ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА
(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем
(повна назва кафедри (предметної, циклової комісії))

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»
(освітньо-кваліфікаційний рівень)

на тему Застосування машинного навчання для рішення медичних задач
Application of machine learning for solving medical problems

Виконав: студент денної форми навчання
спеціальності 123 – Комп'ютерна інженерія
Освітня програма «Комп'ютерна інженерія»
(шифр і назва напряму підготовки, спеціальності)
Осипов Артем Валентинович
(прізвище, ім'я, по-батькові)
Керівник к.ф.-м.н., доц. Шпінарева І. М. Iryna Shpinareva 2023.06.18 18:11:40 +03'00'
(науковий ступінь, вчене звання, прізвище та ініціали, підпис)
Рецензент к.техн.н., доц. Пенко В.Г.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
№__ від «__» червня 2023 р.

Завідувач кафедри
_____ Євгеній МАЛАХОВ
(підпис) (ім'я, прізвище)

Подписано цифровой подписью: Eugene Malakhov
Дата: 2023.06.18 23:05:38 +03'00'

Захищено на засіданні ЕК № ___
протокол № __ від «___» ____ 2023 р.
Оцінка _____/_____/_____
(за національною шкалою, шкалою ECTS, бали)

Голова ЕК
_____ _Алла КОБОЗЄВА
(підпис) (ім'я, прізвище)

Одеса – 2023

АНОТАЦІЯ

Кваліфікаційна робота присвячена застосуванню методів машинного навчання для рішення медичних задач. Метою даної роботи є підвищення точності прогнозування захворювання серця шляхом аналізу алгоритмів машинного навчання.

В сучасних умовах машинне навчання знайшло широке застосування в багатьох галузях, включаючи медицину. Застосування методів машинного навчання може допомогти у покращенні точності діагностики, прогнозуванні розвитку захворювань та підвищенні ефективності лікування.

У роботі проведено аналіз попередніх досліджень, присвячених застосуванню машинного навчання в медицині. Розглянуті різні методи машинного навчання, такі як нейронні мережі та дерева рішень.

Для порівняння різних моделей машинного навчання використані дані з медичних досліджень «Heart Disease», зокрема з областей діагностики захворювань та прогнозування розвитку захворювань. Для кожної моделі проведені тести та порівняння результатів. Розроблен веб-застосунок з інтерфейсом користувача для прогнозування серцевих захворювань

Аналіз методів машинного навчання є дуже важливими для медичної галузі. Застосування даних методів допоможе у покращенні точності діагностики та прогнозування розвитку захворювань.

ABSTRACT

The qualification work is dedicated to the application of machine learning methods for solving medical problems. The aim of this work is to improve the accuracy of predicting heart disease by analyzing machine learning algorithms.

In modern times, machine learning has found wide application in many fields, including medicine. The use of machine learning methods can help improve the accuracy of diagnosis, predict the development of diseases, and increase the effectiveness of treatment.

The work analyzes previous research on the application of machine learning in medicine. Different machine learning methods, such as neural networks and decision trees, are considered.

To compare different machine learning models, data from medical studies on «Heart Disease» were used, particularly in the areas of disease diagnosis and prediction. Tests and comparisons of results were conducted for each model. A web application with a user interface was developed for predicting heart disease.

Analysis of machine learning methods is very important for the medical field. The use of these methods can help improve the accuracy of diagnosis and prediction of disease development.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

CNN (convolutional neural network) – згорткова нейронна мережа.

FCNN (fully Connected Neural Networks) – повнозв'язна нейронна мережа.

GBMs (gradient boosting machine) - градієнтний бустинг.

Random forest classifier – випадковий ліс.

RNN (recurrent neural network) – рекурентна нейронна мережа.

ШІ – штучний інтелект.

ВСТУП

Використання штучного інтелекту в медицині зазнало значного росту і очікує подальшого розширення у майбутньому. Згідно з опитуванням серед провайдерів охорони здоров'я у Сполучених Штатах у квітні 2021 року, 24 відсотки відповідачів повідомили, що в їхній лікарні або системі охорони здоров'я проекти зі штучним інтелектом / машинним навчанням перебувають у стадії тестування і їх впровадження ще має бути вирішено, тоді як ще 22 відсотки заявили, що поки що лише задумуються над використанням подібних технологій [1]. В Європейському Союзі же у 2021 році 42 відсотки організацій охорони здоров'я вже використовували технології штучного інтелекту для діагностики захворювань, ще 19 відсотків мали плани застосувати цю технологію протягом наступних трьох років. Крім того, 33 відсотки організацій охорони здоров'я, опитаних у дослідженні, планували використовувати інструменти штучного інтелекту для моніторингу пацієнтів протягом наступних трьох років [2]. У 2022 році світовий ринок ШІ в галузі охорони здоров'я був оцінений в 15,1 мільярда доларів США, а прогнози передбачають досягнення позначки 187,95 мільярда доларів США до 2030 року. Це вказує на складний річний темп зростання (CAGR) на рівні 35.4% від 2022 до 2030 року і підкреслює величезний потенціал ШІ для революції в медичній практиці [3].

Штучний інтелект знайшов застосування у різних сферах охорони здоров'я, але найпоширенішими є клінічне рішення, медичне зображення та пошук ліків. Клінічні системи підтримки рішень, які працюють на основі ШІ, надають незамінну допомогу лікарям, покращуючи точність діагностики та надаючи рекомендації для оптимальних схем лікування. Медичні системи зображення, підтримані алгоритмами ШІ, дозволяють виявляти хвороби раніше і з більшою точністю при інтерпретації діагностичних зображень. Крім того, машинне навчання сприяє процесам пошуку ліків, допомагаючи дослідникам виявляти нові цілі та проектувати препарати швидше і

ефективніше, що потенційно прискорює розробку життєво важливих медикаментів.

У 2020-2023 роках спостерігаються значні прогреси в інтеграції штучного інтелекту у галузь охорони здоров'я. Наприклад, алгоритми на основі ШІ допомагають лікарям діагностувати рак з більшою точністю, що сприяє поліпшенню результатів лікування пацієнтів. Чат-боти, працюючи на основі ШІ, допомагають пацієнтам керувати хронічними захворюваннями, надаючи персоналізовані поради та підтримку. Використання роботів на основі ШІ в хірургічних процедурах продемонструвало покращену точність та мінімально інвазивні методи, що сприяє швидкому відновленню та зменшенню ризиків для пацієнтів. Крім того, віртуальні помічники на основі ШІ спрощують процес отримання медичних послуг для пацієнтів, допомагаючи в плануванні прийому, поповненні рецептів та отриманні інших важливих медичних відомостей.

Одним з ключових викликів при використанні машинного навчання є вибір найефективнішої моделі для розв'язання конкретної медичної задачі. Кожен алгоритм має свої переваги та обмеження, які необхідно враховувати при виборі моделі для конкретної медичної задачі. Це і є головним напрямком цього проекту.

Метою роботи є підвищення точності прогнозування захворювання серця шляхом аналізу алгоритмів машинного навчання.

Для досягнення поставленої мети необхідно розв’язати такі задачі:

- огляд методів машинного навчання для рішення медичних задач;
- вибір архітектури, технологій і засобів реалізації системи;
- проектування системи для прогнозування серцевих захворювань;
- реалізація системи прогнозування в предметній області;
- тестування системи.

1 ОГЛЯД ІСНУЮЧИХ СИСТЕМ І МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ РІШЕННЯ МЕДИЧНИХ ЗАДАЧ

1.1 Огляд існуючих систем для рішення медичних задач

На даний момент існує величезна кількість продуктів. Розглянемо декілька з них.

1) MedicSen – це медична компанія, яка розвиває систему для моніторингу глюкози для людей з діабетом [4]. Їх система аналізує потік інформації, який поступає з окремого пристрою, і надає користувачу персоналізовані рекомендації щодо стану здоров'я.

Головна перевага системи – зручність моніторингу в порівнянні з альтернативою, яка потребує зразка крові.

Недоліком – система знаходиться у розробці і на даний момент не дуже багато публічних результатів щодо точності вимірювання.

2) Linguamatics – це компанія, яка використовує обробку природньої мови та машинне навчання для виокремлення та структуризації цінної інформації з медичних документів з подальшою обробкою отриманих даних для допомоги медичним організаціям при лікуванні пацієнтів [5].

Окрім очевидних переваг такої системи, таких як економія часу та потенційне покращення якості аналізу інформації, можна виділити можливість налаштування її до особливостей окремих організацій.

3) DeepMind Health – це медична компанія, яка зосереджена на використанні машинного навчання для покращення результатів при лікуванні [6]. Розроблені компанією програми використовуються у лікарнях і медичних організаціях по всьому світу для прогнозування погіршення стану пацієнтів, покращення клінічних робочих процесів і зменшення діагностичних помилок. Серед недоліків можна виділити проблему конфіденційності. DeepMind Health не є незалежної компанією і це викликає занепокоєння щодо того, що її материнська компанія Alphabet Inc. може неправомірно використати дані пацієнтів.

1.2 Обмеження і виклики при використовувані машинного навчання у медицині

У галузі охорони здоров'я виникають кілька значних викликів у процесі впровадження штучного інтелекту. Ці виклики утруднюють безпроблемне поєднання технології ШІ з практикою охорони здоров'я, що заважає реалізації її потенційних переваг. У цьому розділі розглянуті три основні виклики, з якими сьогодні стикаються в процесі впровадження машинного навчання в охорону здоров'я: нестача даних, відсутність стандартів та проблеми приватності та безпеки.

По-перше, одним з найбільш серйозних викликів є недостатня кількість даних для тренування алгоритмів ШІ. Системи штучного інтелекту потребують великого обсягу даних для ефективного навчання та точних прогнозів або рішень. Однак багато організацій охорони здоров'я мають проблеми з отриманням необхідної кількості та якості даних для тренування. Недостатність даних може бути пов'язана з фрагментацією даних, обмеженим доступом до повних медичних записів та відсутністю єдиної протоколу збору даних. В результаті, провайдери охорони здоров'я часто зіштовхуються з труднощами в розкритті повного потенціалу технології ШІ.

По-друге, відсутність стандартизованих рамок та настанов для розробки та використання машинного навчання в охороні здоров'я становить серйозну перешкоду. Без чітких стандартів організації охорони здоров'я стикаються з труднощами в упровадженні єдиної системи ШІ в різних установах. Відсутність стандартів ускладнює взаємодію, робить складним обмін моделями та алгоритмами ШІ між різними медичними установами. Крім того, відсутність стандартизованих процесів оцінки та валідації ставить під сумнів надійність та ефективність штучного інтелекту в охороні здоров'я, ускладнюючи їх широке впровадження.

Крім того, проблеми щодо приватності та безпеки становлять ще одну вагому перешкоду для впровадження ШІ в охорону здоров'я. Алгоритми

штучного інтелекту широко використовують дані пацієнтів, включаючи конфіденційні медичні записи, для тренування та покращення своєї продуктивності. Проте використання таких даних викликає обурення щодо порушень приватності, несанкціонованого доступу та недопущення зловживання даними. Пацієнти та фахівці охорони здоров'я виражають стурбованість щодо потенційних ризиків, пов'язаних зі зберіганням, передачею та аналізом чутливої медичної інформації. Ці стурбованості часто викликають коливання в упровадженні ШІ, оскільки забезпечення приватності пацієнтів та безпеки даних залишається головним пріоритетом.

Загалом, впровадження штучного інтелекту в охорону здоров'я супроводжується кількома викликами, які потребують вирішення для повного використання його потенціалу. Недостатність даних для тренування алгоритмів ШІ, відсутність стандартів та проблеми щодо приватності та безпеки даних пацієнтів – це основні перешкоди. Однак, на шляху до подолання цих викликів з'являються нові можливості та ініціативи.

Для успішного подолання викликів, необхідно спільними зусиллями організацій охорони здоров'я, законодавців та розробників технологій створити надійні механізми обміну даними, розробити стандартизовані настанови та впровадити строгі протоколи приватності та безпеки. При цьому, важливо забезпечити ефективну співпрацю між різними сторонами та обмін знаннями і досвідом.

Тільки завдяки спільним зусиллям можна реалізувати потенційні переваги ШІ в охороні здоров'я. Впровадження штучного інтелекту може призвести до поліпшення догляду за пацієнтами та результатів лікування. Наприклад, алгоритми ШІ можуть допомогти в розпізнаванні патологічних змін на рентгенограмах, виявленні ранніх ознак захворювань та підтримці прийняття рішень лікарів.

Таким чином, впровадження штучного інтелекту в охорону здоров'я має великий потенціал, але вимагає спільних зусиль та забезпечення відповідних механізмів і протоколів для ефективного використання його можливостей.

1.3 Огляд методів машинного навчання для рішення медичних задач

FCNN – повністю зв'язані нейронні мережі. Це найпростіший тип нейронних мереж, який складається з взаємопов'язаних шарів нейронів, що виконують прості обчислення на вхідних даних. Це робить FCNN дуже гнучкою, оскільки вона може використовуватися для вирішення різноманітних проблем.

Є кілька переваг використання FCNN моделей, зокрема:

– легко тренувати: є відносно легкими для тренування порівняно з іншими нейронними мережами, такими як RNN;

– гнучкість: можуть використовуватись для широкого спектру задач, включаючи класифікацію зображень, обробку природньої мови та багато іншого;

– ефективність: можуть бути реалізовані ефективно за допомогою графічних процесорів, що робить їх добре підходящими для великомасштабного тренування.

З недоліків можна виділити наступні:

– потребують багато даних для тренування, що може бути викликом для деяких задач;

– не так ефективні в обробці послідовних даних, як RNN;

– можуть бути обчислювально дорогими.

FCNN використовуються в різноманітних застосування, включаючи:

– класифікація зображень за категоріями;

– обробка природної мови;

– розпізнавання мови.

CNN – це тип нейронних мереж, який спеціально розроблений для обробки даних зі структурою сітки, таких як зображення або звук. CNN складається зі шарів взаємопов'язаних вузлів, кожен з яких виконує операцію

згортки на вхідних даних. Таким чином CNN навчаються виявляти патерни в даних. Перший шар CNN, як правило, виконує операцію згортки на вхідних даних. Ця операція допомагає виявити маленькі патерни в даних, такі як краї або кути. Вихід з шару згортки передається до наступного шару, де також виконується операція згортки. Цей процес продовжується до останнього шару CNN, який виконує операцію класифікації або регресії.

Переваги використання CNN над іншими моделями:

– розрідженість: є розрідженими моделями, що означає, що між нейронами є багато зв'язків, але лише декілька нейронів активні в будь-який момент часу. Це робить їх ефективнішими, ніж FCNN, у яких є багато зв'язків між всіма нейронами;

– локальність: є локальними моделями, що означає, що вони розглядають лише невелике вікно даних за раз. Завдяки чому вони добре підходять для задач, які вимагають обробки даних з просторовою структурою, таких як зображення;

– трансляційна-інваріантність: є інваріантними до трансляції, що означає, що вони можуть навчатися патернам в даних, які не залежать від положення даних у введенні, що допомагає при обробці несинхронізованих даних, таких як обробка природної мови.

Основні недоліки CNN моделей:

– не дуже ефективні при обробці послідовних даних;

– можуть бути складними для тренування та реалізації.

CNN використовуються у широкому спектрі задач, але найчастіше у наступних випадках: класифікація зображень; виявлення об'єктів; обробка природної мови; розпізнавання мови.

RNN добре підходять для задач обробки послідовних даних, таких як обробка природної мови та розпізнавання мови. Вони складаються з взаємопов'язаних шарів нейронів, які виконують рекурентні операції на

вхідних даних. Рекурентні операції добре підходять для обробки даних, що мають темпоральну структуру, наприклад, послідовності слів або звуків.

Переваги RNN над іншими моделями:

– довгострокові залежності: можуть навчатися довгострокових залежностей в даних, що робить їх добре підходящими для задач, які вимагають розуміння контексту послідовності;

– моделювання часових даних: добре підходять для задач, які вимагають моделювання часових даних, таких як розпізнавання мови та обробка природної мови;

– ефективність: можуть бути реалізовані ефективно за допомогою графічних процесорів, що робить їх добре підходящими для великомасштабного тренування.

Недоліки RNN моделей:

– проблеми з градієнтами: схильні до проблем зникаючих та вибухаючих градієнтів, що може ускладнити їх тренування;

– нестійкість: можуть бути нестійкими, що може призводити до перенавчання та інших проблем;

– складність: можуть бути складними для тренування та реалізації, порівняно з іншими нейронними мережами.

Сфери найбільш ефективного застосування RNN:

– обробка природної мови;

– розпізнавання мови;

– створення музики;

– симуляція ігрових агентів.

Random forest classifier – алгоритм навчання з учителем, який використовується для класифікації та регресії. Він працює за допомогою створення кількох дерев рішень, а потім комбінує їх прогнози для прийняття остаточного рішення. Це допомагає зменшити перенавчання та покращити точність моделі.

Випадковий ліс складається з колекції дерев рішень, кожне з яких навчається на різних підмножинах даних, а потім їх прогнози комбінуються для прийняття остаточного прогнозу.

Переваги використання Random Forest Classifier:

– точність: є дуже точним алгоритмом порівняно з іншими алгоритмами машинного навчання;

– робастність: є робастним алгоритмом, що означає що він є менш чутливим до шуму та викидів, ніж інші алгоритми машинного навчання;

– інтерпретовність: є досить простим для інтерпретації алгоритмом, що дозволяє досліджувати прогнози окремих дерев, для розуміння, як модель робить свої прогнози.

Недоліки алгоритму випадкового лісу:

– обчислювальна складність: є обчислювальна витратним алгоритмом, і може бути повільним у тренуванні та передбаченні;

– вимоги до даних: вимагає великої кількості даних для тренування, і може бути непридатним для завдань з невеликими наборами даних;

– перенавчання: схильний до перенавчання, тому важливо налаштувати гіперпараметри моделі, щоб уникнути цього.

Random Forest Classifiers використовують у різноманітних задачах, включаючи:

– класифікація: Random Forest Classifiers часто використовуються для класифікаційних завдань, таких як фільтрація спаму, виявлення шахрайства та медична діагностика;

– регресія: Random Forest Classifiers також можуть використовуватися для регресійних завдань, таких як прогнозування вартості будинків або задоволеності клієнтів;

– Виявлення аномалій: використовується для ідентифікації аномалій в даних, що може бути корисним для виявлення шахрайства або контролю якості.

GBMs – це тип алгоритмів машинного навчання з ансамблевим підходом, який схожий на випадковий ліс. Однак, GBMs використовують інший підхід до комбінування прогнозів окремих дерев. GBMs використовують метод, що називається градієнтним бустингом, який включає ітеративне додавання нових дерев до моделі. Кожне нове дерево додається для корекції помилок, допущених попередніми деревами. Це допомагає зменшити перенавчання та покращити точність моделі.

Переваги:

– точність: GBMs відомі своєю високою точністю та здатністю перевершувати інші алгоритми машинного навчання на деяких завданнях;

– ефективність: можуть бути ефективно реалізовані за допомогою GPU, що робить їх підходящими для масштабного навчання;

– інтерпретованість: є відносно інтерпретованими алгоритмами, і можливо зрозуміти, як модель прийшла до певного прогнозу.

Недоліки:

– обчислювальна складність: навчання GBMs може бути обчислювально витратним, і вони можуть не підходити для завдань з невеликими наборами даних;

– перенавчання: GBMs можуть схильні до перенавчання, тому важливо налаштувати гіперпараметри моделі, для уникнення цього.

GBMs можуть бути застосовані в різних областях, де важлива точність прогнозування і обробка складних взаємозв'язків у даних, наприклад:

– кредитний скоринг і ризик-аналітика: можуть бути використані для оцінки кредитної придатності клієнтів і визначення ризику неповернення кредитів;

– маркетинговий аналіз: можуть бути використані для прогнозування поведінки споживачів, включаючи прогнозування споживчих звичок, передбачення відтоку клієнтів та персоналізацію рекомендацій;

– медична діагностика: можуть бути застосовані для побудови моделей,

які допомагають у виявленні хвороб, класифікації зображень (наприклад, раку на рентгенограмах) і прогнозуванні ризиків;

– прогнозування часових рядів: можуть бути використані для прогнозування цін на нерухомість, акції, валютні курси та інші фінансові показники або погоди;

– обробка природної мови: можуть бути застосовані для різних завдань у обробці тексту, таких як класифікація документів, визначення тональності текстів, автоматичне підсумовування тощо.

1.4 Порівняльні метрики моделей

Одним з важливих аспектів досліджень у сфері нейронних мереж є оцінювання ефективності моделей. Це необхідно для здійснення порівняльного аналізу різних архітектур, оптимізаційних методів та гіперпараметрів. Оцінювання моделей включає в себе використання різних порівняльних метрик, які допомагають кількісно оцінити їхню продуктивність і порівняти результати різних експериментів.

Однією з основних порівняльних метрик є точність (accuracy), яка вимірює відповідність передбачуваних значень моделі до правильних міток у вихідному наборі даних. Формула точності представлена нижче (1.1).

$$Accuracy = \left(\frac{\text{Кількість правильних передбачень}}{\text{Загальна кількість передбачень}}\right) * 100\% \qquad (1.1)$$

Іншою важливою метрикою є втрати (loss), які вимірюють рівень помилок моделі під час тренування. Загальна ідея полягає в тому, що модель намагається мінімізувати значення функції втрат, щоб зменшити помилки передбачень. Серед відомих функцій втрат є бінарна крос-ентропія, яка використовується в задачах бінарної класифікації. Формула для бінарної крос-ентропії має вигляд (1.2).

$$BinaryCrossEntropy = -[y * log(p) + (1 - y) * log(1 - p)], \quad (1.2)$$

де y – цільова мітка;

p – значення передбачення моделі.

Ще одна з популярних функцій втрат – середньоквадратична похибка (MSE). Формула MSE виглядає наступним чином (1.3).

$$\text{MSE} = \left(\frac{1}{n}\right)\Sigma|\text{i} = 1 \text{ to n}|(y_i - \hat{y}_i)^2, \quad (1.3)$$

де y_i – справжнє значення;

$\hat{y}_i$ – передбачене значення;

n – кількість прикладів.

Для деяких задач класифікації можуть бути корисні метрики, такі як відгук (recall), точність передбачення (precision) і F-міра (F-measure).

Відгук вимірює, яку частку правильно класифікованих позитивних екземплярів модель здатна відшукати. Відгук має наступну формулу (1.4).

$$\text{Recall} = \frac{\text{TP}}{TP + FN} \quad (1.4)$$

де TP – кількість правильно передбачених позитивних екземплярів;

FN – кількість неправильно передбачених негативних екземплярів.

Точність передбачення вимірює, яку частку позитивних екземплярів, виявлених моделлю, насправді є позитивними.

Точність обчислюється за наступною формулою (1.5).

$$\text{Precision} = \frac{\text{TP}}{TP + FP} \quad (1.5)$$

де TP – кількість правильно передбачених позитивних екземплярів;

FP – кількість неправильно передбачених позитивних екземплярів.

F-міра об'єднує в собі точність і відгук, щоб отримати одну метрику, яка враховує обидві характеристики. Формула F-міри представлена нижче (1.6).

$$F = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \quad (1.6)$$

Для задач регресії широко використовуються такі метрики, як середня абсолютна похибка (MAE) та середньоквадратична похибка (MSE). MAE вимірює середню абсолютну різницю між передбаченими і справжніми значеннями, тоді як MSE вимірює середню квадратичну різницю між ними. Головна мета полягає в мінімізації обох цих метрик, оскільки це свідчить про більш точні прогнози моделі.

Для оцінки моделей у бінарних задачах класифікації також можна використовувати криві ROC (Receiver Operating Characteristic) та AUC (Area Under the Curve).

Крива ROC відображає залежність між відгуком та долею хибнопозитивних відповідей. AUC вимірює площу під кривою ROC, і чим ближче значення AUC до 1, тим кращою є модель. Ці метрики допомагають оцінити ефективність моделі у відношенні до розпізнавання класів у бінарному контексті.

1.5 Висновки та уточнення задач

На основі розглянутих вище алгоритмів машинного навчання для порівняння обрані наступні моделі [7]:

– повнозв’язна нейронна мережа;

– згорткова нейронна мереж;

– рекурентна нейронна мережа;

– випадковий ліс.

Система для роботи з алгоритмами машинного навчання повинна задовольняти наступним вимогами:

– завантажувати файли для обробки і надавати зручний інструментарій для роботи між даними і алгоритмами;

– приймати будь-які моделі машинного навчання для використання;

– видавати статистику проведеної алгоритмом роботи.

Для реалізації даної роботи потрібно розглянути наступні задачі:

– реалізація обробки даних датасету;

– реалізація повнозв’язної, згорткової і рекурентної нейронних мереж та алгоритму випадкового лісу;

– навчання алгоритмів машинного навчання;

– тестування моделей машинного навчання на тестовій вибірці;

– реалізація інтерфейсу.

Для порівняння ефективності цих алгоритмів використовується один і той же набір даних та обрані наступні метрики: accuracy, loss, recall, precision та f-measure. Результати порівняння допоможуть визначити найефективніший алгоритм для даної задачі та встановити кращі параметри моделі.

2 ПРОЕКТУВАННЯ СИСТЕМИ ДЛЯ ПРОГНОЗУВАННЯ СЕРЦЕВИХ ЗАХВОРЮВАНЬ

2.1 Проектування системи

Система для прогнозування серцевих захворювань складається з трьох етапів, які взаємодіють між собою через інтерфейс, а саме етапу передачі даних, етапу прогнозування і етапу порівняння. Інтерфейс дозволяє користувачеві вводити дані для прогнозування, які при передачі до алгоритмів проходять обробку, та отримувати результати за допомогою вже навчених моделей для їх порівняння. Навчання складається з трьох етапів – етапу обробки даних, етапу розбивки даних та етапу навчання алгоритмів (рис. 2.1).

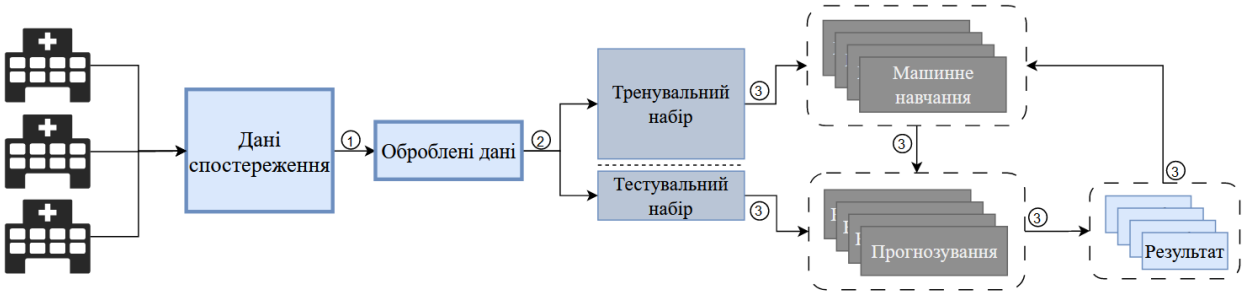

Рисунок 2.1 – Послідовність роботи системи

На етапі обробки, отримані дані нормалізуються для приведення їх до ефективного для використання комп’ютером значень. На етапі розбивки, дані розділяються на два набори: для тренування і для тестування, після чого вони використовуються на наступному етапі навчання моделей.

2.2 Розглянуті алгоритми

Модель FCNN використовує повнозв'язні шари для здатності моделі розрізняти складні шаблони та взаємозв'язки вхідних даних. Основна

структура моделі включає кілька повнозв'язних шарів (Dense) та шарів випадкового відкинення (Dropout) (рис. 2.2).

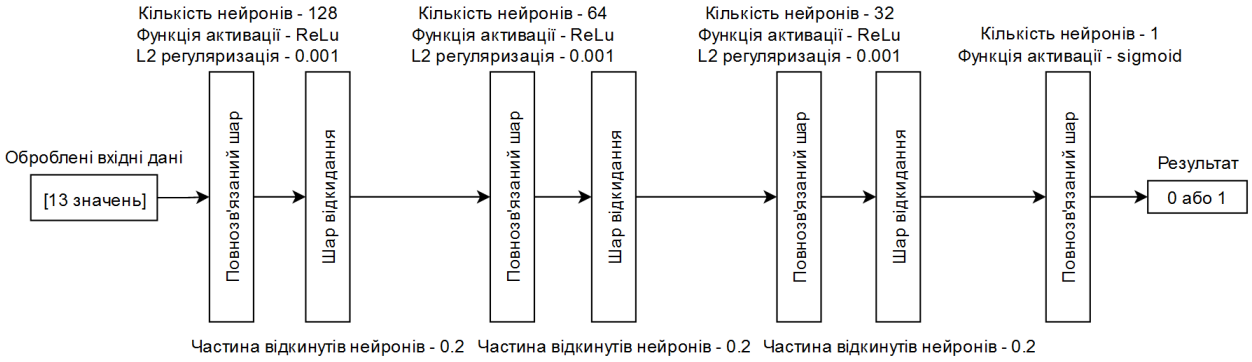

Рисунок 2.2 – Архітектура FCNN

Перший шар є повнозв'язним з 128 нейронами, що дозволить моделі вчитися виокремлювати значущі ознаки з вхідних даних та створювати внутрішнє представлення. Використання активаційної функції relu у цьому шарі допомагає уникнути проблеми з виведенням ваг до нуля, що сприяє швидшому та стабільнішому навчанню. Вона встановлює вихідний сигнал рівним нулю для вхідних значень менших за нуль, тобто активує нейрони тільки при позитивних значеннях вхіду. Ця функція є нелінійною та допомагає моделі навчитися нелінійним залежностям між ознаками.

Далі для запобігання перенавчання моделі буде використані шари випадкового відкидання (Dropout). Випадкове відкидання 20% нейронів після кожного повнозв’язного шару дозволить уникнути того, щоб окремі нейрони або групи нейронів ставали домінуючими та надмірно пристосовувалися до тренувальних даних. Це забезпечить більшу загальну генералізацію та стійкість моделі до нових вхідних даних.

Наступний шар є знову повнозв'язним шаром, але з 64 нейронами та той самою активаційною функцією relu. Після шару відкидання знову йде повнозв’язний шар з 32 нейронами.

Останній повнозв'язний шар з одним нейроном та активаційною функцією sigmoid використовується для вирішення конкретної задачі передбачення класу. Функція активації sigmoid призначена для вирішення задач бінарної класифікації, де модель намагається передбачити ймовірність належності до одного з двох класів.

Модель також використовує параметри регуляризації L2 з коефіцієнтом 0.001 на всіх повнов'язних шарах окрім останнього, які контролюють перенавчання шляхом додавання штрафу до функції втрати, залежного від ваг моделі. Це допомагає зменшити ваги моделі та запобігти перенавчанню, забезпечуючи її загальну здатність до узагальнення.

Модель CNN використовує згорткові шари для виявлення ознак у вхідних даних та здатна розпізнавати складні шаблони та взаємозв'язки між ними. Основною структурою моделі є послідовність згорткових та пулінгових шарів, шарів випадкового відкидання та повнозв'язних шарів (рис. 2.3).

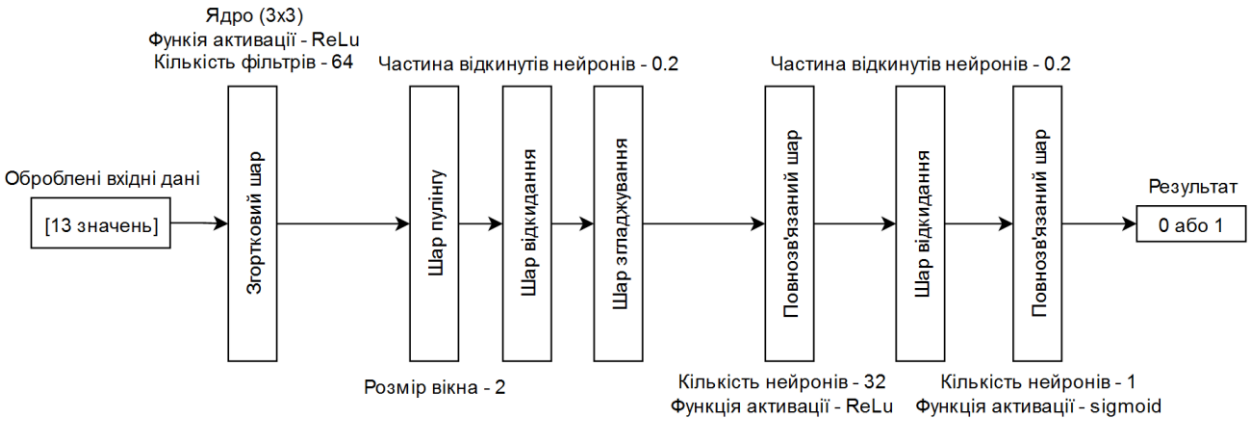

Рисунок 2.3 – Архітектура CNN

Перший згортковий шар має 64 фільтри з ядром розміру 3 та використовує активаційну функцію relu. Після цього шару йде пулінговий шар з розміром пулінгу 2, що зменшує розмір даних та допомагає уникнути перенавчання моделі.

Далі шар випадкового відкидання з відсотком випадкового відкидання 20% застосовується для запобігання перенавчання, після чого йде шар згладжування, щоб підготувати дані для використання у повнозв'язних шарах.

Наступний шар є повнозв'язним з 32 нейронами та активаційною функцією relu, після, для нього, використовується шар випадкового відкидання з 20% нейронів.

Останній повнозв'язний шар з одним нейроном та активаційною функцією sigmoid використовується для вирішення конкретної задачі бінарної класифікації. Функція активації sigmoid дозволяє моделі передбачати ймовірність належності до одного з двох класів.

Модель RNN використовує прості рекурентні, відкидні та повнозв’язні шари (рис. 2.4).

Рисунок 2.4 – Архітектура RNN

Ця модель має 4 SimpleRNN шара з 128, 63, 16 та 8 нейронів відповідно, які використовують функцію активації relu. Між ними розташовані шари відкидання (Dropout) з ймовірністю 20%. Вони допомагають уникнути перенавчання, випадково вимикаючи деякі нейрони під час навчання. Останнім є повнозв’язний шар з одним нейроном і функцією активації sigmoid, який використовується для бінарної класифікації. Перед ним також йде шар відкидання.

Модель RandomForestClassifier (лістинг 2.1) використовує ансамбль дерев для класифікації. Головна структура моделі включає 100 дерев рішень,

які працюють незалежно одне від одного (рис. 2.5). Кожне дерево приймає рішення на основі певних ознак та їх значень із набору даних.

```
model = RandomForestClassifier(criterion="entropy",
n_estimators=100, random_state=42)
```

Лістинг 2.1 – Модель Random forest

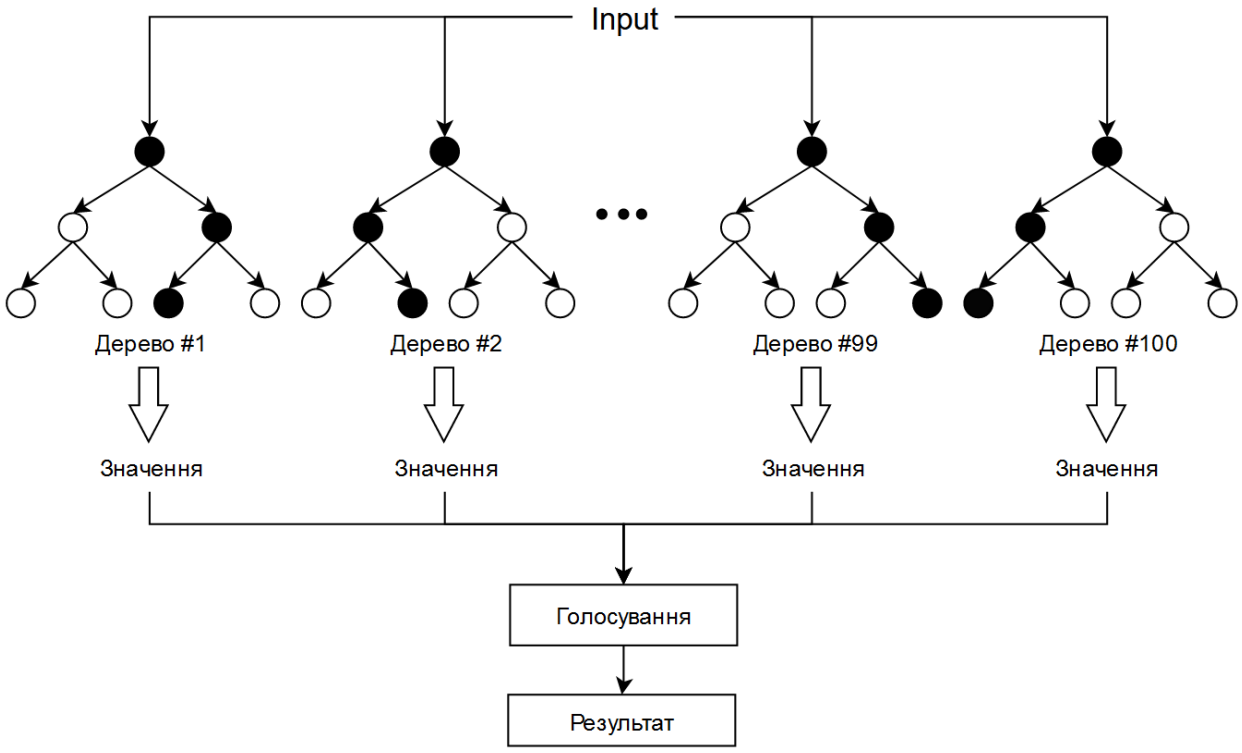

Рисунок 2.5 – Архітектура Random forest classifier

Однією з головних характеристик запланованої моделі є використання ентропії як критерію для визначення якості поділу даних у вузлах дерева. Цей критерій оцінює неоднорідність класів у вузлах, і оптимальний поділ вибирається таким чином, щоб зменшити ентропію та зробити дерево більш «чистим». Модель також використовує параметр n_estimators, який визначає кількість дерев у ансамблі. Більша кількість дерев сприяє зменшенню випадкових варіацій та поліпшенню загальної стійкості та точності моделі.

Також, для забезпечення відтворюваності результатів, встановлено параметр random_state зі значенням 42. Цей параметр використовується для ініціалізації генератора псевдовипадкових чисел, що забезпечує однакову послідовність випадкових чисел при кожному запуску моделі.

2.3 Характеристики вхідного набору даних

У даному дослідженні використано датасет, що складається з чотирьох баз даних: Клівленд, Угорщина, Швейцарія та Лонг Біч, що датуються 1988 роком. Цей датасет був створений Угорським інститутом кардіології, Будапешт: Андрашем Яносі, університетською лікарнею, Цюріх, Швейцарія: Вільямом Штейнбрунн, університетською лікарнею, Базель, Швейцарія: Маттіасом Пфістерером та медичним центром В.А., Лонг-Біч та Фондацією Клівлендської клініки: Робертом Детрано [8].

Загальна кількість атрибутів у датасеті становить 76, проте для всіх опублікованих експериментів використовувався піднабір з 14 атрибутів. Цей піднабір містить ключові характеристики, які допомагають в прогнозуванні наявності хвороби серця у пацієнта.

Один з основних атрибутів – «target», вказує на наявність хвороби серця у пацієнта. Цей параметр має цілочисельне значення, де 0 означає відсутність хвороби, а 1 вказує на наявність хвороби.

Датасет містить 302 унікальних екземпляри, які складаються з 14 атрибутів. Ці атрибути включають такі параметри, як вік, стать, відчуття болю в грудях, тиск, рівень холестерину, рівень цукру в крові, електрокардіографічні зміни у стані спокою, максимальний пульс, наявність стенокардії, кількість великих судин, таласемія та наявність серцево-судинних захворювань. Детальніший опис кожного атрибуту можна знайти в таблиці 2.1.

Таблиця 2.1 – Атрибути датасету

Атрибут	Опис	Значення
age	Вік	29 - 77
sex	Стать	(0)жіноча, (1)чоловіча
cp	Тип болю в грудях	(0)типова стенокардія, (1)атипова стенокардія, (2)біль без стенокардії, (3)безсимптомний
trestbps	Артеріальний тиск у стані спокою у mmHg	94 - 200
chol	Рівень холестерину у сироватці, в мг/дл	126 – 564
fbs	Рівень глюкози в крові натщесерце, більше 120 мг/дл	(0)False (1)True
restecg	Результати електрокардіографії у стані спокою	(0)нормальний, (1)аномалія ST-T, (2)можлива гіпертрофія лівого шлуночка
thalach	Максимальна досягнута частота серцевих скорочень	71 –202
exang	Приступи стенокардії, викликані фізичним навантаженням	(0)Ні (1)Так
oldpeak	Стенокардія, індукована навантажувальним тестом	-2.6 -
slope	Нахил сегмента ST в пік фізичних вправ	(1)підйомний, (2)рівний, (3)знизний
ca	Кількість основних судин, які підсвічуються флюороскопією	0, 1, 2, 3
thal	Таласемія	(3)нормальний, (6)фіксований дефект, (7)оборотний дефект
target	Наявність серцево-судинне захворювання	(0)хвороба серця відсутня, (1)хвороба серця наявна

Усього в датасеті є 1025 об'єктів спостереження. З них у 499 пацієнтів відсутня хвороба серця, а у 526 пацієнтів хвороба присутня. Це розподілення вказує на те, що в наборі даних присутня певна рівновага між позитивними і негативними випадками.

Щодо статевого розподілу в датасеті, 30,4% спостерігається жінок, а 69,6% - чоловіків (рис. 2.6). Це дає змогу вивчити вплив статі на наявність хвороби серця та проводити відповідні аналізи та порівняння.

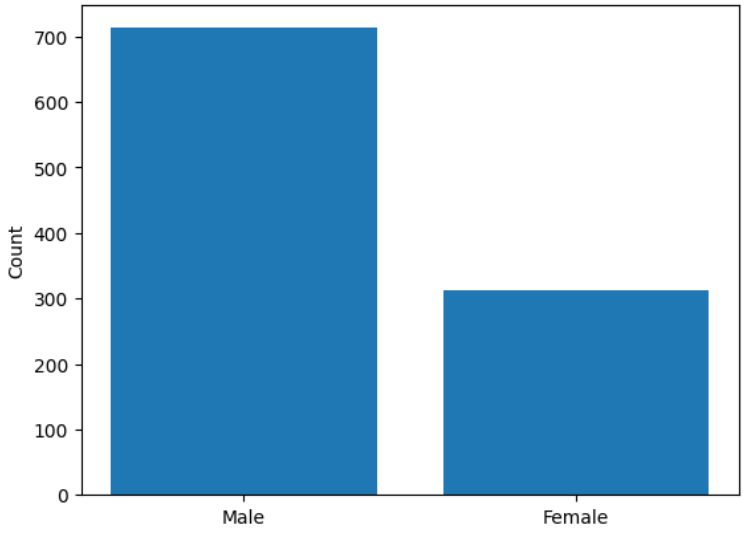

Рисунок 2.6 – Графік кількості пацієнтів за статтю

З 312 жінок, що входять у вибірку, 86 мають серцево-судинні захворювання, а з 713 чоловіків, що були досліджені, хворобу серця діагностовано у 413 осіб. Це свідчить про те, що 58% чоловіків у цьому наборі даних отримали діагноз хвороби серця, тоді як серед жінок ця частка складає лише 28% (рис. 2.7).

Вік пацієнтів, представлених у датасеті, варіюється у проміжку від 29 до 77 років. Це означає, що досліджувана група включає пацієнтів різного вікового спектру, що дозволяє проводити аналіз та вивчення залежностей між віком та наявністю хвороби серця. (рис. 2.8).

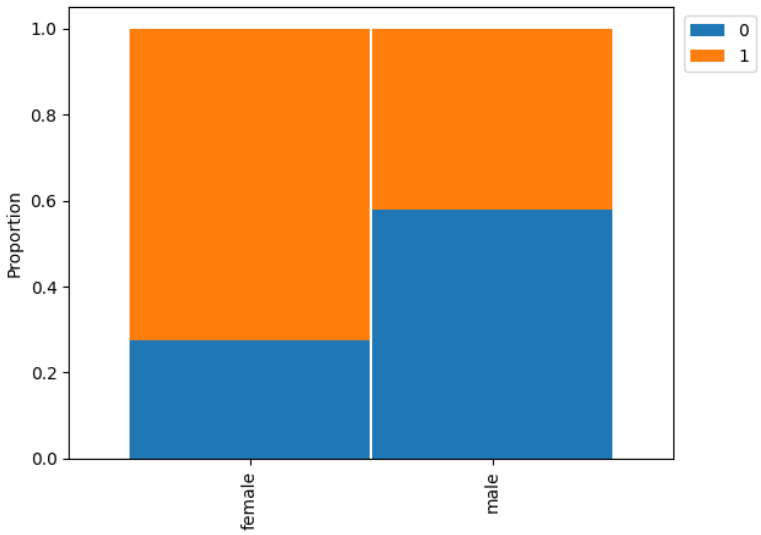

Рисунок 2.7 – Результати дослідження за наявністю хвороби серця в залежності від статі

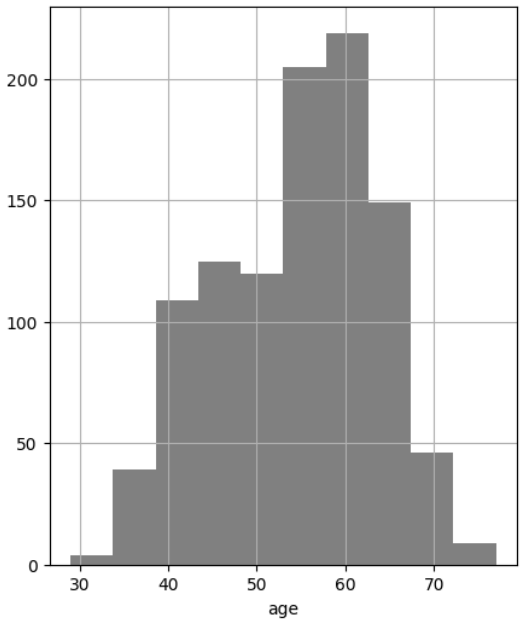

Рисунок 2.8 – Гістограма віку пацієнтів

Датасет обраний за якість представлених даних. Це можна побачити на кореляційній матриці (рис. 2.9). На якій видно, що параметри у середньому мають досить помірну кореляцію між собою, що може позитивним чином вплинути на процес навчання моделей.

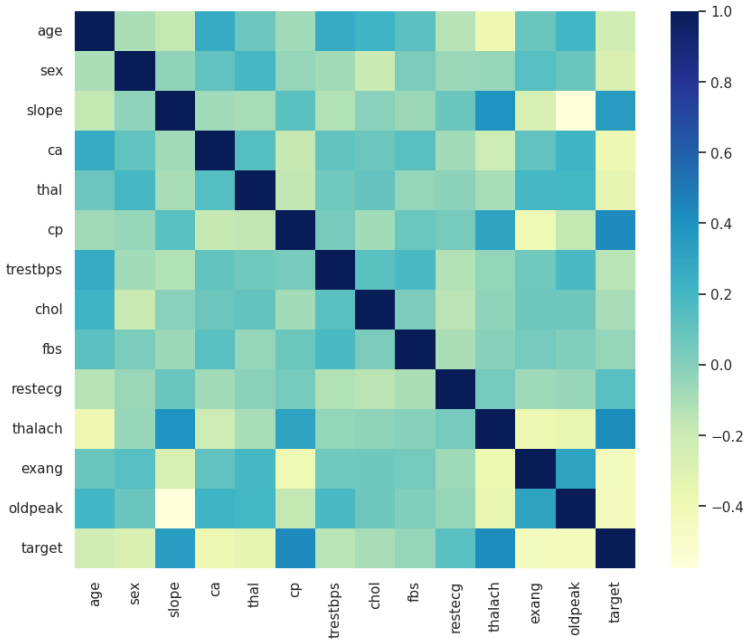

Рисунок 2.9 – Кореляційна матриця

2.4 Обґрунтування мови програмування та платформи для створення інтерфейсу програми

Для розробки та оптимізації алгоритмів використовується мова програмування Python, оскільки вона надає широкий спектр бібліотек та інструментів для роботи з машинним навчанням.

Основні використані бібліотеки:

– tensorFlow – базова бібліотека для створення моделей машинного навчання, яка забезпечує широкі можливості налаштування гіперпараметрів, оптимізації функцій втрат та використання різноманітних оптимізаторів;

– tensorflow.keras.callbacks – запобігання перенавчання моделей;

– tensorflow.keras.optimizers – налаштування гіперпараметру оптимізації моделей;

– pandas – завантаження та обробки вхідних даних для навчання моделі.

– scikit-learn (sklearn) – нормалізація вхідних даних та оцінювання якості навчання;

– Kfold з бібліотеки sklearn.model_selection – навчання моделей.

З метою забезпечення продуктивності та ефективності роботи з даними та моделями машинного навчання інтерфейс розроблений з використанням фреймворку Django. Даний фреймворк обраний через його масштабованість, високу швидкість розробки та надійність інтерфейсу.

2.5 Обробка даних та навчання моделей

Перед тренуванням моделей дані розділяються на тренувальний і тестовий набори за параметром «target». Тренувальний набір нормалізується функцією StandartScaler з бібліотеки sklearn, що стандартизує дані шляхом віднімання середнього значення і ділення на стандартне відхилення.

Після підготовки моделей розпочинається тренування з методом K-кратного перехресного затверджування. Вихідний набір даних розбивається на K рівних частин, де модель навчається на K-1 частині і перевіряється на залишковій. Цей процес повторюється K разів, оцінюючи помилку або точність моделі на всіх K тестових наборах.

Для кожної моделі встановлено тривалість навчання 5 епох з 100 кроків з ранньою зупинкою. Рання зупинка перевіряє покращення результатів навчання і припиняє процес, якщо трата на валідаційному наборі не покращується. Цей підхід допомагає досягти ефективності навчання та уникнути перенавчання моделі.

3 РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Опис програмної реалізації

Перш за все, перед початком навчання необхідно отримати та обробити дані з датасету (лістинг 3.1).

```
# Імпортуємо необхідні бібліотеки
import pandas as pd
from sklearn.preprocessing import StandardScaler
# Завантажуємо дані з файлу 'heart.csv'
df = pd.read_csv('heart.csv')
# Розділяємо дані на ознаки (X) та результат (y)
X = df.drop('target', axis=1).values
y = df['target'].values
# Нормалізуємо ознаки за допомогою StandardScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

Лістинг 3.1 – Завантаження та обробка вхідних даних

Далі перед створенням нейронних мереж CNN, RNN та FCNN необхідно створити параметри для налаштування, а саме функцію оптимізації – AdamW, ранню зупинку, яка слідкує за значенням втрати, і темп навчання (лістинг 3.2).

```
# Встановлюємо оптимізатор AdamW з коефіцієнтом швидкості
навчання 1e-3 та коефіцієнтом зниження ваги 1e-4
optimizer = AdamW(learning_rate=1e-3, weight_decay=1e-4)
# Встановлюємо раннє зупинення з моніторингом функції втрат на
валідаційному наборі та паузою 3 епохи
early_stopping = EarlyStopping(monitor='val_loss', patience=3)
# Встановлюємо регулятор швидкості навчання
LearningRateScheduler, який буде зменшувати швидкість навчання
до 1e-4 після 5 епохи
lr_schedule = LearningRateScheduler(lambda epoch: 1e-3 if epoch
< 5 else 1e-4)
```

Лістинг 3.2 – Тренувальні гіперпараметри

Після підготовки гіперпараметрів створюються моделі FCNN (лістинг 3.3), CNN (лістинг 3.5), RNN (лістинг 3.5) і Random forest (лістинг 3.6).

```
model = Sequential([
    tf.keras.layers.Dense(128, activation='relu',
input_shape=(X.shape[1],),
kernel_regularizer=regularizers.l2(0.001)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(64, activation='relu',
kernel_regularizer=regularizers.l2(0.001)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(32, activation='relu',
kernel_regularizer=regularizers.l2(0.001)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

Лістинг 3.3 – Архітектура FCNN

```
model = Sequential([
        Conv1D(filters=64, kernel_size=3, activation='relu',
input_shape=(X.shape[1], 1)),
        MaxPooling1D(pool_size=2),
        Dropout(0.2),
        Flatten(),
        Dense(32, activation='relu'),
        Dropout(0.2),
        Dense(1, activation='sigmoid')
    ])
```

Лістинг 3.4 – Архітектура CNN

```
model = tf.keras.Sequential([
    tf.keras.layers.SimpleRNN(128, activation='relu',
input_shape=(X.shape[1], 1), return_sequences=True),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.SimpleRNN(64, activation='relu',
return_sequences=True),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.SimpleRNN(16, activation='relu',
return_sequences=True),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.SimpleRNN(8, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

Лістинг 3.5 – Архітектура RNN

```
model = RandomForestClassifier(criterion="entropy",
n_estimators=100, random_state=42)
```

Лістинг 3.6 – Random forest

Після створення і компіляції моделей починається етап навчання. Всі моделі використовують K-кратне перехресне затверджування, при цьому алгоритм однаковий для CNN, RNN та FCNN (лістинг 3.7).

```
model.compile(optimizer=optimizer, loss='binary_crossentropy',
metrics=['accuracy'])
# Використовуємо KFold для крос-валідації
kf = KFold(n_splits=3, shuffle=True, random_state=42)
# Ініціалізуємо змінні для збереження кращої моделі та її метрик
best_model = None
best_val_loss = float('inf')
best_val_accuracy = 0.0
# Проходимо по всіх фолдах та навчаємо модель на кожному з них
for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    # Створюємо потік даних для тренувальних та валідаційних
даних
    train_dataset = tf.data.Dataset.from_tensor_slices((X_train,
y_train))
    train_dataset =
train_dataset.shuffle(buffer_size=1024).batch(32).repeat()
    val_dataset = tf.data.Dataset.from_tensor_slices((X_test,
y_test))
    val_dataset = val_dataset.batch(32)
     # Тренуємо модель
    history = model.fit(train_dataset, epochs=5,
steps_per_epoch=100, validation_data=val_dataset,
callbacks=[early_stopping, lr_schedule])
    # Перевіряємо, чи поточна модель краща попередньої
    val_loss = history.history['val_loss'][-1]
    val_accuracy = history.history['val_accuracy'][-1]
    if val_loss < best_val_loss:
        # Зберігаємо метрики кращої моделі та її ваги
        best_val_loss = val_loss
        best_val_accuracy = val_accuracy
        best_model = model.get_weights()
# Завантажуємо ваги кращої моделі
model.set_weights(best_model)
```

Лістинг 3.7 – Навчання моделей CNN, RNN та FCNN

Для Random forest алгоритм навчання (лістинг 3.8) трохи змінюється, через іншу структуру моделі. Різниця у обробці даних при тренуванні моделі. Тоді як для CNN, FCNN та RNN створюється потоки даних з тренувальних та тестових значень, які використовуються при навчанні, випадковий ліс приймає напряму дані без їх об’єднання та перемішування.

```
# Використовуємо KFold для крос-валідації
kf = KFold(n_splits=3, shuffle=True, random_state=42)
# Ініціалізуємо змінну для збереження кращої моделі та її метрик
best_model = None
best_val_accuracy = 0.0
# Проходимо по всіх фолдах та навчаємо модель на кожному з них
for train_index, val_index in kf.split(X):
    X_train, X_val = X[train_index], X[val_index]
    y_train, y_val = y[train_index], y[val_index]
    # Навчаємо модель RandomForestClassifier на тренувальних
даних
    model.fit(X_train, y_train)
    # Оцінюємо модель RandomForestClassifier на валідаційних
даних
    val_accuracy = model.score(X_val, y_val)
    # Перевіряємо, чи поточна модель краща попередньої
    if val_accuracy > best_val_accuracy:
        # Зберігаємо метрики кращої моделі та її ваги
        best_val_accuracy = val_accuracy
        best_model = model
```

Лістинг 3.8 – Навчання моделі RFC

3.2 Програмна реалізації

Для зручного тестування моделей машинного навчання створений веб додаток за допомогою фреймворку Django. Він підтримує можливості завантаження таблиць даних, моделей у форматі .h5 та .pkl, перегляд даних окремих елементів та загальної інформації щодо завантаженої таблиці.

Для завантаження таблиці використовується функція loadCSVFile (додаток А, лістингА.1). Для викликається функція buildTable (додаток А, лістинг А.2), яка будує таблицю з даних завантаженого файлу.

Для завантаження моделі машинного навчання використовується функція uploadNNFile (додаток А, лістинг А.3), яка дає користувачу можливість вибору файлу навченого алгоритму і збережує цей файл у системі. Вона викликається всередині функції handleNNLoad (додаток А, лістинг А.4), після чого модель і таблиця передається через запит «POST» до функції run_nn_prediction (додаток А, лістинг А.5).

Функція run_nn_prediction у свою чергу розділяє отриману структуру у два об’єкти, модель та таблиця, і викликає функцію nn_prediction (додаток А, лістинг А.6) передаючи ці дані для отримання передбачень моделі. Функція nn_prediction завантажує модель, обробляє дані за допомогою StandardScaler виклкикаючи функцію preprocess (лістинг А.7) та повертає масив передбачень.

3.3 Інструкція користувача

При запуску веб-додатку користувач побачить перед собою два елементи. Той, що знаходиться зліва – пустий, він використовується для відображення завантаженої таблиці. Поле справа має 3 вкладки. За замовчуванням активна вкладка для завантаження таблиці та алгоритму (рис. 3.1).

Рисунок 3.1 – Сторінка веб-додатку на старті на вкладці «Load»

На вкладці «Patient info» розписані всі параметри пацієнту з описом значень (рис. 3.2).

На вкладці «Data info» відображається інформація щодо загальної кількості пацієнтів, кількості потенційно хворих пацієнтів та активного на даний момент алгоритма (рис. 3.3).

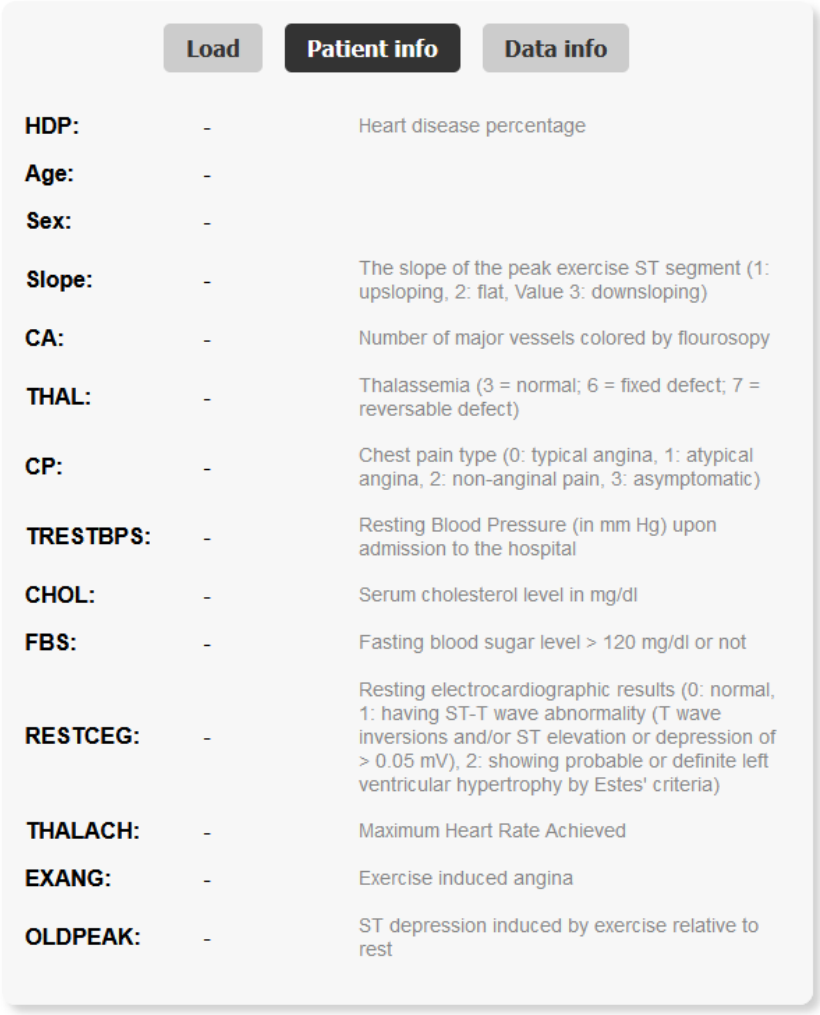

Рисунок 3.2 – Сторінка веб-додатку на старті на вкладці «Patient info»

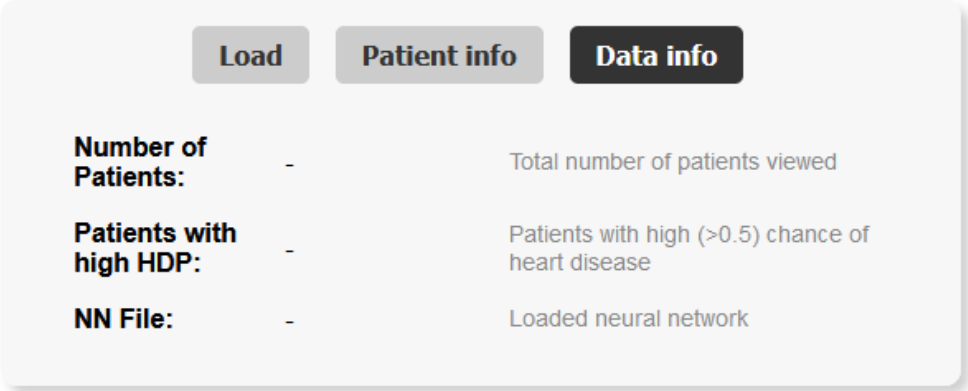

Рисунок 3.3 – Сторінка веб-додатку на старті на вкладці «Data info»

Для завантаження таблиці даних у вкладці «Load» необхідно натиснути на кнопку «Select CSV File» і обрати необхідний файл у форматі .csv у вікні, що з’явиться. Після цього поле зліва відобразить інформацію вибраного файлу (рис. 3.4).

age	sex	slope	ca	thal	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak
52	1	2	2	3	0	125	212	0	1	168	0	1
53	1	0	0	3	0	140	203	1	0	155	1	3.1
70	1	0	0	3	0	145	174	0	1	125	1	2.6
61	1	2	1	3	0	148	203	0	1	161	0	0
62	0	1	3	2	0	138	294	1	1	106	0	1.9
58	0	1	0	2	0	100	248	0	0	122	0	1
58	1	0	3	1	0	114	318	0	2	140	0	4.4
55	1	1	1	3	0	160	289	0	0	145	1	0.8
46	1	2	0	3	0	120	249	0	0	144	0	0.8
54	1	1	2	2	0	122	286	0	0	116	1	3.2
71	0	1	0	2	0	112	149	0	1	125	0	1.6
43	0	1	0	3	0	132	341	1	0	136	1	3
34	0	2	0	2	1	118	210	0	1	192	0	0.7
51	1	1	3	3	0	140	298	0	1	122	1	4.2
52	1	1	0	0	0	128	204	1	1	156	1	1
34	0	2	0	2	1	118	210	0	1	192	0	0.7
51	0	2	1	2	2	140	308	0	0	142	0	1.5
54	1	1	1	3	0	124	266	0	0	109	1	2.2
50	0	2	0	2	1	120	244	0	1	162	0	1.1

Рисунок 3.4 – Поле пацієнтів веб-додатку після завантаження таблиці даних

Для підключення моделі машинного навчання на вкладці «Load» необхідно натиснути на кнопку «Load Neural Network», обрати файл натренованої моделі у форматах .h5 або .pkl, після чого таблиця обновиться и відобразить всіх потенційно хворих пацієнтів. Окрім цього значення ймовірності наявності хвороби можна побачити на вкладці «Patient info» разом з іншими даними пацієнта (рис. 3.5).

На вкладці же «Data info» можна побачити загальну кількість потенційно хворих (рис. 3.6).

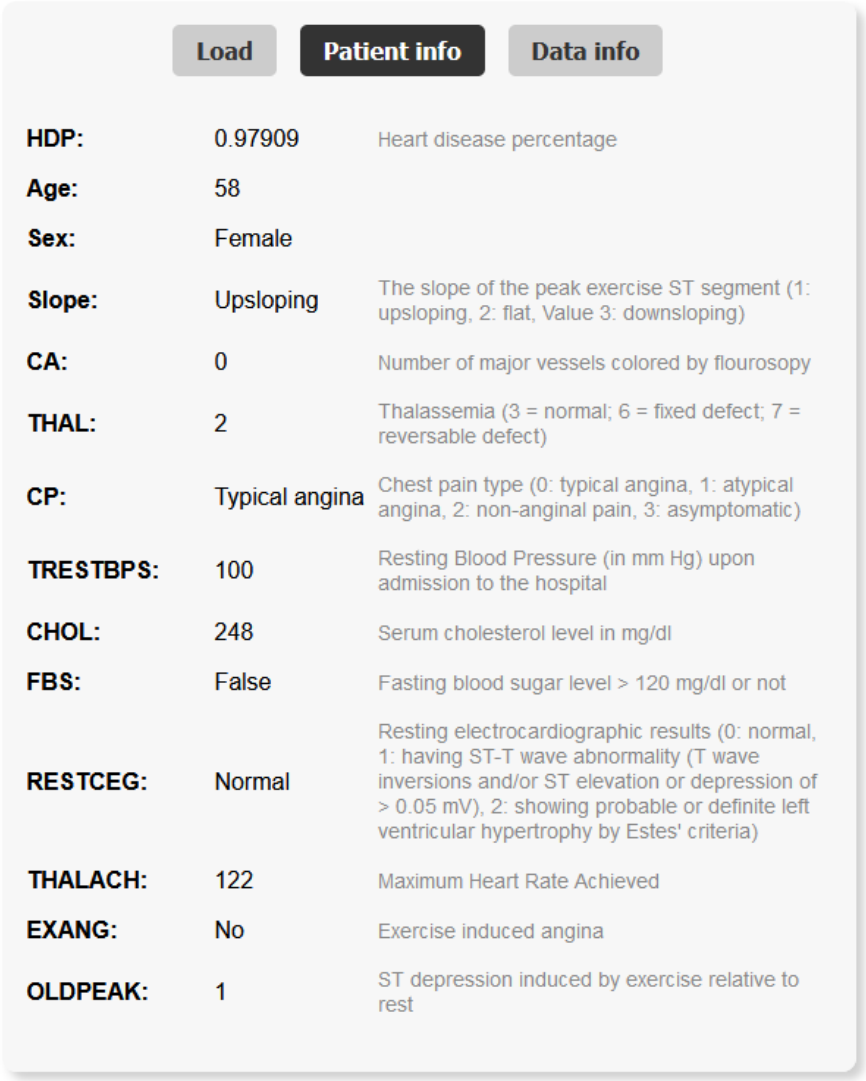

Рисунок 3.5 – Дані вкладки «Patient info» при виборі елементу з таблиці

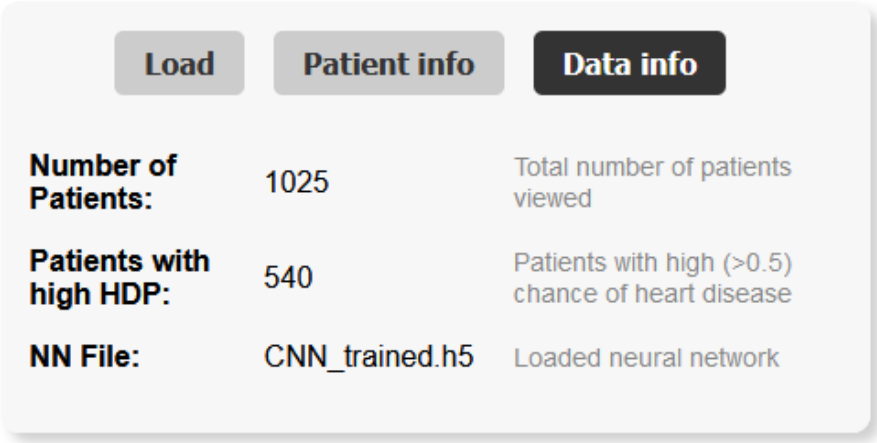

Рисунок 3.6 – Дані вкладки «Data info» після завантаження моделі

4 ТЕСТУВАННЯ СИСТЕМИ

Для тестування створеної системи необхідно порівняти навчені моделі машинного навчання. Для цього потрібно вирахувати метрики кожної моделі. Перед початком рахування метрик необхідно отримати матриці помилок навчених моделей.

Матриця помилок є інструментом для оцінки ефективності класифікації моделі. Бінарний варіант представляється у вигляді 2x2 матриці, де кожен елемент матриці відображає кількість спостережень, які попали в певну категорію класифікації. Матриця має наступні елементи:

– True Positive (TP) – кількість правильно класифікованих позитивних прикладів.

– False Positive (FP) – кількість неправильно класифікованих позитивних прикладів.

– False Negative (FN) – кількість неправильно класифікованих негативних прикладів.

– True Negative (TN) – кількість правильно класифікованих негативних прикладів.

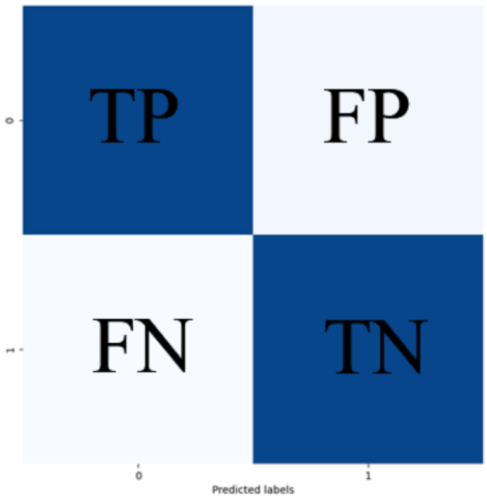

Рисунок 4.1 – Схема матриці помилок

Ці значення дозволяють обчислити різні метрики, що характеризують ефективність класифікації моделі, такі як точність (precision), чутливість (recall), F-мера (F-measure) і специфічність (specificity). За допомогою бінарної матриці помилок також можна визначити загальну точність класифікації моделі та оцінити її здатність розпізнавати певний клас даних.

Бінарна матриця помилок є важливим інструментом для оцінки якості класифікаційних моделей, особливо у випадку задач з двома класами. Вона надає деталізовану інформацію про розподіл помилок і дозволяє зробити висновки щодо сильних і слабких сторін моделі, а також внести необхідні покращення для досягнення більш точних результатів.

Матриця помилок CNN моделі (рис. 4.1):

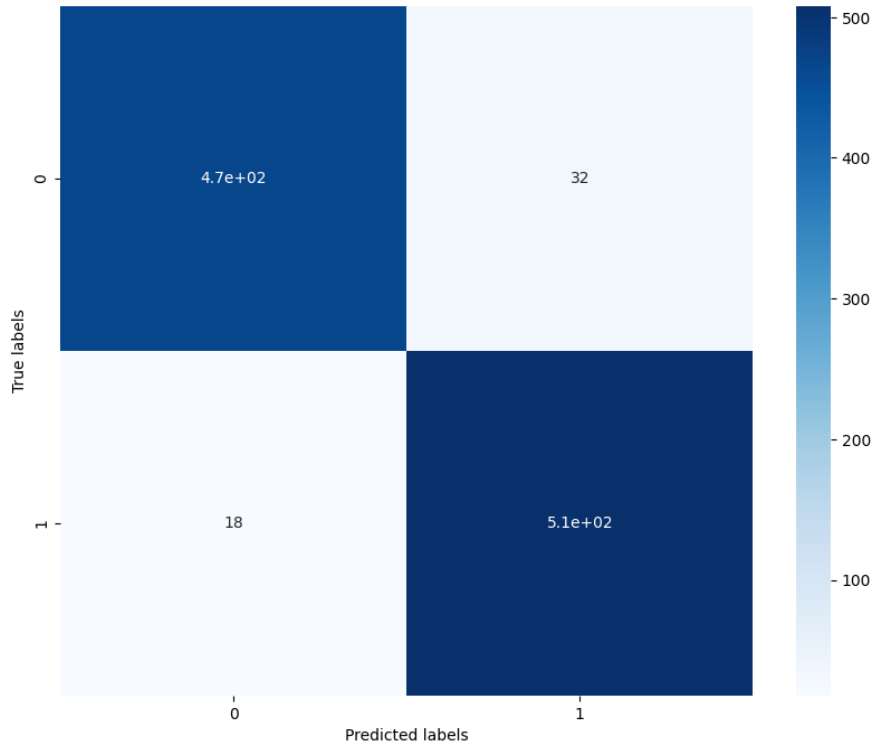

Рисунок 4.1 – Матриця помилок CNN

Матриця помилок FCNN моделі (рис. 4.2):

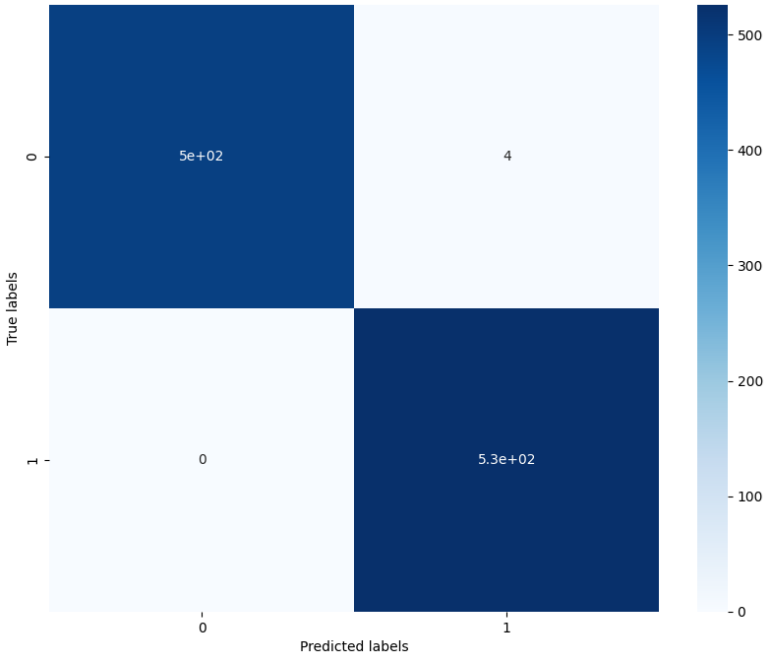

Рисунок 4.2 – Матриця помилок FCNN

Матриця помилок RNN моделі (рис. 4.3):

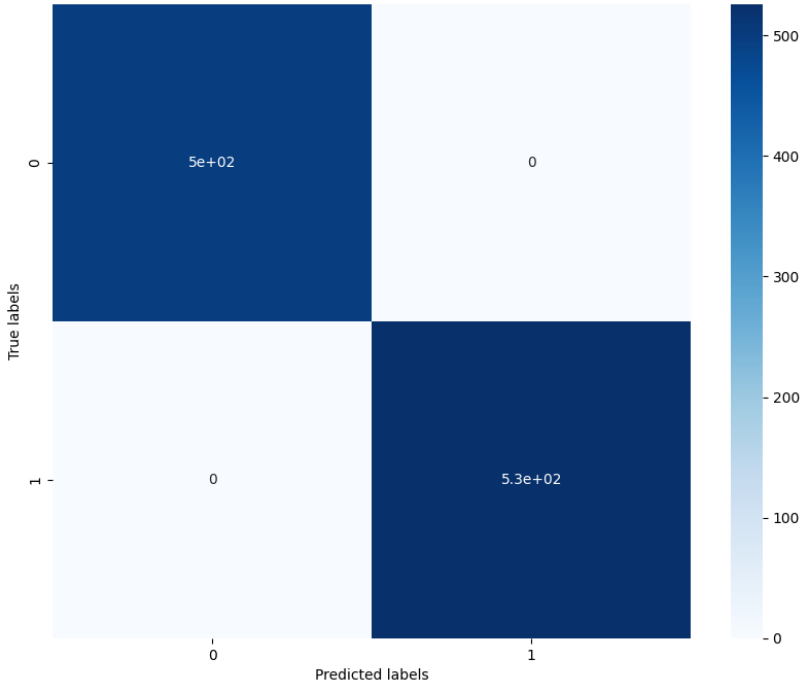

Рисунок 4.3 – Матриця помилок RNN

Матриця помилок RNN моделі (рис. 4.4):

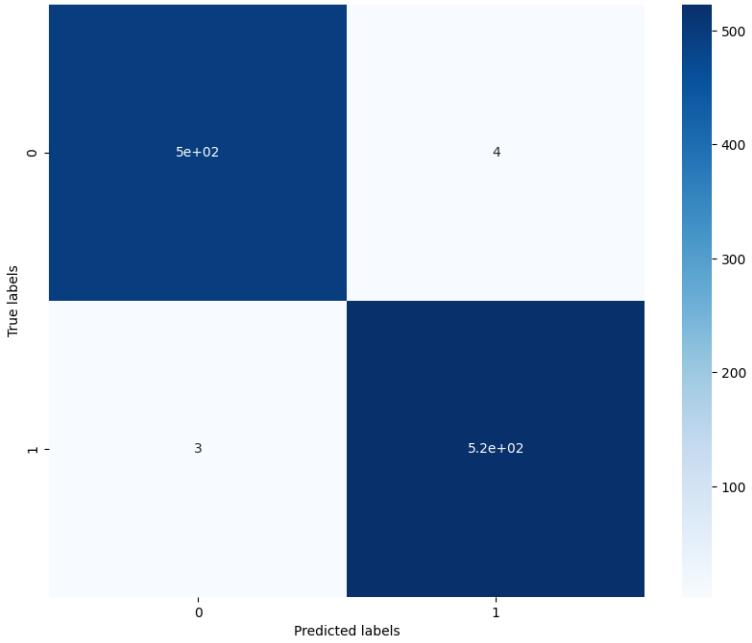

Рисунок 4.4 – Матриця помилок Random Forest

Тепер отримавши значення матриці помилок можна починати пошук наступних метрик. Для початку вирахуємо loss для FCNN, CNN, RNN та Random forest (рис. 4.5).

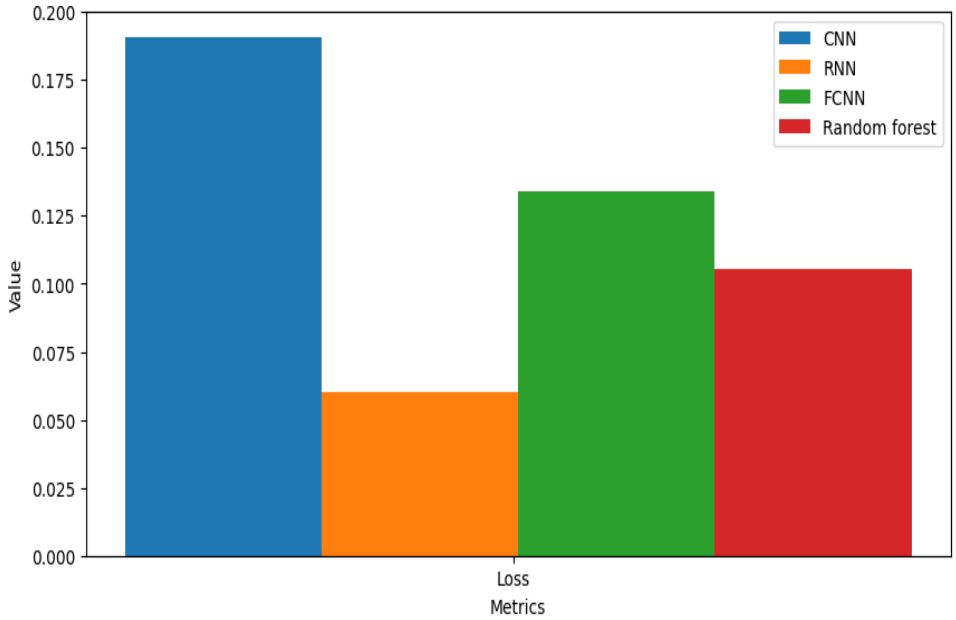

Рисунок 4.5 – Порівняння значення втрати у різних моделей

За даними можна побачити, що найкращою є RNN модель, а найгіршою – CNN.

Далі знайдемо accuracy передбачень (рис. 4.6).

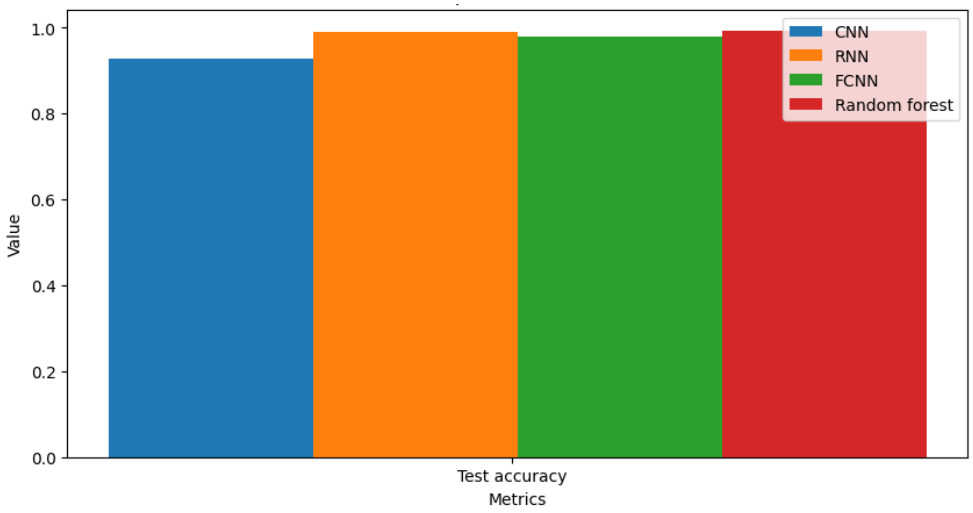

Рисунок 4.6 – Порівняння значення втрати у різних моделей

З отриманого графіку можна зробити висновок, що всі моделі непогано показали себе при тестування точності. Найкращими моделями за метрикою accuracy –RNN та Random Forest.

Наступним знайдемо precision, recall та f-міру (рис. 4.7).

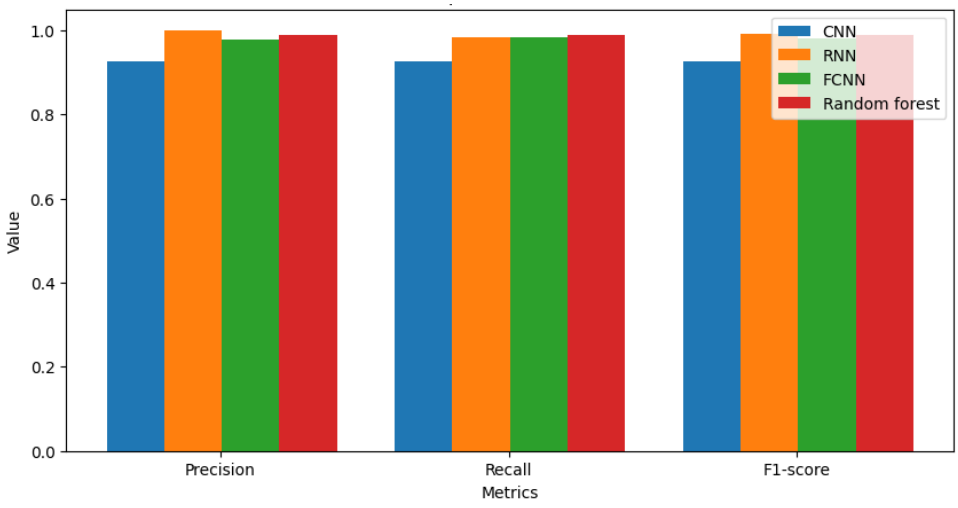

Рисунок 4.7 – Порівняння значення втрати у різних моделей

Аналізуючи отримані діаграми та дані з таблиці 4.1, можна зробити висновок, що всі чотири моделі (CNN, RNN, FCNN та Random Forest) продемонстрували високу ефективність в завданні бінарної класифікації.

Таблиця 4.1 – Тестування моделей

Model	Loss	Accuracy	Precision	Recall	F1-score
CNN	0.1907	0.9269	0.9269	0.9269	0.9269
RNN	0.0603	0.9912	1.0	0.9831	0.9915
FCNN	0.1338	0.9795	0.9776	0.9831	0.9803
Random forest	0.1052	0.993	0.99	0.99	0.99

Таким чином, за результатами тестування та аналізу даних з таблиці 4.1 можна зробити висновок про ефективність моделей. Загальний порядок ефективності моделей у бінарній класифікації від найкращого до найгіршого такий: RNN, Random Forest, FCNN та CNN.

RNN та Random Forest показали найвищі показники матрик серед усіх чотирьох моделей. Ці моделі проявили себе як добре налаштовані та здатні зробити достовірні прогнози у бінарній класифікації.

Наступна за ефективністю модель – FCNN продемонструвала гарні результати, хоча трохи відстала від RNN та Random Forest. Однак, FCNN все ще є потужним інструментом для класифікації та може забезпечити достатньо високу точність.

Нарешті, найменш ефективною моделлю виявилася CNN. Хоча вона все ще показала прийнятні результати, вона має нижчі метрик порівняно з іншими моделями.

ВИСНОВКИ

У даній кваліфікаційній роботі проведено порівняльний аналіз різних моделей машинного навчання для прогнозування серцево-судинних захворювань. Розглянуті моделі включають FCNN, CNN, RNN та Random Forest Classifier. Використано базу даних про пацієнтів з серцевими захворюваннями.

Для розробки забезпечення для прогнозування серцево-судинних захворювань використан фреймворк Django. Розроблений веб застосунок дозволяє завантажувати різні дані правильного формату та різні моделі машинного навчання для прогнозування результатів за вхідними даними.

У роботі досліджено точність кожної моделі на базі Heart Disease, і результати показали, що найкращою моделлю є RNN з 99.1% точністю і 0.06 втратою. Точність і втрата моделей FCNN та CNN склали відповідно 97.9% з 0.134 та 92.7% з 0.2, Random Forest Classifier же показав 99.3% точність і 0.105 втрату.

Отже, модель RNN є найкращою серед розглянутих моделей. Хоча, алгоритм Random Forest Classifier показав другі за якістю значення, які лише трохи поступаються.

За результатами даної роботи опубліковано тези на всеукраїнській конференції [7].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Conor Stewart, State of AI/machine learning in hospitals in the U.S. in 2021. [Електронний ресурс] – Режим доступу: https://www.statista.com/statistics/1249788/ai-machine-learning-in-hospitals-in-the-us/.
2. Conor Stewart, Level of adoption of AI in healthcare in the EU in 2021, by technology. [Електронний ресурс] – Режим доступу: https://www.statista.com/statistics/1312566/adoption-stage-of-ai-in-healthcare-in-the-eu/.
3. Conor Stewart, AI in healthcare market size worldwide 2021-2030. [Електронний ресурс] – Режим доступу: https://www.statista.com/statistics/1334826/ai-in-healthcare-market-size-worldwide/.
4. Medicsen. [Електронний ресурс] – Режим доступу: https://medicsen.com/en/.
5. Linguamatics. [Електронний ресурс] – Режим доступу: https://www.linguamatics.com/.
6. DeepMind Health. [Електронний ресурс] – Режим доступу: https://www.deepmind.com/search?query=Health.
7. Осипов А. В., Шпінарева І. М. Застосування методів машинного навчання для прогнозування серцевих захворювань / Інформатика, інформаційні системи та технології: тези доповідей XX всеукраїнської конференції студентів і молодих науковців. Одеса, 28 квітня 2023 р. – Одеса, 2023. – с.103-104.
8. Heart Disease Dataset. [Електронний ресурс] – Режим доступу: https://www.kaggle.com/johnsmith88/heart-disease-dataset.

ДОДАТОК А

Завантаження і обробка даних у веб-застосунку

```
function loadCSVFile() {
  const fileInput = document.createElement('input');
  fileInput.type = 'file';
  fileInput.accept = '.csv';
  // Додаємо обробник події на зміну вмісту файлу
  fileInput.addEventListener('change', function (e) {
    patientTotalNumber = 0;
    hdp_patients = 0;
    nnFilename = "-"
    // Отримуємо файл, який вибрав користувач
    const file = e.target.files[0];
    // Створюємо об'єкт FileReader для зчитування файлу
    const reader = new FileReader();
    // Додаємо обробник події на завантаження файлу
    reader.onload = function (event) {
      // Отримуємо вміст файлу як текстовий рядок
      const csvData = event.target.result;
      // Створюємо таблицю з вмісту файлу
      const table = buildTable(csvData);
      // Відображаємо таблицю на сторінці
      displayTable(table);
      // Додаємо обробник події на клік
      attachRowClickEvent(table);};
    // Зчитуємо вміст файлу як текст
    reader.readAsText(file);});
  // Симулюємо клік на HTML-елементі <input> для виклику
  fileInput.click();}
```

Лістинг А.1 – Функція loadCSVFile

```
function buildTable(csvData) {
  // Розділяємо вміст файлу на рядки
  const rows = csvData.split('\n');
  // Створюємо HTML-елемент <div> для контейнера таблиці
  var tcontainer = document.createElement('div');
  tcontainer.classList.add('table-container');
  // Створюємо HTML-елемент <div> для контенту таблиці
  var tcontent = document.createElement('div');
  tcontent.classList.add('table-content');
  // Створюємо HTML-елементи <table> для заголовка та вмісту
таблиці
  const header_table = document.createElement('table');
  const content_table = document.createElement('table');
```

Лістинг А.2 – Функція buildTable

```
  // Встановлюємо прапорець, що поточний рядок є заголовком
  var isHeader = true;
  // Проходимо по кожному рядку
  rows.forEach(function (row, index) {
    // Розділяємо рядок на стовпці
    const columns = row.split(',');
    // Створюємо HTML-елемент <tr> для кожного рядка
    const tr = document.createElement('tr');
    // Якщо поточний рядок не є заголовком
    if (index !== 0) {
      // Додаємо обробник події на клік рядка
      tr.setAttribute('onclick', 'rowClicked(this)');
      // Збільшуємо лічильник загальної кількості пацієнтів
      patientTotalNumber++;
    }
    // Проходимо по кожному стовпцю в рядку
    columns.forEach(function (column) {
      // Створюємо HTML-елемент <td> для звичайного стовпця або
<th> для заголовку
      const td = document.createElement(index === 0 ? 'th' :
'td');
      // Заповнюємо вміст стовпця текстом
      td.textContent = column;
      // Додаємо стовпець до рядка
      tr.appendChild(td);});
    // Якщо поточний рядок є заголовком
    if (isHeader) {
      // Додаємо рядок до таблиці заголовка
      header_table.appendChild(tr);
      isHeader = false;
    } else {
      // Додаємо рядок до таблиці вмісту
      content_table.appendChild(tr);
    }});
  // Додаємо таблицю заголовка до контейнера
  tcontainer.appendChild(header_table);
  // Додаємо таблицю вмісту до контенту таблиці
  tcontent.appendChild(content_table);
  // Додаємо контент до контейнера
  tcontainer.appendChild(tcontent);
  // Зменшуємо лічильник загальної кількості пацієнтів на 1,
оскільки перший рядок є заголовком
  patientTotalNumber--;
  // Оновлюємо інформацію про кількість пацієнтів
  updateDataInfo();
  // Повертаємо контейнер таблиці
  return tcontainer;}
```

Лістинг А.2, лист 2

```
function uploadNNFile() {
  // Створюємо новий Promise
  return new Promise(function(resolve, reject) {
    // Створюємо HTML-елемент <input> типу файл
    const fileInput = document.createElement('input');
    fileInput.type = 'file';
    // Вказуємо, що можна вибрати тільки файли з розширенням .h5
та .pkl
    fileInput.accept = '.h5,.pkl'
    // Додаємо обробник події на зміну вмісту файлу
    fileInput.addEventListener('change', function() {
      // Якщо користувач вибрав файл
      if (fileInput.files.length > 0) {
        // Запам'ятовуємо ім'я файлу
        nnFilename = fileInput.files[0].name;
        // Розв'язуємо Promise з файлом як аргументом
        resolve(fileInput.files[0]);
      } else {
        // Якщо користувач не вибрав файл, зберігаємо порожнє
ім'я файлу та відхиляємо Promise з помилкою
        nnFilename = ""
        reject(new Error('No file selected'));}});
    // Симулюємо клік на HTML-елементі <input>, щоб відобразити
вікно вибору файлу
    fileInput.click();});}
```

Лістинг А.3 – Функція uploadNNFile

```
async function handleNNLoad() {
  // Перевіряємо, чи є пацієнти в таблиці
  if(patientTotalNumber != 0){
    try {
      // Отримуємо дані з таблиці
      const table_data = getTableData();
      // Запрошуємо користувача вибрати файл нейромережі
      const file = await uploadNNFile();
      // Створюємо новий об'єкт FormData
      const formData = new FormData();
      // Додаємо файл та дані таблиці до об'єкту FormData
      formData.append('file', file);
      formData.append('input_data', JSON.stringify(table_data));
      // Отримуємо CSRF-токен з HTML-сторінки
      const csrftoken = $('[name=csrfmiddlewaretoken]').val();
      // Відправляємо POST-запит на сервер
      const response = await $.ajax({
        url: '/run_nn_prediction/',
        type: 'POST',
```

Лістинг А.4 – Функція handleNNLoad

```
        headers: {
          'X-CSRFToken': csrftoken
        },
        processData: false,
        contentType: false,
        data: formData,});
      // Обробляємо відповідь сервера
      hdp = response.prediction;
      // Форматуємо дані перед відображенням в таблиці
      hdp = hdp.map(num => num.toLocaleString(undefined, {
        minimumFractionDigits: 5,
        maximumFractionDigits: 5
    }));
      // Відображаємо результати передбачення в таблиці
      displayPredictionInsideTable(hdp);
    } catch (error) {
      console.error('Error:', error);
    }}}
```

Лістинг А.4, лист 2

```
def run_nn_prediction(request):
    # Перевіряємо, чи є запит методом POST
    if request.method == 'POST':
        # Отримуємо дані та файл з запиту
        input_data = json.loads(request.POST.get('input_data'))
        # Конвертуємо рядки таблиці у числа з плаваючою точкою
        input_data = [[float(item) for item in row if
item.strip() != ""] for row in input_data]
        # Отримуємо файл з запиту
        file = request.FILES['file']
        try:
            # Викликаємо функцію nn_prediction для передбачення
            prediction = nn_prediction(input_data, file)
        except ValueError as e:
            # Якщо сталася помилка, повертаємо відповідь з
помилкою та статусом 400
            return HttpResponseBadRequest(str(e))
        # Повертаємо результат передбачення у вигляді JSON-
відповіді
        return JsonResponse({'prediction': prediction})
    # Якщо запит не методом POST, повертаємо відповідь з
помилкою та статусом 400
    return HttpResponseBadRequest('Invalid request method')
```

Лістинг А.5 – Функція run_nn_prediction

```
def nn_prediction(input_data, nn_file):
    # Обробляємо вхідні дані
    preprocessed_data = preprocess(input_data)
    try:
        # Зберігаємо завантажений файл нейромережі у тимчасовому
місці
        file_path = os.path.join(tempfile.gettempdir(),
nn_file.name)
        with open(file_path, 'wb') as f:
            f.write(nn_file.read())
        model = 0
        if(nn_file.name[-2:] == 'h5'):
            # Завантажуємо навчену модель Keras
            with h5py.File(file_path, 'r') as f:
                model = load_model(file_path)
        elif(nn_file.name[-3:] == 'pkl'):
            # Завантажуємо навчену модель з використанням
бібліотеки joblib
            model = joblib.load(file_path)
        # Обробляємо вхідні дані за допомогою моделі
        prediction = model.predict(preprocessed_data)
        return prediction.tolist()
    except OSError as e:
        # Якщо сталася помилка, повертаємо None
        print(f"Unable to load model: {e}")
        return None
```

Лістинг А.6 – Функція nn_prediction

```
def preprocess(input_data):
    # Видаляємо порожні рядки з вхідних даних
    data_list = [lst for lst in input_data if lst]
    # Конвертуємо вхідні дані у масив NumPy
    preprocessed_data = np.array(data_list)
    # Створюємо об'єкт StandardScaler та масштабуємо вхідні дані
    scaler = StandardScaler()
    processed_data = scaler.fit_transform(preprocessed_data)
    return processed_data
```

Лістинг А.7 – Функція preprocess