.predict(input_df)
        prediction = round(prediction[0][0] * 100, 2)

```

```
# prediction_prob = model.predict_proba(input_df)
else:
    model = models[model]
    prediction = model.predict_proba(input_df)
    prediction = round(prediction[0][1] * 100, 2)
    st.markdown(f"Вірогідність наявності серцево-судинної хвороби становить
{prediction}%")
```