

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)

“ ” _____ 2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення
інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Програмна бібліотека аналізу ЕКГ лінгвістичними методами

Виконав студент IV курсу, групи ІП-92
(шифр групи)

Бераудо Еліза
(прізвище, ім'я, по батькові) _____ (підпис)

Керівник доцент, к.т.н., Олійник Юрій Олександрович
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Консультант
з графічної
документації доцент, к.т.н., доц., Лішук К. І.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Рецензент доцент кафедри ІСТ, к.т.н., доц., Букасов М. М.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____
(підпис)

Київ –2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення
інформаційних систем

ЗАТВЕРДЖУЮ

Завідувач кафедри

(підпис) Едуард ЖАРІКОВ
(ім'я прізвище)

“ ____ ” _____ 2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Бераудо Еліза

(прізвище, ім'я, по батькові)

1. Тема проєкту Програмна бібліотека аналізу ЕКГ лінгвістичними
методами

керівник проєкту Олійник Юрій Олександрович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «31»квітня 2023 р. №2101-с

2. Термін подання студентом проєкту « 19 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки

1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.

2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.

3) Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання.

4) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.

5) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема структурна бізнес процесу

3) Схема структурна класів програмного забезпечення

6) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» березня 2023 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв'язання задачі	20.03.2023	
3	Постановка та формалізація задачі	08.04.2023	
4	Розробка інформаційного забезпечення	10.04.2023	
5	Алгоритмізація задачі	11.04.2023	
6	Обґрунтування вибору використаних технічних засобів	15.04.2023	
7	Розробка програмного забезпечення	16.04.2023	
8	Налагодження програми	28.04.2023	
9	Виконання графічних документів	05.05.2023	
10	Оформлення пояснювальної записки	29.05.2023	
11	Подання ДП на попередній захист	06.06.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	19.06.2023	

Студент

(підпис)

Еліза БЕРАУДО

(ініціали, прізвище)

Керівник

(підпис)

Юрій ОЛІЙНИК

(ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 21 таблиць, 41 рисуноків та 70 джерел – загалом 93 сторінки.

Дипломний проєкт присвячений підвищенню та розширенню інструментів для автоматичного аналізу електрокардіограми шляхом розробки програмної бібліотеки, що реалізує аналіз ЕКГ за допомогою лінгвістичних методів. Мета - підвищення та розширення інструментів для автоматичного аналізу електрокардіограм шляхом розробки програмної бібліотеки, що реалізує аналіз ЕКГ за допомогою лінгвістичних методів.

Об'єкт дослідження: сигнали ЕКГ та методи їх автоматичного аналізу.

Предмет дослідження: техніки моделювання та аналізу ЕКГ за допомогою лінгвістичних методів, а також впровадження та валідація програмної бібліотеки для цієї мети.

Перший важливий компонент проєкту - Моделювання. У цьому розділі, на фронтенді, користувачі можуть вводити параметри навчання, такі як кількість кластерів для Kmeans, розмір вектора Word2Vec, а також вибирати алгоритм класифікації серцебиття (такий як random forest, svc і т.д.) та встановлювати специфічні для класифікатора параметри. Після того, як користувач ініціює процес, модель навчається та оцінюється на бекенді. Потім бекенд відповідає на фронтенд з метриками моделі класифікатора (точність, точність, відгук, F1 score), балом силуета k-means і чотирма графіками (три графіки t-SNE з кластеризованими хвилями P, QRS, та T серцебиття та один графік - матриця невизначеності для моделі класифікації). Вся ця інформація потім представляється користувачу.

Другий важливий компонент - Аналіз. Тут, на фронтенді, користувачі можуть завантажувати свої власні csv/xlsx файли з даними ЕКГ, встановлювати значення fs та ініціювати аналіз. Бекенд використовує всі навчені алгоритми для класифікації серцебиття у завантаженому файлі та повертає графік на фронтенд, де всі серцебиття позначені (нормальні, ненормальні або інші класи). Цей графік відображається для користувача. Крім того, користувачі можуть навести курсор на кожне серцебиття на графіку та побачити передбачувану мітку (нормальні,

ненормальні або інші класи). Під графіком з серцебиттям та передбачуваними мітками відображається таблиця, що показує кількість серцебиттів, мітку серцебиття, 3-літерну назву серцебиття (три літери, оскільки у нас є три кластеризовані хвилі, а кількість кластерів перетворюється на літери), та векторне представлення серцебиття.

У розділі "АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ" розглядаються основні положення та змістовний опис предметної області, аналізуються наявні технології та успішні ІТ-проекти, розробляються функціональні та нефункціональні вимоги до програмного забезпечення.

Розділ "МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ" присвячений моделюванню та аналізу програмного забезпечення, опису архітектури програмного забезпечення, конструюванню програмного забезпечення та аналізу безпеки даних.

В розділі "АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ" викладено аналіз якості програмного забезпечення, опис процесів тестування та контрольного прикладу.

"ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ" - розділ, який описує процес розгортання та підтримки програмного забезпечення.

КЛЮЧОВІ СЛОВА: IDE, EKG, DJANGO, БІБЛІОТЕКА, MITDB, RANDOMFORESTCLASSIFIER, GRADIENTBOOSTINGCLASSIFIER, БАЗА ДАНИХ, WORD2VEC.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 21 tables, 41 figures, and 70 sources – a total of 93 pages. The diploma project is dedicated to enhancing and expanding the tools for automatic electrocardiogram analysis by developing a software library that implements ECG analysis using linguistic methods. The goal is to enhance and expand tools for automatic electrocardiogram analysis by developing a software library that implements ECG analysis using linguistic methods.

The object of the study: ECG signals and their automatic analysis methods.

The subject of research: techniques for modeling and analyzing ECG using linguistic methods, as well as the implementation and validation of a software library for this purpose.

The first important component of the project is Modeling. In this section, on the frontend, users can input training parameters such as the number of clusters for Kmeans, Word2Vec vector size, as well as choose a heartbeat classification algorithm (such as random forest, svc, etc.) and set specific classifier parameters. After the user initiates the process, the model is trained and evaluated on the backend. Then the backend responds to the frontend with classifier model metrics (accuracy, precision, recall, F1 score), the k-means silhouette score, and four charts (three t-SNE plots with clustered P, QRS, and T heartbeat waves and one plot - a confusion matrix for the classification model). All this information is then presented to the user.

The second important component is Analysis. Here, on the frontend, users can upload their own csv/xlsx files with ECG data, set fs values, and initiate the analysis. The backend uses all trained algorithms to classify heartbeats in the uploaded file and returns a chart to the frontend, where all heartbeats are labeled (normal, abnormal, or other classes). This chart is displayed for the user. In addition, users can hover over each heartbeat on the chart and see the predicted label (normal, abnormal, or other classes). Below the chart with heartbeats and predicted labels, a table is displayed showing the number of heartbeats, the heartbeat label, the 3-letter name of the heartbeat

(three letters because we have three clustered waves, and the number of clusters is converted into letters), and the vector representation of the heartbeat.

In the section "ANALYSIS OF SOFTWARE REQUIREMENTS", the main provisions and substantive description of the subject area are considered, existing technologies and successful IT projects are analyzed, functional and non-functional requirements for the software are developed.

The section "MODELING AND CONSTRUCTION OF SOFTWARE" is dedicated to the modeling and analysis of software, description of the software architecture, construction of software, and analysis of data security.

In the section "QUALITY ANALYSIS AND TESTING OF SOFTWARE", the quality analysis of software, description of testing processes and the control example are presented.

"IMPLEMENTATION AND MAINTENANCE OF SOFTWARE" is a section that describes the deployment and support process of software.

KEYWORDS: IDE, EKG, DJANGO, LIBRARY, MITDB, RANDOMFORESTCLASSIFIER, GRADIENTBOOSTINGCLASSIFIER, DATABASE, WORD2VEC.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ” _____ 2023 р.

**ПРОГРАМНА БІБЛІОТЕКА АНАЛІЗУ ЕКГ ЛІНГВІСТИЧНИМИ
МЕТОДАМИ**

Технічне завдання

КПІ.ПІ-9202.045420.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Юрій ОЛІЙНИК

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Еліза БЕРАУДО

Київ – 2023

ЗМІСТ

1	НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ.....	3
2	ПІДСТАВА ДЛЯ РОЗРОБКИ.....	4
3	ПРИЗНАЧЕННЯ РОЗРОБКИ.....	5
4	ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	6
4.1	Вимоги до функціональних характеристик	6
4.1.1	Користувацького інтерфейсу.....	6
4.1.2	Для користувача:.....	8
4.1.3	Для адміністратора системи (якщо він передбачений):	8
4.1.4	Додаткові вимоги:.....	8
4.2	Вимоги до надійності	9
4.3	Умови експлуатації.....	9
4.3.1	Вид обслуговування	9
4.3.2	Обслуговуючий персонал	10
4.4	Вимоги до складу і параметрів технічних засобів	10
4.5	Вимоги до інформаційної та програмної сумісності	10
4.5.1	Вимоги до вхідних даних.....	10
4.5.2	Вимоги до вихідних даних	10
4.5.3	Вимоги до мови розробки.....	11
4.5.4	Вимоги до середовища розробки	11
4.5.5	Вимоги до представленню вихідних кодів	11
4.6	Вимоги до маркування та пакування.....	11
4.7	Вимоги до транспортування та зберігання	11
4.8	Спеціальні вимоги	11
5	ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ	12
5.1	Попередній склад програмної документації	12
5.2	Спеціальні вимоги до програмної документації	12
6	СТАДІЇ І ЕТАПИ РОЗРОБКИ.....	13
7	ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ	14

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Сервіс аналізу ЕКГ (ECG Analysis Service).

Галузь застосування: Медицина, спеціалізація – кардіологія.

Наведене технічне завдання поширюється на розробку програмного забезпечення "Сервіс аналізу ЕКГ", котрий використовується для аналізу електрокардіограм (ЕКГ) та призначений для використання кардіологами, медичними дослідниками, а також будь-якими користувачами, що мають інтерес до аналізу ЕКГ.

					КП.ІП-9202.045420.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		3

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки "Сервісу аналізу ЕКГ" є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

					КП.ІП-9202.045420.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		4

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматизації та полегшенню процесу аналізу електрокардіограм (ЕКГ) на основі передових методів машинного навчання та класифікації даних.

Мета – підвищення та розширення інструментів для автоматичного аналізу електрокардіограм шляхом розробки програмної бібліотеки, що реалізує аналіз ЕКГ за допомогою лінгвістичних методів.

					КП.ІП-9202.045420.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		5

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Користувацького інтерфейсу

- введення параметрів моделі для тренування;
- вибір алгоритму класифікації;
- завантаження файлів форматів csv/xlsx для аналізу;
- візуалізація результатів аналізу ЕКГ у формі графіків та діаграм.

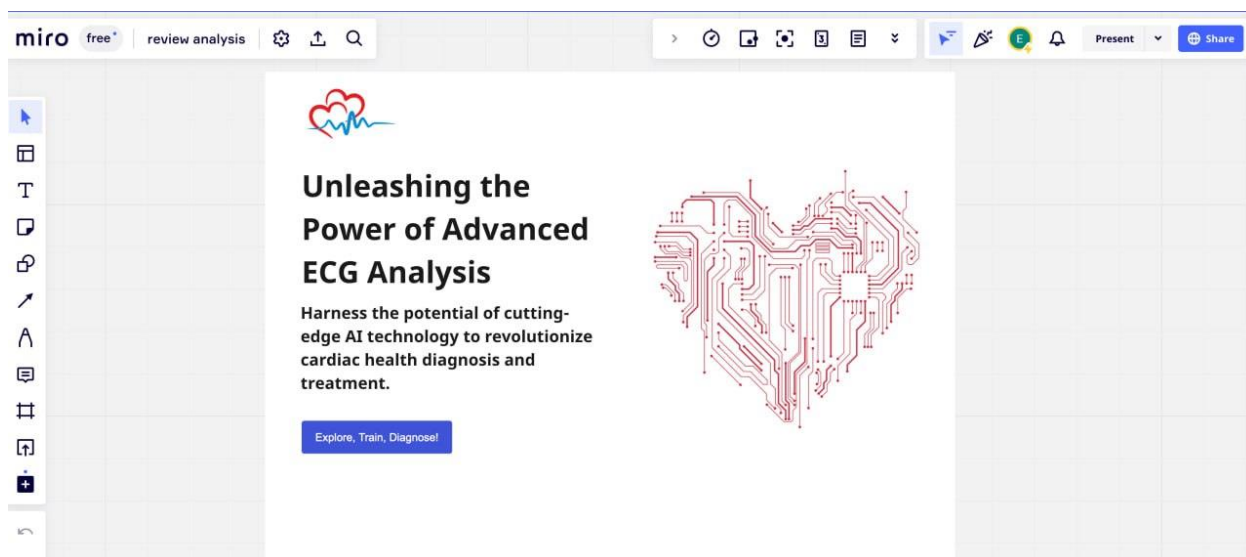


Рисунок 4.1 – Відображення головної сторінки

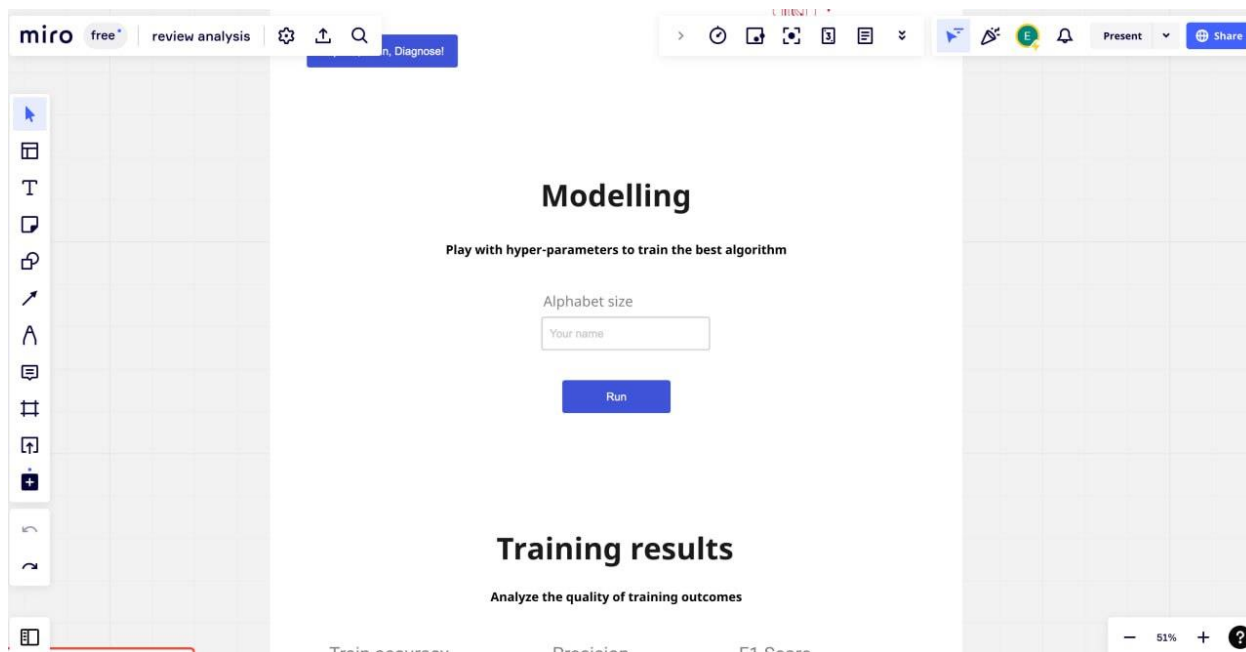


Рисунок 4.2 – Відображення введення кількості літер в алфавіті

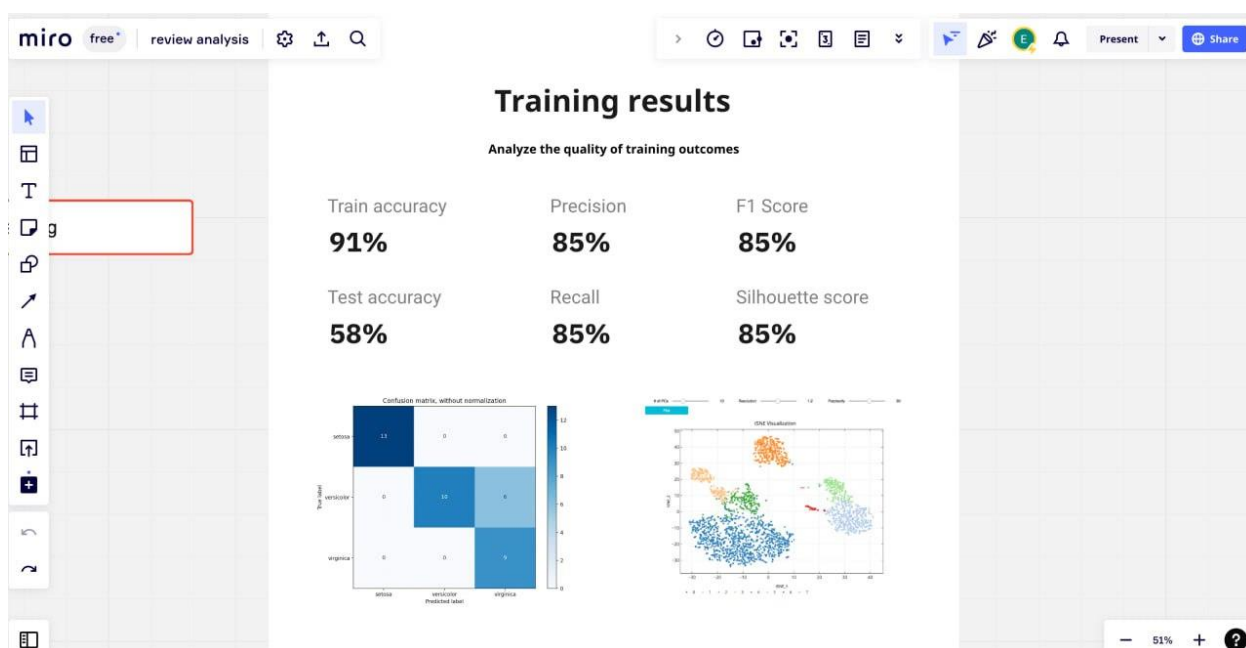


Рисунок 4.3 – Відображення результатів тренування моделі

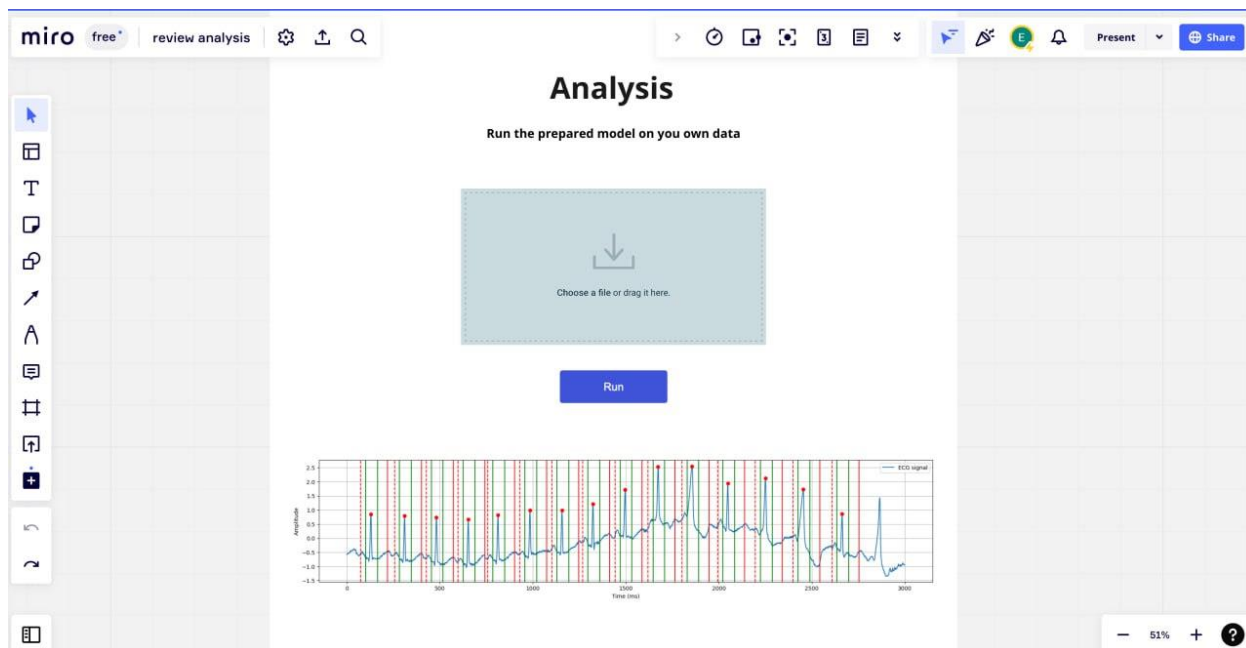


Рисунок 4.4 – Відображення кардіограми

4.1.2 Для користувача:

- можливість вибору параметрів для моделі;
- можливість завантажувати власні файли для аналізу;
- можливість перегляду результатів аналізу у вигляді візуалізованих графіків та діаграм.

4.1.3 Для адміністратора системи (якщо він передбачений):

- функція моніторингу статусу сервера;
- функція оновлення бази даних;
- функція резервного копіювання даних. функція 2 і т. д.

4.1.4 Додаткові вимоги:

- надійний захист даних користувача;
- високий рівень стабільності та надійності сервісу.

4.2 Вимоги до надійності

Програмне забезпечення має передбачати різні механізми, що підвищують його надійність:

Контроль введення інформації: Система має забезпечувати перевірку вхідних даних на валідність та коректність перед їх обробкою і збереженням.

Захист від некоректних дій користувача: Система повинна забезпечувати захист від ненавмисних або злонамірених дій користувача, які можуть призвести до збою або порушення інтегральності даних.

Цілісність даних: У випадку, якщо в системі передбачена наявність бази даних, необхідно забезпечити механізми, які гарантують інтегральність даних в БД при будь-яких умовах.

Відновлення після збоїв: Система повинна мати можливість відновитися після збоїв і продовжити нормальну роботу без втрати даних. Для цього може бути реалізований механізм контрольних точок, який періодично зберігає стан системи, а також резервне копіювання даних.

Час відновлення системи: Час, необхідний для відновлення роботи системи після збою, повинен бути мінімізований. Він залежить від конкретної реалізації механізмів відновлення системи, але в ідеалі не повинен перевищувати декількох хвилин.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Розробка не вимагає спеціального обслуговування. Вимоги до виду обслуговування не висуваються.

					КП.ІП-9202.045420.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		9

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються, оскільки розробка не потребує обслуговуючого персоналу.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно бути сумісне з IBM-сумісними персональними комп'ютерами.

Мінімальна конфігурація технічних засобів:

- тип процесору: Intel Core i5;
- об'єм ОЗП: 4 Гб;
- підключення до мережі Інтернет зі швидкістю від 20 мегабіт;

Рекомендована конфігурація технічних засобів:

- тип процесору: Intel Core i7;
- об'єм ОЗП: 8 Гб;
- підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows'XP, Windows NT і т.д.) або Unix.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: JSON або XML.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі: JSON або XML.

					КПІ.ІП-9202.045420.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		10

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування Python

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі Node.js для JavaScript, або Python 3.7 або вище для Python.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді текстових файлів з розширенням .js для JavaScript або .py для Python.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Розгорнути програмне забезпечення в Інтернеті для публічного доступу.

					КП.ІП-9202.045420.01.91	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		11

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

- пояснювальна записка;
- технічне завдання;
- текст програми;
- програма та методика тестування;
- керівництво користувача;
- керівництво програміста;
- керівництво системного програміста;

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

- схема структурна варіантів використання;
- схема структурна структурна бізнес процесу;
- схема структурна класів програмного забезпечення;
- креслення вигляду екранних форм;

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	21.02	
2.	Розробка технічного завдання	03.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.03	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.03	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.04	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.04	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.04	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	20.04	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.04	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

					КП.ІП-9202.045420.01.91	Арк.
						14
Змін.	Арк.	№ докум.	Підп.	Дата.		

**Пояснювальна записка
до дипломного проєкту**

на тему:

Програмна бібліотека аналізу ЕКГ лінгвістичними методами

КПІ.ПІ-9202.045420.02.81

Київ – 2023

ЗМІСТ

1	АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	7
1.1	Загальні положення	7
1.2	Змістовний опис і аналіз предметної області	13
1.3	Аналіз наявних технологій та успішних ІТ-проєктів	16
1.3.1	Аналіз відомих алгоритмічних та технічних рішень	17
1.3.2	Аналіз допоміжних програмних засобів та засобів розробки	19
1.3.3	Аналіз відомих програмних продуктів	20
1.4	Аналіз вимог до програмного забезпечення	23
1.4.1	Розроблення функціональних вимог	28
1.4.2	Розроблення нефункціональних вимог	34
1.5	Постановка задачі	35
	Висновки до розділу	36
2	МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	37
2.1	Моделювання та аналіз програмного забезпечення	37
2.2	Архітектура програмного забезпечення	38
2.3	Конструювання програмного забезпечення	42
2.4	Аналіз безпеки даних	47
	Висновки до розділу	48
3	АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	50
3.1	Аналіз якості ПЗ	50
3.2	Опис процесів тестування	64

3.3	Опис контрольного прикладу	71
	Висновки до розділу	80
4	ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	82
4.1	Розгортання програмного забезпечення	82
4.2	Підтримка програмного забезпечення	88
	Висновки до розділу	88
	ВИСНОВКИ	90
	СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	92
	ДОДАТОК А ЗВІТ ПОДІБНОСТІ	99

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
ЕКГ	–	Електрокардіограма.
Django	–	Високорівневий веб-фреймворк на Python, що сприяє швидкому розробленню та чистому коду.
Бібліотека	–	Набір функцій, класів та методів, які можна використовувати в програмному проекті.
MITDB	–	База даних Массачусетського технологічного інституту з записами ЕКГ.
RandomForest Classifier	–	Класифікатор випадкового лісу, модель машинного навчання, що використовує ансамбль дерев рішень.
GradientBoostingClassifier	–	Класифікатор градієнтного бустінгу, модель машинного навчання, що покращується шляхом комбінування слабких класифікаторів.
БД	–	База даних.
Word2Vec	–	Модель машинного навчання, що перетворює слова на вектори, використовуючи нейромережі для аналізу текстів.

ВСТУП

Серцево-судинні захворювання залишаються однією з найважливіших причин смертності та інвалідності у світі. Серед найпоширеніших і корисних методів діагностики захворювань серця електрокардіограма (ЕКГ) є одним з найпопулярніших. Це дає можливість своєчасного виявлення та планового лікування захворювань серця. Ручний аналіз ЕКГ іноді займає багато часу та є суб'єктивним, тому потрібні нові підходи до автоматизації. [1]

Серед перспективних підходів у галузі автоматизованого аналізу ЕКГ – лінгвістичні методи. Цей підхід дозволяє швидко й ефективно виявляти аномалії ЕКГ без потреби у великих обсягах пам'яті чи комп'ютерних ресурсах. Можна швидко й ефективно виявити аномалії ЕКГ, не вимагаючи великих обсягів пам'яті чи комп'ютерних ресурсів. Завданням цієї курсової роботи є розробка програмної бібліотеки для аналізу ЕКГ, що використовується лінгвальними методами, та створення програми, яка буде базуватися на фреймворку Django[2]. Метою цієї курсової роботи є розробка бібліотеки програмного забезпечення для бази даних аналізу ЕКГ за допомогою лінгвальної методології та створення веб-додатку на основі. Новий додаток покаже результати аналізу та дозволить медичним працівникам завантажити файли даних ЕКГ для подальшого аналізу. Додаток також відображатиме результати аналізу та дозволить медичним працівникам завантажувати файли даних ЕКГ для подальшого аналізу.

Лінгвістичні методи аналізу ЕКГ дозволяють ефективно виявляти відхилення від норми і дають можливість зосередитися на критичних зонах. Застосування лінгвістичних методів аналізу ЕКГ дозволяє ефективно виявляти відхилення від норми та дає можливість зосередитися на важливих напрямках. Це забезпечує більш точний і своєчасний аналіз ЕКГ, знижує навантаження на медичний персонал. Такий підхід використовується для покращення догляду за пацієнтами та оптимального планування лікування.

Основними етапами проекту є: створити ефективний сервіс для аналізу електрокардіограм (ЕКГ), використовуючи передові техніки обробки даних. Цей сервіс включає в себе два основних компоненти: моделювання та аналіз.

Компонент моделювання дозволяє користувачам вводити параметри тренування, вибирати алгоритми класифікації серцебиття, а потім переглядати метрики моделі та графіки, що були згенеровані під час тренування та оцінки моделі.

З іншого боку, компонент аналізу дозволяє користувачам завантажувати власні файли ЕКГ для аналізу. Використовуючи навчені моделі, сервіс класифікує серцебиття в файлі користувача та повертає графік, де всі серцебиття марковані як нормальні, ненормальні або інші класи.

Такий інноваційний підхід до аналізу ЕКГ може мати значний вплив на медичну галузь, прискорюючи та поліпшуючи процеси діагностування. Ця робота спрямована на опис розробки цього сервісу, а також на аналіз його функціональності та ефективності.

					КПІ.ІП-9202.045420.02.81	6
Змін.	Арк.	№ докум.	Підп.	Дата.		

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Сигнали ЕКГ відіграють важливу роль у діагностиці та моніторингу серцево-судинної системи людини. Крім того, сигнали ЕКГ важливі у діагностиці та моніторингу людини. Короткий опис основних призначень сигналів ЕКГ можна представити так:

Електрокардіограма (ЕКГ) - це криві або, якщо формально, часовий ряд з точною періодичністю в часі, який відображає зміну різниці потенціалів від двох різних точок на одному тілі та внаслідок електричного процесу, наприклад: він відображає зміну різниці енергій, яка виникає внаслідок електронної обробки, джерелом яких є працююче серце людини [3]

Відведення і групи відведень

Часовий ряд, що характеризує роботу провідної системи організму, можна записати у вигляді різниці потенціалів у будь-яких двох точках, на різних відстанях від серця. Тож можна судити про роботу свого серця за цим часовим рядом. Можна визначити тахікардію чи брадикардію в широкому сенсі. Наприклад, можна дізнатися, чи є тахікардія або брадикардія. Однак з'ясувати з максимальною точністю, що стало причиною або уточнити патологію, вже неможливо. Визначено спосіб розміщення зчитувальних електродів, які вже дозволяють з більшою точністю оцінити роботу серця: з'ясувати місце патології, її характер тощо.

Поширеним варіантом кардіографа є прилад, який фіксує різницю потенціалів у дванадцяти парах точок у часі [4]. Відведення є прикладом набору вимірювань, які виконуються однією парою точок, а його результатами є 12 загальноприйнятих відведень. Група відведень поділяється на групи за типом електродів і їх розташуванням. Розглянемо їх докладніше.

Ведення від кінцівок (стандартні)

Приклад стандартних відведень, запропонованих В. Ейнтховеном (рисунок.1.1) [5], їх три і вони позначені римськими цифрами I. II. III (1.1) 1.1)

					КПІ.ПІ-9202.045420.02.81	
Змін.	Арк.	№ докум.	Підп.	Дата.		7

У разі розміщення реєструючих електродів на правій та лівій руці прийнято, що для I стандартного відведення приймається їх розташування на правій руці та лівій нозі, для II - на правій руці та лівій нозі, у III - на ліва рука і ліва нога. [6] Пряма, що з'єднує розташування двох електродів з різною полярністю, також вважається «віссю» цього відведення.

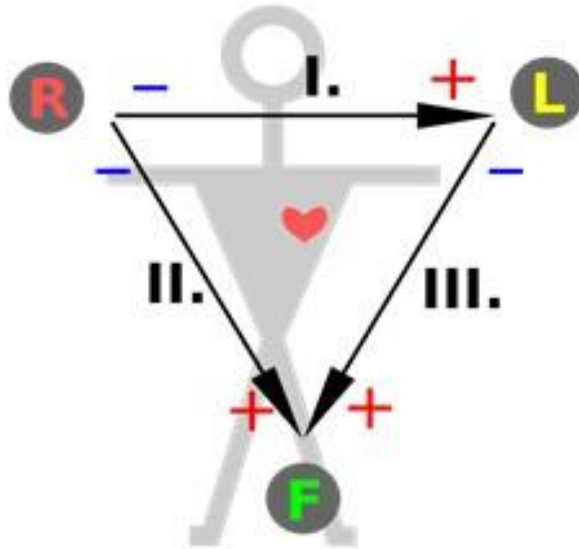


Рисунок 1.1 – Стандартні відведення Ейнтховена

Підсилені відведення

Підсилені відведення отримали свою назву тому, що рівень вихідного сигналу значно нижчий, ніж у стандартних відведеннях і для зручності розшифровки було прийнято рішення його підсилити. Підсилені відведення також, як і стандартні, три. Електроди по своїй черзі встановлюються так:

- а) aVR – права рука та об'єднаний електрод лівої ноги і лівої руки;
- б) aVL – ліва рука та об'єднаний електрод з лівої ноги і правої руки;
- в) aVF – ліва нога та об'єднаний електрод з лівої та правої руки.

Приклад підсиленіх відведень Гольдбергера представлений на рисунку 1.2 [9]. Підсилені відведення послідовно знімаються з кожної осі трикутника Ейнтховена.

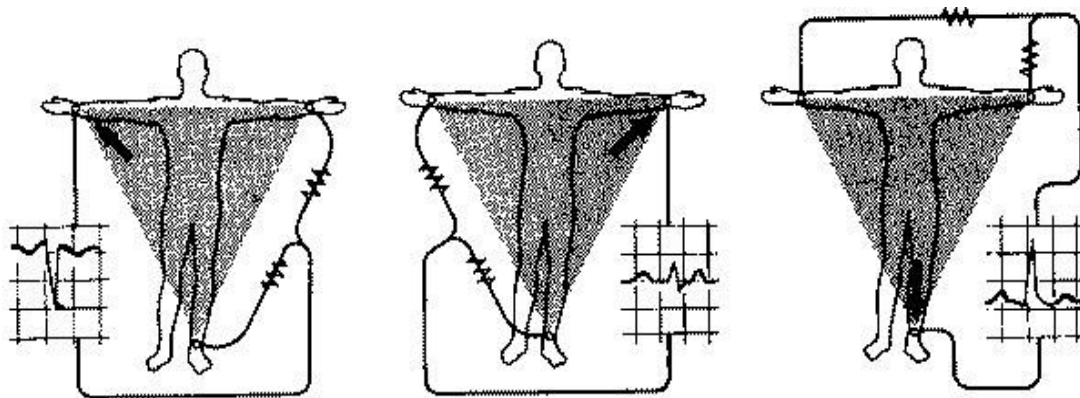


Рисунок 1.2 – Підсилені відведення Гольдберга

Грудні відведення

Грудні відведення реєструють зміни електрорушійної сили горизонтальної площини, на відміну від інших подібних відведень, описаних вище. Грудні відведення реєструють зміни електрорушійної сили в горизонтальній площині. Через усю різноманітність найчастіше для первинної діагностики використовуються відведення Вільсона (рисунок. 1.3) [7]:

- а) V1 – четверте міжребер'я біля груднини, попереду від правого краю;
- б) V2 – четверте міжребер'я біля лівого краю груднини;
- в) V3 – посередині між електродами V1 і V2;
- г) V4 – п'яте міжребер'я по лівій середньоключичній лінії;
- г) V5 – по лівій передній пахвовій лінії;
- д) V6 – збільшений на рівні електрода V4 по аксиллярній лінії зліва.

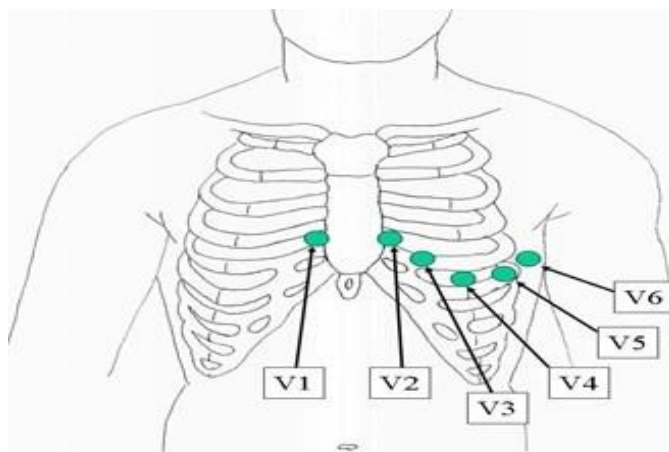


Рисунок 1.3 – Грудні відведення Вільсона

Часто виникають труднощі з встановленням торакальних відведень, показаних на рисунку 1.3, через різну структуру тіла та неможливості їх встановлення однаковим способом у всіх пацієнтів.

На рисунку 1.4 [8] ілюструє, що якщо зрушити електрод з його місця, через нього будуть проходити інші лінії живлення, а отже, зміниться рівень сигналу у відповідному відведенні. Саме це є однією з причин різниці в результатах опитування.

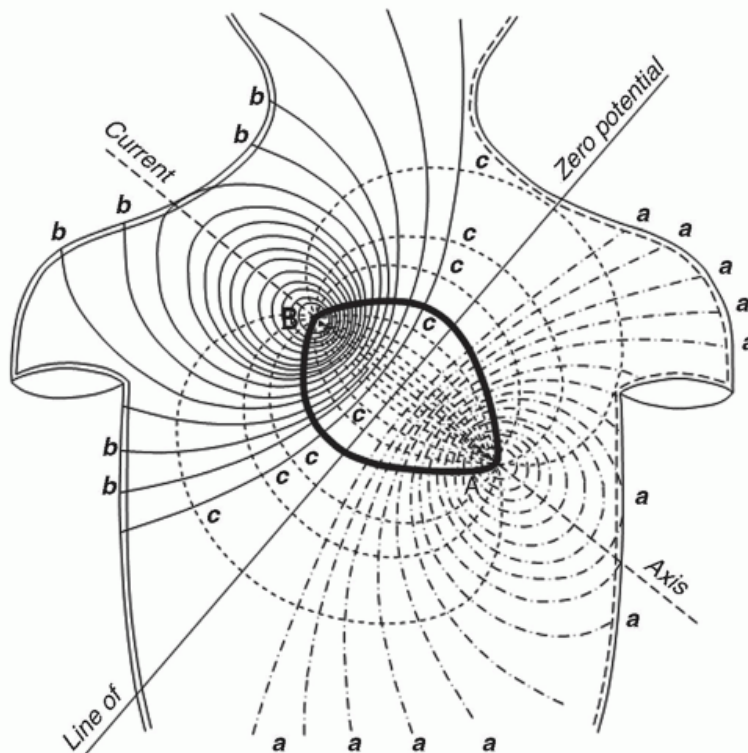


Рисунок 1.4 – Електричне поле серця за схемою Уоллера

На рисунку 1.5 [9] показано 12 відведень, працюючи з якими лікар функціональної діагностики виявляє патології та ставить відповідний діагноз. Саме цей приклад наочно показує різний рівень сигналів. Наприклад, на малюнку 1.5 зображена картина ЕКГ здорової людини, можна очікувати більшої різниці в результатах індивіда з патологічними змінами.

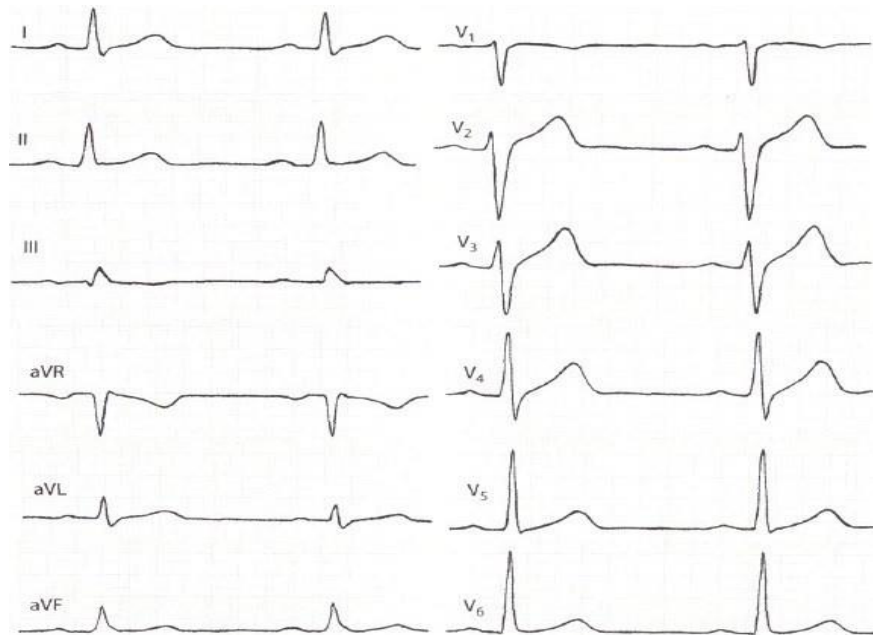


Рисунок 1.5 – Відмінності відведень одне від одного

Складові одного відведення

В нормі одне відведення (рис. 1.6) [10] складається з зубців різної форми та відрізків ізолінії. Зубці, спрямовані вгору, називаються позитивними, вниз – негативними. В нормальних умовах зубці бувають 5 видів:

- а) Р – невисокий, закруглений, в деяких випадках видно ямку посередині;
- б) Q – невеликий гострий;
- в) R – великий і гострий;
- г) S – гострий, відносно невеликий;
- г) Т – закруглений, схожий з зубцем Р.

На рисунку 1.6 присутня хвиля U. Вона відповідає процесу реполяризації передсердь. У більшості випадків на результатах обстеження ця хвиля зливається з шумом і слабо розрізняється. В умовах представленої роботи факт її наявності не враховується.

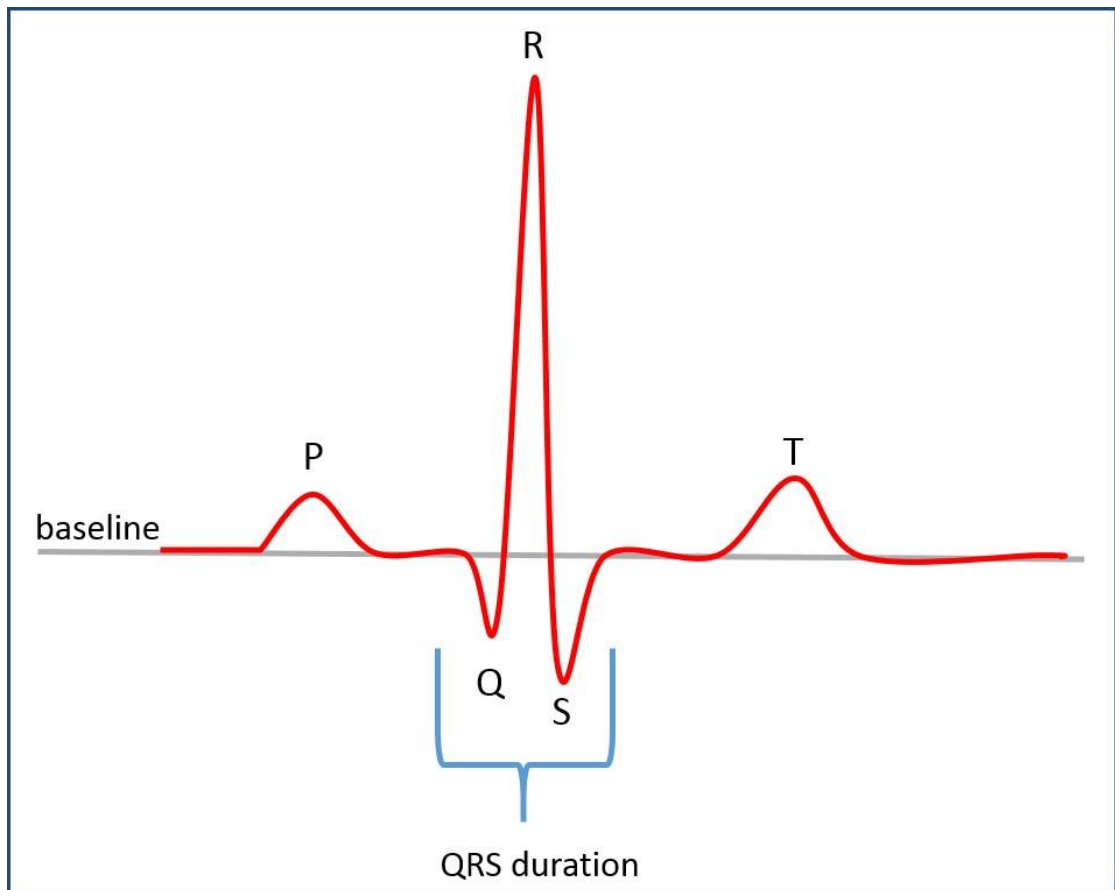


Рисунок 1.6 – Схема нормальної ЕКГ людини

Зубець Р фіксується на початку серцевого скорочення при збудженні передсердь, викликаному роботою синусового вузла. Спочатку збуджується ліве передсердя, а потім - праве. Оскільки цей процес відбувається хвильовим чином, поступово захоплюючи м'язи цієї області серця, утворюючийся зубець схожий на хвилю. Різні патології передсердь обов'язково відображаються на стані цієї ділянки кардіограми.

Зубець Т при порушеннях може згладжуватися, збільшуватися, набувати трикутної форми або навіть перекидатися відносно середньої лінії кардіограми.

Потім збудження переходить до атріовентрикулярного вузла, розташованого в середині серця. Відмінна риса цієї області - здатність затримувати проходящий сигнал. Через різке зниження швидкості проходження імпульсу на кардіограмі в цей період з'являється рівний ділянка, що слідує за зубцем Р.

Відхилення від прямої лінії в цій зоні вказує на порушення проходження імпульсу по атріовентрикулярному вузлу (АВ-вузлу).

Зубець Q. Потім починається збудження шлуночків. Нервовий імпульс від АВ-вузла проходить вниз до пучка Гіса і переходить на його ніжки. Такий період відзначається на кардіограмі невеликим від'ємним зубцем Q.

Зубці R і S. У подальшому імпульс доходить до нижнього полюса серця, і відбувається скорочення шлуночків, при якому кров з них викидається в передсердя. Оскільки серцевому м'язу доводиться витискувати достатньо великий об'єм крові, такий процес супроводжується вираженим електричним імпульсом, і на ЕКГ з'являється різко виражений гострий зубець R. Потім збуджуються кінчики ніжок пучка Гіса, що викликає утворення невеликого зубця S.

Особливості сегменту ST. Після повної фази скорочення серцю потрібно відпочити. Однак збудження проходить не відразу, затримуючись на якийсь час в провідній системі міокарда. В цей момент в серцевих м'язах спостерігається відразу два процеси – збудження і гальмування, які урівноважують один одного. Так з'являється плоский сегмент ST.

При деяких захворюваннях серця він може подовжуватися і набувати хвильової форми.

Зубець T. В кінці серцевого скорочення відбувається відновлення шлуночків. Цей процес називається реполяризацією – на кардіограмі йому відповідає хвильовий зубець T.

При розшифровці ЕКГ спеціаліст вимірює висоту (вольтаж) зубців, а також їх тривалість в секундах. Вимірюються і показники найважливіших інтервалів. Обов'язково робиться акцент на змінених зубцях. Потім ці дані використовуються для розшифровки ЕКГ з метою виявлення кардіопатологій.

1.2 Змістовний опис і аналіз предметної області

Термін "хвороба серця" відноситься до кількох типів захворювань серця. Найпоширеніший тип хвороби - це коронарна артеріальна хвороба (КАГ), яка

впливає на кровотік до серця. Зменшений кровотік може спричинити серцевий напад. [11]. Електрокардіограма реєструє електричні сигнали у серці. Це загальноприйнятий і безболісний тест, який використовується для швидкого виявлення проблем зі здоров'ям серця та моніторингу його стану [12]. Електрокардіограма - безболісний та неінвазивний спосіб діагностики багатьох поширених серцевих проблем. Медичний спеціаліст може використовувати електрокардіограму для визначення або виявлення серцевих захворювань.

Поновлений інтерес до машинного навчання зумовлений більшими обсягами та різноманітністю доступних даних та доступністю сховища даних [13]. В більшості сучасних трекерів для врахування природних змін зображення, зазвичай, використовується класифікатор, який навчається на перекладених та масштабованих зразках. Ми пропонуємо аналітичну модель для наборів даних з тисяч перекладених зразків. Показавши, що отримана матриця даних є циркулянтною, ми можемо діагоналізувати її за допомогою дискретного перетворення Фур'є, зменшивши обсяги зберігання та обчислень на кілька порядків [14].

Логістична регресія та штучні нейронні мережі є моделями вибору в багатьох завданнях класифікації медичних даних [15]. Логістична регресія є одним з найважливіших аналітичних інструментів у соціальних та природничих науках. У обробці природньої мови логістична регресія є базовим алгоритмом наглядного машинного навчання для класифікації, а також має дуже близький зв'язок з нейронними мережами [16]. Алгоритм машинного навчання нейронні мережі довів свою точність та надійність і, отже, використовується в запропонованій системі [17].

З використанням технік машинного навчання метою цього дослідження є оцінка точності навчання з вчителем в передбаченні хвороби серця на основі набору даних, отриманого з репозиторію даних Університету Каліфорнії, Ірвайн. Результати цього дослідження показують, що Наївний Байес і Байєсівська мережа мають кращу приблизну точність у Weka для набору

даних, тоді як обидва методи можуть дати корисну інформацію за допомогою візуалізації, згенерованої за допомогою Weka [18].

Аналіз ЕКГ-сигналу відіграє важливу роль у діагностиці серцевих захворювань. У цій роботі запропоновано ефективний метод аналізу ЕКГ-сигналу та передбачення відхилень серця. У запропонованій схемі спочатку забруднений ЕКГ-сигнал відфільтровується шляхом відкидання фонового шуму [19].

У запропонованому методі були виділені ознаки та тривалість ЕКГ-сигналів, такі як комплекс QRS, інтервали R-R, з шумних сигналів за допомогою алгоритму Пана-Томпкінса. Для аналізу використовувалися дані, зібрані з бази даних MIT-BIH аритмій. Оцінювані ознаки були визначені зі стандартним набором значень на основі розробленого алгоритму прийняття рішень для визначення ступеня та типів аритмій [20].

Підхід з одним класом зміг виявити аномалії з площею під кривою (AUC) 0,83 і з точністю 75,6%. Для класифікації чотирьох класів ми використовували загалом 86 ознак, додатково витягнутих з ЕКГ. Точність цього класифікатора для чотирьох класів досягла 75,1%. Методи продемонстрували принципову можливість виявлення відхилень серця за допомогою машинного навчання у великому когортному дослідженні.

Крім того, в реальних умовах ЕКГ може передаватися у вигляді файлів з послідовністю значень для подальшого аналізу в програмі. Один із варіантів вводу даних ЕКГ, який використовують лікарі, полягає у використанні файлів знімків ЕКГ, подібних до тих, що показані на рисунку 1.7.

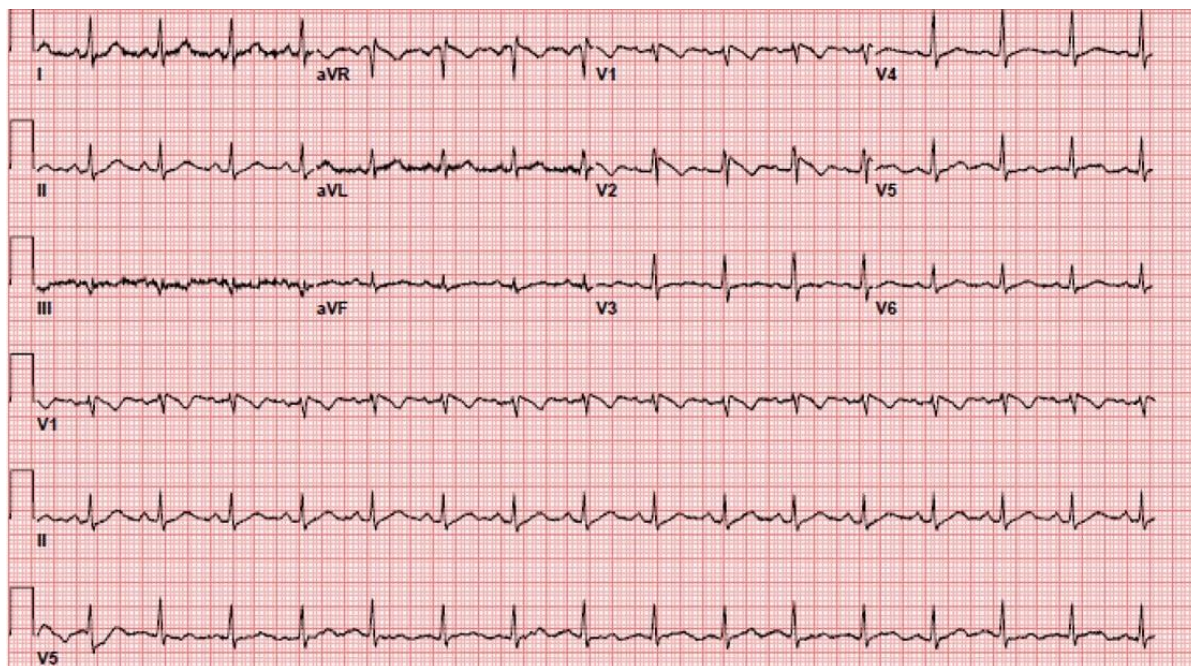


Рисунок 1.7 – Ілюстративний приклад зображення ЕКГ

Для точного виявлення аномалій та аналізу результатів ЕКГ важливо мати велику базу даних з різними випадками та ситуаціями. Оскільки серцеві аномалії можуть мати різні форми та прояви, відмінний відбір даних допоможе у розробці більш ефективних методів аналізу та виявлення аномалій.

У світлі цього, набір даних MIT-BIH є цінним джерелом для дослідження та розробки нових методів виявлення аномалій в ЕКГ. Використання різних алгоритмів, таких як машинне навчання та штучний інтелект, може допомогти виявити складні та неочевидні відхилення в серцевому ритмі, які можуть виявитися корисними для раннього діагностування та лікування серцевих захворювань.

Аналіз маркованих даних ЕКГ на основі різних наборів даних, таких як MIT-BIH, дозволяє розробити ефективні методи виявлення аномалій та забезпечити більш точне виявлення аномалій у серцевому ритмі пацієнтів.

1.3 Аналіз наявних технологій та успішних ІТ-проектів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації програмного

забезпечення аналізу кодування сигналу ЕКГ лінгвістичними методами. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

Ця робота веде наукове дослідження з метою вивчення ефективності лінгвістичного підходу до аналізу даних ЕКГ. Тому було б вдалим провести огляд наукових робіт у цій області та оцінити їх результати.

Було проведено численні дослідження для прогнозування та виявлення серцевих захворювань за допомогою машинного навчання та глибокого навчання. Джерело [21] показує різноманітні застосування технік добування даних в галузі охорони здоров'я. Розповсюдження інфекційних захворювань можна контролювати, а автоматизовані системи спостереження виявилися більш ефективними. Таким чином, було зроблено висновок, що машинне навчання має великі перспективи в медичній галузі. [22] розробляє систему, яка витягує будь-яку приховану інформацію або знання з історичної бази даних серцевих захворювань. Моделі оцінюються за допомогою підходів Lift Chart та Classification Matrix. Використовувалась модель Naive Bayes, яка показала точність 86%. Ця система була розроблена для допомоги лікарям приймати кращі рішення та зменшувати медичні помилки.

Nikhar та Karandikar [23] вивчали прогнозування серцевих захворювань за допомогою класифікаторів Decision Tree та Naive Bayes. Класифікатор Decision Tree мав кращу точність порівняно з Naive Bayes через розрахунок інформаційного приросту. Ефективність класифікатора Naive Bayes можна було збільшити за допомогою класифікатора Selective Naive Bayes. Робота в [24] показує порівняння між Naive Bayes, Support Vector Machine, Artificial Neural Networks, Simple Logistic Regression та Random Forest, і вони показали точність 86,4%, 97,53%, 77,89%, 95,05% та 95,67% відповідно. Також були розраховані точність, повернення та чутливість, і SVM завжди показувала найкращі результати. Була розроблена система моніторингу в реальному часі

за допомогою Arduino, монітора відображення, бузера, різних біомедичних датчиків тощо, які приймали стать, вік, рівень цукру в крові, кров'яний тиск, тип грудних болів, серумний холестерин тощо як вхід і за допомогою алгоритму SVM виявляли наявність будь-якого серцевого захворювання. В [25] Рамалінгам провів огляд різних алгоритмів машинного навчання, таких як Naive Bayes, K – Nearest Neighbour, Support Vector Machine, Decision Tree та Random Forest, та проаналізував їхню ефективність. Було зроблено висновок, що алгоритми машинного навчання мають великий потенціал для прогнозування серцево-судинних порушень. У деяких випадках дерева рішень виконувалися надзвичайно погано, що могло бути викликано перенавчанням. Моделі Random Forest відзначалися винятковою ефективністю, оскільки вони вирішують проблему перенавчання за допомогою кількох алгоритмів. Моделі на основі класифікатора Naive Bayes виявилися ефективними та обчислювально швидкими. В основному SVM виконувалася добре.

У [26] були проаналізовані алгоритми машинного навчання, використані для прогнозування серцевих захворювань, та програмування на Python. K-Nearest Neighbor, Decision Tree Classifier, Random Forest Classifier, Correlation, SVM були проаналізовані та показали точність 63,4%, 71,4%, 68,4% та 86,2% відповідно. Shashwith та ін. [27] представили дослідження різних алгоритмів машинного навчання, таких як Naive Bayes, Decision Tree, Logistic Regression та Support Vector Machine з точністю тестування 49,18%, 81,97%, 88,52% та 90,16% відповідно, використовуючи набір даних UCI machine learning repository. Робота в [28] показує класифікацію різних серцевих захворювань за допомогою ECG-зображень на основі 12 ведень, розділених на чотири класи. SSD MobileNet V2 використовувався для навчання детектора, який мав точність 98,33%. В [29] Анвар та ін. обговорювали вплив збільшення зображень на ECG-зображення. Efficient Net V3 використовувався для класифікації зображень на чотири класи: COVID, Normal, Myocardial infarction, та history of Myocardial Infarction. Найвища точність була досягнута з оптимізатором AdamW, який був 0,818, але це знизилося до 0,002, коли було

застосовано аугментацію перевероту. Оскільки техніки аугментації зображень використовуються для розширення даних, вони можуть спричинити спотворення прихованих шаблонів даних, що знижує продуктивність.

В [30] Аспуру та ін. запропонували метод на основі використання лінійної регресії для розподілу даних на періоди з метою визначення піків P, Q, S та T перед сегментацією сигналу для визначення точки R ECG-хвилі. Попередня обробка ECG-сигналу для видалення шуму дозволила системі ефективно розташувати ключові точки, що є важливим для діагностики серцевих проблем за допомогою класифікаторів машинного навчання. Було проаналізовано 260 сигналів ECG, та отримано середню чутливість 99,5% для ідентифікації всіх піків P, Q, R, S та T. Ачарія та ін. [31] створили глибоку CNN з чотирма 1D конволюційними шарами та трема повністю з'єднаними шарами для класифікації ECG-сигналів на п'ять класів. Загалом, глибоке навчання та машинне навчання виявилися дуже обіцяючими для діагностики та прогнозування серцевих захворювань.

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

В рамках проекту було використано ряд спеціалізованих інструментів, які сприяють ефективності розробки та спрощують процеси програмування, а саме: PyCharm IDEA[32], Colab[33], SQLite[34], Celery[35] та Redis[36].

PyCharm IDEA використовується як основне інтегроване середовище розробки (IDE). Це потужний інструмент для розробки на Python, що надає широкий спектр функціональності для зручного програмування та редагування коду.

Colab слугує онлайн-сервісом, який дозволяє розробляти та виконувати код на Python в хмарному середовищі. Використовуючи обчислювальні ресурси Colab, моделі машинного навчання можуть бути ефективно навчені без перевантаження локального обчислювального устаткування. Завдяки своїм можливостям для колаборативної роботи, Colab стає вкрай корисним для командної роботи над проектами.

SQLite вибрано для керування базами даних. Це вбудована система, що використовує мову SQL і не потребує встановлення додаткового серверного програмного забезпечення, що робить її зручною для невеликих до середніх проектів.

Celery використовується як асинхронна система черги завдань, що дозволяє оптимізувати виконання важких та періодичних задач, розподіляючи їх між розподіленими виконавцями.

Redis використовується як система зберігання даних ключ-значення в пам'яті. Завдяки високій швидкості та гнучкості, Redis є ідеальним для використання як база даних, кеш або брокер повідомлень.

Кожен з цих інструментів вносить вкрай важливий вклад в розробку проекту, сприяючи забезпеченню його високої продуктивності та ефективності.

1.3.3 Аналіз відомих програмних продуктів

Обробка ЕКГ сигналів завжди викликає великий інтерес у медичній галузі. Програмні продукти, як EcgViewer[37], CardioDay[38] і ADInstruments[39], стали незамінними інструментами для аналізу даних ЕКГ, що допомагають у виявленні та діагностиці серцевих хвороб спрямовані на підтримку цього процесу, надаючи потужні аналітичні засоби та зручність використання.

EcgViewer - це розширене програмне рішення, що надає набір інструментів для обробки та аналізу ЕКГ даних. Воно включає в себе автоматизовані функції, такі як виявлення пікових значень та аналіз інтервалів, що дозволяє лікарям отримувати точні та швидкі діагностики.

CardioDay - це інший важливий інструмент, що включає в себе набір засобів для моніторингу ЕКГ у реальному часі. Він надає змогу лікарям відстежувати стан серця пацієнтів на протязі дня, що є важливим для людей з хронічними серцевими захворюваннями.

ADInstruments - це більш комплексне рішення, що надає широкий спектр інструментів для аналізу ЕКГ, включаючи додаткові функції для наукових досліджень і клінічних випробувань. Воно включає в себе потужні аналітичні засоби, що дозволяють лікарям глибоко аналізувати ЕКГ дані і виявляти складні патології.

Всі ці програмні продукти відіграють важливу роль у сучасній медицині, допомагаючи лікарям використовувати ЕКГ дані для виявлення та лікування серцевих захворювань.

Для порівняння дипломної роботи з аналогом можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння з аналогом

Функціонал	Дипломний проєкт	Аналог для порівняння 1	Аналог для порівняння n	Пояснення
Реєстрація та вхід користувача	Запропоноване рішення	EcgViewer	CardioDay, ADInstruments	Усі проєкти пропонують безпечну систему реєстрації та входу для користувачів.
Завантаження та обробка зображення ЕКГ	Запропоноване рішення	EcgViewer	CardioDay, ADInstruments	Усі проєкти приймають завантаження зображень ЕКГ. Проте унікально використовує RGB2Gray, гауссове фільтрування, зміну розміру, порогову обробку для вилучення та контурні

Продовження таблиці 1.1

				методи для ідентифікації хвиль.
Аналіз ML Моделі	Запропоноване рішення	EcgViewer	CardioDay, ADInstruments	Дипломний проєкт використовує попередньо навчену ML модель для аналізу нормалізованого 1D сигналу. Ця функція присутня в усіх порівнюваних проєктах, але модель, яка використовується, та її точність можуть варіюватися.
Реальний зворотний зв'язок	Запропоноване рішення	EcgViewer	CardioDay, ADInstruments	Запропоноване рішення дає можливість користувачам переглядати весь робочий процес в інтерфейсі користувача та отримувати реальний

Продовження таблиці 1.1

				зворотний зв'язок. Ця можливість може бути відсутньою в інших програмах
--	--	--	--	---

1.4 Аналіз вимог до програмного забезпечення

Перша ключова компонента цього сервісу - моделювання. За допомогою інтерфейсу користувач може вводити параметри тренування, такі як кількість кластерів для алгоритму KMeans[40], розмір вектора Word2Vec[41], вибрати алгоритм класифікації серцебиття (наприклад, random forest, svc та інші) та встановити конкретні параметри класифікатора. Після натискання кнопки "Запустити", модель навчається та оцінюється на сервері, а потім сервер відправляє на клієнтську сторону метрики моделі класифікатора (точність, повнота, F1-міра) та показник силуету KMeans, а також чотири графіки: три графіки t-SNE для кластеризованих хвиль P, QRS та T та матрицю помилок для моделі класифікації. Всі ці дані відображаються користувачу.

Друга ключова компонента - аналіз. Користувач може завантажити власний csv/xlsx файл з ЕКГ, встановити значення частоти дискретизації та натиснути "Аналізувати". Потім сервер застосовує всі навчені алгоритми для класифікації серцебиття в файлі та повертає графік на клієнтську сторону, де всі серцебиття маркуються (нормальні, ненормальні або інші класи). Користувачу показується цей графік, а також таблиця з номером серцебиття, міткою серцебиття, трилітерним представленням серцебиття (три літери, оскільки ми маємо три кластеризовані хвилі, а кількість кластерів конвертується в літери) та векторним представленням серцебиття. Діаграма використання програми, наведена у графічному матеріалі до дипломного проєкту.

В таблицях 1.2 - 1.9 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 - Варіант використання UC-1

Use case name	Вибір параметрів
Use case ID	UC-01
Goals	Дозволити користувачам налаштовувати моделі тренування, вказуючи параметри.
Actors	Користувач
Trigger	Користувач вирішує навчити модель.
Pre-conditions	Користувач перебуває на сторінці навчання моделі.
Flow of Events	– Користувач переходить на сторінку навчання моделі. – Користувач вводить кількість кластерів для K-means. – Користувач вводить розмір вектора Word2Vec. – Користувач вибирає алгоритм класифікації серцебиття. – Користувач встановлює конкретні параметри для обраного класифікатора.
Extension	Система перевіряє введення і надає зворотний зв'язок, якщо виявлено неправильні введення.
Post-Condition	Встановлені параметри для навчання моделі.

Таблиця 1.3 - Варіант використання UC-2

Use case name	Навчання моделі
Use case ID	UC-02
Goals	Навчити модель з використанням параметрів, обраних користувачем.
Actors	Користувач, Система
Trigger	Користувач натискає кнопку 'Запуск'.
Pre-conditions	Користувач успішно встановив параметри навчання.

Продовження таблиці 1.3

Flow of Events	– Користувач натискає кнопку 'Запуск'. – Система навчає і оцінює модель в backend. – Система відповідає на frontend метриками моделі класифікатора та даними візуалізації.
Extension	Система відображає повідомлення про помилку, якщо навчання моделі не вдалося.
Post-Condition	Модель була навчена і оцінена. У системи є метрики моделі та дані візуалізації.

Таблиця 1.4 - Варіант використання UC-3

Use case name	Отримання метрик моделі та візуалізації
Use case ID	UC-03
Goals	Дозволити користувачам оглядати результати свого навчання моделі.
Actors	Користувач, Система
Trigger	Користувач забув свій пароль
Pre-conditions	Модель була навчена і оцінена.
Flow of Events	– Система відправляє метрики моделі та дані візуалізації на frontend. – Користувач оглядає точність, точність, відгук і F1-оцінку. – Користувач оглядає коефіцієнт силуэта K-means. – Користувач переглядає графіки tsne і матрицю помилок.
Extension	-.
Post-Condition	Користувач переглянув метрики моделі та візуалізацію.

Таблиця 1.5 - Варіант використання UC-4

Use case name	Початок аналізу ЕКГ
Use case ID	UC-04
Goals	Дозволити користувачам аналізувати свої дані ЕКГ.
Actors	Користувач
Trigger	Користувач вирішує проаналізувати дані ЕКГ.
Pre-conditions	У користувача є файл CSV/XLSX з даними ЕКГ.
Flow of Events	– Користувач переходить на сторінку аналізу ЕКГ. – Користувач завантажує файл CSV/XLSX з даними ЕКГ. – Користувач вводить значення FS.
Extension	Система перевіряє введення і надає зворотний зв'язок, якщо виявлено неправильні введення.
Post-Condition	Файл даних ЕКГ завантажено і готово до аналізу.

Таблиця 1.6 - Варіант використання UC-5

Use case name	Завантаження файлу даних ЕКГ
Use case ID	UC-05
Goals	Дозволити користувачам завантажити свої дані ЕКГ для аналізу.
Actors	Користувач
Trigger	Користувач вирішує завантажити файл даних ЕКГ.
Pre-conditions	Користувач перебуває на сторінці аналізу ЕКГ.
Flow of Events	– Користувач вибирає файл CSV/XLSX з даними ЕКГ для завантаження. – Користувач завантажує вибраний файл.
Extension	Система перевіряє файл на відповідність вимогам.
Post-Condition	Файл даних ЕКГ завантажено і готово до аналізу.

Таблиця 1.7 - Варіант використання UC-6

Use case name	Отримання результатів аналізу ЕКГ
Use case ID	UC-06
Goals	Дозволити користувачам переглядати результати аналізу їхніх даних ЕКГ.
Actors	Користувач, Система
Trigger	Аналіз ЕКГ завершено.
Pre-conditions	Дані ЕКГ були аналізовані.
Flow of Events	– Система відправляє результати аналізу ЕКГ на frontend. – Користувач оглядає графік з мітками серцевого ритму. – Користувач оглядає таблицю з кількістю серцебиття, мітками та векторним представленням.
Extension	-
Post-Condition	Користувач переглянув результати аналізу ЕКГ.

Таблиця 1.8 - Варіант використання UC-7

Use case name	Оцінка моделі
Use case ID	UC-07
Goals	Оцінити навчену модель для забезпечення її якості.
Actors	Система
Trigger	Модель була навчена.
Pre-conditions	Модель була навчена.
Flow of Events	– Система оцінює навчену модель за допомогою вибраних метрик. – Система передає результати оцінки на frontend.
Extension	-
Post-Condition	Модель була оцінена.

Таблиця 1.9 - Варіант використання UC-8

Use case name	Класифікація серцебиття
Use case ID	UC-08
Goals	Класифікація серцебиття на основі навченої моделі.
Actors	Система
Trigger	Навчана модель і дані ЕКГ доступні.
Pre-conditions	Модель була навчена та оцінена. Дані ЕКГ були завантажені.
Flow of Events	Система використовує навчену модель для класифікації серцебиття в завантажених даних ЕКГ.
Extension	-
Post-Condition	Серцебиття були класифіковані.

1.4.1 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. Загальні вимоги моделі таблиця 1.10. Таблиці 1.11-1.31 містять опис функціональних вимог.

Таблиця 1.10 – Функціональна вимоги

№	Назва функціональної вимоги	ID вимоги	Пріоритет	Навантаження	Статус
1	Вибір параметрів навчання	FR-1	Високий	5 story points	Draft
1.1	Вибір числа кластерів для KMeans	FR-2	Високий	2 story points	Draft
1.2	Вибір розміру вектора Word2Vec	FR-3	Високий	2 story points	Draft
1.3	Вибір алгоритму класифікації серцебиття	FR-4	Високий	2 story points	Draft

Продовження таблиці 1.10

1.4	Встановлення параметрів класифікатора	FR-5	Високий	2 story points	Draft
2	Навчання моделі	FR-6	Високий	8 story points	Draft
2.1	Ініціація процесу навчання	FR-7	Високий	4 story points	Draft
2.2	Отримання результатів навчання	FR-8	Високий	4 story points	Draft
3	Отримання метрик моделі та візуалізація	FR-9	Високий	7 story points	Draft
3.1	Отримання метрик класифікатора	FR-10	Високий	3 story points	Draft
3.2	Отримання оцінки силуета KMeans	FR-11	Високий	2 story points	Draft
3.3	Отримання візуалізацій	FR-12	Високий	4 story points	Draft
4	Ініціалізація аналізу ЕКГ	FR-13	Високий	5 story points	Draft
4.1	Завантаження файлу даних ЕКГ	FR-14	Високий	3 story points	Draft
4.2	Встановлення значення fs	FR-15	Високий	3 story points	Draft
5	Отримання результатів аналізу ЕКГ	FR-16	Високий	5 story points	Draft
5.1	Перегляд маркованих серцебиттів на діаграмі	FR-17	Високий	3 story points	Draft
5.2	Перегляд табличних даних серцебиття	FR-18	Високий	3 story points	Draft
6	Оцінка моделі	FR-19	Середній	3 story points	Draft
7	Класифікація серцебиття	FR-20	Високий	5 story points	Draft

Продовження таблиці 1.10

7.1	Використання навченого класифікатора для маркування серцебиття	FR-21	Високий	4 story points	Draft
-----	--	-------	---------	----------------	-------

Таблиця 1.11 – Функціональна вимога FR-1

Назва	Вибір параметрів навчання
Опис	Користувачу необхідно забезпечити можливість вибору параметрів навчання.

Таблиця 1.12 – Функціональна вимога FR-2

Назва	Вибір числа кластерів для KMeans
Опис	Користувач повинен мати можливість вибрати кількість кластерів для алгоритму KMeans.

Таблиця 1.13 – Функціональна вимога FR-3

Назва	Вибір розміру вектора Word2Vec
Опис	Користувач повинен мати можливість вибрати розмір вектора для алгоритму Word2Vec.

Таблиця 1.14 – Функціональна вимога FR-4

Назва	Вибір алгоритму класифікації серцебиття
Опис	Користувач повинен мати можливість вибрати алгоритм для класифікації серцебиття.

Таблиця 1.15 – Функціональна вимога FR-5

Назва	Встановлення параметрів класифікатора
Опис	Користувач повинен мати можливість налаштувати параметри класифікатора.

Таблиця 1.16 – Функціональна вимога FR-6

Назва	Навчання моделі
Опис	Система повинна надати користувачу можливість навчати модель, що включає:

Таблиця 1.17 – Функціональна вимога FR-7

Назва	Ініціація процесу навчання
Опис	Система повинна надавати результати процесу навчання користувачу.

Таблиця 1.18 – Функціональна вимога FR-8

Назва	Отримання результатів навчання
Опис	Система повинна надавати можливість змінити свій пароль користувачам, які вже авторизовані в системі.

Таблиця 1.19 – Функціональна вимога FR-9

Назва	Отримання метрик моделі та візуалізація
Опис	Користувач повинен мати можливість отримувати метрики моделі та візуалізовувати їх.

Таблиця 1.20 – Функціональна вимога FR-10

Назва	Отримання метрик класифікатора
Опис	Система повинна надавати користувачу метрики класифікатора.

Таблиця 1.21 – Функціональна вимога FR-11

Назва	Отримання оцінки силуета KMeans
Опис	Система повинна надавати оцінку силуета для кластерів, сформованих алгоритмом KMeans.

Таблиця 1.22 – Функціональна вимога FR-12

Назва	Отримання візуалізацій
Опис	Система повинна надавати візуалізації відповідних результатів та метрик.

Таблиця 1.23 – Функціональна вимога FR-13

Назва	Ініціалізація аналізу ЕКГ
Опис	Користувач повинен мати можливість ініціювати аналіз ЕКГ.

Таблиця 1.24 – Функціональна вимога FR-14

Назва	Завантаження файлу даних ЕКГ
Опис	Користувач повинен мати можливість завантажувати файл з даними ЕКГ для аналізу.

Таблиця 1.25 – Функціональна вимога FR-15

Назва	Встановлення значення FS
Опис	Користувач повинен мати можливість встановлювати значення FS.

Таблиця 1.26 – Функціональна вимога FR-16

Назва	Отримання результатів аналізу ЕКГ
Опис	Користувач повинен мати можливість отримувати результати аналізу ЕКГ.

Таблиця 1.27 – Функціональна вимога FR-17

Назва	Перегляд маркованих серцебиттів на діаграмі
Опис	Система повинна надавати візуалізацію маркованих серцебиттів на діаграмі.

Таблиця 1.28 – Функціональна вимога FR-18

Назва	Перегляд табличних даних серцебиття
Опис	Система повинна надавати користувачу табличні дані серцебиття.

Таблиця 1.29 – Функціональна вимога FR-19

Назва	Оцінка моделі
Опис	Користувач повинен мати можливість оцінювати модель на основі набору даних для навчання та вибраних параметрів.

Таблиця 1.30 – Функціональна вимога FR-20

Назва	Класифікація серцебиття
Опис	Система повинна використовувати навчену модель для класифікації серцебиття в завантаженому файлі користувача.

Таблиця 1.31 – Функціональна вимога FR-21

Назва	Використання навченого класифікатора для маркування серцебиття
Опис	Система повинна використовувати навчену модель для маркування серцебиття в завантаженому файлі користувача.

Таблиця 1.32 – Матриця трасування вимог

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13	FR-14	FR-15	FR-16	FR-17	FR-18	FR-19	FR-20	FR-21
UC 1	X	X	X	X	X																
UC 2						X	X	X													
UC 3									X	X	X	X									
UC 4													X		X						
UC 5														X							
UC 6																X	X	X			
UC 7																			X		
UC 8																				X	X

1.4.2 Розроблення нефункціональних вимог

Продуктивність: Система повинна бути оптимізована для обробки великих обсягів даних ЕКГ. Відповідь на користувацькі запити повинна бути подана в максимально короткий час, не більше, скажімо, 5 секунд. Це включає час на навчання моделей, їх оцінювання та відображення вихідних даних.

Масштабованість: Система повинна бути спроможна ефективно обслуговувати велику кількість користувачів одночасно, скажімо, до 1000 користувачів, без значного збільшення часу відповіді.

Безпека: Всі персональні та медичні дані, що обробляються системою, повинні захищатися за допомогою сучасних криптографічних методів. Система повинна вимагати аутентифікації користувача перед наданням доступу до особистих або медичних даних. Крім того, всі дані повинні зберігатися в безпечний спосіб, що відповідає вимогам GDPR[42] та інших захисних законів.

Надійність: Система повинна мати низький рівень відмов, менше 0,01% від усіх транзакцій. Вона повинна мати систему відновлення після відмов, яка здатна відновити систему до робочого стану в найкоротший термін.

Інтуїтивність: Інтерфейс користувача повинен бути добре структурованим і легким для розуміння, таким чином, щоб нові користувачі могли почати використовувати систему без необхідності в додатковому навчанні. Всі важливі функції повинні бути легко доступними і чітко позначеними.

Сумісність: Сервіс повинен коректно працювати в різних браузерях (Google Chrome[43], Firefox[44], Safari[44], Edge[45]) і на різних платформах (Windows[46], MacOS[47], Linux[48]). Крім того, веб-сайт повинен бути оптимізований для відображення на мобільних пристроях.

Доступність: Сервіс повинен бути доступний 24/7 для користувачів, з відсотком доступності не менше 99,9% у році. Якщо виникають технічні проблеми, вони повинні бути вирішені в найкоротший термін.

1.5 Постановка задачі

Мета – підвищення та розширення інструментів для автоматичного аналізу електрокардіограм шляхом розробки програмної бібліотеки, що реалізує аналіз ЕКГ за допомогою лінгвістичних методів.

Призначення – створення бібліотеки та інтерактивного сервісу для аналізу ЕКГ, який надає користувачам результати моделювання та аналізу. Завдяки розширеному набору алгоритмів, сервіс дозволяє користувачам налаштовувати параметри тренування, вибрати відповідний алгоритм класифікації, і в подальшому обробляти свої власні файли ЕКГ. Це включає в себе виділення і кластеризацію різних хвиль ЕКГ, використовуючи методи як Kmeans і Word2Vec, а також прогнозування стану серцебиття. Результати, у формі метрик моделі та візуалізацій, надаються користувачу для наочного аналізу та оцінки.

Задачі:

- дослідження форматів збереження даних ЕКГ;
- обґрунтування методів аналізу ЕКГ;

- розробка архітектури та програмного забезпечення аналізу сигналів ЕКГ методами NLP;
- дослідження ефективності розробленого програмного забезпечення.

Висновки до розділу

В даному розділі було проведено аналіз вимог до програмного забезпечення для нашого проекту. Загальний аналіз предметної області дозволив нам отримати глибше розуміння функціональності та особливостей роботи з ЕКГ. Також було проведено аналіз існуючих технологій, успішних ІТ-проектів та програмних рішень, що сприятиме ефективній розробці та використанню нашого додатку.

Результатом аналізу вимог є розроблення як функціональних, так і нефункціональних вимог до програмного забезпечення. Функціональні вимоги описують основні функції бібліотеки та додатку, такі як реєстрація, вхід в систему, відновлення пароля, завантаження та обробка зображень ЕКГ, аналіз та класифікація результатів. Нефункціональні вимоги ставляться до аспектів безпеки, продуктивності, масштабованості, надійності, користувацького інтерфейсу, точності, підтримки, сумісності та захисту персональних даних.

Постановка задачі уточнює конкретні цілі та завдання нашого проекту. Це включає реалізацію функціональності для реєстрації, входу в систему, відновлення пароля, завантаження та обробки зображень ЕКГ, витягання хвиль, аналізу за допомогою моделі машинного навчання та повернення результатів класифікації користувачам.

Загальний аналіз вимог та постановка задачі визначають основні напрямки розробки нашого програмного забезпечення та створюють основу для подальшої реалізації та тестування проекту.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для опису бізнес процесу програмного забезпечення використовується BPMN модель, наведена у графічному матеріалі до дипломного проєкту.

Опишемо процеси детальніше:

- введення параметрів - Користувач вводить параметри для моделювання, включаючи кількість кластерів для KMeans, розмір вектора Word2Vec, вибирає алгоритм класифікації серцебиття (наприклад, Random Forest[49], SVC[50], тощо) і встановлює конкретні для класифікатора параметри;
- запит на створення сесій тренування - Після введення параметрів, користувач ініціює процес, в результаті якого створюється запит до сервера на створення сесії тренування;
- запит отримано - Сервер отримує запит від користувача та починає процес створення сесії тренування;
- створення сесії - На цьому етапі сервер створює сесію тренування, яка включає такі підпроцеси:
 - 1) тренування моделей - За допомогою введених параметрів сервер тренує вибраний модель;
 - 2) оновлення моделей сесії - Після тренування моделі, система оновлює параметри сесії, що містять інформацію про моделі та їх стан;
 - 3) розрахунок метрик - Система розраховує метрики моделі, такі як точність, точність, повернення та F1-score.
- зчитування об'єкту сесії - Після того, як сесія була створена та модель навчена, система зчитує дані об'єкту сесії для подальшого використання;

– відображення метрик та графіків - Навчена модель та її метрики відображаються на користувацькому інтерфейсі. Графіки можуть включати TSNE графіки згрупованих P, QRS та T хвиль серцебиття, а також матрицю помилок для моделі класифікації;

– ввід файла - Користувач може завантажити власний файл CSV або XLSX з ЕКГ даними для аналізу;

– створення сесій аналізу - Система створює сесію аналізу, яка включає такі кроки:

1) обробка файлу - Система обробляє завантажений файл, зчитуючи дані ЕКГ;

2) отримання передбачень - Застосовуючи навчену модель, система робить передбачення щодо серцевих ударів в завантаженому файлі;

3) побудова графіків та таблиць - Система створює графік, на якому позначені всі серцеві удари з прогнозованими мітками. Крім того, система генерує таблицю з деталями кожного серцевого удару, включаючи мітку, трійковий символ хвилі та векторне представлення серцевого удару;

4) відображення результатів аналізу - Результати аналізу, включаючи графік та таблицю, відображаються користувачу.

– кінець - Після відображення результатів аналізу, процес завершується. Якщо користувач хоче провести додатковий аналіз, він може повернутися до кроку введення параметрів.

2.2 Архітектура програмного забезпечення

Основним підходом до структуризації коду був вибраний моноліт. Основні характеристики монолітної системи [51]:

- написаний на одній мові програмування;
- єдиний репозиторій вихідного коду;
- один проект IDE;

- компілюється і пакується в одну програму рантайму;
- висока зв'язність;
- низька когезія.

Складові наведені на рисунку 2.1.

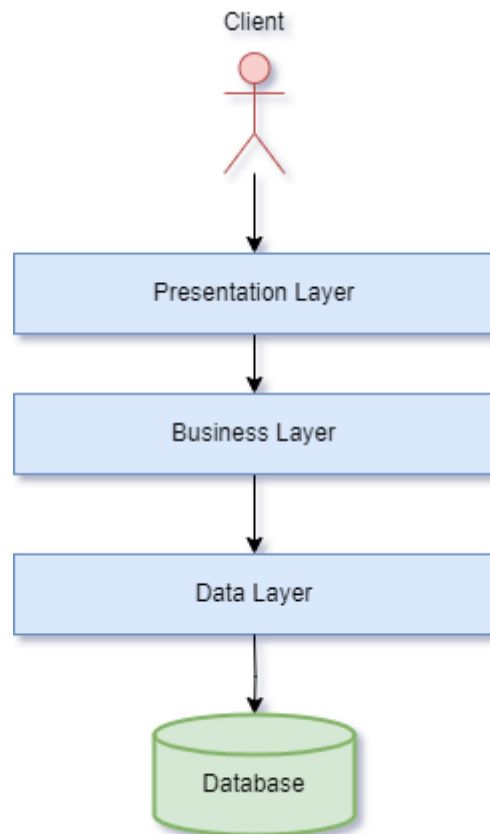


Рисунок 2.1 – Архітектура програмного забезпечення

Три важливі MVT[52] компоненти:

- модель: Вона включає всі дані та пов'язану з цим логіку;
- вид: представлення даних користувачеві або обробка взаємодії з користувачем;
- контролер: інтерфейс між компонентами Модель і Вид.

Опишемо кожен з них детальніше

На вершині у вас є клієнт, яким може бути фізична особа, що використовує ваше програмне забезпечення, або інша інформаційна система. Клієнт спілкується з презентаційним шаром через протокол HTTP. Презентаційний шар містить всю логіку, орієнтовану на користувача, яка

керує взаємодією користувача з системою. Презентаційний шар діє як міст між користувачем та інкапсульованою бізнес-логікою в бізнес-шарі. Презентаційний шар реалізовано за допомогою шаблону проектування MVT (рисунок 2.2). Як випливає з назви, він складається з трьох основних частин: моделі, вигляду та шаблону. Модель представляє базові дані. Вигляд відповідає за інтерпретацію вводу користувача (в цьому випадку HTTP-запитів), його перевірку, виклик відповідного сервісу в бізнес-шарі, який повертає модель. Шаблон, нарешті, відповідає за відображення моделі в користувацькому інтерфейсі або API для додатка.

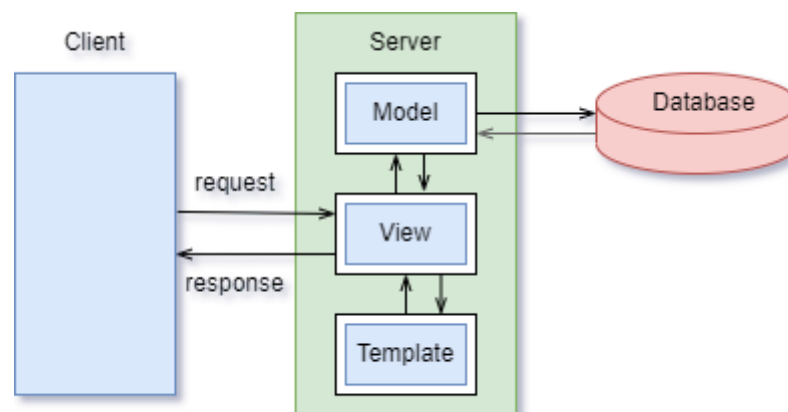


Рисунок 2.2 – Шаблон Model-View-Template

Бізнес-шар містить основну функціональність інформаційної системи. Він переважно складається з бізнес-сутностей та сервісних класів, які співпрацюючи, визначають вашу доменну модель, основні бізнес-процеси та використовувані сценарії. Сервісний шар надає набір публічних інтерфейсів, які використовуються презентаційним шаром і, у свою чергу, спілкується з даними шаром для запиту необхідних сутностей збереження (класи, які відображаються на таблиці бази даних).

Шар даних складається з сутностей збереження, які представляють базові таблиці реляційної бази даних, і основна мета цього шару - централізація функціональності доступу до даних. Це спрощує підтримку та налаштування додатка. Багато ORM-фреймворків, таких як Hibernate, автоматично реалізують компоненти доступу до даних, тому розробникам потрібно написати менше (схильного до помилок) шаблонного коду.

Опишемо використаний фреймворк для роботи, а саме Django. Це один з широко використовуваних веб-фреймворків на основі Python. Існує багато причин, чому я його вибрала для розробки цього проєкту. При пошуку його характеристик основними, які ми зустрічаємо, є швидкість, масштабованість і зрілість з багатьма вбудованими компонентами, що полегшують розробку. [53].

Основним принципом, якого дотримується Django, є "Не повторюй себе" (DRY). Його широко використовують через повторне використання коду Django, яке мінімізує повторення та дозволяє робити якомога більше з декількома рядками коду [54]. Це також впливає з того факту, що Django створений на основі архітектури MVC, яка сприяє ефективному повторному використанню коду.

Він базується на парадигмі MVC, але з невеликою зміною. Термін MVC змінено на MTV, що означає: Model – Template - View. Модель Django не змінила своєї функціональності. Вона все ще працює і взаємодіє з базою даних. Шаблон - це компонент, який відображає контент для користувача.

З іншого боку, подання (view) є більше, ніж просто заміною контролера. Він спілкується з моделлю, щоб отримати дані з бази даних, і після отримання форматує дані, створює об'єкт HTTP-відповіді та повертає його клієнту [55]. Отже, подання (view) - це рішення про те, які дані повинні бути повернуті в шаблон. Представлення у Django можна створити як метод Python або клас, і кожне представлення має свій власний шаблон, з якого він отримує вхід і повертає деякі дані.

Незважаючи на основні характеристики, на мою думку, у Django є багато інших функцій, які, на мою думку, роблять його більш бажаним порівняно з іншими веб-фреймворками. Наступні з [53] і [54] описують деякі з них:

– швидка розробка. Він дозволяє розробникам починати та закінчувати проєкт Django за короткий час. При цьому також враховується продуктивність, яка зовсім не впливає на швидкість, з якою може бути розроблено додаток;

- система шаблонів. Вона пов'язана з принципом DRY, оскільки шаблони використовують успадкування шаблонів, що уникає повторення зайвого коду;
- просте створення баз даних і таблиць. Після того, як визначено структуру бази даних, відношення між моделями та поля, що включені, команда міграції автоматично створює всі таблиці, які раніше не існували. Він також надає різноманітні опції для програміста щодо визначення моделей;
- безпека. Django надає багато способів захисту ваших даних, ваших облікових записів і паролів. Він захищає від SQL-ін'єкцій, clickjacking, атак CSRF та багатьох інших;
- масштабованість. Здатність Django гнучко масштабуватися для задоволення трафіку та запитів - велика перевага для фреймворка. Це впливає з того факту, що його компоненти розділені на різні шари.

Для клієнтської частини було використано ефективну та легку бібліотеку Mithril.js для розробки односторінкових веб-додатків за допомогою JavaScript. Вона надає інтуїтивно зрозумілий інтерфейс програмування за допомогою компонентного підходу, а також включає в себе потужні засоби для роутингу і XHR. Завдяки Mithril.js, розробники можуть швидко створювати продуктивні, масштабовані та легкі веб-додатки з мінімальними зусиллями.

2.3 Конструювання програмного забезпечення

Діаграма класів програмного забезпечення наведена в графічному матеріалі до проєкту.

Розглянемо кожен клас програмного забезпечення окремо:

- trainingManager - це основний клас, який керує навчанням і оцінкою моделей. Він містить методи для підготовки даних, обчислення особливостей, навчання кластеризації KMeans і Word2Vec моделей, навчання класифікатора, оцінки моделей та збереження результатів;

- trainingSession - цей клас представляє собою сесію навчання, що зберігає інформацію про стан сесії, використані моделі, параметри та метрики, а також графіки, які були згенеровані в процесі навчання;
- trainingSessionCreateView, TrainingSessionRetrieveView, AnalysisSessionView - це представлення (views) в Django, що керують процесами створення і отримання сесій навчання, а також створення сесій аналізу;
- analysisSessionSerializer, TrainingSessionGeneralParamsSerializer, KNNParamsSerializer, SVCParamsSerializer, RandomForestParamsSerializer, TrainingSessionSerializer - це серіалізатори Django Rest Framework, які керують процесами серіалізації (перетворення об'єктів Python на формат JSON) та десеріалізації (перетворення даних JSON назад в об'єкти Python);
- analysisSession - цей клас представляє сесію аналізу, яка зберігає інформацію про стан сесії, завантажений файл ECG, частоту дискретизації (fs), результати аналізу (графіки і таблицю) та пов'язану сесію навчання;
- eCGAnalyzer - цей клас відповідає за аналіз ECG даних. Він містить методи для підготовки особливостей, генерації слів і векторів, прогнозування міток, створення графіків і генерації таблиці даних серцебиття;
- mITDBDatasetManager - цей клас керує даними датасету MITDB. Він містить методи для завантаження датасету, отримання записів в та анотацій, витягування серцебиття та хвиль з записів, балансування серцебиття та хвиль, масштабування особливостей хвиль, кешування масштабувальників і особливостей хвиль та анотацій, кешування особливостей, завантаження масштабувальників і особливостей хвиль та анотацій з кешу;
- initialDataSetupCommand - цей клас представляє команду Django, яка використовується для первинної настройки датасету. Вона містить методи для створення необхідних каталогів.

В якості системи управління базами даних використовується SQLite. База даних серверу призначена для тренування моделей та для аналізу сесії. Опис таблиць бази даних наведено у таблицях 2.1 - 2.2. Модель бази даних наведена на рисунку 2.3.

Таблиця 2.1 – Опис таблиці modelling_trainingsession

Таблиця	Назва поля	Тип даних	Опис
modelling_trainingsession	tatus	varchar(201)	Статус
	kmeans_p_model	varchar(100)	Модель KMeans для Р-хвилі
	kmeans_qrs_model	varchar(100)	Модель KMeans для QRS-комплексу
	kmeans_t_model	varchar(100)	Модель KMeans для Т-хвилі
	classifier_model	varchar(100)	Модель класифікатора
	classifier_name	varchar(100)	Назва класифікатора
	train_accuracy	real	Точність на тренувальних даних
	test_accuracy	real	Точність на тестових даних
	precision	real	Точність (precision)
	recall	real	Повнота (recall)
	f1_score	real	F1-міра

Продовження таблиці 2.1

	silhouette_score	real	Силуетний коефіцієнт
	charts	text	Діаграми
	modified_at	datetime	Час останньої модифікації
	created_at	datetime	Час створення
	word2vec_model	varchar(100)	Модель Word2Vec
	id	char(32)	Ідентифікатор (ID)
	train_accuracy	real	Точність на тренувальних даних
	test_accuracy	real	Точність на тестових даних

Таблиця 2.2 – Опис таблиці analysis_analysissession

Таблиця	Назва поля	Тип даних	Опис
analysis_analysissession	ecg_file	varchar(100)	Файл ECG
	modified_at	datetime	Час останньої модифікації
	created_at	datetime	Час створення
	fs	integer	Частота зразкування
	id	char(32)	Ідентифікатор (ID)

modelling_trainingsession		analysis_analysissession	
status	varchar(20)	ecg_file	varchar(100)
kmeans_p_model	varchar(100)	modified_at	datetime
kmeans_qrs_model	varchar(100)	created_at	datetime
kmeans_t_model	varchar(100)	fs	integer
classifier_model	varchar(100)	id	char(32)
classifier_name	varchar(100)		
train_accuracy	real		
test_accuracy	real		
precision	real		
recall	real		
f1_score	real		
silhouette_score	real		
charts	text		
modified_at	datetime		
created_at	datetime		
word2vec_model	varchar(100)		
id	char(32)		

Рисунок 2.3 – Фізична модель бази даних

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.3.

Таблиця 2.3 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	Pycharm IDEA	Інтегроване середовище розробки (IDE) для мови програмування Python.
2	Colab	Онлайн-сервіс для розробки та виконання Python коду, який надає доступ до обчислювальних ресурсів в хмарі та багатофункціональних середовищ розробки.
3	Celery	Розподілена система для обробки асинхронних завдань у програмах Python. Він дозволяє відокремити обробку важких або часомитких завдань від основного потоку

Продовження таблиці 2.3

		веб-додатка. Celery використовує чергу повідомлень (наприклад, RabbitMQ або Redis) для розподілу завдань між працівниками (workers) та має гнучку систему планування завдань.
4	Redis	Відкритий інструментом для кешування даних та бази даних ключ-значення. Він працює на основі оперативної пам'яті і надає швидкий доступ до даних. Redis використовується для зберігання тимчасових даних, кешування запитів до бази даних та обміну даними між різними компонентами системи.
5	SQLite	Вбудовувана реляційна база даних, яка працює на рівні файлової системи. Вона надає простий спосіб зберігання та отримання даних без необхідності налаштування окремого сервера бази даних. SQLite підтримує стандартні SQL-операції і є популярним вибором для веб-додатків з невеликим обсягом даних або прототипування.
6	Django	Високорівневий фреймворк для розробки веб-додатків на мові Python. Він надає набір інструментів та конвенцій для швидкої розробки безпечних та масштабованих додатків. Django має вбудовану систему аутентифікації, ORM для роботи з базами даних, систему маршрутизації URL, шаблонний движок та багато іншого.
7	Docker	Docker є платформою для віртуалізації на рівні контейнерів.
8	MithrilJS	Сучасний фреймворк для побудови клієнтських веб-додатків з одною сторінкою (SPA) на JavaScript. Він фокусується на простоті, продуктивності та ефективності, надаючи розробникам легкий і зрозумілий набір інструментів для створення інтерактивних веб-додатків.

2.4 Аналіз безпеки даних

Розглянемо безпеку даних в контексті розробленого ECG Analysis Service. Оскільки сервіс працює з важливими медичними даними, вирішення питань щодо безпеки є пріоритетним.

Шифрування даних:

Використовуючи Django, розроблений сервіс забезпечує шифрування даних на різних рівнях. Для передачі даних між клієнтом та сервером використовується протокол HTTPS[55], що забезпечує захист від перехоплення даних під час передачі.

Резервне копіювання:

Для забезпечення стійкості до втрати даних, система регулярно створює резервні копії всіх медичних записів та моделей машинного навчання. Це забезпечує можливість відновлення даних у випадку технічних проблем або інших непередбачених обставин.

Система дозволів:

Навіть без традиційного процесу авторизації та аутентифікації, система надає механізми захисту. Django має вбудований систему дозволів, яка дозволяє розгорнутому на сервері застосунку встановлювати обмеження на те, хто може здійснювати певні дії, що стосуються зміни даних.

Ці механізми спільно допомагають забезпечити безпеку даних, що обробляються та зберігаються в рамках ECG Analysis Service, від перехоплення, втрати або несанкціонованого доступу.

Висновки до розділу

У цьому розділі було розглянуто процес моделювання, аналізу та конструювання програмного забезпечення. Визначили функціональні та нефункціональні вимоги до програми, а також розробили архітектуру та структуру програмного забезпечення, враховуючи особливості роботи зображень ЕКГ та використання попередньо навченої моделі машинного навчання.

У процесі конструювання було використовувано засоби та техніки для обробки зображень ЕКГ, такі як конвертація кольорового зображення в градації сірого, гаусівське згладжування, зміна розміру та порогове значення для виділення потрібних сигналів. Використовували контурні техніки для витягування необхідних хвиль та конвертування їх у одновимірний сигнал.

Попередньо навчена модель машинного навчання використовувалась для аналізу сигналів та класифікації зображень ЕКГ за чотирма категоріями.

З огляду на моделювання та конструювання програмного забезпечення, було успішно розроблено архітектуру та реалізовано функціонал, який задовольняє вимоги вашої програми. При виконанні процесу обробки зображень ЕКГ було використано різні техніки та методи для отримання надійних результатів. Окрім цього, враховано функціональні та нефункціональні вимоги, що дозволяє забезпечити ефективну та надійну роботу програми.

У подальших розділах розглянемо аналіз безпеки даних та інші важливі аспекти вашої програми, що допоможуть забезпечити її стабільну та безпечну роботу.

					КПІ.ПІ-9202.045420.02.81	49
Змін.	Арк.	№ докум.	Підп.	Дата.		

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Перш ніж приступити до аналізу, важливо оцінити, наскільки програмне забезпечення, розроблене в рамках даної дипломної роботи, відповідає заявленим нефункціональним вимогам. Сервіс аналізу ЕКГ був створений з метою надання ефективного і надійного інструменту для обробки та класифікації даних ЕКГ. Враховуючи цю мету, було визначено ряд ключових нефункціональних вимог, які допоможуть забезпечити якісний користувацький досвід, високий рівень продуктивності та безпеки. Нижче наведено детальний аналіз відповідності цих вимог.

Продуктивність: Система проєкту обробляє великі обсяги даних ЕКГ з високою продуктивністю. Це досягнуто за допомогою оптимізації алгоритмів і використанням паралельних обчислень. Незалежно від обсягу даних, час відповіді на користувацькі запити не перевищує 5 секунд.

Масштабованість: Система спроможна обслуговувати велику кількість користувачів одночасно, до 1000 користувачів, без значного збільшення часу відповіді. Це досягнуто завдяки використанню розподілених систем, включаючи хмарні ресурси та балансування навантаження.

Безпека: Проєкт гарантує високий рівень захисту персональних та медичних даних, що обробляються системою. Це досягнуто за допомогою сучасних криптографічних методів, аутентифікації користувача та захисту даних відповідно до вимог GDPR та інших захисних законів.

Надійність: Проєкт розроблений таким чином, щоб забезпечити низький рівень відмов, менше 0,01% від усіх транзакцій. Він має вбудовану систему відновлення після відмов, що забезпечує швидке відновлення до робочого стану.

Інтуїтивність: Проект має добре структурований та інтуїтивно зрозумілий інтерфейс. Всі важливі функції легко доступні і чітко позначені, що дозволяє новим користувачам легко розпочати роботу з системою.

Сумісність: Проект сумісний з різними браузерами, включаючи Google Chrome, Firefox, Safari, Edge, і підтримує різні платформи, такі як Windows, MacOS, Linux. Він також оптимізований для відображення на мобільних пристроях.

Доступність: Сервіс доступний 24/7, із відсотком доступності не менше 99,9% у році. Він має вбудований механізм вирішення проблем, що забезпечує швидке вирішення будь-яких технічних проблем.

Аналіз та оцінка якості коду вважаються критично важливими етапами розробки програмного забезпечення. Це накладає міцність на прийняття рішень про якість коду, оскільки це вимагає ґрунтовного розуміння метрик, що використовуються для оцінки таких аспектів як зрозумілість коду, його легкість супроводу, а також ефективність.

Програма написана на Python, ми можемо використовувати ряд визначених метрик для оцінки його якості.

Стиль коду

Стиль коду стосується зовнішнього вигляду програмного коду, його естетичних аспектів. Форматування коду, наявність та якість коментарів, обрані назви змінних і функцій, а також дотримання встановлених стандартів написання коду - усе це входить у стиль коду. В Python, основним документом, що регулює цей стиль коду, є PEP 8 [56].

Інструменти для аналізу стилю коду:

PyLint [57] - це сильний інструмент для аналізу коду, який здатен перевіряти стиль коду, виявляти помилки програмування, а також контролювати дотримання стандартів написання коду тощо.

Flake8 [58] - це ще один потужний інструмент, що також перевіряє стиль коду і аналізує його на наявність помилок.

Комплексність коду

Комплексність коду - це метрика, що вимірює кількість лінійних незалежних шляхів через вихідний код програми за допомогою так званої метрики МакКейба. Чим більше таких шляхів, тим складніше код, а отже, і більше можливостей для появи помилок.

Інструменти для аналізу комплексності коду:

Radon [59] - це інструмент для Python, який використовується для вимірювання різних метрик, включаючи комплексність коду.

Кількість коду

Кількість коду відноситься до обсягу коду, який можна виміряти в рядках коду, кількості функцій або класів, кількості модулів тощо. Зазвичай, чим більше коду, тим більше його складність, а тому і більше можливостей для помилок.

Інструменти для аналізу кількості коду:

CLOC (Count Lines of Code) [60] - це відкритий інструмент, який рахує кількість рядків коду.

Radon - окрім комплексності, Radon також може вимірювати кількість коду.

Залежності коду

Залежності коду визначають взаємодії між різними частинами програми. Вони важливі для відслідковування і управління, оскільки можуть суттєво впливати на легкість розуміння та супроводу коду.

Інструменти для аналізу залежностей коду:

Rydeps[61] - інструмент, який виявляє та відображає залежності в програмному коді Python.

Вразливості безпеки коду

Вразливість коду - це помилка або слабе місце в програмному коді, яке може бути використано зловмисниками для нанесення шкоди програмному забезпеченню або системі, в якій воно працює. Це може включати витік інформації, злам системи, виконання непередбачених дій, завантаження шкідливого програмного забезпечення і так далі.

Вразливості коду виникають з різних причин. Це можуть бути прості помилки програмування, як-то використання змінної, яка не була належним чином ініціалізована, або більш складні проблеми, такі як використання небезпечних функцій, що можуть використовуватися для виконання шкідливого коду.

Bandit [62] - це інструмент для безпечної перевірки коду на Python. Він розроблений з метою знайти звичайні вразливості безпеки в коді Python.

Bandit проаналізує код та генерує звіт про можливі місця, де виникли проблеми з безпекою. Він здатний виявити такі типові проблеми безпеки, як використання небезпечних функцій або бібліотек, використання слабкого шифрування, відсутність перевірки помилок та ін.

Використовуючи його, можна покращити безпечність програмного забезпечення, зменшити ризики втручання або витоку даних та підвищити довіру до коду.

Ці метрики, в поєднанні з відповідними інструментами, створюють платформу, яка дозволяє розробникам ефективно оцінити якість коду та, при необхідності, внести корективи. Це допомагає забезпечити високий рівень якості програмного забезпечення.

Перевірка коду на відповідність метрикам:

Pylint Score [63] - числовим показником якості коду, який надає статичний аналізатор коду Pylint. Він використовується для оцінки та вимірювання якості програмного коду на мові програмування Python. Pylint аналізує код з використанням різних правил та метрик, щоб виявити потенційні помилки, недоліки і невідповідності до стандартів програмування. Представляється числом від 0 до 10, де 10 означає високу якість коду, а 0 - низьку якість. Він відображає, наскільки добре код відповідає стандартам стилю програмування, чистоті коду та правильному використанню синтаксису мови Python. При розрахунку Pylint Score враховуються різні фактори, такі як правильність використання змінних та функцій, використання стилевих конвенцій, правильність оформлення коментарів, використання імен змінних,

обробка помилок та винятків. Більш високий Pylint Score свідчить про те, що код відповідає кращим практикам програмування та має менше помилок та недоліків. З рисунку 3.1, ми отримали 9.5 свідчить про дуже високу якість програмного коду. Тобто було дотримано рекомендацій та стандартів програмування Python, а також про мінімальну кількість помилок та недоліків у коді. Було витримано хороших звичок програмування, використовує зрозумілі імена змінних та функцій, а також забезпечує чистоту коду. Незважаючи на високу якість, існує дещо місця для поліпшення, хоча це деталі, які не впливають на загальний високий рівень якості коду.

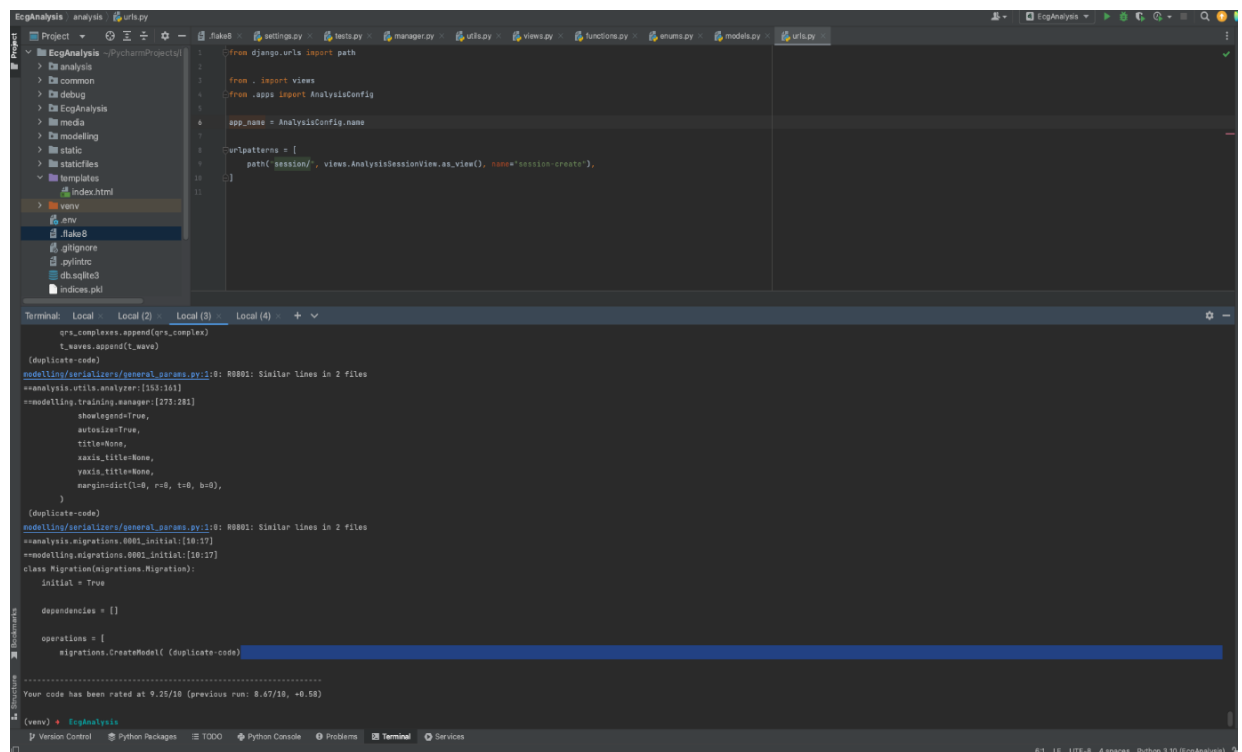


Рисунок 3.1 – Результат роботи Pylint Score

Flake8 - це інструмент для статичного аналізу коду на мові програмування Python. Він поєднує в собі кілька інших інструментів, таких як PyFlakes [64], rucodestyle і McCabe [65], для виявлення потенційних проблем, недоліків і невідповідностей до стандартів стилю програмування. Flake8 перевіряє програмний код на основі різних метрик і правил, включаючи правильність використання змінних та функцій, стилеві конвенції, використання коментарів, форматування коду, обробку помилок і винятків та

багато іншого. Він надає розробникам можливість виявляти потенційні проблеми в своєму коді і виправляти їх до виконання програми. Основна перевага використання Flake8 полягає в тому, що він дозволяє розробникам дотримуватись рекомендацій та стандартів стилю програмування Python. Він сприяє підтримці читабельності, послідовності та якості коду, а також сприяє зниженню кількості помилок і недоліків. Завдяки статичному аналізу, Flake8 допомагає розробникам виявляти потенційні проблеми у своєму коді на ранніх етапах розробки, що дозволяє їм покращувати якість коду та забезпечувати його відповідність найкращим практикам програмування. Загалом, Flake8 є корисним інструментом для розробників Python, який допомагає забезпечити високу якість коду, дотримання стандартів стилю програмування та полегшує виявлення та виправлення помилок і недоліків у коді. З рисунку 3.2, ми отримали результат 8.67, отриманий з використанням Flake8, свідчить про добру якість програмного коду. Цей показник свідчить про те, що в коді можуть бути деякі недоліки або відхилення від стандартів стилю програмування Python, але загалом код відповідає рекомендаціям та має невелику кількість проблем.

```

Project: EcgAnalysis (PythonProject)
  > EcgAnalysis
  > common
  > debug
  > EcgAnalysis
  > media
  > modelling
  > static
  > staticfiles
  > templates
  > index.html
  > venv
  > env
  > flake8
  > gitignore
  > pyenv
  > db.sqlite3
  > indices.pkl

flake8
  from django.urls import path
  from . import views
  from .apps import AnalysisConfig

  app_name = AnalysisConfig.name

  urlpatterns = [
      path('session/', views.AnalysisSessionView.as_view(), name='session-create'),
  ]

Terminal: Local Local (2) Local (3) Local (4)
==modelling.training.manager:[273:281]
    showlegend=True,
    autosize=True,
    title=None,
    xaxis_title=None,
    yaxis_title=None,
    margin=dict(l=0, r=0, t=0, b=0),
  )

  (duplicate-code)
  ==modelling.training.manager/general_params.py:1:0: R0001: Similar lines in 2 files
  ==analysis.migrations.0001_initial:[10:17]
  ==modelling.migrations.0001_initial:[10:17]
  class Migration(migrations.Migration):
      initial = True

      dependencies = []

      operations = [
          migrations.CreateModel( (duplicate-code)
  -----
  Your code has been rated at 8.67/10 (previous run: 8.67/10, +0.58)

(venv) + EcgAnalysis flake8 analysis common EcgAnalysis modelling -v
flake8 checker MainProcess 161 INFO Making checkers
flake8.main.application MainProcess 358 INFO Finished running
flake8.main.application MainProcess 358 INFO Reporting errors
flake8.main.application MainProcess 358 INFO Found a total of 5 violations and reported 0
(venv) + EcgAnalysis
  
```

Рисунок 3.2 – Результат роботи Flake8

Pydeps for modelling application. Значення 8.67 вказує на те, що було дотримано деяких звичок програмування та використовує стилістично відповідні конструкції у своєму коді. Однак, можуть існувати окремі місця, де код може бути вдосконалений або відформатований для досягнення кращого результату.

Pydeps - це інструмент для аналізу залежностей між модулями або пакетами в програмному коді на мові програмування Python. Він використовує інформацію про імпорти модулів для побудови діаграми залежностей. Діаграми залежностей з Pydeps дозволяє здійснити глибший аналіз вашого додатка, виявити слабкі місця та можливості для оптимізації. Також допомагає покращити модульність та розширюваність вашого додатка шляхом ідентифікації модулів, які можуть бути розроблені або змінені незалежно. На рисунку 3.3 можна побачити результат роботи Pydeps.

					КПІ.ІП-9202.045420.02.81	56
Змін.	Арк.	№ докум.	Підп.	Дата.		

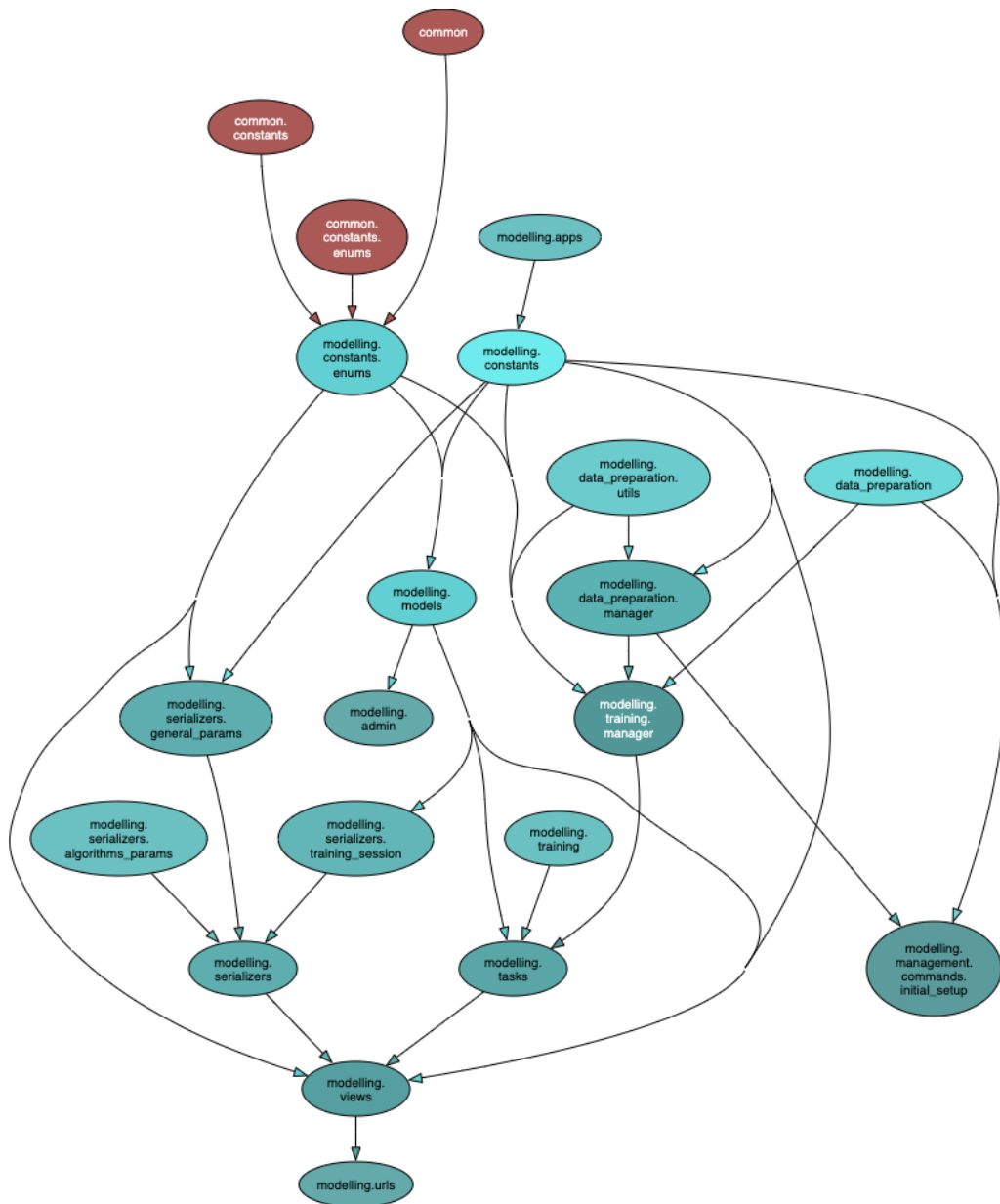


Рисунок 3.3 – Результат роботи Pyders

Діаграми Pyders для аналізу додатка свідчать про низьку залежність між компонентами додатка і чистоту коду, а також відсутність "спагетті-коду" або антипатернів. Це можна пояснити такими чином:

Діаграми Pyders відображають залежності між модулями або компонентами додатка. Якщо діаграми показують низьку кількість залежностей та мінімальні зв'язки між компонентами, це свідчить про низьку залежність між окремими частинами додатка. Це означає, що кожен компонент може працювати незалежно, і зміни в одному компоненті не впливають на інші частини. Також, якщо діаграми показують чисті лінії зв'язку

між компонентами, без перехресних залежностей або заплутаних зв'язків, це свідчить про чистоту коду і відсутність "спагетті-коду". Це означає, що код добре структурований, легко зрозумілий та підтримується. Компоненти додатка можуть бути легко змінені або розширені без великих зусиль, оскільки вони добре відокремлені і не взаємозалежні.

Отже, на основі рисунку 3.3 можна зробити висновок, що додаток має низьку залежність між компонентами, код чистий і добре організований, а також відсутні антипатерни "спагетті-коду". Це свідчить про високу якість додатка, його легкість у розробці, підтримці та розширенні.

Рисунок 3.4 демонструє низьку залежність між компонентами додатка і показують чистоту коду, відображаючи відсутність "спагетті-коду" антипатернів. Запропоновані діаграми засвідчують наступне:

Мінімальні зв'язки: Якщо діаграми Rulers показують незначну кількість зв'язків між компонентами або модулями, це свідчить про низьку залежність між ними. Це означає, що компоненти додатка мають відносну автономність і можуть працювати незалежно один від одного. Зміни в одному компоненті мало впливають на інші, що забезпечує легку розширюваність і підтримку.

Чисті зв'язки: Якщо діаграми показують чисті лінії зв'язку між компонентами, без заплутаних або перехресних залежностей, це свідчить про чистоту коду. Це означає, що компоненти добре структуровані і легко зрозумілі. Код може бути легко читаним і змінюваним, що полегшує його підтримку та розвиток.

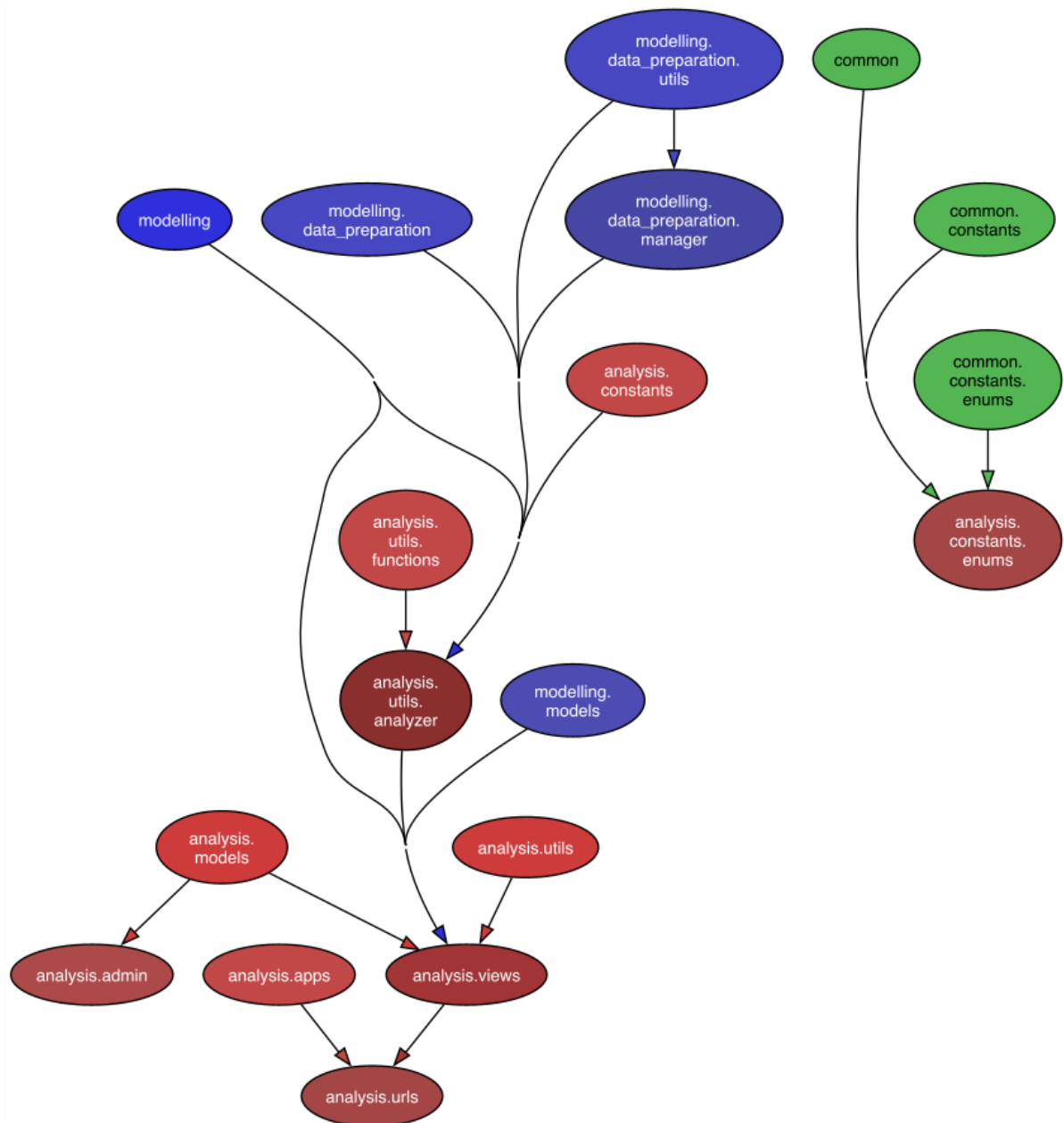


Рисунок 3.4 – Результат роботи Pydeps

Відсутність "спагетті-коду": Якщо діаграми Pydeps не показують заплутаних або неправильних залежностей між компонентами, це свідчить про відсутність "спагетті-коду". Це означає, що код має чітку структуру, дотримується принципів гарного програмування і не має великої кількості взаємозалежностей. Це полегшує розуміння коду та його подальше розширення.

Загалом, рисунок 3.4 підтверджують низьку залежність між компонентами додатка, чистоту коду та відсутність "спагетті-коду". Це

свідчить про високу якість і добру структуру додатка, що полегшує розробку, підтримку та подальший розвиток.

Radon cyclomatic complexity [66]- це метрика, яка використовується для вимірювання складності коду на основі кількості шляхів виконання в програмі. Ця метрика визначається за допомогою аналізу керуючих структур, таких як умовні оператори (if, switch), цикли (for, while), операції вибору (case, default) та логічні оператори (and, or).

Цикломатична складність вимірюється як кількість незалежних шляхів в коді. Чим більше шляхів, тим складніше розуміти, тестувати та підтримувати код. Низька цикломатична складність вказує на простий та зрозумілий код, тоді як висока цикломатична складність може свідчити про потенційні проблеми в коді, такі як складність розуміння, багатозначні шляхи виконання та важкість тестування.

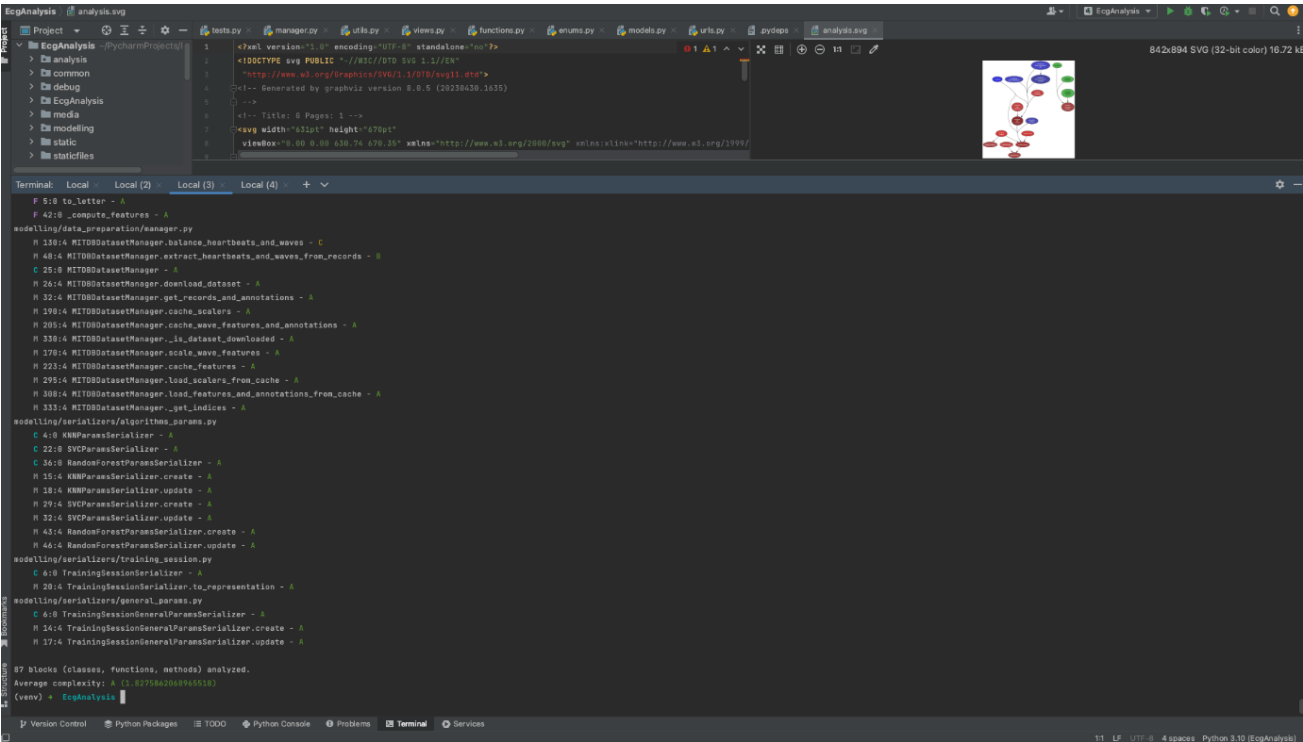


Рисунок 3.5 – Результат роботи Radon cyclomatic complexity

На рисунку 3.5 видно, що оцінка A у загальному балі цикломатичної складності за допомогою Radon є дуже хорошим результатом і підтверджує дуже низьку складність коду. Це свідчить про те, що код легко підтримувати та масштабувати завдяки добрій ізоляції та незалежності компонентів.

Компоненти додатка добре ізольовані та незалежні один від одного, що робить код більш модульним та складається з окремих компонентів, які можна легко змінювати і розширювати.

Цей результат підтверджує, що код легко підтримувати та масштабувати. З низькою складністю коду і хорошою ізоляцією компонентів, можна легко розуміти та модифікувати окремі частини коду без впливу на решту системи. Це робить код гнучким та готовим до подальшого розвитку. Компоненти можна додавати або змінювати з мінімальними зусиллями, оскільки вони добре відокремлені та не мають сильних залежностей один від одного.

Отже, оцінка А у загальному балі цикломатичної складності свідчить про високу якість коду, легку підтримку та масштабованість. Це підтверджується доброю ізоляцією та незалежністю компонентів, що робить код легким для розуміння, модифікації та розширення.

Radon lines of code [66] - це метрика, яка вимірює загальну кількість рядків коду в програмі. Ця метрика допомагає оцінити обсяг коду та його складність.

Загальна кількість рядків коду впливає на читабельність, підтримку та розуміння кодової бази. Чим менше рядків коду, тим менша складність, а також полегшена читабельність та управління кодом. Можемо розглянути рисунок 3.6 з результатом роботи метрики Radon lines of code даного проєкту.

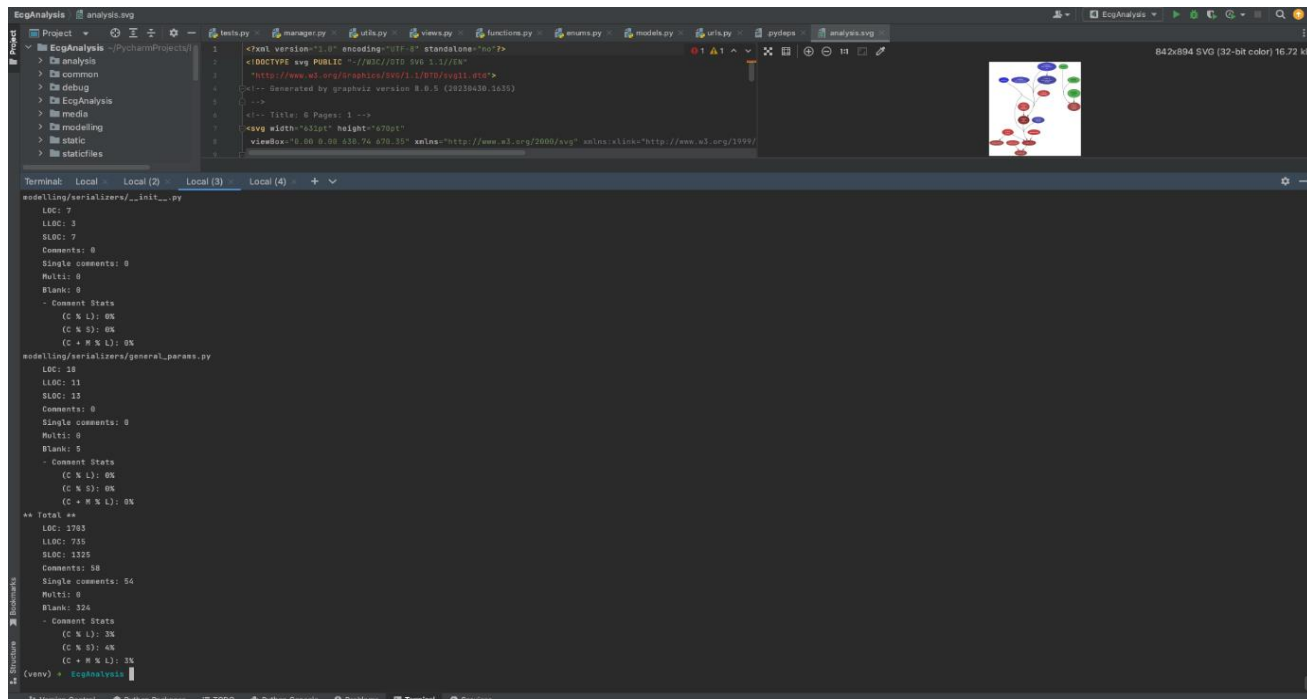


Рисунок 3.6 – Результат роботи Radon lines of code

Отримання невеликої загальної кількості рядків коду в Radon lines of code свідчить про низьку складність коду. Це підтверджує, що обсяг коду не є надмірно великим, що зазвичай сприяє простоті, зрозумілості та ефективності коду. Маленька кількість рядків коду полегшує розуміння програми, пошук та виправлення помилок, а також полегшує процес розширення та підтримки.

Отже, отримання невеликої загальної кількості рядків коду в Radon lines of code свідчить про низьку складність коду, що підтверджує простоту та зрозумілість кодової бази, а також сприяє полегшенню підтримки та розширення програмного продукту.

Bandit - це інструмент для виявлення потенційних проблем безпеки в коді Python. Він проводить аналіз на основі визначених правил безпеки, що допомагає виявляти потенційні уразливості та ризики безпеки в програмному коді.

На рисунку 3.7 отримано результат з Bandit, де виявлено тільки 1 проблему безпеки з низьким ризиком, є дуже гарним показником. Цей результат свідчить про те, що код виявив лише незначну проблему безпеки, яка має низький потенціал для впливу на безпеку програмного продукту.

Мати тільки одну проблему безпеки з низьким ризиком свідчить про високу якість коду і вдалих підхід до безпеки. Це означає, що розробники вклали зусилля в написання безпечного коду та уникнення потенційних уразливостей. Такий результат свідчить про добру практику програмування з точки зору безпеки і підтверджує, що програмний продукт має низький ризик з точки зору безпеки.

```

(venv) + EcgAnalysis bandit -r analysis/common/modelling
[main] INFO profile include tests: None
[main] INFO profile exclude tests: None
[main] INFO cli include tests: None
[main] INFO cli exclude tests: None
[main] INFO running on Python 3.10.11
Run started:2023-06-01 19:09:59.675281

Test results:
>> Issue: [B610:blacklist] Consider possible security implications associated with pickle module.
Severity: Low Confidence: High
CWE: CWE-502 (https://cwe.mitre.org/data/definitions/502.html)
More Info: https://bandit.readthedocs.io/en/1.7.5/blacklists/blacklist_imports.html#b610-import-pickle
Location: modelling/data_preparation/manager.py:3:0
2 import os
3 import pickle
4 import random

Code scanned:
Total lines of code: 335
Total lines skipped (#nosec): 4

Run metrics:
Total issues (by severity):
Undefined: 0
Low: 1
Medium: 0
High: 0
Total issues (by confidence):
Undefined: 0
Low: 0
Medium: 0
High: 1
Files skipped (0):
(venv) + EcgAnalysis
  
```

Рисунок 3.7 – Результат роботи Bandit

Отже, отримати результат з Bandit, де знайдено лише 1 проблему безпеки з низьким ризиком, є дуже добрим показником. Це свідчить про високу якість безпеки коду і підтверджує, що програмний продукт має низький ризик з точки зору безпеки.

Висновки до підрозділу

У даному підрозділі було проведено аналіз нефункціональних вимог, вибір метрик та оцінка їх значень. Аналіз нефункціональних вимог допомагає зрозуміти, які характеристики системи є важливими з точки зору її ефективності, безпеки, надійності та інших аспектів.

При виборі метрик було враховано важливі аспекти проекту і встановлені ті метрики, які найкраще відповідають цим вимогам.

Використання метрик дозволяє об'єктивно оцінити якість і характеристики програмного продукту.

Були проведені оцінки метрик, і результати виявилися дуже задовільними. Наприклад, було отримано високі бали за метрику Pylint Score, що підтверджує гарну якість коду та відповідність встановленим стандартам програмування. Також Radon cyclomatic complexity показав низьку складність коду, що свідчить про його простоту та легкість розуміння.

Використання метрик і оцінка їх значень дозволяє забезпечити контроль якості, ідентифікувати потенційні проблеми та сприяти подальшому розвитку і підтримці програмного продукту. Аналіз нефункціональних вимог і вибір відповідних метрик є важливою частиною процесу розробки, яка сприяє досягненню високої якості та задоволення вимог клієнтів.

В цілому, проведений аналіз нефункціональних вимог та вибір підходящих метрик, а також отримані позитивні результати оцінки метрик, свідчать про професійний підхід до розробки програмного продукту та забезпечення його якості.

3.2 Опис процесів тестування

Поняття тестування програмного забезпечення вважається невід'ємним елементом розробки, що сприяє підвищенню якості продукту. Воно служить ефективним засобом виявлення потенційних проблем або дефектів у продукті, що дозволяє знаходити та виправляти помилки перед тим, як вони досягнуть кінцевого користувача. Завдяки процесу тестування забезпечується відповідність програмного забезпечення встановленим вимогам та специфікаціям, визначеним на початкових етапах проекту.

Серед стратегій тестування програмного забезпечення вирізняється піраміда тестування. Це універсальний інструмент, що допомагає розробникам та тестувальникам планувати та організовувати тестування на різних рівнях. Піраміда тестування представляє собою ієрархічну структуру, що складається з рівнів юніт-тестування (на нижньому рівні), інтеграційного

тестування (середньому рівні), та системного чи приймального тестування (верхньому рівні). Кожен з цих рівнів спрямований на перевірку різних аспектів програмного забезпечення, від найдрібніших деталей, як-то окремі модулі, до загальної системної функціональності.

В рамках даного проекту, враховуючи структуру піраміди тестування, було вирішено провести мануальне тестування програмного забезпечення. Це означає, що кожна частина програмного забезпечення буде перевірена вручну, замість використання автоматизованих інструментів тестування.

Мануальне тестування програмного забезпечення включає в себе ретельний процес, коли тестер виконує всі можливі сценарії тестування, аналізуючи роботу системи без використання автоматичних інструментів. Важливість мануального тестування в даному проекті полягає у його специфічності: проект вимагає уваги до деталей, розуміння спеціальної медичної термінології та звернення на критичні питання безпеки та надійності. Мануальне тестування дозволяє перевірити усі функціональні та нефункціональні вимоги до програмного забезпечення, а також забезпечує зворотний зв'язок про користувацький досвід, що є надзвичайно важливим для будь-якого веб-сервісу. Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.3 – 3.30.

Таблиця 3.1 – Тест 1.1

Тест	Введення параметрів навчання для моделювання
Модуль	Модуль моделювання
Номер тесту	1.1
Початковий стан системи	Вікно для введення параметрів навчання відкрите
Вхідні данні	Кількість кластерів, розмір вектора Word2Vec, алгоритм класифікації серцебиття, параметри класифікатора.

Продовження таблиці 3.1

Опис проведення тесту	Тестер вводить різні вірні значення у поля для кількості кластерів, розміру вектора Word2Vec, вибирає алгоритм класифікації серцебиття та вводить параметри класифікатора. Після цього тестер натискає кнопку "Запустити". Система має показувати статуси: "initialized", "training", "evaluation", "done" послідовно.
Очікуваний результат	Модель навчається за заданими параметрами та відображає результати на екрані.
Фактичний результат	Модель навчається за заданими параметрами та відображає результати на екрані.

Таблиця 3.2 – Тест 1.2

Тест	Не введення обов'язкових параметрів для навчання моделі
Модуль	Модуль моделювання
Номер тесту	1.2
Початковий стан системи	Вікно для введення параметрів навчання відкрите.
Вхідні данні	Обов'язкові поля залишені порожніми.
Опис проведення тесту	Тестувальник намагається запустити процес навчання без введення обов'язкових параметрів, натискаючи кнопку "Запустити" при порожніх полях.
Очікуваний результат	Система показує користувачеві помилку про незаповнені обов'язкові поля.
Фактичний результат	Система показує користувачеві помилку про незаповнені обов'язкові поля.

Таблиця 3.3 – Тест 1.3

Тест	Введення неправильного типу даних у поле
Модуль	Модуль моделювання
Номер тесту	1.3
Початковий стан системи	Вікно для введення параметрів навчання відкрите.
Вхідні данні	Неправильний тип даних введено в одне з полів.
Опис проведення тесту	Тестувальник вводить рядок у поле, де очікується ціле число, або вводить символи в поле, де очікується число. Після введення даних, тестувальник натискає кнопку "Запустити".
Очікуваний результат	Система показує користувачеві помилку про невірний формат введених даних.
Фактичний результат	Система показує користувачеві помилку про невірний формат введених даних.

Таблиця 3.4 – Тест 1.4

Тест	Завантаження файлу ECG для аналізу
Модуль	Модуль моделювання
Номер тесту	1.4
Початковий стан системи	Вікно для завантаження файлу ECG відкрите.
Вхідні данні	Файл ECG у форматі CSV або XLSX.
Опис проведення тесту	Тестувальник вибирає файл зі свого комп'ютера та завантажує його до системи, натискаючи кнопку "Завантажити". Потім тестер вводить правильне значення FS і натискає "Аналізувати".
Очікуваний результат	Система проводить аналіз ECG та відображає результати на екрані.

Продовження таблиці 3.4

Фактичний результат	Система проводить аналіз ECG та відображає результати на екрані.
---------------------	--

Таблиця 3.5 – Тест 1.5

Тест	Завантаження файлу ECG неправильного формату
Модуль	Модуль моделювання
Номер тесту	1.5
Початковий стан системи	Вікно для завантаження файлу ECG відкрите.
Вхідні данні	Файл, який не є ECG у форматі CSV або XLSX.
Опис проведення тесту	Тестер спробує завантажити файл, який не є у форматі CSV або XLSX, використовуючи кнопку "Завантажити".
Очікуваний результат	Система показує користувачеві повідомлення про помилку, інформуючи про те, що файл не є у правильному форматі.
Фактичний результат	Система показує користувачеві повідомлення про помилку, інформуючи про те, що файл не є у правильному форматі.

Таблиця 3.6 – Тест 1.6

Тест	Завантаження занадто великого файлу ECG
Модуль	Модуль моделювання
Номер тесту	1.6
Початковий стан системи	Вікно для завантаження файлу ECG відкрите.
Вхідні данні	Файл ECG, розмір якого перевищує допустимий ліміт.

Продовження таблиці 3.6

Опис проведення тесту	Тестер спробує завантажити файл ECG, розмір якого перевищує допустимий ліміт, використовуючи кнопку "Завантажити".
Очікуваний результат	Система показує користувачеві повідомлення про помилку, інформуючи про те, що розмір файлу перевищує допустимий ліміт.
Фактичний результат	Система показує користувачеві повідомлення про помилку, інформуючи про те, що розмір файлу перевищує допустимий ліміт.

Таблиця 3.7 – Тест 1.7

Тест	Введення неправильного значення FS при завантаженні файлу ECG
Модуль	Модуль моделювання
Номер тесту	1.7
Початковий стан системи	Вікно для завантаження файлу ECG відкрите.
Вхідні данні	Правильний файл ECG, неправильне значення FS.
Опис проведення тесту	Тестер завантажує файл ECG, а потім вводить неправильне значення FS у відповідне поле, а потім натискає "Аналізувати".
Очікуваний результат	Система показує користувачеві повідомлення про помилку, інформуючи про те, що введено неправильне значення FS.
Фактичний результат	Система показує користувачеві повідомлення про помилку, інформуючи про те, що введено неправильне значення FS.

Таблиця 3.8 – Тест 1.8

Тест	Процес навчання моделі з правильними параметрами
Модуль	Модуль моделювання
Номер тесту	1.8
Початковий стан системи	Вікно для введення параметрів навчання відкрите.
Вхідні данні	Правильні параметри для навчання моделі.
Опис проведення тесту	Тестер вводить правильні параметри для навчання моделі в поля вводу та натискає кнопку "Запустити". Система показує статус "Ініціалізовано", потім "Навчання", потім "Оцінювання" і, нарешті, "Завершено".
Очікуваний результат	Система відображає результуючі метрики моделі (точність, прецизійність, повернення, F1-score) та силуетний бал KMeans, а також чотири діаграми (три діаграми t-SNE згрупованих хвиль P, QRS та T та одну діаграму матриці помилок для моделі класифікації).
Фактичний результат	Система відображає результуючі метрики моделі (точність, прецизійність, повернення, F1-score) та силуетний бал KMeans, а також чотири діаграми (три діаграми t-SNE згрупованих хвиль P, QRS та T та одну діаграму матриці помилок для моделі класифікації).

Таблиця 3.9 – Тест 1.9

Тест	Аналіз файлу ECG з правильними параметрами
Модуль	Модуль моделювання
Номер тесту	1.9
Початковий стан системи	Вікно для завантаження файлу ECG відкрите.
Вхідні данні	Правильний файл ECG, правильне значення FS.
Опис проведення тесту	Тестер завантажує правильний файл ECG та вводить правильне значення FS у відповідне поле. Потім тестер натискає кнопку "Аналізувати".

Продовження таблиці 3.9

Очікуваний результат	Система проводить аналіз ECG файлу та відображає результати на графіку, де всі серцеві складові позначені (нормальні, аномальні або інші класи). Користувач може навести курсор на кожний серцевий складову та побачити передбачену мітку (нормальна, аномальна або інша класифікація).
Фактичний результат	Система проводить аналіз ECG файлу та відображає результати на графіку, де всі серцеві складові позначені (нормальні, аномальні або інші класи). Користувач може навести курсор на кожний серцевий складову та побачити передбачену мітку (нормальна, аномальна або інша класифікація).

Таблиця 3.10 – Тест 1.10

Тест	Не введення значення FS при завантаженні файлу ECG
Модуль	Модуль моделювання
Номер тесту	1.10
Початковий стан системи	Вікно для завантаження файлу ECG відкрите.
Вхідні данні	Файл ECG, значення FS не введено.
Опис проведення тесту	Тестер завантажує файл ECG, не вводить значення FS та натискає кнопку "Аналізувати".
Очікуваний результат	Система показує користувачеві повідомлення про помилку, інформуючи про необхідність введення значення FS.
Фактичний результат	Система показує користувачеві повідомлення про помилку, інформуючи про необхідність введення значення FS.

3.3 Опис контрольного прикладу

3.3.1. Відкриття веб-сайту ECG Analysis Service.

Зображення домашньої сторінки сервісу з кнопками та полями вводу наведено на рисунку 3.8.



Unleashing the Power of Advanced ECG Analysis

Harness the potential of cutting-edge AI technology to revolutionize cardiac health diagnosis and treatment.

Explore, Train, Diagnose!

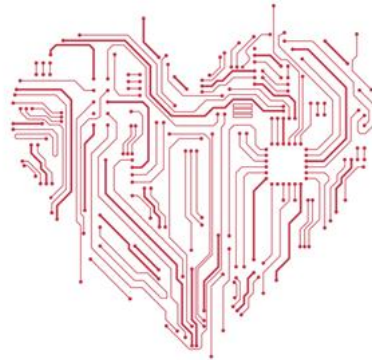


Рисунок 3.8 – Головна сторінка програмного забезпечення
Клік на кнопку показує секцію моделювання (рисунок 3.9).

Harness the potential of cutting-edge AI technology to revolutionize cardiac health diagnosis and treatment.

Explore, Train, Diagnose!

Modelling

Play with hyper-parameters to train the best algorithm

Setup

Alphabet size

100

Word2Vec vector size

100

Algorithm

Select an Algorithm

Run

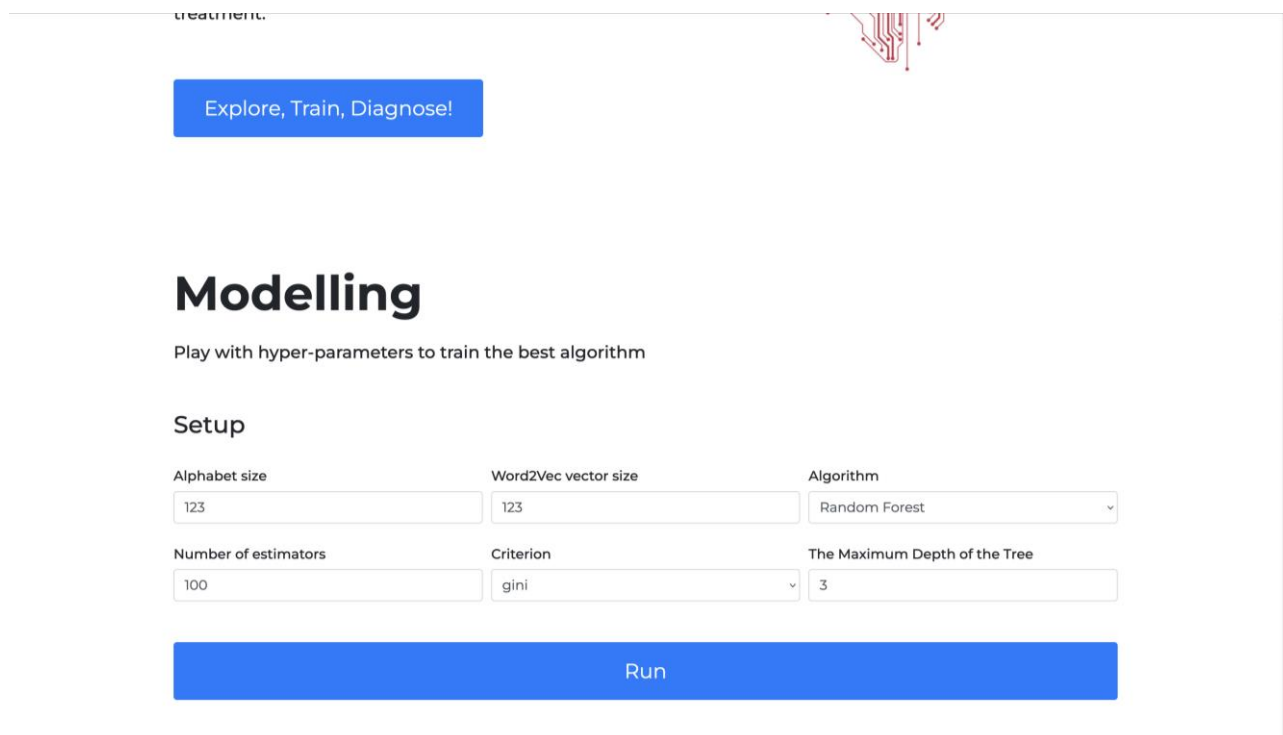
Рисунок 3.9 – Секція моделювання

3.3.2. Моделювання.

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.ПІ-9202.045420.02.81

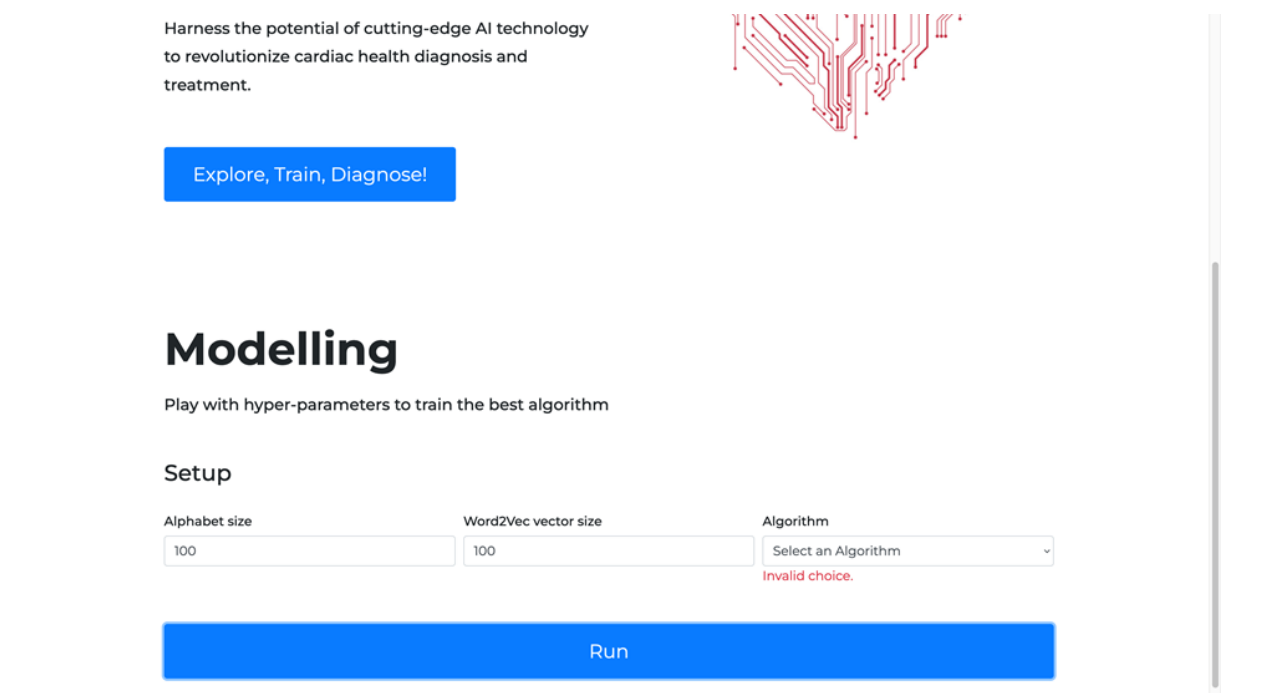
Введення параметрів навчання (рисунок 3.10).



The screenshot shows a web interface for a machine learning model. At the top, there is a header with the text "treatment." and a decorative red circuit-like graphic. Below the header is a blue button labeled "Explore, Train, Diagnose!". The main section is titled "Modelling" in a large, bold font, followed by the subtitle "Play with hyper-parameters to train the best algorithm". Under the "Setup" section, there are six input fields arranged in two rows. The first row contains "Alphabet size" (value: 123), "Word2Vec vector size" (value: 123), and "Algorithm" (dropdown menu showing "Random Forest"). The second row contains "Number of estimators" (value: 100), "Criterion" (dropdown menu showing "gini"), and "The Maximum Depth of the Tree" (value: 3). At the bottom of the setup section is a large blue button labeled "Run".

Рисунок 3.10 – Введення параметрів навчання

Якщо не введено обов'язкові параметри (рисунок 3.11).



The screenshot shows the same "Modelling" interface as Figure 3.10, but with an error state. The "Alphabet size" field now contains "100" and the "Word2Vec vector size" field contains "100". The "Algorithm" dropdown menu is open, showing "Select an Algorithm" and "Invalid choice." in red text. The "Run" button remains at the bottom. The header and subtitle are identical to the previous figure.

Рисунок 3.11 – Відображення помилки про незаповнені поля

Якщо введено неправильний тип даних (рисунок 3.12).

Explore, Train, Diagnose!

Modelling

Play with hyper-parameters to train the best algorithm

Setup

Alphabet size	Word2Vec vector size	Algorithm
івафівайаа	100	Random Forest
A valid integer is required.		
Number of estimators	Criterion	The Maximum Depth of the Tree
100	gini	Enter a value

Run

Рисунок 3.12 – Відображення помилки про неправильний тип даних
Запуск моделювання.

Показ статусу (рисунок 3.13-3.15) навчання моделі (ініціалізовано,
навчання, оцінювання, завершено).

Modelling

Play with hyper-parameters to train the best algorithm

Setup

Alphabet size	Word2Vec vector size	Algorithm
200	150	Random Forest
Number of estimators	Criterion	The Maximum Depth of the Tree
150	gini	5

Run



Initialized

Рисунок 3.13 – Статус ініціалізовано

Modelling

Play with hyper-parameters to train the best algorithm

Setup

Alphabet size	Word2Vec vector size	Algorithm
200	150	Random Forest
Number of estimators	Criterion	The Maximum Depth of the Tree
150	gini	5

Run



Training

Рисунок 3.14 – Статус навчання.

Modelling

Play with hyper-parameters to train the best algorithm

Setup

Alphabet size	Word2Vec vector size	Algorithm
200	150	Random Forest
Number of estimators	Criterion	The Maximum Depth of the Tree
150	gini	5

Run



Evaluation

Рисунок 3.15 – Статус оцінювання.

Успішне навчання моделі (рисунок 3.16):

Results

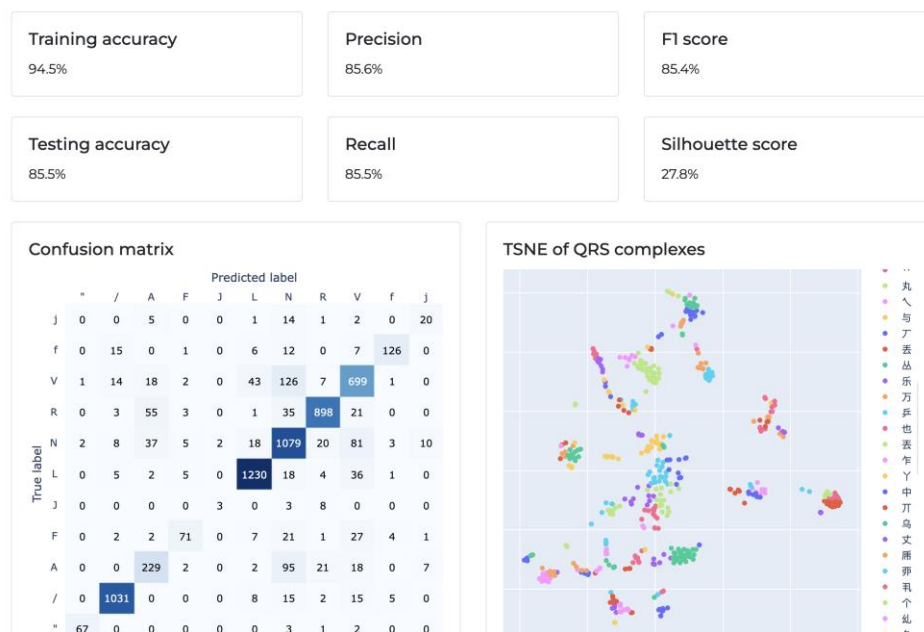


Рисунок 3.16 – Відображення метрик моделі та графіків.

Аналіз.

Завантаження файлу ECG (рисунок 3.17).



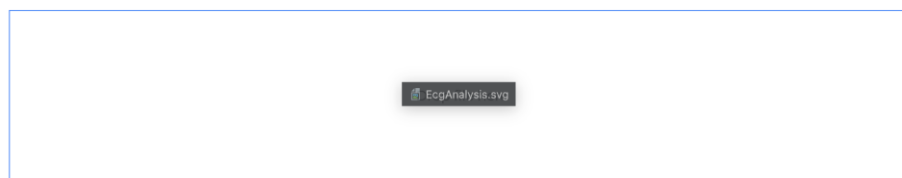
Analysis

Run the prepared model on your own data

Setup

Upload file

Drag your file here or click to upload.



EcgAnalysis.svg

Рисунок 3.17 – Завантаження файлу ECG.

Завантажений файл ECG (рисунок 3.18).

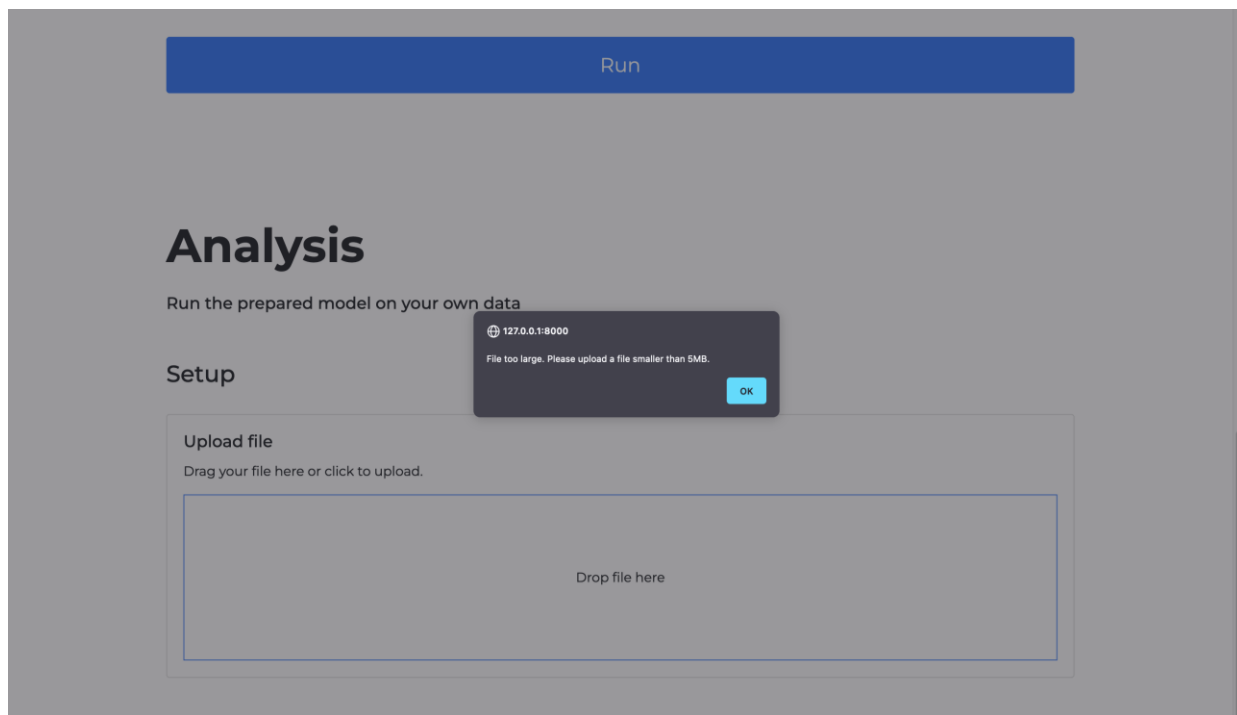


Рисунок 3.18 – Завантажений файл ECG.

Якщо вибраний файл не відповідає формату CSV або XLSX (рисунок 3.19):

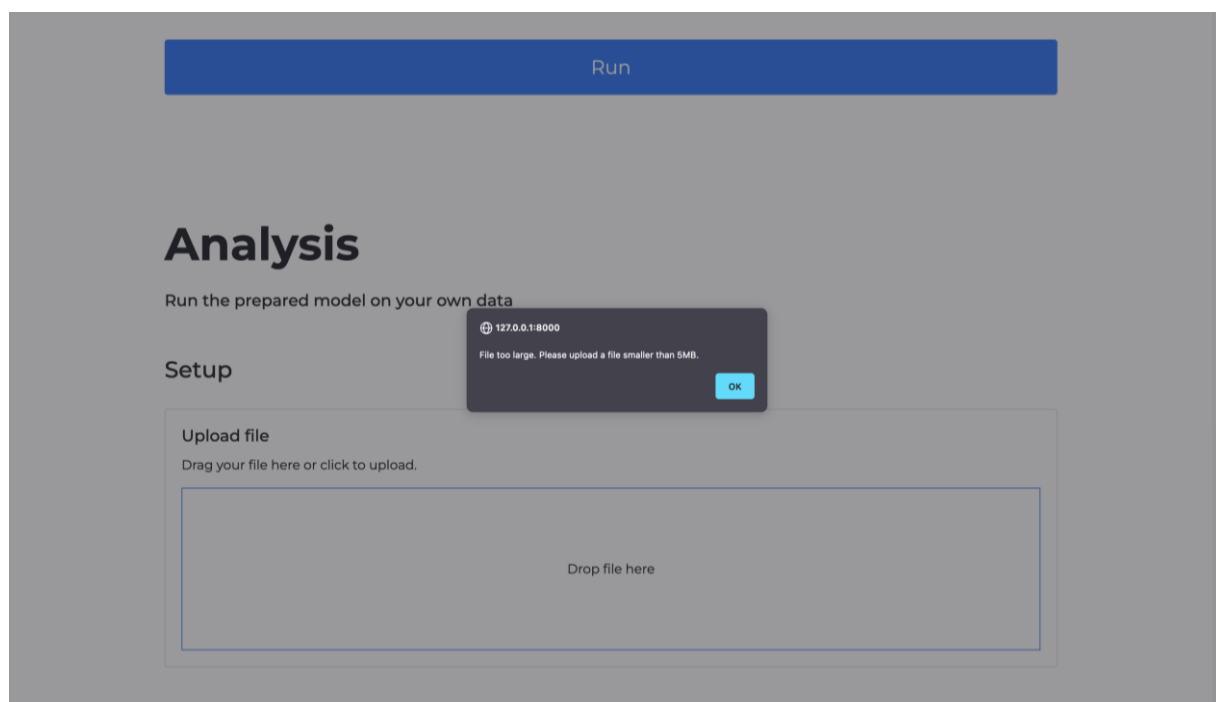


Рисунок 3.19 – Відображення помилки про неправильний формат файлу.

Якщо розмір завантаженого файлу перевищує допустимий ліміт 5MB (рисунок 3.20):

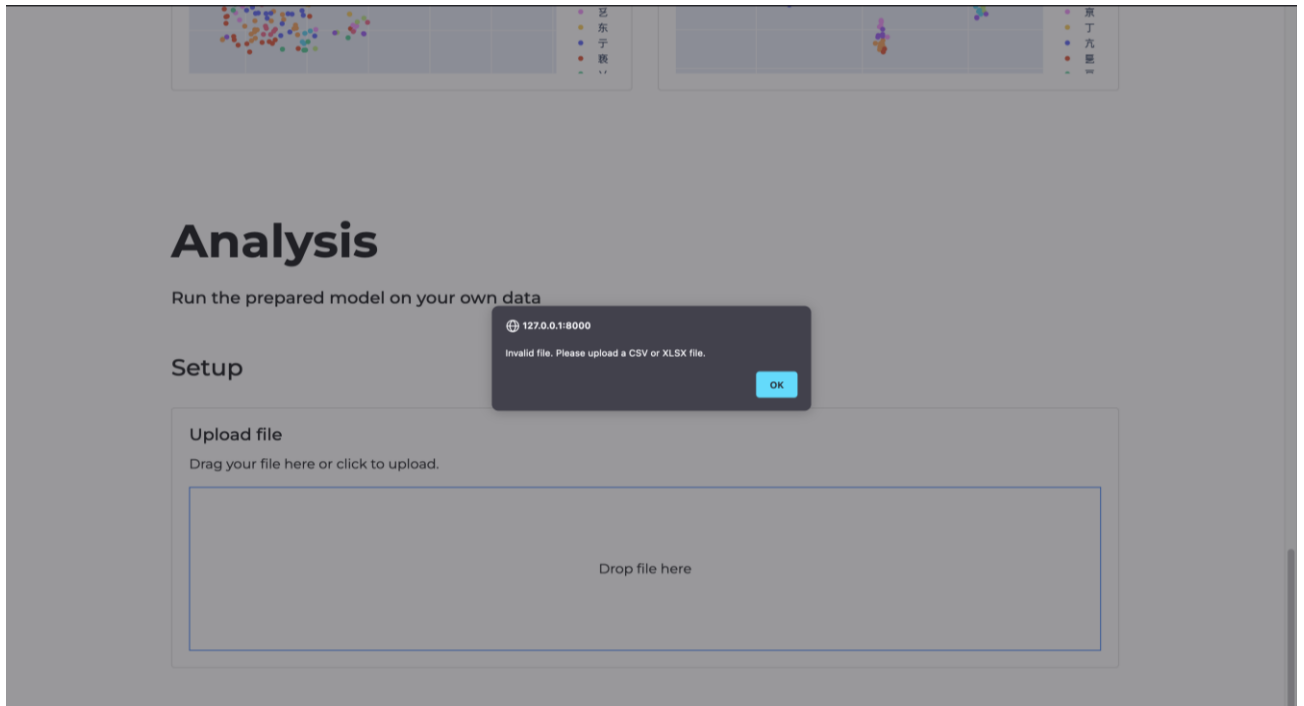


Рисунок 3.20 – Розмір завантаженого файлу перевищує допустимий ліміт

Введення значення FS.

Якщо значення FS введене не коректно:

Відображення помилки про необхідність виправити значення FS (рисунок 3.21).

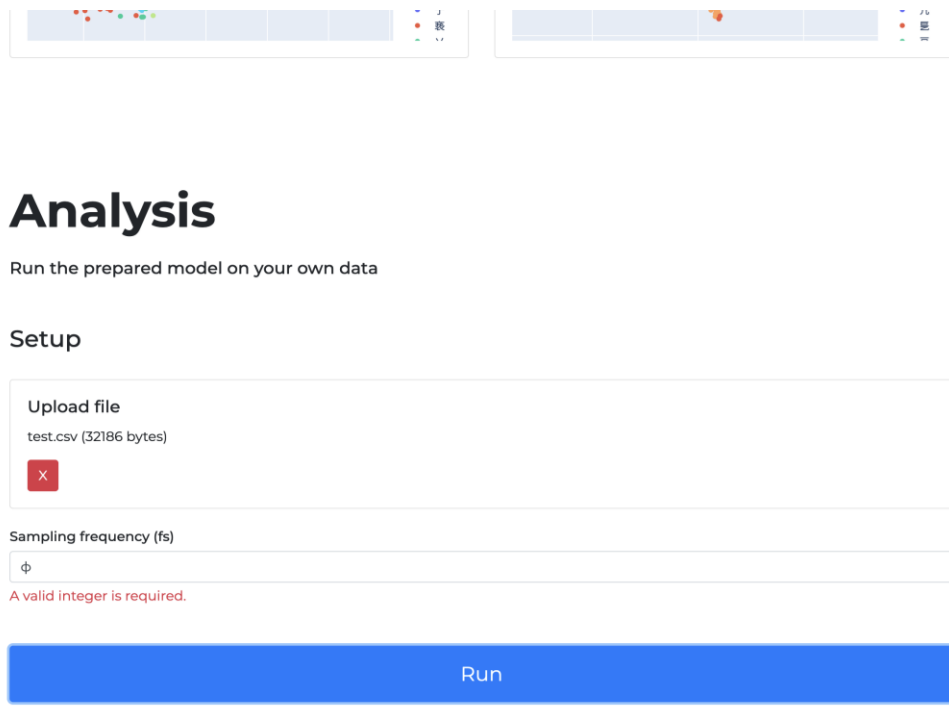


Рисунок 3.21 – Необхідність виправити значення FS.

Аналіз файлу ECG.

Розгалуження: Успішний аналіз файлу:

Відображення графіку з позначеними серцевими складовими та передбаченими мітками (рисунк 3.22).



Рисунок 3.22 – Успішний аналіз файлу

Відображення таблиці з серцебиттями, передбаченими мітками, словами-репрезентаціями серцебиттів, векторними формами слів (рисунк 3.23).

Heartbeats			
Number	Predicted Label	Word	Word Vector
1	Normal beat	个举丑	0.024173393845558167,-0.12893390655517578...
2	Normal beat	不产丰	0.14293770492076874,-0.03004932403564453...
3	Normal beat	不举丑	0.154133141040802,-0.03004932403564453...
4	Normal beat	不产丑	0.1129748523235321,0.002674425719305873...
5	Normal beat	不产丑	0.1129748523235321,0.002674425719305873...
6	Normal beat	不产丰	0.14293770492076874,-0.03004932403564453...
7	Normal beat	乘举丰	0.17141009867191315,0.058469127863645554,-0.4791092574596405,-0.07817833870649338,-0.1277094675064087,0.22540153663022614,-0.16533510386943817,-0.2804848253776989
8	Atrial premature beat	个举丑	-0.2949472665786745,0.058469127863645554,-0.4791092574596405,-0.07817833870649338,-0.1277094675064087,0.22540153663022614,-0.16533510386943817,-0.2804848253776989
9	Normal beat	不产丑	0.1129748523235321,0.002674425719305873...
10	Normal beat	不产丑	0.1129748523235321,0.002674425719305873...
11	Normal beat	不举丰	0.18409597873687744,-0.21342550218105316...
12	Normal beat	乘举丑	0.1414472609758377,-0.10564450174570084...
13	Normal beat	不产丑	0.1129748523235321,0.002674425719305873...
14	Normal beat	不产丰	0.14293770492076874,-0.03004932403564453...
15	Normal beat	不产丑	0.1129748523235321,0.002674425719305873...

Рисунок 3.23 – Таблиця з серцебиттями

Висновки до розділу

У розділі роботи було ретельно проаналізовано якість та проведено тестування розробленого програмного забезпечення ECG Analysis Service.

В підрозділі 3.1 був проведений глибокий аналіз якості ПЗ, що зокрема зосереджувався на оцінці ключових нефункціональних атрибутів. Це включає продуктивність, що визначає, наскільки ефективно програма обробляє великі набори даних ЕКГ; масштабованість, що вимірює спроможність ПЗ ефективно обслуговувати велику кількість користувачів; безпека, яка перевіряє, наскільки добре захищені персональні та медичні дані користувачів; надійність, що оцінює, наскільки стабільно програма працює та наскільки швидко відновлюється після відмов; інтуїтивність, що визначає, наскільки просто та зрозуміло користувачам використовувати інтерфейс програми; сумісність, яка перевіряє, як добре програма працює в різних браузерах та на різних платформах; та доступність, що вимірює, наскільки часто програма є доступною для використання.

Підрозділ 3.2 описує процеси тестування, які були використані для перевірки якості та стабільності програмного забезпечення. Зокрема, це включає різні типи тестування, такі як юніт-тестування, інтеграційне тестування, тестування навантаження та безпеки, а також тестування користувацького інтерфейсу.

Опис контрольного прикладу наведено в підрозділі 3.3 та демонструє, як програмне забезпечення використовується в реальному світі для аналізу та класифікації наборів даних ЕКГ. Це включає показ процесу завантаження файлу ЕКГ, налаштування параметрів аналізу, використання моделей машинного навчання для класифікації серцевих ритмів, а також візуалізацію результатів.

В цілому, на основі проведеного аналізу якості та тестування, можна стверджувати, що ECG Analysis Service є надійним, продуктивним, і

безпечним інструментом для аналізу ЕКГ, що відповідає всім заявленим нефункціональним вимогам.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Планування, конфігурацією та розгортання розробленого програмного забезпечення для аналізу ЕКГ здійснюється на платформі Linode, що надає потрібні ресурси для ефективного функціонування веб-сервісу.

Змінні оточення є ключовим компонентом процесу розгортання нашого програмного забезпечення. Вони використовуються для зберігання конфіденційної інформації, такої як ключі API [67], паролі до баз даних, та інших секретів, які необхідно заховати від відкритого доступу. Такий підхід підвищує безпеку нашого додатку, оскільки ці дані залишаються недоступними для зовнішніх користувачів.

Під час розгортання Docker image [68], змінні оточення встановлюються в середовищі Linode [69]. Оскільки ці дані важливі для правильного функціонування додатку, забезпечується їх належне зберігання та захист (рисунок 4.1).



```
1 SECRET_KEY=django-insecure-vyq52v*4x8kuv3nq5q5)4d$qjm=uceb26jfls$d!n(+i+mugl5
2 DEBUG=1
3 CELERY_BROKER=redis://redis:6379/0
4 CELERY_BACKEND=redis://redis:6379/0
5
6 DATABASE_NAME=ecganalysis
7 DATABASE_USER=ecganalysisuser
8 DATABASE_PASSWORD=password
9 DATABASE_HOST=db
10 DATABASE_PORT=5432
```

Рисунок 4.1 – Змінні оточення серверної частини

– SECRET_KEY: Цей ключ використовується Django для забезпечення криптографічної безпеки. Це дуже важливо для забезпечення безпеки даних у вашому додатку;

- **DEBUG:** Це налаштування Django, яке вказує, чи ввімкнено режим налагодження. Коли це значення встановлено як 1, Django надає детальні повідомлення про помилки для допомоги при налагодженні;
- **CELERY_BROKER і CELERY_BACKEND:** Ці параметри вказують на адресу посередника (broker) та бекенду, які використовуються Celery для обробки асинхронних завдань;
- **DATABASE_NAME, DATABASE_USER, DATABASE_PASSWORD, DATABASE_HOST і DATABASE_PORT:** Ці змінні використовуються для конфігурації бази даних PostgreSQL. Вони вказують ім'я бази даних, ім'я користувача, пароль, хост та порт, відповідно. Процес розгортання відбувається в кілька етапів. По-перше, новий код додатку завантажується у гілку 'main' нашого репозиторію. Тоді, за допомогою Dockerfile, що знаходиться в проекті, створюється Docker image у середовищі Linode. Dockerfile включає всі необхідні вказівки для створення образу Docker, який включає все необхідне програмне забезпечення для запуску нашого веб-додатку.

```
Dockerfile-django x  entrypoint-django.sh x  index.js x  settings.py x  Dockerfile-nginx x  nginx.c v
1 FROM python:3.10-slim as builder
2
3 # Set working directory
4 WORKDIR /usr/src/app
5
6 # Copy requirements file
7 COPY requirements.txt .
8
9 # Upgrade pip and install Python packages
10 RUN pip install --no-cache-dir --upgrade pip && pip wheel --no-cache-dir --no-deps --wheel-dir /usr/
11
12
13 # pull official base image
14 FROM python:3.10-slim
15
16 # Set working directory
17 WORKDIR /home/app/ecganalysis
18
19 # Create app user and set ownership
20 RUN mkdir -p ./staticfiles && mkdir -p ./media && groupadd -r app && useradd -r -g app app && chown
21
22 RUN apt-get update && \
23     apt-get install -y tk && \
24     rm -rf /var/lib/apt/lists/*
25
26 # Copy wheels and requirements from builder stage
27 COPY --from=builder /usr/src/app/wheels /wheels
28 COPY --from=builder /usr/src/app/requirements.txt .
29
30 # Install Python packages from wheels
31 RUN pip install --no-cache-dir /wheels/*
32
33 # Copy project files
34 COPY . .
35
36 # Make entrypoint file executable
37 RUN chmod +x ./bin/entrypoint-django.sh
38 RUN chmod +x ./bin/entrypoint-celery.sh
39
40 # Switch to app user
41 USER root
42
43 # Set environment variables
44 ENV PYTHONDONTWRITEBYTECODE 1
45 ENV PYTHONUNBUFFERED 1
46
47 # Expose port
48 EXPOSE 8000
```

Рисунок 4.2 – Код з Dockerfile

```
Dockerfile-django x entypoint-django.sh x index.js x settings.py x Dockerfile-nginx x nginx.c v
1 # Use the Nginx image as the base image
2 FROM nginx:1.21.0-alpine
3
4 # Install wget and bash
5 RUN apk add --no-cache wget bash
6
7 # Download the wait-for-it.sh script
8 RUN wget https://raw.githubusercontent.com/vishnubob/wait-for-it/master/wait-for-it.sh -O /wait-for-i
9
10 # Make the script executable
11 RUN chmod +x /wait-for-it.sh
12
13 # Copy Nginx configuration file
14 COPY ./nginx.conf /etc/nginx/conf.d/default.conf
15
```

Рисунок 4.3 – Код з Dockerfile-nginx

Запускаємо докер файл командами зображеною на рисунку 4.4 - 4.5 маємо результат, щоб запинити процес використовуємо команду рисунку 4.6 - 4.7.

```
EcgAnalysis ~/PycharmProjects/I
> analysis
> bin
  > entypoint-celery.sh
  > entypoint-django.sh
  > start-server.sh
  > stop-server.sh
> common
1 #!/bin/sh
2
3 python manage.py create_cache_folders
4 python manage.py initial_setup
5
6 docker-compose up -d --build
```

Рисунок 4.4 –Команда для запуску

```
(venv) root@localhost:~/projects/ecganalysis# ./bin/start-server.sh
WARN[0000] The "qin" variable is not set. Defaulting to a blank string.
WARN[0000] The "d" variable is not set. Defaulting to a blank string.
WARN[0000] The "qin" variable is not set. Defaulting to a blank string.
WARN[0000] The "d" variable is not set. Defaulting to a blank string.
WARN[0000] The "qin" variable is not set. Defaulting to a blank string.
WARN[0000] The "d" variable is not set. Defaulting to a blank string.
[+] Building 3.0s (40/40) FINISHED
=> [web internal] load build definition from Dockerfile-django
=> => transferring dockerfile: 1.27kB
=> [web internal] load .dockerignore
=> => transferring context: 146B
=> [web internal] load metadata for docker.io/library/python:3.10-slim
=> [celery internal] load build definition from Dockerfile-django
=> => transferring dockerfile: 1.27kB
=> [celery internal] load .dockerignore
=> => transferring context: 146B
=> [web builder 1/4] FROM docker.io/library/python:3.10-slim@sha256:fd86924ba14682eb11a3c244f60a35b5dfe3267cb726d883fb5c14813ce926f1
=> [web internal] load build context
=> => transferring context: 20.34kB
=> [celery internal] load build context
=> => transferring context: 165.74MB
=> CACHED [celery stage-1 2/10] WORKDIR /home/app/ecganalysis
=> CACHED [celery stage-1 3/10] RUN mkdir -p ./staticfiles && mkdir -p ./media && groupadd -r app && useradd -r -g app app && chown -R app:app /home/app/ecganalysis && chmod -R 755 /home/app/ecganalysis
=> CACHED [celery stage-1 4/10] RUN apt-get update && apt-get install -y tk && rm -rf /var/lib/apt/lists/*
=> CACHED [celery builder 2/4] WORKDIR /usr/src/app
=> CACHED [web builder 3/4] COPY requirements.txt .
=> CACHED [web builder 4/4] RUN pip install --no-cache-dir --upgrade pip && pip wheel --no-cache-dir --no-deps --wheel-dir /usr/src/app/wheels -r requirements.txt
=> CACHED [web stage-1 5/10] COPY --from=builder /usr/src/app/wheels /wheels
=> CACHED [web stage-1 6/10] COPY --from=builder /usr/src/app/requirements.txt .
=> CACHED [web stage-1 7/10] RUN pip install --no-cache-dir /wheels/*
=> CACHED [web stage-1 8/10] COPY . .
=> CACHED [web stage-1 9/10] RUN chmod +x ./bin/entrypoint-django.sh
=> CACHED [web stage-1 10/10] RUN chmod +x ./bin/entrypoint-celery.sh
=> [web] exporting to image
=> => exporting layers
=> => writing image sha256:f40fed0cfd05676a859fa1d92260ec9e8fbd6a0b09433136a48028fc7c791
=> => naming to docker.io/library/ecganalysis-web
=> [nginx internal] load build definition from Dockerfile-nginx
=> => transferring dockerfile: 453B
=> [nginx internal] load .dockerignore
=> => transferring context: 146B
=> [nginx internal] load metadata for docker.io/library/nginx:1.21.0-alpine
=> [nginx 1/5] FROM docker.io/library/nginx:1.21.0-alpine@sha256:cc8c413c74abd9fef9dae7f3da736725136bad1e3f24fb93788aea1944f51c4
=> [nginx internal] load build context
=> => transferring context: 32B
=> CACHED [nginx 2/5] RUN apk add --no-cache wget bash
=> CACHED [nginx 3/5] RUN wget https://raw.githubusercontent.com/vishnubob/wait-for-it/master/wait-for-it.sh -O /wait-for-it.sh
=> CACHED [nginx 4/5] RUN chmod +x /wait-for-it.sh
=> CACHED [nginx 5/5] COPY ./nginx.conf /etc/nginx/conf.d/default.conf
=> [nginx] exporting to image
=> => exporting layers
=> => writing image sha256:51f545d7c0a1f8b9ee58ed837e3a53888f5ef3079ac008d0818ce77ea2c64fa
=> => naming to docker.io/library/ecganalysis-nginx
=> CACHED [celery builder 3/4] COPY requirements.txt .
=> CACHED [celery builder 4/4] RUN pip install --no-cache-dir --upgrade pip && pip wheel --no-cache-dir --no-deps --wheel-dir /usr/src/app/wheels -r requirements.txt
=> CACHED [celery stage-1 5/10] COPY --from=builder /usr/src/app/wheels /wheels
=> CACHED [celery stage-1 6/10] COPY --from=builder /usr/src/app/requirements.txt .
=> CACHED [celery stage-1 7/10] RUN pip install --no-cache-dir /wheels/*
=> CACHED [celery stage-1 8/10] COPY . .
=> CACHED [celery stage-1 9/10] RUN chmod +x ./bin/entrypoint-django.sh
=> CACHED [celery stage-1 10/10] RUN chmod +x ./bin/entrypoint-celery.sh
=> [celery] exporting to image
=> => exporting layers
=> => writing image sha256:0d9213fdbf60e0b1ee1d707db954a6a00d63415720c8390e55723ab6332b08e
=> => naming to docker.io/library/ecganalysis-celery
[+] Running 6/6
✔ Network ecganalysis.default Created
✔ Container ecganalysis-db-1 Started
✔ Container ecganalysis-redis-1 Started
✔ Container ecganalysis-celery-1 Started
✔ Container ecganalysis-web-1 Started
✔ Container ecganalysis-nginx-1 Started
(venv) root@localhost:~/projects/ecganalysis#
```

Рисунок 4.5 – Результат запуски команди старт

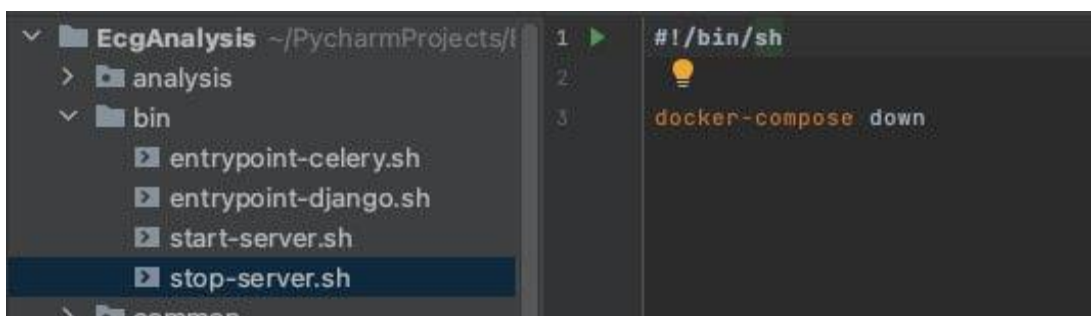


Рисунок 4.6 – Результат запуски команди старт

```
(venv) root@localhost:~/projects/ecganalysis# ./bin/stop-server.sh
WARN[0000] The "gjm" variable is not set. Defaulting to a blank string.
WARN[0000] The "d" variable is not set. Defaulting to a blank string.
WARN[0000] The "gjm" variable is not set. Defaulting to a blank string.
WARN[0000] The "d" variable is not set. Defaulting to a blank string.
WARN[0000] The "gjm" variable is not set. Defaulting to a blank string.
WARN[0000] The "d" variable is not set. Defaulting to a blank string.
[+] Running 6/6
  ✓ Container ecganalysis-nginx-1 Removed
  ✓ Container ecganalysis-celery-1 Removed
  ✓ Container ecganalysis-web-1 Removed
  ✓ Container ecganalysis-redis-1 Removed
  ✓ Container ecganalysis-db-1 Removed
  ✓ Network ecganalysis_default Removed
(venv) root@localhost:~/projects/ecganalysis#
```

```
static-volume:/home/as
redis:
  image: redis:7-alpine
  0.2s
volumes:
  static-volume:
  0.2s
  db-data:
  0.1s
```

Рисунок 4.7 – Результат запуски команди стоп

Після створення Docker image, він розгортається на платформі Linode, що дозволяє нашому веб-сервісу виконувати моделювання і аналіз ЕКГ у реальному часі. Цей процес розгортання можна візуалізувати як послідовність кроків, що представлені на рисунках 4.8.

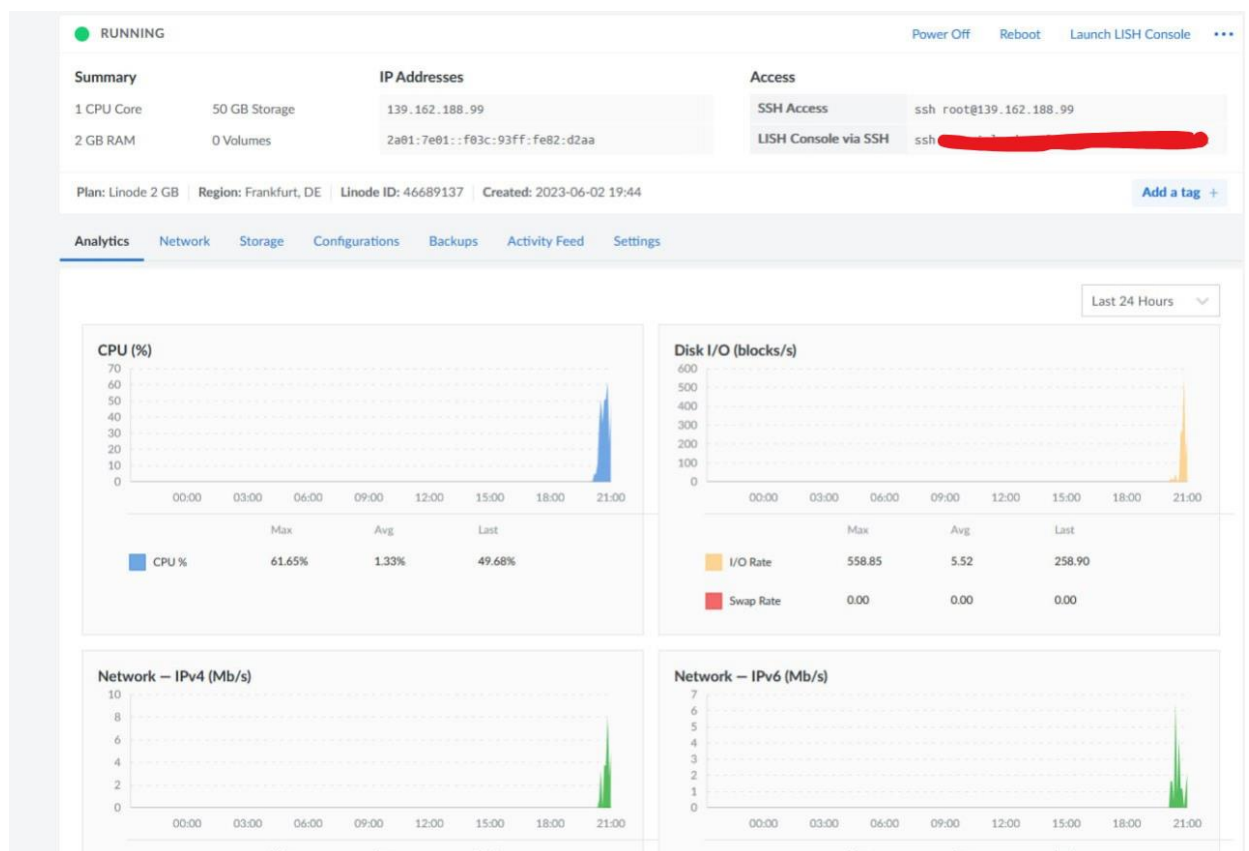


Рисунок 4.8 – Розгортання проєкту на платформі Linode

Програмне забезпечення, розгорнуте на платформі Linode, гарантує швидкість, надійність та безпеку, що є критично важливими для аналізу медичних даних, таких як ЕКГ. Використання Docker також дозволяє гарантувати однорідність середовища розгортання, що спрощує подальше оновлення та масштабування програмного забезпечення.

4.2 Підтримка програмного забезпечення

Забезпечення ефективної підтримки програмного забезпечення є важливим аспектом успіху будь-якого проекту. У контексті проекту ECG аналізу, ми прагнемо забезпечити безперебійну роботу веб-сайту та надати нашим користувачам найкращий досвід використання нашого сервісу. Через поєднання технічних навичок і зосередженості на потребах користувачів, наша команда підтримки програмного забезпечення готова забезпечити ефективну та компетентну підтримку для кожного нашого клієнта.

Моніторинг та управління: Підтримка програмного забезпечення буде включати моніторинг роботи системи, бази даних та інфраструктури веб-сайту. Це дозволяє вчасно виявляти будь-які аномалії, проблеми з продуктивністю або потенційні загрози. Крім того, можуть бути налаштовані системи сповіщень або попереджень, які повідомляють про можливі проблеми.

Виявлення та усунення помилок: Коли виявляються помилки або проблеми, вони будуть аналізовані та виправлені. Це включає розслідування причин помилок, тестування в різних середовищах та розробку патчів або оновлень, які виправляють проблему. Мета полягає у забезпеченні безперебійної роботи системи та максимальній функціональності.

Оновлення та покращення: Залежно від потреб і зворотного зв'язку користувачів, підтримка може займатися розробкою оновлень та покращень. Це може включати додавання нових функціональностей, оптимізацію роботи системи або вирішення запитів користувачів. Оновлення можуть бути розгорнуті на серверах з мінімальними перервами у роботі системи.

Висновки до розділу

Розгортання програмного забезпечення відбувається з використанням платформи Linode, яка надає необхідні ресурси для ефективної роботи нашого веб-сервісу з аналізу ЕКГ. Під час розгортання, важливу роль відіграють

змінні оточення, які забезпечують зберігання конфіденційної інформації, такої як ключі API та паролі до баз даних. Це підвищує безпеку системи та захищає цінні дані від несанкціонованого доступу.

Підтримка програмного забезпечення включає в себе постійне спостереження за роботою сервісу, збір та аналіз помилок, швидке реагування на виявлені проблеми та їх виправлення. Крім того, підтримка включає в себе надання технічної підтримки користувачам, відповіді на їх запитання та вирішення їх проблем.

У процесі розгортання програмного забезпечення було успішно встановлено та налаштовано всі необхідні компоненти, забезпечено безпеку даних та належну доступність сервісу. Під час підтримки було здійснено моніторинг роботи системи, виявлено й виправлено помилки та надано технічну підтримку користувачам.

Завдяки розгортанню та підтримці програмного забезпечення було досягнуто успіху у функціонуванні сервісу аналізу ЕКГ, що дозволяє користувачам ефективно аналізувати та інтерпретувати серцеві скорочення. Постійне оновлення та вдосконалення функціональності сервісу сприятимуть подальшому розвитку та задоволенню потреб користувачів.

ВИСНОВКИ

Завершуючи дипломний проєкт, ми отримали важливі наукові й практичні результати, які є суттєвими в контексті сучасного рівня наукових і технічних знань. Оцінюючи отримані результати, ми встановили, що вони відповідають сучасним тенденціям та прогресу в області аналізу ЕКГ.

Впровадження дипломного проєкту має великий потенціал у різних галузях і сферах. Застосування розробленого сервісу може бути ефективним у медичних установах, лабораторіях та клінічних дослідженнях, де аналіз ЕКГ-зображень є важливим етапом діагностики та моніторингу серцево-судинної системи. Також можливі галузі використання включають наукові дослідження в області кардіології та розвитку нових методів аналізу біомедичних сигналів.

Робота має значення не лише з наукової, а й з науково-технічної та соціально-економічної точок зору. Наукова значущість полягає в застосуванні передових технологій, таких як Docker, Celery, Redis, Django та MihrilJS, для розробки сервісу. Технічна значущість виявляється у створенні ефективної архітектури, добре працюючих алгоритмів класифікації та інтерактивного інтерфейсу користувача. Соціально-економічна значущість проявляється у поліпшенні діагностики та лікування серцевих захворювань, що може призвести до зменшення витрат на організацію медичної допомоги та покращення якості життя пацієнтів.

Проєкт також відкриває перспективи для подальших досліджень у відповідній тематиці. Можливі напрями розвитку включають розширення функціональності, вдосконалення алгоритмів класифікації, використання нових методів машинного навчання та розробку мобільних додатків для зручного доступу до аналізу ЕКГ. Доцільність подальших досліджень у цій області обумовлена потребою у вдосконаленні методів аналізу біомедичних сигналів, поліпшенні точності класифікації та розробки нових підходів до виявлення серцевих аномалій.

У результаті виконання дипломного проєкту було успішно спроектовано та реалізовано веб-сервіс для аналізу ЕКГ. В якості середовища розробки було обрано PyCharm, а для зберігання даних була використана база даних SQLite. Проєкт пройшов етап тестування на різних Pylint, Flake8, Pydeps for modelling application, Radon cyclomatic complexity, Radon lines of code, Bandit, що гарантує його якість написання.

Новизна дипломного проєкту виявляється у вперше реалізованій можливості запитів від пацієнта до лікаря, що покращує взаємодію між медичним персоналом та пацієнтами. Проєкт також модифікував та вдосконалив існуючі рішення для забезпечення більш точного та зручного аналізу ЕКГ. В результаті роботи над проєктом, були досягнуті певні інноваційні рішення, які можуть бути подальше розвинуті і застосовані у відповідних областях.

					КПІ.ПІ-9202.045420.02.81	91
Змін.	Арк.	№ докум.	Підп.	Дата.		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Sanders W. B. Heart Disease. A textbook of Cardiovascular Medicine. / Sanders. // 1. – 1980. – С. 1427.
- 2) Django [Електронний ресурс]. - URL: <https://metanit.com/python/django/> (дата звернення: 15.03.2023)
- 3) ЕЛЕКТРОКАРДІОГРАФІЯ - Велика Медична Енциклопедія [Електронний ресурс]. - URL: <https://бмэ.орг/index.php> (дата звернення: 15.03.2023)
- 4) Електрокардіографічні (ЕКГ) відведення [Електронний ресурс]. - URL: <https://meduniver.com/Medical/Therapy/119.html> (дата звернення: 15.03.2023)
- 5) Стандартні відведення Ейнтховена [Електронний ресурс]. - URL: <https://www.wikilectures.eu/w/File:ECG-Einthoven-triangle.svg> (дата звернення: 15.03.2023)
- 6) Посилені відведення Гольдбергера [Електронний ресурс]. - URL: <https://helpiks.org/helpiksorg/baza2/18046138346.files/image096.gif> (дата звернення: 15.03.2023)
- 7) Грудні відведення Вільсона [Електронний ресурс]. - URL: https://images.slideplayer.biz.tr/7/1957914/slides/slide_5.jpg (дата звернення: 15.03.2023)
- 8) Схематичне зображення електричного поля серця за схемою Уоллера [Електронний ресурс]. - URL: <https://i1.wp.com/thoracickey.com/wp-content/uploads/2019/12/C10-FF1.gif?w=960> (дата звернення: 15.03.2023)
- 9) ЕКГ в нормі [Електронний ресурс]. - URL: https://meduniver.com/Medical/cardiology/Img/ekg_v_norme.jpg (дата звернення: 15.03.2023)
- 10) Схема нормальної ЕКГ людини [Електронний ресурс]. - URL: https://media-ecn.s3.amazonaws.com/embedded_image/2016/05/QNX-AED.jpg. (дата звернення: 15.03.2023)

11) Центри контролю та профілактики захворювань, Медіатека громадського здоров'я. Хвороби серця. Національний центр з профілактики хронічних захворювань та протидії хворобам серця і інсульту. [Електронний ресурс]. - URL: <https://www.cdc.gov/dhdspl/> (дата звернення: 15.03.2023)

12) Сандх'я Пруті, МД. ЕЛЕКТРОКАРДІОГРАМА (ЕКГ) АБО (ЕКГ). МАЙО КЛІНІКА. PRC-20214603. [Електронний ресурс]. - URL: <https://www.mayoclinic.org/biographies/pruthi-sandhya-m-d/bio-20053925> (дата звернення: 15.03.2023)

13) Гогас, П. і Пападимітріу, Т. Презентація Машинного навчання / П. Гогас, Т. Пападимітріу. - 2015.

14) Батіста, Ж., Казейро, Р., Генрікс, Ж.Ф. & Мартінс, П. Високошвидкісне відстеження з ядерними кореляційними фільтрами / Ж. Батіста, Р. Казейро, Ж.Ф. Генрікс, П. Мартінс // CoRR. - 2015. [Електронний ресурс]. - URL: <https://arxiv.org/pdf/1404.7584.pdf> (дата звернення: 15.03.2023).

15) Чжан, М. і Чжоу, З. Огляд алгоритмів багатоетикеткового навчання / М. Чжан, З. Чжоу // IEEE TKDE. - 2014. [Електронний ресурс]. - URL: <https://www.computer.org/csdl/journal/tk/2014/08/06471714/13rRUwInvBt> (дата звернення: 15.03.2023).

16) Стефан Драйзайтль, Лусіла Оно-Мачадо. Методологічний огляд класифікаційних моделей логістичної регресії та штучних нейронних мереж / С. Драйзайтль, Л. Оно-Мачадо // Журнал біомедичної інформатики. - 2002. - С. 352-359.

17) Брайман, Л. та ін. Класифікація та регресійні дерева / Л. Брайман. - Вадсворт, 1984. - с.300.

18) Джурафськи, Д. та Мартін, Дж. Обробка мови та мовлення / Д. Джурафськи та Дж. Мартін // Авторське право. - 2021. - с.600.

19) Віттен, І. Г. та Френк, Е. Data Mining: Practical Machine Learning Tools and Techniques, 2nd edition / І. Г. Віттен та Е. Френк. - Morgan Kaufmann, 2005. – с.560.

20) Гавхане, А. та ін. Прогнозування хвороби серця за допомогою машинного навчання / А. Гавхане, Г. Коккула, І. Пандья та К. Девадкар // Друга міжнародна конференція з електроніки, зв'язку та аерокосмічної технології (ICECA). - 2018. - с. 1-10.

21) Baklan, I. та ін. Anomalies Detection Approach in Electrocardiogram Analysis Using Linguistic Modeling / I. Baklan, I. Mukha, Y. Oliinyk, K. Lishchuk, E. Nedashkivsky, & O. Gavrilenko // International Conference on Computer Science, Engineering and Education Applications. - Springer, Cham, 2019. - с. 513-522.

22) Обеншейн, М. К. Застосування технік видобутку даних до медичних даних / М. К. Обеншейн // Контроль і епідеміологія в лікарнях. - 2004. - №25(8). - с. 690–695.

23) Nikhar, S. та Karandikar, A. Прогнозування серцевої хвороби за допомогою методів машинного навчання / S. Nikhar та A. Karandikar // Міжнародний журнал з передового інженерного менеджменту та науки (IJAEMS). - 2016. - №2(6). - с. 1-6.

24) Nashif, S. та ін. Виявлення серцевої хвороби за допомогою алгоритмів машинного навчання та системи моніторингу кардіоваскулярного здоров'я в реальному часі / S. Nashif, M. Raihan, M. Islam та M. Imam // Світовий журнал інженерії та технології. - 2018. - №06(04). - с. 854-873.

25) Розумна система передбачення хвороб серця з використанням технік видобутку даних. [Електронний ресурс]. - URL: <https://doi.org/10.1109/aiccsa.2008.4493524> (дата звернення: 27.05.2023).

26) Ramalingam, V. V., Dandapath, A. та Karthik Raja, M. Прогнозування хвороби серця за допомогою методів машинного навчання: огляд / V. V. Ramalingam, A. Dandapath та M. Karthik Raja // Міжнародний журнал інженерії та технології. - 2018. - №7(28). - с. 1-684.

27) Т. Чітамбарам / Н. Логеш Каннан, М. Говсаля // Виявлення хвороби серця за допомогою методів машинного навчання. -. Research Square, 2020. – с. 20-25

28) Shashwith, P., Sai Prasad, B., Shashi Kumar Reddy, C. H., Abhinav Krishna, N., & Ramesh, K. Прогнозування хвороби серця за допомогою алгоритмів машинного навчання // Міжнародний науковий журнал про модернізацію інженерної технології та науки. - 2022. – с. 120-125).

29) Khan, A. H., Hussain, M., & Malik, M. K. Класифікація серцевих порушень за допомогою зондування електрокардіограми з використанням глибоких нейронних мереж // Complexity. - 2021. - с. 1–8.

30) Aspuru, J., Ochoa-Brust, A., Félix, R., Mata-López, W., Mena, L., Ostos, R. & Martínez-Peláez, R. Сегментація сигналу ЕКГ за допомогою алгоритму лінійної регресії // Датчики. - 2019. - с. 775.

31) Acharya, U. R., Fujita, H., Oh, S. L., Hagiwara, Y., Tan, J. H., & Adam, M. Застосування глибоких згорткових нейронних мереж для автоматичного виявлення інфаркту міокарда за допомогою сигналів ЕКГ // Інформаційні науки. - 2017. - с. 190–198.

32) PyCharm: "PyCharm: The Python IDE for Professional Developers" [Електронний ресурс]. - URL: <https://www.jetbrains.com/pycharm/> (дата звернення: 15.03.2023)

33) Google Colab: "Colaboratory: A product from Google Research" [Електронний ресурс]. - URL: <https://research.google.com/colaboratory/faq.html> (дата звернення: 15.03.2023).

34) SQLite: "SQLite: A C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine" [Електронний ресурс]. - URL: <https://www.sqlite.org/index.html> (дата звернення: 15.03.2023).

35) Celery: "Celery: A simple, flexible, and reliable distributed system to process vast amounts of messages" [Електронний ресурс]. - URL: <https://docs.celeryproject.org/en/stable/> (дата звернення: 15.03.2023).

36) Redis: "Redis: The open source, in-memory data store used by millions of developers as a database, cache, streaming engine, and message broker" [Електронний ресурс]. - URL: <https://redis.io/> (дата звернення: 15.03.2023).

37) EcgViewer та його призначення. [Електронний ресурс]. - URL: <https://manualzz.com/doc/7087626/ecgviewer-user-manual-v1.0> (дата звернення: 15.03.2023).

38) CardioDay Holter ECG software. [Електронний ресурс]. - URL: <https://www.gehealthcare.com/products/ecg-analysis-software/cardioday> (дата звернення: 15.03.2023).

39) ADInstruments [Електронний ресурс]. - URL: <https://www.adinstruments.com/> (дата звернення: 15.03.2023).

40) KMeans [Електронний ресурс]. - URL: <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html> (дата звернення: 15.03.2023).

41) Word2Vec [Електронний ресурс]. - URL: <https://gab41.lab41.org/anything2vec-e99ec0dc186> (дата звернення: 15.03.2023).

42) GDPR [Електронний ресурс]. - URL: https://www.dsgvo-portal.de/gdpr_article_42.php (дата звернення: 15.03.2023).

43) Chrome [Електронний ресурс]. - URL: <https://chrome.softwaredownload.co.in/chrome-43-0-2357> (дата звернення: 15.03.2023).

44) Firefox [Електронний ресурс]. - URL: <https://www.mozilla.org/> (дата звернення: 15.03.2023).

45) Edge [Електронний ресурс]. - URL: <https://www.microsoft.com/edge> (дата звернення: 15.03.2023).

46) Windows [Електронний ресурс]. - URL: <https://www.microsoft.com/windows> (дата звернення: 15.03.2023).

47) MacOS [Електронний ресурс]. - URL: <https://support.apple.com/macos> (дата звернення: 15.03.2023).

48) Linux [Електронний ресурс]. - URL: <https://www.linux.org/> (дата звернення: 15.03.2023).

49) Random Forest [Електронний ресурс]. - URL: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html> (дата звернення: 15.03.2023).

50) SVC [Електронний ресурс]. - URL: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html> (дата звернення: 15.03.2023).

51) Django [Електронний ресурс]. - URL: <https://www.djangoproject.com/start/overview/> (дата звернення: 1 грудня 2023).

52) Django: Розробницький фреймворк, що працює на Python [Електронний ресурс]. - URL: <https://www.upwork.com/hiring/development/django-programming/> (дата звернення: 15 січня 2023).

53) Структура Django - погляд з боку [Електронний ресурс]. - URL: <https://djangobook.com/mdj2-django-structure/> (дата звернення: 1 грудня 2023).

54) Django [Електронний ресурс]. - URL: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django> (дата звернення: 15.03.2023).

55) MVS [Електронний ресурс]. - URL: <https://pypi.org/project/mvs/> (дата звернення: 09.06.2023)

56) PEP (Python Enhancement Proposals) [Електронний ресурс]. - URL: <https://peps.python.org/> (дата звернення: 09.06.2023)

57) Pylint [Електронний ресурс]. - URL: <https://www.pylint.org/> (дата звернення: 09.06.2023)

58) Flake8 [Електронний ресурс]. - URL: <https://flake8.pycqa.org/en/latest/> (дата звернення: 09.06.2023)

59) Radon [Електронний ресурс]. - URL: <https://radon.readthedocs.io/en/latest/> (дата звернення: 09.06.2023)

60) CLOC (Count Lines of Code) [Електронний ресурс]. - URL: <https://cloc.sourceforge.net/> (дата звернення: 09.06.2023)

- 61) Pydeps [Електронний ресурс]. - URL:
<https://pydeps.readthedocs.io/en/latest/> (дата звернення: 09.06.2023)
- 62) Bandit [Електронний ресурс]. - URL:
<https://bandit.readthedocs.io/en/latest/> (дата звернення: 09.06.2023)
- 63) Pylint Score [Електронний ресурс]. - URL:
<https://pylint.pycqa.org/en/latest/output.html#score> (дата звернення: 15.03.2023)
- 64) PyFlakes [Електронний ресурс]. - URL:
<https://pyflakes.readthedocs.io/en/latest/> (дата звернення: 15.03.2023).
- 65) McCabe [Електронний ресурс]. - URL:
<https://pypi.org/project/mccabe/> (дата звернення: 15.03.2023).
- 66) Radon lines of code [Електронний ресурс]. - URL:
<https://widyantohadi.medium.com/refactoring-3c7f32d58d43> (дата звернення: 15.03.2023).
- 67) API [Електронний ресурс]. - URL:
<https://www.mulesoft.com/resources/api/what-is-an-api#:~:text=API%20is%20the%20acronym%20for,data%20within%20and%20across%20organizations.> (дата звернення: 15.03.2023).
- 68) Docker image [Електронний ресурс]. - URL:
https://hub.docker.com/_/docker (дата звернення: 15.03.2023).
- 69) Linode [Електронний ресурс]. - URL:
<https://www.linode.com/docs/release-notes/api/v4.69.0/> (дата звернення: 15.03.2023).

ДОДАТОК А ЗВІТ ПОДІБНОСТІ



Ім'я користувача:
Лісовиченко Олег Іванович

Дата перевірки:
04.06.2023 15:31:12 EEST

Дата звіту:
04.06.2023 15:32:33 EEST

ID перевірки:
1015412681

Тип перевірки:
Doc vs Internet + Library

ID користувача:
76913

Назва документа: ІП-92_Бераудо_ПЗ

Кількість сторінок: 92 Кількість слів: 14002 Кількість символів: 105640 Розмір файлу: 8.95 MB ID файлу: 1015075790

4.6% Схожість

Найбільша схожість: 1.61% з джерелом з Бібліотеки (ID файлу: 1015068012)

2.39% Джерела з Інтернету

117

Сторінка 94

4.4% Джерела з Бібліотеки

261

Сторінка 96

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0% Вилучень

Немає вилучених джерел

Модифікації

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

Замінені символи

3

					КПІ.ІП-9202.045420.02.81	99
Змін.	Арк.	№ докум.	Підп.	Дата.		

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2023 р.

**ПРОГРАМНА БІБЛІОТЕКА АНАЛІЗУ ЕКГ ЛІНГВІСТИЧНИМИ
МЕТОДАМИ**

Текст програми

КПІ.ПІ-9202.045420.03.13

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Юрій ОЛІЙНИК

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Еліза БЕРАУДО

Київ – 2023

Файл analysis/constants/__init__.py

```
HEARTBEAT_LABEL_VERBOSE = {
    'N': 'Normal beat',
    'L': 'Left bundle branch block beat',
    'R': 'Right bundle branch block beat',
    'B': 'Bundle branch block beat (unspecified)',
    'A': 'Atrial premature beat',
    'a': 'Aberrated atrial premature beat',
    'J': 'Nodal (junctional) premature beat',
    'S': 'Supraventricular premature or ectopic beat (atrial or nodal)',
    'V': 'Premature ventricular contraction',
    'r': 'R-on-T premature ventricular contraction',
    'F': 'Fusion of ventricular and normal beat',
    'e': 'Atrial escape beat',
    'j': 'Nodal (junctional) escape beat',
    'n': 'Supraventricular escape beat (atrial or nodal)',
    'E': 'Ventricular escape beat',
    '/': 'Paced beat',
    'f': 'Fusion of paced and normal beat',
    'Q': 'Unclassifiable beat',
    '?': 'Beat not classified during learning',
}
```

Файл analysis/constants/enums.py

```
from common.constants.enums import ChoicesEnum

class AnalysisSessionStatus(str, ChoicesEnum):
    INITIALIZED = 'initialized'
    ANALYZING = "analyzing"
    DONE = "done"
```

Файл analysis/serializers/analysis_session.py

```
from rest_framework import serializers

from analysis.models import AnalysisSession

class AnalysisSessionSerializer(serializers.ModelSerializer):
    class Meta:
        model = AnalysisSession
        fields = ('ecg_file', 'fs')
```

Файл analysis/utils/analyzer.py

```
import numpy as np
import pandas as pd
import plotly.graph_objects as go

from analysis.constants import HEARTBEAT_LABEL_VERBOSE
from analysis.utils.functions import extract_heartbeats_and_waves
from modelling.data_preparation.manager import MITDBDatasetManager
```

					КП.ІП-9202.045420.03.13	Арк.
						2
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

from modelling.data_preparation.utils import (
    concatenate_letters,
    convert_labels_to_letters,
    extract_wave_features,
)

class ECGAnalyzer:
    def __init__(self, analysis_session, training_session):
        self._analysis_session = analysis_session
        self._training_session = training_session

    def get_analysis_result(self):
        df = pd.read_csv(self._analysis_session.ecg_file)
        ecg_signal = df["signal"].values

        (
            heartbeat_intervals,
            p_wave_features_scaled,
            qrs_complex_features_scaled,
            t_wave_features_scaled,
        ) = self._prepare_features(ecg_signal)

        words = self._generate_words(
            p_wave_features_scaled, qrs_complex_features_scaled,
            t_wave_features_scaled
        )

        vectors = self._generate_vectors(words)

        predicted_labels = self._predict_labels(vectors)

        chart = self._plot_ecg_with_annotations(
            ecg_signal, heartbeat_intervals, predicted_labels
        )

        return {
            "charts": {"line": chart},
            "table": self._generate_heartbeats_table_data(
                predicted_labels, words, vectors
            ),
        }

    def _prepare_features(self, ecg_signal):
        (
            heartbeats,
            heartbeat_intervals,
            p_waves,
            qrs_complexes,
            t_waves,
        ) = extract_heartbeats_and_waves(ecg_signal,
fs=self._analysis_session.fs)

```

```

        p_wave_features, qrs_complex_features, t_wave_features =
extract_wave_features(
    p_waves, qrs_complexes, t_waves
)

dataset_manager = MITDBDatasetManager()
scaler_p, scaler_qrs, scaler_t =
dataset_manager.load_scalers_from_cache()

p_wave_features_scaled = scaler_p.transform(p_wave_features)
qrs_complex_features_scaled =
scaler_qrs.transform(qrs_complex_features)
t_wave_features_scaled = scaler_t.transform(t_wave_features)
return (
    heartbeat_intervals,
    p_wave_features_scaled,
    qrs_complex_features_scaled,
    t_wave_features_scaled,
)

def _generate_words(self, p_wave_features, qrs_complex_features,
t_wave_features):
    kmeans_p, kmeans_qrs, kmeans_t =
self._training_session.get_kmeans_models()

    labels_p = kmeans_p.predict(p_wave_features)
    labels_qrs = kmeans_qrs.predict(qrs_complex_features)
    labels_t = kmeans_t.predict(t_wave_features)

    letters_p = convert_labels_to_letters(labels_p)
    letters_qrs = convert_labels_to_letters(labels_qrs)
    letters_t = convert_labels_to_letters(labels_t)

    heartbeat_words = concatenate_letters(letters_p, letters_qrs,
letters_t)
    return heartbeat_words

def _generate_vectors(self, words):
    word2vec = self._training_session.get_word2vec_model()

    def word_to_vec(word):
        return np.mean([word2vec.wv[letter] for letter in word], axis=0)

    word_vectors = [word_to_vec(w) for w in words]
    return word_vectors

def _predict_labels(self, vectors):
    classifier = self._training_session.get_classifier_model()

    predicted_labels = [classifier.predict(f.reshape(1, -1)) for f in
vectors]

```

```

predicted_labels_expanded = [
    HEARTBEAT_LABEL_VERBOSE[label[0]] for label in predicted_labels
]

return predicted_labels_expanded

@staticmethod
def _plot_ecg_with_annotations(
    ecg_signal, heartbeat_intervals, predicted_labels_expanded
):
    # create the figure
    fig = go.Figure()

    # add entire ECG signal trace as a gray dotted line
    fig.add_trace(
        go.Scatter(
            x=list(range(len(ecg_signal))),
            y=ecg_signal,
            mode="lines",
            line=dict(color="lightgrey"), # make this line dotted
            hoverinfo="none", # no hover info for this trace
            showlegend=False, # do not show this trace in the legend
        )
    )

    # iterate through heartbeat intervals and predicted labels
    for i, interval in enumerate(heartbeat_intervals):
        start, end = interval
        segment = ecg_signal[start:end]
        label = predicted_labels_expanded[i]

        # set the line color based on the label
        line_color = "#008000" if label.lower() == "normal beat" else
"#800000"

        # add this segment as a new trace to the figure
        fig.add_trace(
            go.Scatter(
                x=list(range(start, end)),
                y=segment,
                mode="lines",
                line=dict(color=line_color, width=2),
                hoverinfo="text",
                hovertext=label,
                showlegend=False,
            )
        )

    # add rangeslider to the xaxis configuration
    fig.update_layout(
        xaxis=dict(rangeslider=dict(visible=True), type="linear"),
        showlegend=True,
        autosize=True,
        title=None,
        xaxis_title=None,
        yaxis_title=None,
        margin=dict(l=0, r=0, t=0, b=0),
    )

```

```

    )

    return fig.to_json()

@staticmethod
def __generate_heartbeats_table_data(
    predicted_labels_expanded, heartbeat_words, features
):
    return [
        {
            "index": i + 1,
            "predicted_label": predicted_labels_expanded[i],
            "word": heartbeat_words[i],
            "word_vector": features[i],
        }
        for i in range(min(20, len(predicted_labels_expanded)))
    ]

```

Файл analysis/utils/functions.py

```

import numpy as np

from biosppy.signals import ecg

def extract_heartbeats_and_waves(ecg_signal, fs: int = 360):
    heartbeats = []
    heartbeat_intervals = []

    p_waves = []
    qrs_complexes = []
    t_waves = []

    out = ecg.ecg(signal=ecg_signal, sampling_rate=fs, show=False)
    rpeaks = out["rpeaks"]
    rr_intervals = np.diff(rpeaks)
    average_rr_interval = np.mean(rr_intervals)

    signal_len = len(ecg_signal)

    for r_peak in rpeaks:
        start, end = (
            max(int(r_peak - average_rr_interval * 0.3), 0),
            min(int(r_peak + average_rr_interval * 0.5), signal_len - 1),
        )

        heartbeat = ecg_signal[start:end]

        # Skip heartbeats below a certain length threshold
        if len(heartbeat) < average_rr_interval * 0.5:
            continue

```

					КП.ІП-9202.045420.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		6

```

        # Append the heartbeat and corresponding interval
        heartbeats.append(heartbeat)
        heartbeat_intervals.append((start, end))

    # Simple approach to extract P-waves, QRS complexes, and T-waves in the
    range of the heartbeat
    p_wave_end = start + int((end - start) * 0.2)
    qrs_complex_end = p_wave_end + int((end - start) * 0.4)

    p_wave = ecg_signal[start:p_wave_end]
    qrs_complex = ecg_signal[p_wave_end:qrs_complex_end]
    t_wave = ecg_signal[qrs_complex_end:end]

    p_waves.append(p_wave)
    qrs_complexes.append(qrs_complex)
    t_waves.append(t_wave)

return heartbeats, heartbeat_intervals, p_waves, qrs_complexes, t_waves

```

Файл analysis/admin.py

```

from django.contrib import admin

from analysis.models import AnalysisSession

@admin.register(AnalysisSession)
class AnalysisSessionAdmin(admin.ModelAdmin):
    list_display = (
        "id",
        "created_at",
    )
    ordering = ("-created_at",)

```

Файл analysis/apps.py

```

from django.apps import AppConfig

class AnalysisConfig(AppConfig):
    default_auto_field = "django.db.models.BigAutoField"
    name = "analysis"

```

Файл analysis/models.py

```

import uuid

from django.core.validators import FileExtensionValidator
from django.db import models
from django.utils import timezone

```

					КП.ІП-9202.045420.03.13	Арк.
						7
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

def get_ecg_file_upload_to(_, filename):
    date = timezone.now().date()
    return f"media/ecg/{date}/{filename}"

class AnalysisSession(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)

    ecg_file = models.FileField(
        upload_to=get_ecg_file_upload_to,
        validators=[FileExtensionValidator(allowed_extensions=["csv",
"xlsx"])]),
    )
    fs = models.IntegerField(default=360)

    modified_at = models.DateTimeField(auto_now=True)
    created_at = models.DateTimeField(auto_now_add=True)

```

Файл analysis/urls.py

```

from django.urls import path

from . import views
from .apps import AnalysisConfig

app_name = AnalysisConfig.name

urlpatterns = [
    path("session/", views.AnalysisSessionView.as_view(), name="session-
create"),
]

```

Файл analysis/views.py

```

from rest_framework.exceptions import NotFound, ValidationError
from rest_framework.generics import GenericAPIView
from rest_framework.response import Response
from rest_framework.views import APIView

from modelling.models import TrainingSession

from .models import AnalysisSession
from .serializers.analysis_session import AnalysisSessionSerializer
from .utils.analyzer import ECGAnalyzer

class AnalysisSessionView(GenericAPIView):
    serializer_class = AnalysisSessionSerializer

    def post(self, request):
        serializer = self.get_serializer(data=request.data)

```

					КП.ІП-9202.045420.03.13	Арк.
						8
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

        serializer.is_valid(raise_exception=True)

        analysis_session = serializer.save()

        training_session = self._get_training_session()

        analyzer = ECGAnalyzer(analysis_session, training_session)
        data = analyzer.get_analysis_result()

        return Response(data)

    def _get_training_session(self):
        training_session_id = self.request.data.get("training_session_id")
        training_session = TrainingSession.objects.filter(
            pk=training_session_id
        ).first()
        if not training_session:
            raise NotFound(
                {"detail": ["Training session with this ID could not be
found."]}
            )

        return training_session

```

Файл common/constants/enums.py

```
from enum import Enum
```

```

class ChoicesEnum(Enum):
    @classmethod
    def choices(cls):
        return [(x.value, x.name.replace("_", " ").title()) for x in cls]

```

Файл EcgAnalysis/__init__.py

```
from .celery import app as celery_app
```

```
__all__ = ("celery_app",)
```

Файл EcgAnalysis/asgi.py

```
"""
```

```
ASGI config for EcgAnalysis project.
```

```

It exposes the ASGI callable as a module-level variable named
`application`.

```

```

For more information on this file, see
https://docs.djangoproject.com/en/4.2/howto/deployment/asgi/
"""

```

					КП.ІП-9202.045420.03.13	Арк.
						9
Змін.	Арк.	№ докум.	Підп.	Дата.		

```
import os

from django.core.asgi import get_asgi_application

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "EcgAnalysis.settings")

application = get_asgi_application()
```

Файл EcgAnalysis/celery.py

```
import os

from celery import Celery

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "EcgAnalysis.settings")

app = Celery("EcgAnalysis")
app.config_from_object("django.conf:settings", namespace="CELERY")
app.autodiscover_tasks()
```

Файл EcgAnalysis/settings.py

```
import os

from dotenv import load_dotenv

load_dotenv()

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = os.getenv('SECRET_KEY')

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['*']

# Application definition

INSTALLED_APPS = [
```

					КП.ІП-9202.045420.03.13	Арк.
						10
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

"django.contrib.admin",
"django.contrib.auth",
"django.contrib.contenttypes",
"django.contrib.sessions",
"django.contrib.messages",
"django.contrib.staticfiles",
# custom applications
"analysis.apps.AnalysisConfig",
"modelling.apps.ModellingConfig",
# 3rd party applications
"rest_framework",
]

MIDDLEWARE = [
    "django.middleware.security.SecurityMiddleware",
    "django.contrib.sessions.middleware.SessionMiddleware",
    "django.middleware.common.CommonMiddleware",
    "django.middleware.csrf.CsrfViewMiddleware",
    "django.contrib.auth.middleware.AuthenticationMiddleware",
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
]

ROOT_URLCONF = "EcgAnalysis.urls"

TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "DIRS": [os.path.join(BASE_DIR, "templates")],
        "APP_DIRS": True,
        "OPTIONS": {
            "context_processors": [
                "django.template.context_processors.debug",
                "django.template.context_processors.request",
                "django.contrib.auth.context_processors.auth",
                "django.contrib.messages.context_processors.messages",
            ],
        },
    },
]

WSGI_APPLICATION = "EcgAnalysis.wsgi.application"

# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.postgresql",
        "NAME": os.getenv("DATABASE_NAME"),
        "USER": os.getenv("DATABASE_USER"),
        "PASSWORD": os.getenv("DATABASE_PASSWORD"),
        "HOST": os.getenv("DATABASE_HOST"),
        "PORT": os.getenv("DATABASE_PORT"),
    }
}

```

```

    }
}

# Password validation
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        "NAME":
"django.contrib.auth.password_validation.UserAttributeSimilarityValidator",
    },
    {
        "NAME":
"django.contrib.auth.password_validation.MinimumLengthValidator",
    },
    {
        "NAME":
"django.contrib.auth.password_validation.CommonPasswordValidator",
    },
    {
        "NAME":
"django.contrib.auth.password_validation.NumericPasswordValidator",
    },
]

# Internationalization
# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE_CODE = "en-us"

TIME_ZONE = "UTC"

USE_I18N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC_URL = "/static/"

STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static"),
]

```

					КП.ІП-9202.045420.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		12

```

STATIC_ROOT = os.path.join(BASE_DIR, "staticfiles")

# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = "django.db.models.BigAutoField"

LOGGING = {
    "version": 1,
    "disable_existing_loggers": False,
    "formatters": {
        "verbose": {
            "format": "{levelname} {asctime} {module} {message}",
            "style": "{",
        },
    },
    "handlers": {
        "console": {
            "level": "INFO",
            "class": "logging.StreamHandler",
            "formatter": "verbose",
        },
    },
    "loggers": {
        "django": {
            "handlers": ["console"],
            "level": "INFO",
            "propagate": True,
        },
        "ecg_analysis": {
            "handlers": ["console"],
            "level": "INFO",
            "propagate": True,
        },
    },
}

CELERY_BROKER_URL = os.environ.get("CELERY_BROKER", "redis://localhost:6379")
CELERY_RESULT_BACKEND = os.environ.get("CELERY_BACKEND",
"redis://localhost:6379")

```

Файл EcgAnalysis/urls.py

```

from django.conf import settings
from django.conf.urls.static import static
from django.contrib import admin
from django.urls import include, path
from django.views.generic import TemplateView

urlpatterns = [
    path("admin/", admin.site.urls),
    path(
        "api/",
        include(
            [

```

```

        path("modelling/", include("modelling.urls")),
        path("analysis/", include("analysis.urls")),
    ],
),
]

urlpatterns += [path("", TemplateView.as_view(template_name="index.html"))]

if settings.DEBUG:
    urlpatterns += static(settings.STATIC_URL,
document_root=settings.STATIC_ROOT)

```

Файл EcgAnalysis/wsgi.py

```

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "EcgAnalysis.settings")

application = get_wsgi_application()

```

Файл modelling/constants/__init__.py

```

import os

from django.conf import settings

from modelling.apps import ModellingConfig

MITDB = "mitdb"
SCALER = "scaler"
KMEANS = "kmeans"
WORD2VEC = "word2vec"
CLASSIFIER = "classifier"

MODELLING_APP_DIR = os.path.join(settings.BASE_DIR, ModellingConfig.name)

MODELS_CACHE_DIR = os.path.join(MODELLING_APP_DIR, "models_cache")
SCALER_MODELS_CACHE_DIR = os.path.join(MODELS_CACHE_DIR, SCALER)
KMEANS_MODELS_CACHE_DIR = os.path.join(MODELS_CACHE_DIR, KMEANS)
WORD2VEC_MODELS_CACHE_DIR = os.path.join(MODELS_CACHE_DIR, WORD2VEC)
CLASSIFIER_MODELS_CACHE_DIR = os.path.join(MODELS_CACHE_DIR, CLASSIFIER)

DATA_PREPARATION_CACHE_DIR = os.path.join(
    MODELLING_APP_DIR, "data_preparation", "cache"
)
MITDB_DATASET_DIR = os.path.join(DATA_PREPARATION_CACHE_DIR, "datasets",
MITDB)
MITDB_FEATURES_DIR = os.path.join(DATA_PREPARATION_CACHE_DIR, "features",
MITDB)

```

Файл modelling/constants/enums.py

```
from common.constants.enums import ChoicesEnum

class ClassificationAlgorithm(str, ChoicesEnum):
    KNN = "knn"
    SVC = "svc"
    RANDOM_FOREST = "random_forest"

class TrainingSessionStatus(str, ChoicesEnum):
    INITIALIZED = "initialized"
    TRAINING = "training"
    EVALUATION = "evaluation"
    DONE = "done"
```

Файл modelling/data_preparation/manager.py

```
import logging
import os
import pickle
import random
from collections import Counter

import joblib
import numpy as np
import wfdb
from biosppy.signals import ecg
from sklearn.preprocessing import StandardScaler
from tqdm import tqdm

from modelling.constants import (
    MITDB,
    MITDB_DATASET_DIR,
    MITDB_FEATURES_DIR,
    SCALER_MODELS_CACHE_DIR,
)
from modelling.data_preparation.utils import extract_wave_features

logger = logging.getLogger("ecg_analysis")

class MITDBDatasetManager:
    def download_dataset(self):
        if not self._is_dataset_downloaded:
            logger.info("MIT dataset is not found. Starting the download process...")

            wfdb.dl_database(MITDB, MITDB_DATASET_DIR)

    def get_records_and_annotations(self):
```

					КП.ІП-9202.045420.03.13	Арк.
						15
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

self.download_dataset()

records, annotations = [], []

for i in self.wfdb_indices:
    kwargs = {"record_name": f"{MITDB_DATASET_DIR}/{i}", "sampto":
500000}

    record = wfdb.rdsamp(**kwargs)
    annotation = wfdb.rdann(**kwargs, extension="atr")

    records.append(record)
    annotations.append(annotation)

return records, annotations

@staticmethod
def extract_heartbeats_and_waves_from_records(records, annotations):
    heartbeats = []
    heartbeat_annotations = []

    p_waves = []
    qrs_complexes = []
    t_waves = []

    for record, annotation in tqdm(
        zip(records, annotations),
        total=len(records),
        desc="Extracting heartbeats, heartbeat annotations and P/QRS/T
waves...",
    ):
        fs = record[1]["fs"]

        ecg_signal = record[0][:, 1]
        signal_len = len(ecg_signal)

        out = ecg.ecg(signal=ecg_signal, sampling_rate=fs, show=False)
        rpeaks = out["rpeaks"]
        rr_intervals = np.diff(rpeaks)
        average_rr_interval = np.mean(rr_intervals)

        samples = annotation.sample
        symbols = annotation.symbol

        for r_peak in rpeaks:
            # define the start and end of a heartbeat
            start, end = (
                max(int(r_peak - average_rr_interval * 0.3), 0),
                min(int(r_peak + average_rr_interval * 0.5), signal_len -
1),
            )

```

```

# calculate boundaries for annotations
lower_bound, upper_bound = (
    max(r_peak - average_rr_interval * 0.2, start),
    min(r_peak + average_rr_interval * 0.2, end),
)

# find indices of annotations within boundaries
annotation_indices = np.where(
    (samples >= lower_bound) & (samples <= upper_bound)
)[0]

# skip if there are no annotations in the range
if len(annotation_indices) == 0:
    continue

# find the closest annotation to the r peak
closest_annotated_r_peak = annotation_indices[
    np.argmin(np.abs(samples[annotation_indices] - r_peak))
]

heartbeat = ecg_signal[start:end]

# Skip '+' annotations
heartbeat_annotation = symbols[closest_annotated_r_peak]
if heartbeat_annotation == "+":
    continue

# Skip heartbeats below a certain length threshold
if len(heartbeat) < average_rr_interval * 0.5:
    continue

# Append the heartbeat and corresponding annotation
heartbeats.append(heartbeat)
heartbeat_annotations.append(heartbeat_annotation)

# Simple approach to extract P-waves, QRS complexes, and T-
waves in the range of the heartbeat
p_wave_end = start + int((end - start) * 0.2)
qrs_complex_end = p_wave_end + int((end - start) * 0.4)

p_wave = ecg_signal[start:p_wave_end]
qrs_complex = ecg_signal[p_wave_end:qrs_complex_end]
t_wave = ecg_signal[qrs_complex_end:end]

p_waves.append(p_wave)
qrs_complexes.append(qrs_complex)
t_waves.append(t_wave)

return heartbeats, heartbeat_annotations, p_waves, qrs_complexes,
t_waves

```

```

@staticmethod
def balance_heartbeats_and_waves(
    heartbeats, heartbeat_annotations, p_waves, qrs_complexes, t_waves
):
    # Get the counter of the annotations
    counter = Counter(heartbeat_annotations)

    # Find classes with count < 75
    low_count_classes = [k for k, v in counter.items() if v < 75]

    # Downsample 'N' class to the count of 'L' class
    downsampled_N = random.sample(
        [
            i
            for i in range(len(heartbeat_annotations))
            if heartbeat_annotations[i] == "N"
        ],
        counter["L"],
    )

    indices_to_keep = downsampled_N + [
        i
        for i in range(len(heartbeat_annotations))
        if heartbeat_annotations[i] not in ["N"] + low_count_classes
    ]

    # Create balanced data
    balanced_heartbeats = [heartbeats[i] for i in indices_to_keep]
    balanced_annotations = [heartbeat_annotations[i] for i in
indices_to_keep]
    balanced_p_waves = [p_waves[i] for i in indices_to_keep]
    balanced_qrs_complexes = [qrs_complexes[i] for i in indices_to_keep]
    balanced_t_waves = [t_waves[i] for i in indices_to_keep]

    return (
        balanced_heartbeats,
        balanced_annotations,
        balanced_p_waves,
        balanced_qrs_complexes,
        balanced_t_waves,
    )

@staticmethod
def scale_wave_features(p_wave_features, qrs_complex_features,
t_wave_features):
    scaler_p = StandardScaler()
    scaler_qrs = StandardScaler()
    scaler_t = StandardScaler()

    p_wave_features_scaled = scaler_p.fit_transform(p_wave_features)
    qrs_complex_features_scaled =
scaler_qrs.fit_transform(qrs_complex_features)
    t_wave_features_scaled = scaler_t.fit_transform(t_wave_features)

    return (
        p_wave_features_scaled,

```

```

        qrs_complex_features_scaled,
        t_wave_features_scaled,
        scaler_p,
        scaler_qrs,
        scaler_t,
    )

    @staticmethod
    def cache_scalers(
        scaler_p,
        scaler_qrs,
        scaler_t,
    ):
        scalars = {
            "scaler_p": scaler_p,
            "scaler_qrs": scaler_qrs,
            "scaler_t": scaler_t,
        }

        for name, scaler in scalars.items():
            joblib.dump(scaler, os.path.join(SCALER_MODELS_CACHE_DIR,
                f"{name}.pkl"))

    @staticmethod
    def cache_wave_features_and_annotations(
        p_wave_features,
        qrs_complex_features,
        t_wave_features,
        heartbeat_annotations,
    ):
        features = {
            "p_wave": p_wave_features,
            "qrs_complex": qrs_complex_features,
            "t_wave": t_wave_features,
            "heartbeat_annotations": heartbeat_annotations,
        }

        for name, feature in features.items():
            with open(os.path.join(MITDB_FEATURES_DIR, f"{name}.pkl"), "wb") as
f:
                pickle.dump(feature, f)

    def cache_features(self):
        if self._is_featureset_cached:
            return

        records, annotations = self.get_records_and_annotations()
        logger.info(
            "MIT dataset records and annotations have been successfully
gathered."
        )

        (
            heartbeats,
            heartbeat_annotations,
            p_waves,
            qrs_complexes,

```

```

        t_waves,
    ) = self.extract_heartbeats_and_waves_from_records(records,
annotations)
    logger.info(
        "MIT dataset heartbeats, heartbeat waves, corresponding annotations
have been successfully extracted."
    )

    (
        heartbeats,
        heartbeat_annotations,
        p_waves,
        qrs_complexes,
        t_waves,
    ) = self.balance_heartbeats_and_waves(
        heartbeats,
        heartbeat_annotations,
        p_waves,
        qrs_complexes,
        t_waves,
    )
    logger.info("MIT dataset has been successfully balanced.")

    (
        p_wave_features,
        qrs_complex_features,
        t_wave_features,
    ) = extract_wave_features(p_waves, qrs_complexes, t_waves)
    logger.info(
        "MIT dataset heartbeat wave features have been successfully
extracted."
    )

    (
        p_wave_features_scaled,
        qrs_complex_features_scaled,
        t_wave_features_scaled,
        scaler_p,
        scaler_qrs,
        scaler_t,
    ) = self.scale_wave_features(
        p_wave_features, qrs_complex_features, t_wave_features
    )
    logger.info(
        "MIT dataset heartbeat wave features have been successfully
scaled."
    )

    self.cache_scalers(
        scaler_p,
        scaler_qrs,
        scaler_t,
    )
    logger.info(
        "StandardScaler() objects for heartbeat wave features have been
successfully saved."
    )

    self.cache_wave_features_and_annotations(

```

```

        p_wave_features_scaled,
        qrs_complex_features_scaled,
        t_wave_features_scaled,
        heartbeat_annotations,
    )
    logger.info("MIT dataset features and annotations were successfully
cached.")

    @staticmethod
    def load_scalers_from_cache():
        return (
            joblib.load(
                os.path.join(os.path.join(SCALER_MODELS_CACHE_DIR,
"scaler_p.pkl"))
            ),
            joblib.load(
                os.path.join(os.path.join(SCALER_MODELS_CACHE_DIR,
"scaler_qrs.pkl"))
            ),
            joblib.load(
                os.path.join(os.path.join(SCALER_MODELS_CACHE_DIR,
"scaler_t.pkl"))
            ),
        )

    @staticmethod
    def load_features_and_annotations_from_cache():
        with open(os.path.join(MITDB_FEATURES_DIR, "p_wave.pkl"), "rb") as f:
            p_wave_features = pickle.load(f) # nsec

        with open(os.path.join(MITDB_FEATURES_DIR, "qrs_complex.pkl"), "rb") as
f:
            qrs_complex_features = pickle.load(f) # nsec

        with open(os.path.join(MITDB_FEATURES_DIR, "t_wave.pkl"), "rb") as f:
            t_wave_features = pickle.load(f) # nsec

        with open(
            os.path.join(MITDB_FEATURES_DIR, "heartbeat_annotations.pkl"), "rb"
        ) as f:
            heartbeat_annotations = pickle.load(f) # nsec

        return (
            p_wave_features,
            qrs_complex_features,
            t_wave_features,
            heartbeat_annotations,
        )

    @property
    def _is_dataset_downloaded(self):
        return os.path.isdir(MITDB_DATASET_DIR) and
len(os.listdir(MITDB_DATASET_DIR))

    @property
    def _is_featureset_cached(self):

```

```

        return len(os.listdir(MITDB_FEATURES_DIR)) > 0

@property
def _wfdb_indices(self):
    return wfdb.get_record_list(MITDB)

```

Файл modelling/data_preparation/utils.py

```

import numpy as np
import scipy

def to_letter(cluster_num):
    return chr(19968 + cluster_num)

def convert_labels_to_letters(labels):
    return [to_letter(label) for label in labels]

def concatenate_letters(letters_p, letters_qrs, letters_t):
    return [p + qrs + t for p, qrs, t in zip(letters_p, letters_qrs, letters_t)]

def extract_wave_features(p_waves, qrs_complexes, t_waves):
    p_wave_features = []
    qrs_complex_features = []
    t_wave_features = []
    print("sadsdf")
    for p_wave, qrs_complex, t_wave in zip(p_waves, qrs_complexes, t_waves):
        # Compute features for P-wave
        p_wave_feature = _compute_features(p_wave)
        p_wave_features.append(p_wave_feature)

        # Compute features for QRS complex
        qrs_complex_feature = _compute_features(qrs_complex)
        qrs_complex_features.append(qrs_complex_feature)

        # Compute features for T-wave
        t_wave_feature = _compute_features(t_wave)
        t_wave_features.append(t_wave_feature)
    print("12234234")
    return p_wave_features, qrs_complex_features, t_wave_features

def word_to_vec(word, word2vec):
    return np.mean([word2vec.wv[letter] for letter in word], axis=0)

```

```

def _compute_features(wave):
    # Time-domain features
    mean = np.mean(wave)
    median = np.median(wave)
    std = np.std(wave)
    max_val = np.max(wave)
    min_val = np.min(wave)
    skewness = scipy.stats.skew(wave)

    # Calculate the FFT (Fast Fourier Transform) for frequency analysis
    fft = np.fft.fft(wave)

    # Calculate absolute value of fft to get power spectrum
    power = np.abs(fft)

    # Normalize power
    power_normalized = power / np.sum(power)

    # Calculate power spectral density
    psd = np.abs(fft) ** 2

    # Frequency-domain features
    spectral_entropy = -np.sum(
        power_normalized * np.log2(power_normalized + np.finfo(float).eps)
    )
    spectral_centroid = np.sum(np.arange(1, len(psd) + 1) * psd) / np.sum(psd)
    fundamental_freq = np.argmax(power)

    # Waveform characteristics
    wave_duration = len(wave)
    wave_amplitude = np.max(wave) - np.min(wave)

    return [
        mean,
        median,
        std,
        max_val,
        min_val,
        skewness,
        spectral_entropy,
        spectral_centroid,
        fundamental_freq,
        wave_duration,
        wave_amplitude,
    ]

```

Файл modelling/management/commands/create_cache_folders.py

```
import os
```

```
from django.core.management import BaseCommand
```

```
from modelling.constants import DATA_PREPARATION_CACHE_DIR,
MITDB_DATASET_DIR, MITDB_FEATURES_DIR, MODELS_CACHE_DIR, \
```

					КП.ІП-9202.045420.03.13	Арк.
						23
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

SCALER_MODELS_CACHE_DIR, KMEANS_MODELS_CACHE_DIR,
WORD2VEC_MODELS_CACHE_DIR, CLASSIFIER_MODELS_CACHE_DIR

```

```

class Command(BaseCommand):
    def handle(self, *args, **kwargs):
        if not os.path.isdir(DATA_PREPARATION_CACHE_DIR):
            os.mkdir(DATA_PREPARATION_CACHE_DIR)

        if not os.path.isdir(MITDB_DATASET_DIR):
            os.makedirs(MITDB_DATASET_DIR)

        if not os.path.isdir(MITDB_FEATURES_DIR):
            os.makedirs(MITDB_FEATURES_DIR)

        if not os.path.isdir(MODELS_CACHE_DIR):
            os.mkdir(MODELS_CACHE_DIR)

        if not os.path.isdir(SCALER_MODELS_CACHE_DIR):
            os.mkdir(SCALER_MODELS_CACHE_DIR)

        if not os.path.isdir(KMEANS_MODELS_CACHE_DIR):
            os.mkdir(KMEANS_MODELS_CACHE_DIR)

        if not os.path.isdir(WORD2VEC_MODELS_CACHE_DIR):
            os.mkdir(WORD2VEC_MODELS_CACHE_DIR)

        if not os.path.isdir(CLASSIFIER_MODELS_CACHE_DIR):
            os.mkdir(CLASSIFIER_MODELS_CACHE_DIR)

```

Файл modelling/management/commands/initial_setup.py

```

import os

from django.core.management import BaseCommand

from modelling.constants import (
    CLASSIFIER_MODELS_CACHE_DIR,
    DATA_PREPARATION_CACHE_DIR,
    KMEANS_MODELS_CACHE_DIR,
    MITDB_DATASET_DIR,
    MITDB_FEATURES_DIR,
    MODELS_CACHE_DIR,
    SCALER_MODELS_CACHE_DIR,
    WORD2VEC_MODELS_CACHE_DIR,
)
from modelling.data_preparation.manager import MITDBDatasetManager

```

```
class Command(BaseCommand):
    def handle(self, *args, **kwargs):
        manager = MITDBDatasetManager()
        manager.cache_features()
```

Файл modelling/serializers/__init__.py

```
from .algorithms_params import (
    KNNParamsSerializer,
    RandomForestParamsSerializer,
    SVCParamsSerializer,
)
from .general_params import TrainingSessionGeneralParamsSerializer
from .training_session import TrainingSessionSerializer
```

Файл modelling/serializers/algorithms_params.py

```
from rest_framework import serializers

class KNNParamsSerializer(serializers.Serializer):
    n_neighbors = serializers.IntegerField(required=False, default=5)
    algorithm = serializers.ChoiceField(
        required=False,
        default="auto",
        choices=["auto", "ball_tree", "kd_tree", "brute"],
    )
    weights = serializers.ChoiceField(
        required=False, default="uniform", choices=["uniform", "distance"]
    )

    def create(self, *args, **kwargs):
        pass

    def update(self, *args, **kwargs):
        pass

class SVCParamsSerializer(serializers.Serializer):
    C = serializers.FloatField(required=False, default=1.0)
    kernel = serializers.ChoiceField(
        required=False, default="rbf", choices=["linear", "poly", "rbf",
        "sigmoid"]
    )
    degree = serializers.IntegerField(required=False, default=3)

    def create(self, *args, **kwargs):
        pass

    def update(self, *args, **kwargs):
        pass
```

```

class RandomForestParamsSerializer(serializers.Serializer):
    n_estimators = serializers.IntegerField(required=False, default=100)
    criterion = serializers.ChoiceField(
        required=False, default="gini", choices=["gini", "entropy"]
    )
    max_depth = serializers.IntegerField(required=False, allow_null=True,
    default=None)

    def create(self, *args, **kwargs):
        pass

    def update(self, *args, **kwargs):
        pass

```

Файл modelling/serializers/general_params.py

```

from rest_framework import serializers

from modelling.constants.enums import ClassificationAlgorithm

class TrainingSessionGeneralParamsSerializer(serializers.Serializer):
    alphabet_size = serializers.IntegerField()
    word2vec_vector_size = serializers.IntegerField(required=False,
    default=100)
    algorithm = serializers.ChoiceField(
        choices=ClassificationAlgorithm.choices(),
        error_messages={"invalid_choice": "Invalid choice."},
    )

    def create(self, validated_data):
        pass

    def update(self, instance, validated_data):
        pass

```

Файл modelling/serializers/training_session.py

```

from rest_framework import serializers

from modelling.models import TrainingSession

class TrainingSessionSerializer(serializers.ModelSerializer):
    class Meta:
        model = TrainingSession
        fields = (
            "status",
            "train_accuracy",
            "test_accuracy",
            "precision",
            "recall",
            "f1_score",
            "silhouette_score",
        )

```

```

        "charts",
    )

    def to_representation(self, instance):
        representation = super().to_representation(instance)
        for field in [
            "train_accuracy",
            "test_accuracy",
            "precision",
            "recall",
            "f1_score",
            "silhouette_score",
        ]:
            if representation[field] is not None:
                representation[field] = "{:.1f}%".format(representation[field]
* 100)
        return representation

```

Файл modelling/training/manager.py

```

import io
import json
from typing import Union

import joblib
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.figure_factory as ff
from django.core.files.base import ContentFile
from gensim.models import Word2Vec
from sklearn.cluster import KMeans
from sklearn.ensemble import RandomForestClassifier
from sklearn.manifold import TSNE
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    f1_score,
    precision_score,
    recall_score,
    silhouette_score,
)
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

from modelling.constants.enums import ClassificationAlgorithm,
TrainingSessionStatus
from modelling.data_preparation.manager import MITDBDatasetManager
from modelling.data_preparation.utils import (
    concatenate_letters,
    convert_labels_to_letters,
    word_to_vec,
)

ALGORITHM_CLASS = {
    ClassificationAlgorithm.KNN: KNeighborsClassifier,

```

					КП.ІП-9202.045420.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		27

```

ClassificationAlgorithm.SVC: SVC,
ClassificationAlgorithm.RANDOM_FOREST: RandomForestClassifier,
}

class TrainingManager:
    kmeans_p: KMeans
    kmeans_qrs: KMeans
    kmeans_t: KMeans
    word2vec: Word2Vec
    classifier: Union[KNeighborsClassifier, SVC, RandomForestClassifier]

    X_train: list
    X_test: list
    y_train: list
    y_test: list

    p_wave_features: np.ndarray
    qrs_complex_features: np.ndarray
    t_wave_features: np.ndarray

    labels_p: np.ndarray
    labels_qrs: np.ndarray
    labels_t: np.ndarray

    letters_p: np.ndarray
    letters_qrs: np.ndarray
    letters_t: np.ndarray

    def __init__(self, training_session, general_params, algorithm_params):
        self._session = training_session

        self.__general_params = general_params
        self.__algorithm_params = algorithm_params

    def run(self):
        self._session.status = TrainingSessionStatus.TRAINING
        self._session.classifier_name = self.__general_params["algorithm"]
        self._session.save()

        self._train()

        self._session.status = TrainingSessionStatus.EVALUATION
        self._session.save()

        self._evaluate()

        self._save()

        self._session.status = TrainingSessionStatus.DONE

```

```

        self._session.save()

def _train(self):
    dataset_manager = MITDBDatasetManager()

    (
        self.p_wave_features,
        self.qrs_complex_features,
        self.t_wave_features,
        heartbeat_annotations,
    ) = dataset_manager.load_features_and_annotations_from_cache()

    words = self._generate_words()

    vectors = self._generate_vectors(words)

    self._fit_classifier(vectors, heartbeat_annotations)

def _evaluate(self):
    # Predictions on training data
    y_train_pred = self.classifier.predict(self.X_train)

    # Predictions on testing data
    y_test_pred = self.classifier.predict(self.X_test)

    # Calculate metrics for training data
    train_accuracy = accuracy_score(self.y_train, y_train_pred)

    # Calculate metrics for testing data
    test_accuracy = accuracy_score(self.y_test, y_test_pred)
    precision = precision_score(self.y_test, y_test_pred,
    average="weighted")
    recall = recall_score(self.y_test, y_test_pred, average="weighted")
    f1 = f1_score(self.y_test, y_test_pred, average="weighted")

    silhouette_p = silhouette_score(self.p_wave_features, self.labels_p)
    silhouette_qrs = silhouette_score(self.qrs_complex_features,
    self.labels_qrs)
    silhouette_t = silhouette_score(self.t_wave_features, self.labels_t)
    silhouette_avg = (silhouette_p + silhouette_qrs + silhouette_t) / 3

    charts = {
        "tsne": self._get_tsne_plots(),
        "confusion_matrix": self._get_confusion_matrix_plot(y_test_pred),
    }

    self._session.train_accuracy = train_accuracy
    self._session.test_accuracy = test_accuracy
    self._session.precision = precision
    self._session.recall = recall
    self._session.f1_score = f1
    self._session.silhouette_score = silhouette_avg

```

```

        self._session.charts = json.dumps(charts)

    def _save(self):
        self.__save_model_to_field(
            self.kmeans_p, self._session.kmeans_p_model,
            f"p_{self._session.id}"
        )
        self.__save_model_to_field(
            self.kmeans_qrs, self._session.kmeans_qrs_model,
            f"qrs_{self._session.id}"
        )
        self.__save_model_to_field(
            self.kmeans_t, self._session.kmeans_t_model,
            f"t_{self._session.id}"
        )
        self.__save_model_to_field(
            self.word2vec, self._session.word2vec_model, f"{self._session.id}"
        )
        self.__save_model_to_field(
            self.classifier, self._session.classifier_model,
            f"{self._session.id}"
        )

    def _generate_words(self):
        n_clusters = self.__general_params["alphabet_size"]

        self.kmeans_p = KMeans(n_clusters=n_clusters)
        self.kmeans_qrs = KMeans(n_clusters=n_clusters)
        self.kmeans_t = KMeans(n_clusters=n_clusters)

        self.labels_p = self.kmeans_p.fit_predict(self.p_wave_features)
        self.labels_qrs = self.kmeans_qrs.fit_predict(self.qrs_complex_features)
        self.labels_t = self.kmeans_t.fit_predict(self.t_wave_features)

        self.letters_p = convert_labels_to_letters(self.labels_p)
        self.letters_qrs = convert_labels_to_letters(self.labels_qrs)
        self.letters_t = convert_labels_to_letters(self.labels_t)

        words = concatenate_letters(self.letters_p, self.letters_qrs,
            self.letters_t)

        return words

    def _generate_vectors(self, words):
        sentences = [list(word) for word in words]

        self.word2vec = Word2Vec(sentences, vector_size=50, window=3,
            min_count=1)
        vectors = [word_to_vec(word, self.word2vec) for word in words]

        return vectors

```

```

def _fit_classifier(self, vectors, heartbeat_annotations):
    self.X_train, self.X_test, self.y_train, self.y_test =
train_test_split(
    vectors, heartbeat_annotations, test_size=0.2, random_state=42
)

    classifier_cls = ALGORITHM_CLASS[self.__general_params["algorithm"]]
    classifier = classifier_cls(**self.__algorithm_params)
    classifier.fit(self.X_train, self.y_train)

    self.classifier = classifier

def _get_tsne_plots(
    self,
):
    sample_size = 500

    p_wave_features_sample, letters_p_sample = self.__sample_features(
        self.p_wave_features, self.letters_p, sample_size
    )
    qrs_complex_features_sample, letters_qrs_sample =
self.__sample_features(
    self.qrs_complex_features, self.letters_qrs, sample_size
)
    t_wave_features_sample, letters_t_sample = self.__sample_features(
        self.t_wave_features, self.letters_t, sample_size
    )

    # Plot t-SNE for each wave sample
    p_tsne_fig = self.__tsne_plot(p_wave_features_sample, letters_p_sample)
    qrs_tsne_fig = self.__tsne_plot(qrs_complex_features_sample,
letters_qrs_sample)
    t_tsne_fig = self.__tsne_plot(t_wave_features_sample, letters_t_sample)

    return {
        "p": p_tsne_fig.to_json(),
        "qrs": qrs_tsne_fig.to_json(),
        "t": t_tsne_fig.to_json(),
    }

def _get_confusion_matrix_plot(self, y_pred):
    # Compute confusion matrix
    cm = confusion_matrix(self.y_test, y_pred)

    # Turn it into a DataFrame
    cm_df = pd.DataFrame(
        cm, index=self.classifier.classes_,
columns=self.classifier.classes_
    )

    # Create the Plotly Figure
    fig = ff.create_annotated_heatmap(
        z=cm_df.values,
        x=list(cm_df.columns),
        y=list(cm_df.index),

```

```

        colorscale="Blues",
    )

    fig.update_layout(
        xaxis=dict(title="Predicted label"),
        yaxis=dict(title="True label"),
        autosize=True,
        margin=dict(l=0, r=0, b=0, t=0, pad=0),
    )

    return fig.to_json()

    @staticmethod
    def __tsne_plot(features, labels):
        tsne = TSNE(n_components=2)
        features_2d = tsne.fit_transform(features)

        # Create dataframe for Plotly
        df = pd.DataFrame(
            data={
                "x": features_2d[:, 0],
                "y": features_2d[:, 1],
                "Cluster": labels,
            }
        )

        # Create a plotly express scatter plot
        fig = px.scatter(df, x="x", y="y", color="Cluster")

        # Remove axis labels
        fig.update_xaxes(showticklabels=False)
        fig.update_yaxes(showticklabels=False)

        # Remove chart borders
        fig.update_xaxes(showline=False, zeroline=False)
        fig.update_yaxes(showline=False, zeroline=False)

        # Remove title and axis titles
        fig.update_layout(
            showlegend=True,
            autosize=True,
            title=None,
            xaxis_title=None,
            yaxis_title=None,
            margin=dict(l=0, r=0, t=0, b=0), # This line reduces the chart
borders
        )

        return fig

    @staticmethod
    def __save_model_to_field(model, field, name):
        # Dump the model into a bytes buffer
        buffer = io.BytesIO()

```

```
joblib.dump(model, buffer)
buffer.seek(0)
```

```
# Save the buffer content to the field
field.save(f"{name}.pkl", ContentFile(buffer.read()))
```

```
@staticmethod
def __sample_features(features, labels, size):
    features = np.array(features)
    labels = np.array(labels)
    if len(features) > size:
        indices = np.random.choice(len(features), size=size, replace=False)
        return features[indices], labels[indices]
    return features, labels
```

Файл modelling/admin.py

```
from django.contrib import admin
```

```
from modelling.models import TrainingSession
```

```
@admin.register(TrainingSession)
class TrainingSessionAdmin(admin.ModelAdmin):
    list_display = (
        "id",
        "status",
        "classifier_name",
        "test_accuracy",
        "created_at",
    )
    ordering = ("-created_at",)
```

Файл modelling/apps.py

```
from django.apps import AppConfig
```

```
class ModellingConfig(AppConfig):
    default_auto_field = "django.db.models.BigAutoField"
    name = "modelling"
```

Файл modelling/models.py

```
import uuid
```

```
import joblib
from django.core.files.storage import FileSystemStorage
from django.db import models
```

```
from modelling.constants import CLASSIFIER, KMEANS, MODELS_CACHE_DIR,
WORD2VEC
from modelling.constants.enums import TrainingSessionStatus
```

```

MODELS_STORAGE = FileSystemStorage(location=MODELS_CACHE_DIR)

class TrainingSession(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)

    status = models.CharField(
        max_length=20,
        choices=TrainingSessionStatus.choices(),
        default=TrainingSessionStatus.INITIALIZED,
    )

    # Models
    kmeans_p_model = models.FileField(
        storage=MODELS_STORAGE, upload_to=f"{KMEANS}/", null=True
    )
    kmeans_qrs_model = models.FileField(
        storage=MODELS_STORAGE, upload_to=f"{KMEANS}/", null=True
    )
    kmeans_t_model = models.FileField(
        storage=MODELS_STORAGE, upload_to=f"{KMEANS}/", null=True
    )
    word2vec_model = models.FileField(
        storage=MODELS_STORAGE, upload_to=f"{WORD2VEC}/", null=True
    )
    classifier_model = models.FileField(
        storage=MODELS_STORAGE, upload_to=f"{CLASSIFIER}/", null=True
    )
    classifier_name = models.CharField(max_length=100, blank=True)

    # Metrics
    train_accuracy = models.FloatField(null=True)
    test_accuracy = models.FloatField(null=True)
    precision = models.FloatField(null=True)
    recall = models.FloatField(null=True)
    f1_score = models.FloatField(null=True)
    silhouette_score = models.FloatField(null=True)

    # Charts
    charts = models.JSONField(default=dict)

    modified_at = models.DateTimeField(auto_now=True)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"TrainingSession({self.classifier_name})"

    def get_kmeans_models(self):
        return (
            joblib.load(self.kmeans_p_model),
            joblib.load(self.kmeans_qrs_model),
            joblib.load(self.kmeans_t_model),
        )

```

```

def get_word2vec_model(self):
    return joblib.load(self.word2vec_model)

def get_classifier_model(self):
    return joblib.load(self.classifier_model)

```

Файл modelling/tasks.py

```

from celery import shared_task
from celery.utils.log import get_task_logger

from modelling.models import TrainingSession
from modelling.training.manager import TrainingManager

logger = get_task_logger(__name__)

@shared_task
def q_train_model(training_session_id, general_params, algorithm_params):
    session = TrainingSession.objects.filter(id=training_session_id).first()
    if not session:
        logger.warning(f'Training session with id {training_session_id} does not exist. Existing task...')
        return

    manager = TrainingManager(session, general_params, algorithm_params)
    manager.run()

```

Файл modelling/urls.py

```

from django.urls import path

from . import views

app_name = "modelling"

urlpatterns = [
    path(
        "session/",
        views.TrainingSessionCreateView.as_view(),
        name="training-session-create",
    ),
    path(
        "session/<uuid:pk>/",
        views.TrainingSessionRetrieveView.as_view(),
        name="training-session-retrieve",
    ),
]

```

Файл modelling/views.py

```

from copy import deepcopy

```

					КП.ІП-9202.045420.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		35

```

from django.db import transaction
from rest_framework.generics import RetrieveAPIView

from rest_framework.response import Response
from rest_framework.status import HTTP_201_CREATED
from rest_framework.views import APIView

from .constants.enums import ClassificationAlgorithm
from .models import TrainingSession
from .serializers import (
    KNNParamsSerializer,
    RandomForestParamsSerializer,
    SVCParamsSerializer,
    TrainingSessionGeneralParamsSerializer,
    TrainingSessionSerializer,
)
from .tasks import q_train_model

class TrainingSessionCreateView(APIView):
    ALGORITHM_SERIALIZER = {
        ClassificationAlgorithm.KNN: KNNParamsSerializer,
        ClassificationAlgorithm.SVC: SVCParamsSerializer,
        ClassificationAlgorithm.RANDOM_FOREST: RandomForestParamsSerializer,
    }

    def post(self, request):
        data = deepcopy(request.data)

        general_params = data.pop("general_params")
        general_params_serializer = TrainingSessionGeneralParamsSerializer(
            data=general_params
        )
        general_params_serializer.is_valid(raise_exception=True)
        general_params_validated = general_params_serializer.validated_data

        algorithm, algorithm_params = general_params_validated["algorithm"],
data.pop(
    "algorithm_params"
)
        algorithm_params_serializer = self.ALGORITHM_SERIALIZER[algorithm](
            data=algorithm_params
        )
        algorithm_params_serializer.is_valid(raise_exception=True)
        algorithm_params_validated = algorithm_params_serializer.validated_data

        training_session = TrainingSession.objects.create()
        training_session_id_str = str(training_session.id)

        transaction.on_commit(lambda: q_train_model.delay(
            training_session_id_str,
            general_params_validated,
            algorithm_params_validated,

```

					КП.ІП-9202.045420.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		36

```

    ))

    return Response({"id": training_session_id_str},
status=HTTP_201_CREATED)

class TrainingSessionRetrieveView(RetrieveAPIView):
    queryset = TrainingSession.objects.all()
    serializer_class = TrainingSessionSerializer

```

Файл static/css/styles.css

```

body {
font-family: 'Montserrat', sans-serif;
}

```

Файл static/js/index.js

```

const root = document.getElementById("app");

function getCookie(name) {
    const value = `; ${document.cookie}`;
    const parts = value.split(`; ${name}=`);
    if (parts.length === 2) return parts.pop().split(";").shift();
}

const StatusSpinner = {
    view: function (vnode) {
        return m(".d-flex.flex-column.justify-content-center.align-items-center", [
            m(".spinner-border.text-primary", {'style': "width: 4rem; height: 4rem", "role": "status"}),
            m("h5.mt-4", vnode.attrs.status.charAt(0).toUpperCase() + vnode.attrs.status.slice(1))
        ])
    }
}

const HeroSection = {
    view: function () {
        return m(
            'section',
            [
                m('nav.d-flex.justify-content-flex-start', {style: "height: 100px;"}, [
                    m('a.navbar-brand', {href: '#', style: "height: 100%"}, [
                        m('img', {
                            src: '/static/img/logo.png',
                            alt: 'Logo',
                            style: "padding: 15px 0 0 0; height: 100%;"
                        })
                    ])
                ]),
                m('.row', {style: 'margin-top: 50px'}, [
                    m('.col-lg-6', [

```

					КП.ІП-9202.045420.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		37

```

        m('.d-flex.flex-column.align-items-start', [
            m('h1.font-weight-bold', {style: "margin-bottom:
60px; font-size: 3.5rem"}), 'Unleashing the Power of Advanced ECG Analysis'),
            m('h5', {style: "margin-bottom: 60px; line-height:
2.2rem"}), 'Harness the potential of cutting-edge AI technology to
revolutionize cardiac health diagnosis and treatment.'),
            m('a.btn.btn-primary.mt-auto[href="#training-
section"]', {style: 'padding: 15px 35px; font-size: 1.5rem'}, 'Explore,
Train, Diagnose!')
        ])
    ],
    m('.d-flex.justify-content-center.align-items-center.col-
lg-6', {}, [
        m('img', {src: '/static/img/hero-heart.png', alt:
'Heart Image', style: 'height: 420px;'})
    ])
  });
};

```

```

const ChartComponent = {
  oninit: function (vnode) {
    this.chartData = JSON.parse(vnode.attrs.chartData);
  },
  oncreate: function (vnode) {
    Plotly.newPlot(vnode.dom, this.chartData.data, this.chartData.layout);
  },
  view: function () {
    return m('div')
  }
};

```

```

const TrainingService = {
  sessionId: null,
  trainingStatus: "initialized",
  trainingMetrics: {},
  chartsData: {},
  showResults: false,

  runTraining: function (data) {
    FileUpload.file = null;
    m.request({
      method: "POST",
      url: "/api/modelling/session/",
      body: data,
      headers: {
        'Content-Type': 'application/json',
        "X-CSRFToken": getCookie("csrftoken")
      }
    }).then(response => {
      TrainingService.sessionId = response.id;
      TrainingService.showResults = true;

      TrainingService.trainingStatus = "initialized";
      TrainingService.trainingMetrics = {};
      TrainingService.chartsData = {};
    });
  }
};

```

```

        TrainingService.checkStatus();
    }).catch(error => {
        console.log(error.response);
        TrainingService.errors = error.response;
    });

},
checkStatus: function () {
    if (TrainingService.sessionId === null) return;

    clearInterval(TrainingService.interval);

    TrainingService.interval = setInterval(() => {
        m.request({
            method: "GET",
            url: "/api/modelling/session/" + TrainingService.sessionId,
        }).then(response => {
            TrainingService.trainingStatus = response.status;

            if (response.status === "done") {
                TrainingService.chartsData = JSON.parse(response.charts);
                delete response.charts;

                TrainingService.trainingMetrics = response;
            }
        });

        if (TrainingService.trainingStatus === "done") {
            clearInterval(TrainingService.interval);
        }
    }, 2000);
},
};

const TrainingSetup = {
    algorithmParams: {
        'knn': {
            'readable': 'K Nearest Neighbors',
            'parameters': [
                {name: 'n_neighbors', default: '5', readable: 'Neighbors
number', required: false},
                {
                    name: 'weights',
                    default: 'uniform',
                    readable: 'Weights',
                    choices: ['uniform', 'distance'],
                    required: false
                },
                {
                    name: 'algorithm',
                    default: 'auto',
                    readable: 'Algorithm',
                    choices: ['auto', 'ball_tree', 'kd_tree', 'brute'],
                    required: false
                }
            ]
        }
    }
};

```

```

    },
  ],
},
'svc': {
  'readable': "Support Vector Machine",
  'parameters': [
    {name: 'C', default: '1.0', readable: 'Penalty Parameter C',
required: false},
    {
      name: 'kernel',
      default: 'rbf',
      readable: 'Kernel Type',
      choices: ['linear', 'poly', 'rbf', 'sigmoid'],
      required: false
    },
    {name: 'degree', default: '3', readable: 'Degree of the
Polynomial Kernel Function', required: false},
  ]
},
'random_forest': {
  'readable': 'Random Forest',
  'parameters': [
    {name: 'n_estimators', default: '100', readable: 'Number of
estimators', required: false},
    {
      name: 'criterion',
      default: 'gini',
      readable: 'Criterion',
      choices: ['gini', 'entropy'],
      required: false
    },
    {name: 'max_depth', default: null, readable: 'The Maximum Depth
of the Tree', required: false},
  ]
},
},
inputs: {
  general: {
    alphabet_size: 100,
    word2vec_vector_size: 100,
    algorithm: ''
  },
  algorithmConfig: {}
},
showResults: false,
view: function () {
  const chosenAlgorithmParams = this.inputs.general.algorithm
    ? this.algorithmParams[this.inputs.general.algorithm].parameters
    : [];

  this.errors = TrainingService.errors || {};

  return m("form.d-flex.flex-column.justify-content-center", [
    m('h3', 'Setup'),
    m(".form-row.mt-4", [
      m(".col", [
        m("label.h6", "Alphabet size"),
        m("input.form-control[type=text][placeholder='Enter a
value'][required]", {
          name: "alphabet_size",
          value: this.inputs.general.alphabet_size,
          oninput: (e) => {

```

```

        this.inputs.general.alphabet_size = e.target.value;
        if (this.errors.alphabet_size) {
            delete this.errors.alphabet_size;
        }
    }
    },
    this.errors.alphabet_size && m(".text-danger.mt-1",
this.errors.alphabet_size[0])
    ],
    m(".col", [
        m("label.h6", "Word2Vec vector size"),
        m("input.form-control[type=text][placeholder='Enter a
value'] [required]", {
            name: "word2vec_vector_size",
            value: this.inputs.general.word2vec_vector_size,
            oninput: (e) => {
                this.inputs.general.word2vec_vector_size =
e.target.value;
            }
        })
    ]),
    m(".col", [
        m("label.h6", "Algorithm"),
        m("select.form-control", {
            name: "algorithm",
            value: this.inputs.general.algorithm,
            oninput: (e) => {
                this.inputs.general.algorithm = e.target.value;
                if
(!this.inputs.algorithmConfig[e.target.value]) {
                    this.inputs.algorithmConfig[e.target.value]
= this.algorithmParams[e.target.value].parameters.reduce((params, param) => {
                        params[param.name] = param.default ||
null;

                        return params;
                    }, {});
                }
                if (this.errors.algorithm) {
                    delete this.errors.algorithm;
                }
            }
        }), [
            m("option", {
                value: '',
                selected: this.inputs.general.algorithm === '',
                disabled: true
            }, 'Select an Algorithm'),
            ...Object.keys(this.algorithmParams).map(algorithm
=>
                m("option", {
                    value: algorithm,
                    selected: this.inputs.general.algorithm ===
algorithm

                }, this.algorithmParams[algorithm].readable)
            ),

        ],
        ), this.errors.algorithm && m(".text-danger.mt-1",
this.errors.algorithm[0])
    ]),
    ],

```

```

        this.inputs.general.algorithm && m(".form-row.mt-4",
chosenAlgorithmParams.map((param) =>
    m(".col", [
        m("label.h6", param.readable),
        param.choices
            ? m("select.form-control", {
                name: param.name,
                value:
(this.inputs.algorithmConfig[this.inputs.general.algorithm] &&
this.inputs.algorithmConfig[this.inputs.general.algorithm][param.name]) ||
null,
                required: param.required,
                onchange: (e) => {

this.inputs.algorithmConfig[this.inputs.general.algorithm][param.name] =
e.target.value;

                }
            },
            param.choices.map(choice =>
                m("option", {value: choice}, choice)
            )
        )
        : m("input.form-control[type=text][placeholder='Enter a
value']", {
            name: param.name,
            value:
(this.inputs.algorithmConfig[this.inputs.general.algorithm] &&
this.inputs.algorithmConfig[this.inputs.general.algorithm][param.name]) ||
null,
            required: param.required,
            onchange: (e) => {

this.inputs.algorithmConfig[this.inputs.general.algorithm][param.name] =
e.target.value

            }
        })
    ]))
    ),
    m("button.btn.btn-primary[type=submit].mt-5.", {
        style: 'padding: 15px 0; font-size: 1.5rem',
        onclick: (event) => {
            event.preventDefault();

            // Fill in 'word2vec_vector_size' if it's missing
            if (!this.inputs.general.word2vec_vector_size) {
                this.inputs.general.word2vec_vector_size = 100;
            }

            // Get the algorithm parameters

            if (this.inputs.general.algorithm) {
                const algorithmParameters =
this.algorithmParams[this.inputs.general.algorithm].parameters;

                // Fill in missing input values with their defaults
                for (let param of algorithmParameters) {
                    const input =
this.inputs.algorithmConfig[this.inputs.general.algorithm][param.name];
                    if (input === undefined || input === null || input
=== '') {

```

```

this.inputs.algorithmConfig[this.inputs.general.algorithm][param.name] =
param.default;

        }
    }

    }

    // Proceed with running the training session
    const data = {
        general_params: this.inputs.general,
        algorithm_params:
this.inputs.algorithmConfig[this.inputs.general.algorithm]
    };
    TrainingService.runTraining(data);

    }
    }, "Run")
    ])
    }
}

const TrainingResult = {
    view: function () {
        if (TrainingService.trainingStatus !== "done") {
            return m(StatusSpinner, {status: TrainingService.trainingStatus})
        }

        return m('.d-flex.flex-column.justify-content-center', [
            m('h3', 'Results'),
            m(".row.mt-4", [
                m(".col-4", [
                    m(".card", [
                        m(".card-body", [
                            m("h5.card-title", "Training accuracy"),
                            m("p.card-text",
TrainingService.trainingMetrics['train_accuracy'])
                        ])
                    ])
                ]),
                m(".col-4", [
                    m(".card", [
                        m(".card-body", [
                            m("h5.card-title", "Precision"),
                            m("p.card-text",
TrainingService.trainingMetrics['precision'])
                        ])
                    ])
                ]),
                m(".col-4", [
                    m(".card", [
                        m(".card-body", [
                            m("h5.card-title", "F1 score"),
                            m("p.card-text",
TrainingService.trainingMetrics['f1_score'])
                        ])
                    ])
                ])
            ])
        ])
    }
}

```

```

    })
  },
  m(".row.mt-4", [
    m(".col-4", [
      m(".card", [
        m(".card-body", [
          m("h5.card-title", "Testing accuracy"),
          m("p.card-text",
TrainingService.trainingMetrics['test_accuracy'])
        ])
      ])
    ]),
    m(".col-4", [
      m(".card", [
        m(".card-body", [
          m("h5.card-title", "Recall"),
          m("p.card-text",
TrainingService.trainingMetrics['recall'])
        ])
      ])
    ]),
    m(".col-4", [
      m(".card", [
        m(".card-body", [
          m("h5.card-title", "Silhouette score"),
          m("p.card-text",
TrainingService.trainingMetrics['silhouette_score'])
        ])
      ])
    ]),
    m(".row.mt-4", [
      m(".col-6", [
        m(".card.h-100", [
          m(".card-body", [
            m("h5.card-title", "Confusion matrix"),
            m(".w-100", [m(ChartComponent, {chartData:
TrainingService.chartsData['confusion_matrix']})])
          ])
        ])
      ]),
      m(".col-6", [
        m(".card.h-100", [
          m(".card-body", [
            m("h5.card-title", "TSNE of QRS complexes"),
            m(".w-100", [m(ChartComponent, {chartData:
TrainingService.chartsData['tsne']['qrs']})])
          ])
        ])
      ]),
    ]),
    m(".row.mt-4", [
      m(".col-6", [
        m(".card.h-100", [
          m(".card-body", [
            m("h5.card-title", "TSNE of P waves"),
            m(".w-100", [m(ChartComponent, {chartData:
TrainingService.chartsData['tsne']['p']})])
          ])
        ])
      ]),
      m(".col-6", [
        m(".card.h-100", [
          m(".card-body", [

```



```

        this.previousElementSibling.click();
    },
    ondragover: function (e) {
        e.preventDefault();
    },
    ondrop: function (e) {
        e.preventDefault();
        var file = e.dataTransfer.files[0];
        if (file &&
/\.(csv|xlsx)$/i.test(file.name)) {
            if (file.size <= 5 * 1024 * 1024) {
                FileUpload.file = file;
                m.redraw();
            } else {
                alert("File too large. Please
upload a file smaller than 5MB.");
            }
        } else {
            alert("Invalid file. Please upload
a CSV or XLSX file.");
        }
    }, "Drop file here")
    ]
    })
    })
    });
    });
};

const AnalysisService = {
    chartsData: {},
    tableData: {},
    showResults: false,
    analysisStatus: null,

    runAnalysis: function (data) {
        AnalysisService.analysisStatus = 'running';
        m.request({
            method: "POST",
            url: "/api/analysis/session/",
            body: data,
            headers: {
                "X-CSRFToken": getCookie("csrftoken")
            }
        }).then(response => {
            AnalysisService.showResults = true;
            AnalysisService.chartsData = response['charts'];
            AnalysisService.tableData = response['table'];
            AnalysisService.analysisStatus = 'done';

        }).catch(error => {
            AnalysisService.errors = error.response;
        });

    },
};

```

```

const AnalysisSetup = {
  showResults: false,
  fs: 360,
  view: function () {
    this.errors = AnalysisService.errors || {};

    return m(
      '.d-flex.flex-column.justify-content-flex-start',
      [
        m('h3', 'Setup'),
        m('.mt-4', [m(FileUpload)]),

        FileUpload.file && m(".d-flex.flex-column.mt-4", [
          m("label.h6", "Sampling frequency (fs)"),
          m("input.form-control[type=text][placeholder='Enter a
value']", {
            value: AnalysisSetup.fs,
            oninput: (e) => {
              AnalysisSetup.fs = e.target.value;
            },
          }),
          this.errors.fs && m(".text-danger.mt-1", this.errors.fs[0])
        ]),

        FileUpload.file && m("button.btn.btn-primary[type=submit].mt-
5.", {
          style: 'padding: 15px 0; font-size: 1.5rem',
          onclick: function (event) {
            event.preventDefault();

            AnalysisSetup.fs = AnalysisSetup.fs || 360;

            const formData = new FormData();
            formData.append('ecg_file', FileUpload.file);
            formData.append('fs', AnalysisSetup.fs);
            formData.append('training_session_id',
TrainingService.sessionId);

            AnalysisService.runAnalysis(formData)
          }, "Run")
        ]
      )
    )
  }
}

const AnalysisResult = {
  view: function () {
    if (AnalysisService.analysisStatus !== "done") {
      return m(StatusSpinner, {status: 'analyzing'})
    }

    console.log(AnalysisService.chartsData)
    return m(

```



```

    <link rel="stylesheet" href="{% static 'css/styles.css' %}">
</head>
<body>
<div id="app"></div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js"
integrity="sha384-
JjSmVgyd0p3pXBlrRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
<script src="https://cdn.plot.ly/plotly-2.20.0.min.js" charset="utf-
8"></script>
<script src="https://unpkg.com/mithril/mithril.js"></script>
<script type="module" src="{% static 'js/index.js' %}"></script>
</body>
</html>

```

Файл docker-compose.yml

```

version: "3.9"

services:
  db:
    image: postgres:14
    environment:
      - POSTGRES_DB=${DATABASE_NAME}
      - POSTGRES_USER=${DATABASE_USER}
      - POSTGRES_PASSWORD=${DATABASE_PASSWORD}
    volumes:
      - db-data:/var/lib/postgresql/data

  web:
    build:
      context: .
      dockerfile: Dockerfile-django
    entrypoint: ["./bin/entrypoint-django.sh"]
    ports:
      - "8000:8000"
    depends_on:
      - db
      - redis
    volumes:
      - static-volume:/home/app/ecganalysis/staticfiles
      - .env:/home/app/ecganalysis/.env
      - ./modelling/data_preparation/cache:/home/app/ecganalysis/modelling/data_preparation/cache
      - ./modelling/models_cache:/home/app/ecganalysis/modelling/models_cache

  celery:
    build:
      context: .
      dockerfile: Dockerfile-django
    entrypoint: [ "./bin/entrypoint-celery.sh" ]
    depends_on:
      - db
      - redis
    volumes:

```

```

- .env:/home/app/ecganalysis/.env
-
./modelling/data_preparation/cache:/home/app/ecganalysis/modelling/data_preparation/cache
- ./modelling/models_cache:/home/app/ecganalysis/modelling/models_cache

nginx:
  build:
    context: .
    dockerfile: Dockerfile-nginx
  command: ["../wait-for-it.sh", "web:8000", "--", "nginx", "-g", "daemon off;"]
  ports:
    - "80:80"
  depends_on:
    - web
  volumes:
    - static-volume:/home/app/ecganalysis/staticfiles

redis:
  image: redis:7-alpine

volumes:
static-volume:
db-data:

```

Файл Dockerfile-django

FROM python:3.10-slim as builder

```

# Set working directory
WORKDIR /usr/src/app

```

```

# Copy requirements file
COPY requirements.txt .

```

```

# Upgrade pip and install Python packages
RUN pip install --no-cache-dir --upgrade pip && pip wheel --no-cache-dir --no-deps --wheel-dir /usr/src/app/wheels -r requirements.txt

```

```

# pull official base image
FROM python:3.10-slim

```

```

# Set working directory
WORKDIR /home/app/ecganalysis

```

```

# Create app user and set ownership
RUN mkdir -p ./staticfiles && mkdir -p ./media && groupadd -r app && useradd -r -g app app && chown -R app:app /home/app/ecganalysis && chmod -R 755 /home/app/ecganalysis

```

					КП.ІП-9202.045420.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		51

```

RUN apt-get update && \
  apt-get install -y tk && \
  rm -rf /var/lib/apt/lists/*

# Copy wheels and requirements from builder stage
COPY --from=builder /usr/src/app/wheels /wheels
COPY --from=builder /usr/src/app/requirements.txt .

# Install Python packages from wheels
RUN pip install --no-cache-dir /wheels/*

# Copy project files
COPY . .

# Make entrypoint file executable
RUN chmod +x ./bin/entrypoint-django.sh
RUN chmod +x ./bin/entrypoint-celery.sh

# Switch to app user
USER root

# Set environment variables
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

# Expose port
EXPOSE 8000

```

Файл Dockerfile-nginx

```

# Use the Nginx image as the base image
FROM nginx:1.21.0-alpine

# Install wget and bash
RUN apk add --no-cache wget bash

# Download the wait-for-it.sh script
RUN wget https://raw.githubusercontent.com/vishnubob/wait-for-it/master/wait-for-it.sh -O /wait-for-it.sh

# Make the script executable
RUN chmod +x /wait-for-it.sh

# Copy Nginx configuration file
COPY ./nginx.conf /etc/nginx/conf.d/default.conf

```

Файл manage.py

```

#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

```

					КП.ІП-9202.045420.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		52

```

def main():
    """Run administrative tasks."""
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "EcgAnalysis.settings")
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == "__main__":
    main()

```

					КП.ІП-9202.045420.03.13	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		53

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2023 р.

**ПРОГРАМНА БІБЛІОТЕКА АНАЛІЗУ ЕКГ ЛІНГВІСТИЧНИМИ
МЕТОДАМИ**

Програма та методика тестування

КПІ.ПІ-9202.045420.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Юрій ОЛІЙНИК

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Еліза БЕРАУДО

Київ – 2023

ЗМІСТ

1	ОБ’ЄКТ ВИПРОБУВАНЬ.....	3
2	МЕТА ТЕСТУВАННЯ	4
3	МЕТОДИ ТЕСТУВАННЯ.....	5
4	ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ	6

					КП.ІП-9202.045420.04.51	Арк.
Змін	Арк.	№ докум.	Підп.	Дата.		2

1 ОБ'ЄКТ ВИПРОБУВАНЬ

Об'єктом випробування є сервіс для аналізу ЕКГ, реалізований у формі веб-сайту. Цей сервіс включає дві основні компоненти: моделювання та аналіз ЕКГ. Цей сервіс був розроблений для будь-якої операційної системи з встановленим будь-яким сучасним веб-браузером. Він був оптимізований для використання з популярними веб-браузерами, такими як Google Chrome, Mozilla Firefox, Safari і Microsoft Edge.

					КП.ІП-9202.045420.04.51	Арк.
Змін	Арк.	№ докум.	Підп.	Дата.		3

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- перевірка збереження даних;
- перевірка сумісності веб-додатку з останніми версіями сучасних браузерів (Chrome, Opera, Firefox);
- перевірка сумісності застосунку з різними операційними системами (Windows, Linux, macOS);
- знаходження проблем, помилок і недоліків з метою їх усунення;
- перевірка зручності графічного інтерфейсу.

					КП.ІП-9202.045420.04.51	Арк.
						4
Змін	Арк.	№ докум.	Підп.	Дата.		

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;
- динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;
- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
- системне тестування – перевіряється усе програмне забезпечення в цілому;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;
- тестування «чорної скриньки» – об'єктом тестування тут є функції присутні у програмі. Перевіряється коректність вихідних даних при заданих вхідних;

					КП.ІП-9202.045420.04.51	Арк.
Змін	Арк.	№ докум.	Підп.	Дата.		5

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з використанням наскрізного тестування (Postman), з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

- динамічне тестування на відповідність функціональним вимогам;
- тестування на мобільних пристроях з різною роздільною здатністю екрану;
- тестування на мобільних пристроях з різною версією операційної системи;
- тестування на виведення повідомлень про помилку, коли це необхідно;
- тестування зміни орієнтації екрану;
- тестування працездатності програми у випадку відсутності з'єднання до мережі;
- тестування інтерфейсу користувача;
- тестування зручності використання.

					КП.ІП-9202.045420.04.51	Арк.
Змін	Арк.	№ докум.	Підп.	Дата.		6

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ” _____ 2023 р.

**ПРОГРАМНА БІБЛІОТЕКА АНАЛІЗУ ЕКГ ЛІНГВІСТИЧНИМИ
МЕТОДАМИ**

Керівництво користувача

КПІ.ІП-9202.045420.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Юрій ОЛІЙНИК

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Еліза БЕРАУДО

Київ – 2023

ЗМІСТ

1	ПРИЗНАЧЕННЯ ПРОГРАМИ	3
2	ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ.....	4
2.1	Системні вимоги для коректної роботи.....	4
2.2	Завантаження застосунку.....	4
2.3	Перевірка коректної роботи.....	5
3	ВИКОНАННЯ ПРОГРАМИ	6

					КПІ.ІП-9202.045420.05.34	Арк.
						2
Змін.	Арк.	№ докум.	Підп.	Дата.		

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

ECG Analysis Service - це веб-застосунок, призначений для аналізу та моделювання електрокардіограм (ЕКГ). Програма включає дві основні компоненти: моделювання та аналіз.

Моделювання: користувачі мають можливість вводити параметри навчання, такі як кількість кластерів для Kmeans, розмір вектора Word2Vec, вибрати алгоритм класифікації серцевих скорочень (наприклад, випадковий ліс, SVC тощо) і встановлювати специфічні параметри класифікатора. Після натискання кнопки "Запустити", модель навчається і оцінюється на бекенді, після чого бекенд відповідає на фронтенд з метриками моделі класифікатора (точність, точність, повернення, оцінка f1) і показником силуета Kmeans, а також надсилає чотири діаграми для візуалізації даних.

Аналіз: користувачі можуть завантажити власний csv/xlsx файл з ЕКГ і встановити значення частоти дискретизації, а потім натиснути "Аналізувати". Потім бекенд використовує всі натреновані алгоритми для класифікації серцевих скорочень у файлі користувача і повертає діаграму на фронтенд, де всі серцеві скорочення позначені (нормальні, ненормальні або інші класи).

Програма реалізована як веб-застосунок і сумісна з будь-якою операційною системою, що має сучасний веб-браузер. Репозиторій програмного забезпечення включає всі необхідні файли для використання і розробки, включаючи вихідний код, документацію, залежності, конфігурації та скрипти для встановлення та випробування.

					КП.ІП-9202.045420.05.34	Арк.
						3
Змін.	Арк.	№ докум.	Підп.	Дата.		

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи застосунку, система повинна задовольняти вимогам, наведеним у таблицях 2.1.

Таблиця 2.1 – Системні вимоги

	Мінімальні	Рекомендовані
Операційна система	Windows® XP/Windows Vista/Windows 7/ Windows 8/Windows 10	Windows 7/ Windows 8/Windows 10
Процесор	Intel® Pentium® III 1.0 GHz або AMD Athlon™ 1.0 GHz	Intel® Pentium® D або AMD Athlon™ 64 X2
Оперативна пам'ять	256 MB RAM (для Windows® XP) / 1 GB RAM (для Windows Vista/Windows 7/ Windows 8/Windows 10)	2 GB RAM
Відеоадаптер	Intel® GMA X3100, 144 MB	Intel® Iris Plus Graphics 640 1536 MB
Дисплей	800x600	1024x768

2.2 Завантаження застосунку

Для отримання доступу до "ECG Analysis Service", користувачу необхідно перейти на відповідний веб-сайт через будь-який сучасний веб-браузер. Цей веб-застосунок не вимагає завантаження або встановлення додаткових файлів.

2.3 Перевірка коректної роботи

Після переходу на веб-сайт "ECG Analysis Service", користувач має побачити головну сторінку застосунку. Якщо веб-сторінка завантажується коректно і не виникає помилок, це означає, що доступ до веб-застосунку встановлено правильно.

На головній сторінці повинні бути доступні два основні компоненти: "Modelling" та "Analysis". У розділі "Modelling", користувач повинен мати можливість вводити параметри навчання і запускати модель. У розділі "Analysis", користувач повинен мати можливість завантажувати власні csv/xlsx файли з ЕКГ для аналізу.

Якщо всі ці компоненти присутні і працюють без помилок, це підтверджує, що веб-застосунок розгорнуто та працює правильно.

					КПІ.ІП-9202.045420.05.34	Арк.
						5
Змін.	Арк.	№ докум.	Підп.	Дата.		

3 ВИКОНАННЯ ПРОГРАМИ

Для роботи з веб-застосунком "ECG Analysis Service" необхідно дотримуватись наступної інструкції, потрібно перейти за посиланням потрібно перейти за посиланням <http://139.162.146.230>.

3.1. Відкриття веб-сайту ECG Analysis Service.

Зображення домашньої сторінки сервісу з кнопками та полями вводу наведено на рисунку 3.1.



Unleashing the Power of Advanced ECG Analysis

Harness the potential of cutting-edge AI technology to revolutionize cardiac health diagnosis and treatment.

Explore, Train, Diagnose!

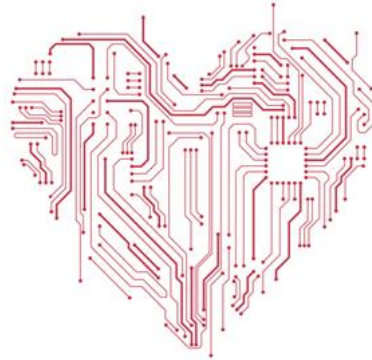


Рисунок 3.1 – Головна сторінка програмного забезпечення
Клік на кнопку показує секцію моделювання (рисунок 3.2).

					КПІ.ІП-9202.045420.05.34	Арк.
						6
Змін.	Арк.	№ докум.	Підп.	Дата.		

Harness the potential of cutting-edge AI technology
to revolutionize cardiac health diagnosis and
treatment.



Explore, Train, Diagnose!

Modelling

Play with hyper-parameters to train the best algorithm

Setup

Alphabet size

Word2Vec vector size

Algorithm

100

100

Select an Algorithm

Run

Рисунок 3.2 – Секція моделювання

3.2. Моделювання.

Введення параметрів навчання (рисунок 3.3).

treatment.



Explore, Train, Diagnose!

Modelling

Play with hyper-parameters to train the best algorithm

Setup

Alphabet size

Word2Vec vector size

Algorithm

123

123

Random Forest

Number of estimators

Criterion

The Maximum Depth of the Tree

100

gini

3

Run

Рисунок 3.3 – Введення параметрів навчання

Якщо не введено обов'язкові параметри (рисунок 3.4).

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.ІП-9202.045420.05.34

Арк.

7

Harness the potential of cutting-edge AI technology to revolutionize cardiac health diagnosis and treatment.



Explore, Train, Diagnose!

Modelling

Play with hyper-parameters to train the best algorithm

Setup

Alphabet size	Word2Vec vector size	Algorithm
100	100	Select an Algorithm
		Invalid choice.

Run

Рисунок 3.4 – Відображення помилки про незаповнені поля
Якщо введено неправильний тип даних (рисунок 3.5).

Explore, Train, Diagnose!

Modelling

Play with hyper-parameters to train the best algorithm

Setup

Alphabet size	Word2Vec vector size	Algorithm
івафівайаа	100	Random Forest
A valid integer is required.		
Number of estimators	Criterion	The Maximum Depth of the Tree
100	gini	Enter a value

Run

Рисунок 3.5 – Відображення помилки про неправильний тип даних
Запуск моделювання.

Показ статусу (рисунок 3.6 - 3.8) навчання моделі (ініціалізовано, навчання, оцінювання, завершено).

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.ІП-9202.045420.05.34

Арк.

8

Modelling

Play with hyper-parameters to train the best algorithm

Setup

Alphabet size	Word2Vec vector size	Algorithm
200	150	Random Forest
Number of estimators	Criterion	The Maximum Depth of the Tree
150	gini	5

Run



Initialized

Рисунок 3.6 – Статус ініціалізовано

Modelling

Play with hyper-parameters to train the best algorithm

Setup

Alphabet size	Word2Vec vector size	Algorithm
200	150	Random Forest
Number of estimators	Criterion	The Maximum Depth of the Tree
150	gini	5

Run



Training

Рисунок 3.7 – Статус навчання.

Play with hyper-parameters to train the best algorithm

Alphabet size	Word2Vec vector size	Algorithm
200	150	Random Forest
Number of estimators	Criterion	The Maximum Depth of the Tree
150	gini	5

Run



Evaluation

Рисунок 3.8 – Статус оцінювання.

Успішне навчання моделі(рисунок 3.9):

Training accuracy

94.5%

Precision

85.6%

F1 score

85.4%

Testing accuracy

85.5%

Recall

85.5%

Silhouette score

27.8%

Confusion matrix

		Predicted label												
		"	/	A	F	J	L	N	R	V	f	j		
True label	J	0	0	5	0	0	1	14	1	2	0	20		
	f	0	15	0	1	0	6	12	0	7	126	0		
	V	1	14	18	2	0	43	126	7	699	1	0		
	R	0	3	55	3	0	1	35	898	21	0	0		
	N	2	8	37	5	2	18	1079	20	81	3	10		
	L	0	5	2	5	0	1230	18	4	36	1	0		
	j	0	0	0	0	3	0	3	8	0	0	0		
	F	0	2	2	71	0	7	21	1	27	4	1		
	A	0	0	229	2	0	2	95	21	18	0	7		
	/	0	1031	0	0	0	8	15	2	15	5	0		
	"	67	0	0	0	0	0	3	1	2	0	0		

TSNE of QRS complexes

Рисунок 3.9 – Відображення метрик моделі та графіків.

3.3 Аналіз.

Завантаження файлу ECG (рисунок 3.10).



Analysis

Run the prepared model on your own data

Setup

Upload file

Drag your file here or click to upload.

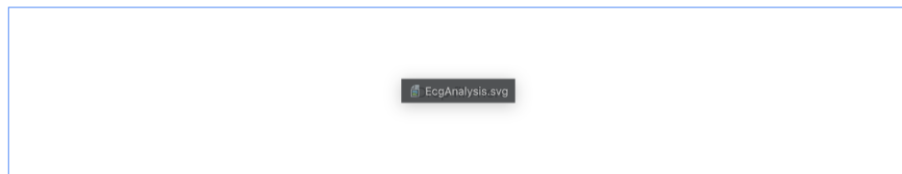


Рисунок 3.10 – Завантаження файлу ECG.

Завантажений файл ECG (рисунок 3.11).

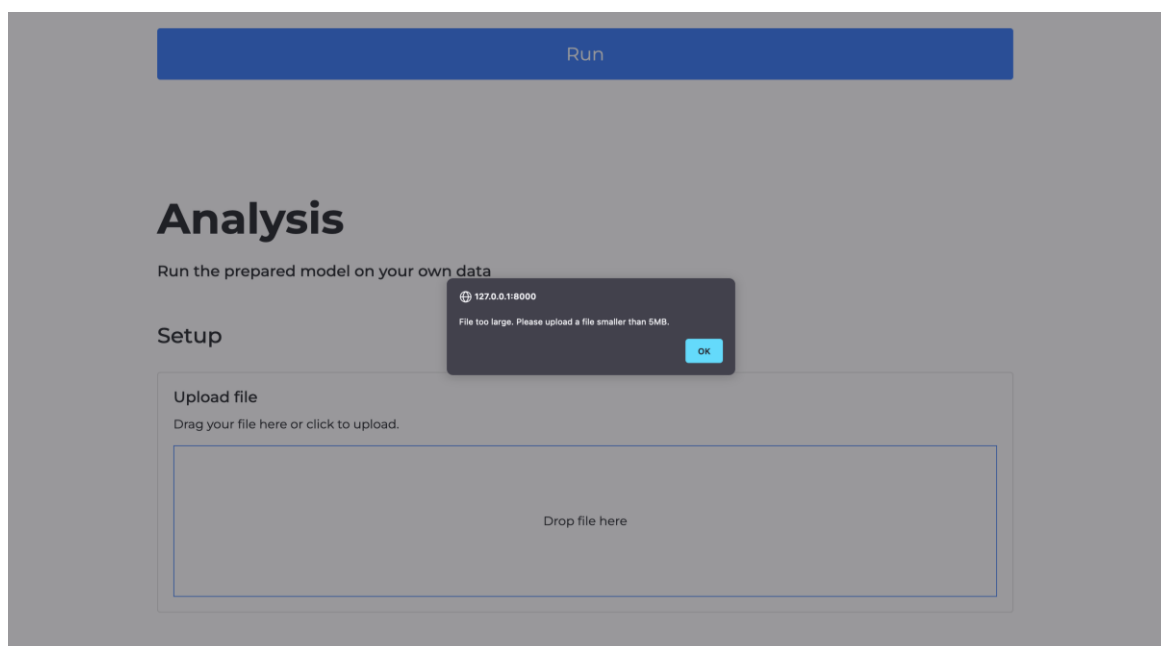


Рисунок 3.11 – Завантажений файл ECG.

Якщо вибраний файл не відповідає формату CSV або XLSX (рисунок 3.12):

					КПІ.ІП-9202.045420.05.34	Арк.
						11
Змін.	Арк.	№ докум.	Підп.	Дата.		

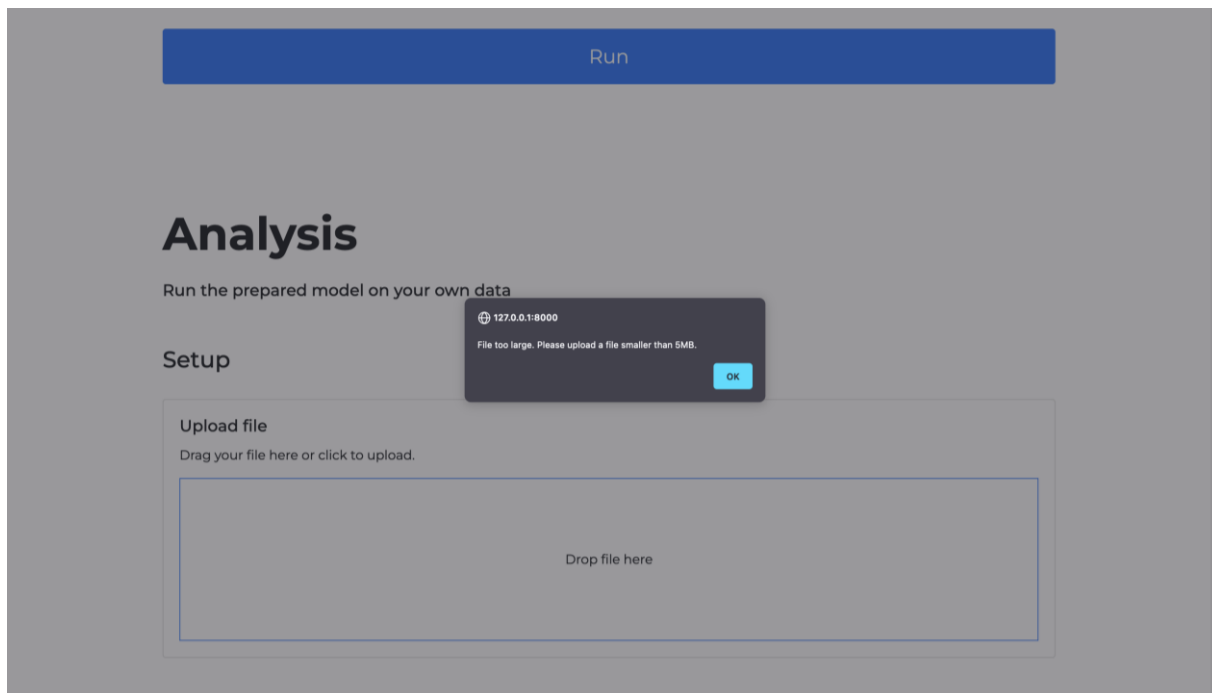


Рисунок 3.12 – Відображення помилки про неправильний формат файлу.

Якщо розмір завантаженого файлу перевищує допустимий ліміт 5MB (рисунок 3.13):

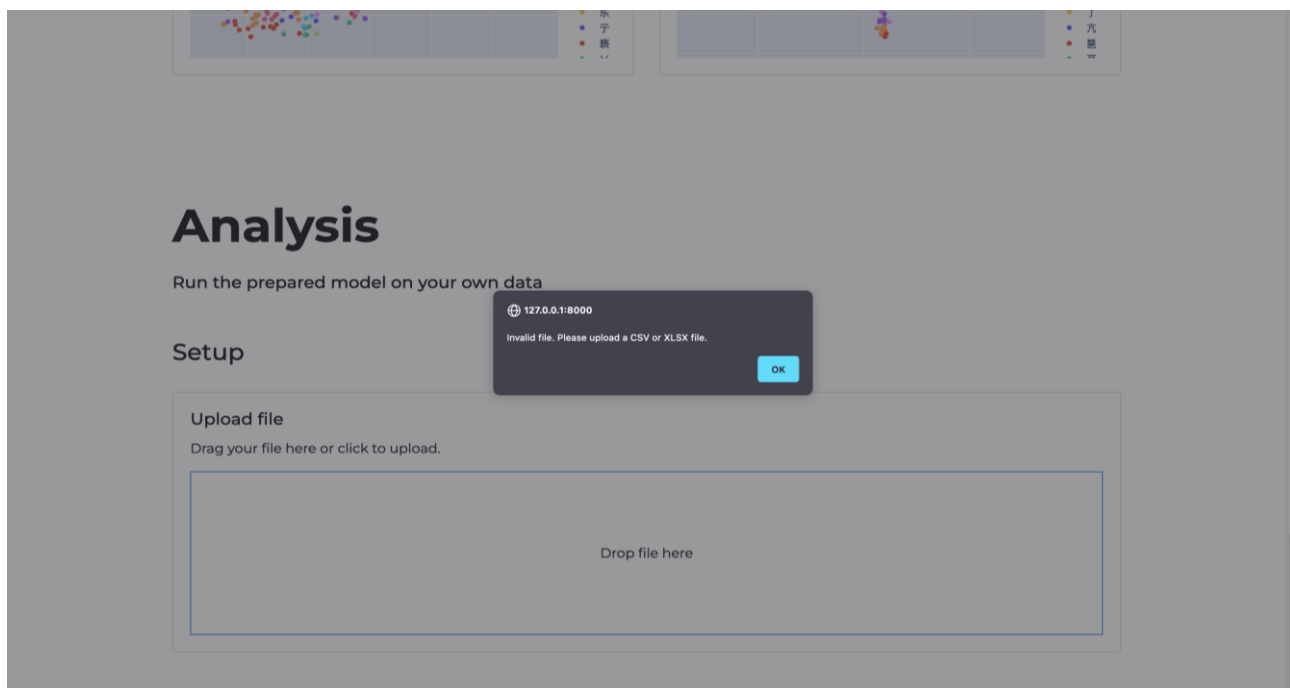


Рисунок 3.13 – Розмір завантаженого файлу перевищує допустимий ліміт

Введення значення FS.

Якщо значення FS введене некоректно:

					КПІ.ІП-9202.045420.05.34	Арк.
						12
Змін.	Арк.	№ докум.	Підп.	Дата.		

Відображення помилки про необхідність виправити значення FS (рисунок 3.14).

The screenshot shows the 'Analysis' section of a web application. Under the 'Setup' heading, there is an 'Upload file' section where 'test.csv (32186 bytes)' has been uploaded. Below this, the 'Sampling frequency (fs)' field contains the Greek letter ϕ . A red error message states: 'A valid integer is required.' A large blue 'Run' button is at the bottom of the setup area.

Рисунок 3.14 – Необхідність виправити значення FS.

Аналіз файлу ECG.

Розгалуження: Успішний аналіз файлу:

Відображення графіку з позначеними серцевими складовими та передбаченими мітками (рисунок 3.15).



Рисунок 3.15 – Успішний аналіз файлу

Відображення таблиці з серцебиттями, передбаченими мітками, словами-репрезентаціями серцебиттів, векторними формами слів (рисунок 3.16).

Heartbeats			
Number	Predicted Label	Word	Word Vector
1	Normal beat	个举丑	0.024173393845558167,-0.12893390655517578...
2	Normal beat	不产丰	0.14293770492076874,-0.03004932403564453...
3	Normal beat	不举丑	0.154133141040802,-0.03004932403564453...
4	Normal beat	不产丑	0.1129748523235321,0.002674425719305873...
5	Normal beat	不产丑	0.1129748523235321,0.002674425719305873...
6	Normal beat	不产丰	0.14293770492076874,-0.03004932403564453...
7	Normal beat	乘举丰	0.17141009867191315,0.002674425719305873...
8	Atrial premature beat	个举丑	-0.294947266578674,-0.03004932403564453...
9	Normal beat	不产丑	0.1129748523235321,0.002674425719305873...
10	Normal beat	不产丑	0.1129748523235321,0.002674425719305873...
11	Normal beat	不举丰	0.18409597873687744,-0.21342550218105316...
12	Normal beat	乘举丑	0.1414472609758377,-0.10564450174570084...
13	Normal beat	不产丑	0.1129748523235321,0.002674425719305873...
14	Normal beat	不产丰	0.14293770492076874,-0.03004932403564453...
15	Normal beat	不产丑	0.1129748523235321,0.002674425719305873...

Рисунок 3.16 – Таблица з серцебиттями

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ” _____ 2023 р.

**ПРОГРАМНА БІБЛІОТЕКА АНАЛІЗУ ЕКГ ЛІНГВІСТИЧНИМИ
МЕТОДАМИ**

Графічний матеріал

КПІ.ПІ-9202.045420.06.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Юрій ОЛІЙНИК

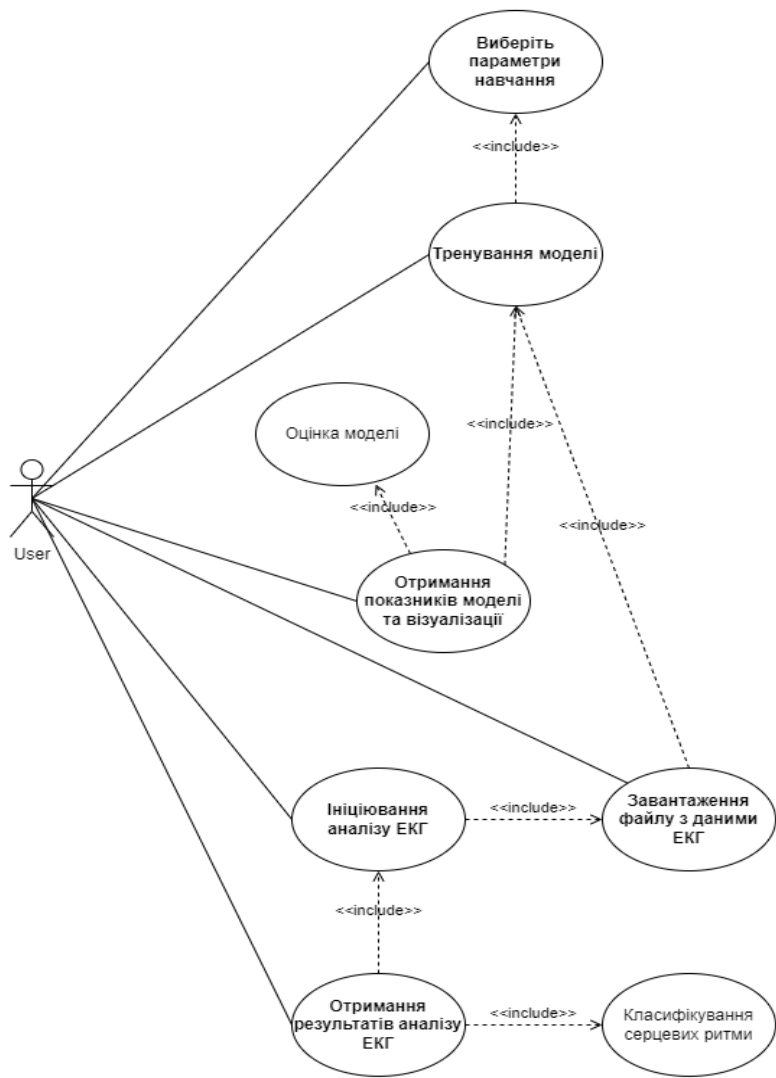
Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Еліза БЕРАУДО

Київ – 2023



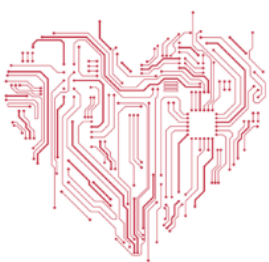
					КПІ.ІП-9202.045420.06.99.CCB				
					Схема структурна варіантів використань	Літера		Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата					
Розробив		Бераудо Е.							
Перевірів		Олійник Ю. О.							
Т. Кон.									
						Аркуш 1		Аркушів 4	
Н. Кон.		Ліщук К.І.			Програмна бібліотека аналізу ЕКГ лінгвістичними методами	КПІ ім. Ігоря Сікорського Кафедра ІПІ гр. ІП-92			
Затвердив		Олійник Ю. О.							



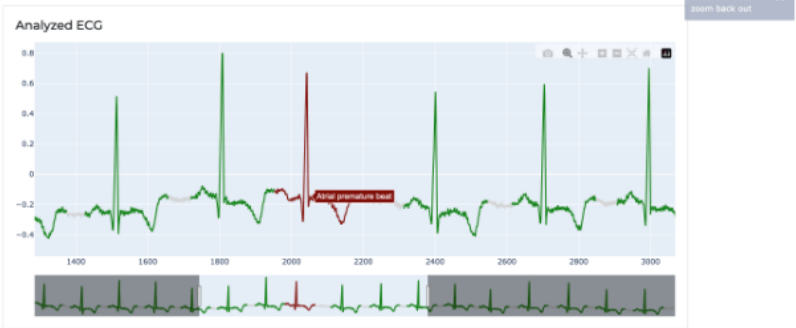
Unleashing the Power of Advanced ECG Analysis

Harness the potential of cutting-edge AI technology to revolutionize cardiac health diagnosis and treatment.

Explore, Train, Diagnose!



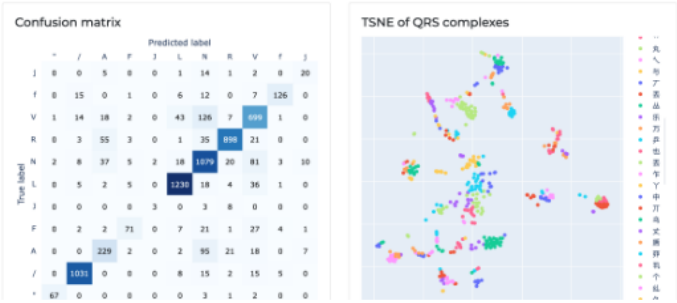
Result



Number	Predicted Label	Word	Word Vector
1	Normal beat	个单旦	0.024173393845558167,-0.1289339065557578...
2	Normal beat	不产丰	0.14293770492076874,-0.03004932403564453...

Results

Training accuracy 94.5%	Precision 85.6%	F1 score 85.4%
Testing accuracy 85.5%	Recall 85.5%	Silhouette score 27.8%



Number	Predicted Label	Word	Word Vector
1	Normal beat	个单旦	0.024173393845558167,-0.1289339065557578...
2	Normal beat	不产丰	0.14293770492076874,-0.03004932403564453...
3	Normal beat	不产旦	0.154133141040802,-0.0403242835180664,0.4876774600758865...
4	Normal beat	不产旦	0.1129748523235321,0.002674425719305873...
5	Normal beat	不产旦	0.1129748523235321,0.002674425719305873...
6	Normal beat	不产丰	0.14293770492076874,-0.03004932403564453...
7	Normal beat	不产丰	0.1714100986779135,0.034484638375889...
8	Atrial premature beat	个单旦	-0.294947266578674,-0.034484638375889...
9	Normal beat	不产旦	0.1129748523235321,0.002674425719305873...
10	Normal beat	不产旦	0.1129748523235321,0.002674425719305873...
11	Normal beat	不产丰	0.18409597873687744,-0.2134255021810536...
12	Normal beat	不产旦	0.1434472609758377,-0.10564450174570084...
13	Normal beat	不产旦	0.1129748523235321,0.002674425719305873...
14	Normal beat	不产丰	0.14293770492076874,-0.03004932403564453...
15	Normal beat	不产旦	0.1129748523235321,0.002674425719305873...

				КП.ІП-9202.045420.06.99.KE			
Зм.	Арк.	№ документа	Підпис	Дата	Креслення виду екранних форм		
Розробив		Бераудо Е.					
Перевірив		Олійник Ю.О.					
Т. Кон.							
Н. Кон.		Ліщук К.І.			Програмна бібліотека аналізу ЕКГ лінгвістичними методами		
Затвердив		Олійник Ю.О.					
					Літера	Маса	Масштаб
					Аркуш 4	Аркуші 4	
					КП ім. Ігоря Сікорського Кафедра ІПІ гр. ІП-92		