НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.4

«До захисту допущено»
В.о. завідувача кафедри
_____ Едуард ЖАРІКОВ
«___»_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра

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

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Програмне та математичне забезпечення аналізу ЕКГ на основі WORD2VEC моделі»

Виконав:
студент II курсу, групи ІП-02мп
Язенок Михайло Сергійович _____

Керівник:
доц., к.т.н., Олійник Юрій Олександрович _____

Рецензент:

_____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення інформаційних систем»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Едуард ЖАРІКОВ

«___»_____2021р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Язенку Михайлу Сергійовичу

1. Тема дисертації «Програмне та математичне забезпечення аналізу ЕКГ на основі WORD2VEC моделі», науковий керівник дисертації доц., к.т.н., Олійник Юрій Олександрович, затверджені наказом по університету від «27» жовтня 2021 р. № 3587-с

2. Термін подання студентом дисертації «6» грудня 2021 р.

3. Об’єкт дослідження – Аналіз сигналу електрокардіограми із застосуванням методів NLP.

4. Предмет дослідження – Методи та засоби створення програмного забезпечення для рішення задач аналізу сигналу електрокардіограми з можливістю застосування методів обробки текстових даних.

5. Перелік завдань, які потрібно розробити:

– проаналізувати наявні програмні засоби імпортування, обробки та збереження даних сигналів електрокардіограми;

– розробити програмну бібліотеку для полегшення розробки програмного забезпечення аналізу даних за допомогою моделі Word2Vec;

– застосувати розроблену бібліотеки для аналізу даних сигналу ЕКГ на основі методів Random Forest, Gradient Boosting, TextRank, Word2Vec;

– проаналізувати ефективність розробленої бібліотеки та точність моделі Word2Vec.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу – 3 плакати

7. Орієнтовний перелік публікацій – одна публікація

8. Консультанти розділів дисертації

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

9.Дата видачі завдання «30» вересня 2021 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Вивчення рекомендованої літератури	01.09.2021	
2	Аналіз існуючих методів розв’язання задачі	13.09.2021	
3	Постановка та формалізація задачі	19.09.2021	
4	Аналіз вимог до програмного забезпечення	23.10.2021	
5	Моделювання програмного забезпечення	01.10.2021	
6	Розробка програмного забезпечення	21.10.2021	
7	Виконання експериментальних досліджень	08.11.2021	
8	Оформлення пояснювальної записки	16.11.2021	
9	Подання дисертації на попередній захист	22.11.2021	
10	Подання дисертації на захист	6.12.2021	

Студент Михайло ЯЗЕНОК

Науковий керівник Юрій ОЛІЙНИК

РЕФЕРАТ

Магістерська дисертація: 116 сторінок, 21 рисунок, 31 таблиця, 2 додатки, 31 джерело

Актуальність теми. На сьогоднішній день перспективним напрямком розвитку програмного забезпечення є розробка алгоритмів із застосуванням методів машинного навчання. Дані методи надають змогу прогресувати точність рішень задач різних напрямків, де раніше неможливо було зробити хоч якісь висновки без наявності висококваліфікованого персоналу, не говорячи вже про те, що для досягання рішення деяких задач, навість при наявності найінтелектуальнішої групи дослідників, необхідно витратити значний проміжок часу.

Однією із таких сфер є комплекс задач основаних на даних сигналу електрокардіограми з можливістю застосування NLP, тому дуже важливо мати відповідний набір програмних інструментів для розробки програмного забезпечення, які б забезпечили швидке та ефективне досягання результатів.

Мета дослідження. Покращення можливостей бібліотек, призначених для розробки програмного забезпечення для аналізу ЕКГ сигналу із застосуванням методів NLP.

Об’єкт дослідження. Аналіз сигналу електрокардіограми із застосуванням методів NLP.

Предмет дослідження. Методи та засоби створення програмного забезпечення для рішення задач аналізу сигналу електрокардіограми з можливістю застосування методів обробки текстових даних.

Наукова новизна отриманих результатів. Вперше створено програмну бібліотеку аналізу сигналу ЕКГ методами NLP.

Практичне значення отриманих результатів полягає у застосуванні розробленої бібліотеку для більш ефективної розробки програмного забезпечення та ведення досліджень сигналу електрокардіограми

Результати проведених досліджень лексичного представлення ЕКГ надають можливість розроблення нестандартних алгоритмів .

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась на кафедрі інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» в рамках теми «Методи та технології високопродуктивних обчислень та обробки надвеликих масивів даних». Державний реєстраційний номер 0117U000924.

Апробація. Результати роботи доповідалися на «Першій Всеукраїнській науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології»(SoftTech-2021)»: секція кафедри інформатики та програмної інженерії. Матеріали конференції. – Київ. – 2021. 22-26 листопада 2021р. – С.53.

Публікації. Наукові положення опубліковані в тезах наукової конференції «Перша Всеукраїнська науково-практична конференція молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології»(SoftTech-2021)». Секція кафедри інформатики та програмної інженерії. Матеріали конференції. – Київ. – 2021. 22-26 листопада 2021р. – С.53.

Ключові слова: NLP, МАШИННЕ НАВЧАННЯ, АНАЛІЗ ЕКГ, ЛІНГВІСТИЧНЕ ПРЕДСТАВЛЕННЯ ЕКГ, БІБЛІОТЕКА

ABSTRACT

Master's dissertation: 116 pages, 21 images, 31 tables, 2 attachments, 31 referring sources.

Topicality. To date, a promising area of software development is the development of algorithms using machine learning methods. These methods allow to progress the accuracy of solving problems in different areas, where previously it was impossible to draw any conclusions without highly qualified staff, not to mention that to achieve some problems, even with the most intelligent group of researchers, it takes a long time.

One of these areas is a set of tasks based on the data of the electrocardiogram signal with the possibility of using NLP, so it is very important to have an appropriate set of software tools for software development that would ensure rapid and effective results.

The purpose of the dissertation research. Improving the capabilities of libraries designed to develop software for ECG signal analysis using NLP methods.

Object of study. Analysis of the electrocardiogram signal using NLP methods.

Subject of research. Methods and means of creating software for solving problems of electrocardiogram signal analysis with the possibility of using methods of text data processing.

Scientific novelty. For the first time, a software library for ECG signal analysis by NLP methods was created.

The practical value of the obtained results is that developed library can be used for more effective software development and electrocardiogram signal research.

The results of studies of the lexical representation of the ECG provide an opportunity to develop non-standard algorithms .

Relationship with working with scientific programs, plans, topics. The work was performed at the Department of Informatics and Software Engineering of the National Technical University of Ukraine "Kyiv Polytechnic Institute. Igor Sikorsky ”in the framework of the topic“ Methods and technologies of high-performance

computing and processing of ultra-large data sets ”. State registration number 0117U000924.

Testing. The results of the work were reported at the "First All-Ukrainian scientific-practical conference of young scientists and students" Software Engineering and Advanced Information Technologies "(SoftTech-2021)": section of the department of informatics and software engineering. Conference materials. – Kyiv. – 2021. November 22-26, 2021. – P.53.

Publications. Scientific provisions were published in the abstracts of the scientific conference "The First All-Ukrainian Scientific and Practical Conference of Young Scientists and Students" Software Engineering and Advanced Information Technologies "(SoftTech-2021)". Section of the department of informatics and software engineering. Conference materials. – Kyiv. – 2021. November 22-26, 2021. – P.53.

Keywords: NLP, MACHINE LEARNING, ECG ANALYSIS, LINGUISTIC ECG REPRESENTATION, LIBRARY

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЕКГ (електрокардіограма)	–	графічне представлення серцебиттів
Бібліотека	–	програмне забезпечення, яке можна використати для розробки іншого програмного забезпечення
Модель-класифікатор (алгоритм-класифікатор)	–	алгоритм, який може визначити приналежність об’єкту до того або іншого класу. Зазвичай такі алгоритми потрібно навчати (надавати вже класифіковані дані) перед використанням
Лінгвістичний вигляд	–	представлення об’єкту за допомогою літер природної мови
Природна мова (Natural Language)	–	мова, за допомогою якої спілкуються люди
Natural Language Processing	–	(скорочено: NLP - обробка природної мови) процеси обробки природної мови (зазвичай деякими програмними алгоритмами з метою дослідження)
Задача класифікації	–	задача, яка потребує визначення приналежність певного об’єкту до певного класу (кількість та назви класів відомі заздалегідь)
Задача бінарної класифікації	–	задача класифікації з двома класами

Кластер	–	згруповані дані за певними характеристиками
Рівень абстракції	–	термін в програмному забезпеченні, який позначає ступінь абстрагування відношення до певного об’єкту або класу
Датасет	–	набір даних (зазвичай великого розміру) єдиного типу
Кешування	–	процес використання пам’яті комп’ютера для можливості прискорення процесу повторного виконання дій

ВСТУП

На сьогоднішній день застосування алгоритмів машинного навчання в різних сферах є ключовим елементом дослідження даних різного характеру для отримання поштовхів прогресу науки або ж для автоматизації системи формування висновків, досягання яких зазвичай потребує присутності людського рішення.

Програмні бібліотеки для машинного навчання, які існують сьогодні надають широкий спектр можливостей для розробників, полегшуючи процес написання продуктів. Саме тому, покращення даних програмних засобів є ключовим для прискорення етапу реалізації наукових ідей.

Однією зі сфер застосування алгоритмів машинного навчання є аналіз сигналу електрокардіограми з метою визначення медичних висновків, або ж для відстеження інших характеристик, які можуть призвести до наукового відкриття або ж програмного засобу для полегшення роботи лікарів. Також, одним з цікавих напрямків дослідження сигналів електрокардіограм є застосування засобів NLP на лінгвістичному представлені сигналів.

Процес дослідження лінгвістичного представлення сигналу ЕКГ включає в себе застосування програмних засобів, які надають можливості: отримання, форматування та збереження датасетів; перетворення стандартного чисельного представлення ЕКГ в лінгвістичне; самого аналізу методами машинного навчання та проміжних етапів обробки. Найпопулярнішими бібліотеки, які можуть забезпечити дані потреби розробника є: бібліотека Scikit Learn, в якій присутні алгоритми та моделі машинного навчання для векторизованих даних, бібліотека Gensim, яка надає засоби NPL та бібліотека WFDB яка надає доступ до множини датасетів сигналів електрокардіограм.

Перелічені програмні засоби мають ряд недоліків в контексті обробки ЕКГ, серед цих недоліків: занадто низький рівень абстракції, відсутність системи

автоматичного кешування, відсутність додаткових інструментів для обробки сигналів ЕКГ.

Отже, для покращення інструментів, було вирішено виконати розробку бібліотеки, яка б виправила перелічені недоліки, скоротила кількість коду для рішення простих задач та надала б змогу зосередитися розробнику на самій суті дослідження.

1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ДЛЯ АНАЛІЗУ СИГНАЛУ ЕКГ

1.1 ЕКГ

Електрокардіограма використовується для перевірки активності серцевого м’яза. Виявлення результатів зазвичай може проводитися на основі людини, яка знаходиться у стані покою, виконує фізичні вправи або після застосування конкретних лікарських засобів, які впливають на точність отриманих результатів та надають змогу оцінити зміни в різних умовах. Результатом дослідження є набір даних, який допомагає визначити серцевий ритм, стан серцевого м’яза та кровотоку.

Електрокардіографія - це метод графічного запису електричних явищ, що відбуваються під час діяльності серцевого м’яза з поверхні тіла [8]. Крива, що відображає електричну активність серця, називається електрокардіограмою (ЕКГ). Тому електрокардіограму можна назвати зареєстрованими коливаннями різниці потенціалів, що виникають під час збудження серця (рис. 1.1).

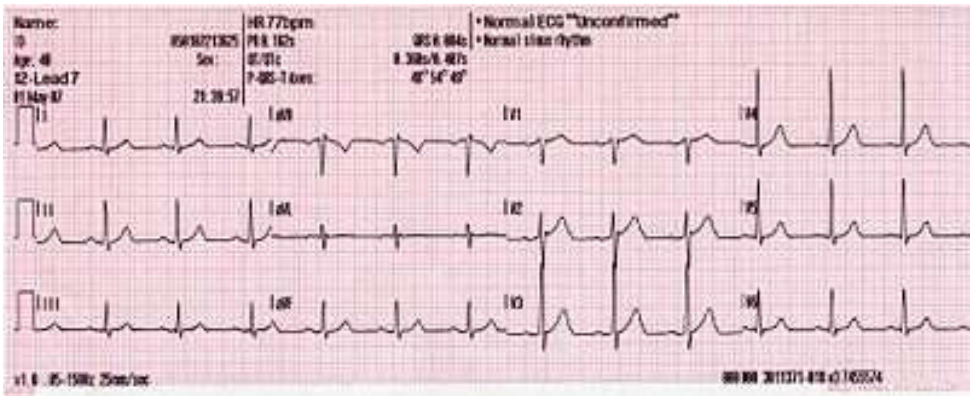


Рисунок 1.1 – Приклад ЕКГ на міліметровому папері

Електрокардіограма - один з найефективніших методів дослідження серця та діагностики серцево-судинних захворювань [8]. Електрокардіограма необхідна для діагностики аритмії та ішемічної хвороби серця. За допомогою цього методу ми можемо точно спостерігати і вивчати зміни міокарда, його поширеність, глибину та час виникнення. Електрокардіограма може виявити

порушення харчування та загартовування в міокарді, а також порушення електролітного обміну, викликане впливом різних токсичних речовин. Електрокардіограма широко використовується для перевірки роботи серцево-судинної системи. Поєднання електрокардіограмного обстеження та функціонального тестування допомагає виявити потенційну недостатність коронарної артерії, минущу аритмію та диференціальну діагностику функціональних та органічних захворювань серця.

1.2 Комплекс QRS ЕКГ сигналу

Комплекс QRS, або короткошлуночковий комплекс, скорочено QRS, являє собою групу зубів, що відображаються на електрокардіограмі (ЕКГ), він відображає складний процес поширення збудження, викликаний деполяризацією двох шлуночків серця [12].

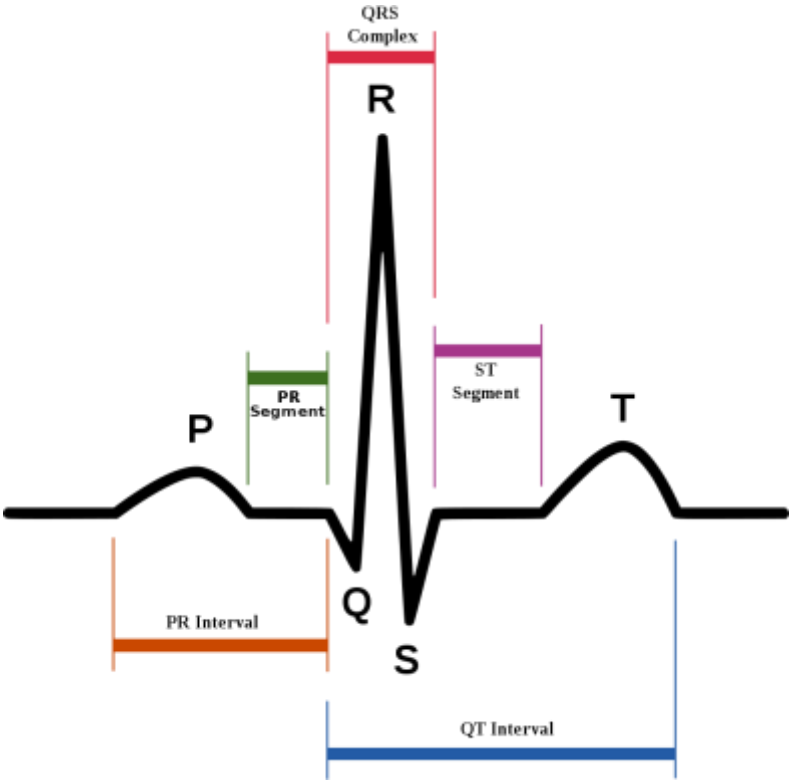


Рисунок 1.2 – QRS комплекс сигналу ЕКГ

Комплекс QRS має один або два позитивні зубці R та один або два негативні зубці S та може мати негативний зубець Q. В залежності від розміщення електричної осі серця, амплітуда зубців змінюється. Також, на ширину та тривалість QRS впливають вік та стать аналізованого пацієнта. Найширша верхня межа становить 0,12 секунди (120 мілісекунд), а нормальна тривалість QRS зазвичай становить 100-120 мілісекунд. В залежності від амплітуди використовується різні літерні позначення, так, якщо вона більша за 5 мм, то позначається літерами верхнього регістру: Q, R, S, якщо ж менша, то літерами нижнього регістру [10].

1.3 Метод перетворення ЕКГ у лінгвістичний вигляд

Для аналізу чисельних даних графіку електрокардіограми методами NLP, необхідно виконати перетворення у лінгвістичний вигляд. Існує багато способів для виконання даного перетворення. Одним з таких є кластеризація комплексу QRS [28].

Для початку необхідно розділити весь ЕКГ сигнал на окремі серцебиття, в кожному з яких можна виділити комплекс QRS. В кожному такому комплексі необхідно виконати розбиття на ключові ділянки: P, QRS, T, згідно опису зазначеному в попередньому підрозділу. Далі, ці ділянки передаються в алгоритм кластеризації, наприклад K-means [15], де кожному кластеру ставиться відповідність своя літера. Після виконання кластеризації зможемо отримати лінгвістичне представлення кожного серцебиття у трьох-літерному вигляді. Поєднавши серцебиття, отримаємо речення, які вже можна використовувати у значній кількості алгоритмів обробки природної мови. Використання лінгвістичного методу описано на конференції «Computational Linguistics and Intelligent Systems (COLINS 2020)» [30]

Перетворення сигналу ЕКГ в модель word2vec описано в магістерській роботі «АНАЛІЗ ЕКГ СИГНАЛІВ ЗА ДОПОМОГОЮ МОДЕЛІ WORD2VEC» [31]

1.4 Метод обробки природної мови Word2Vec

Word2Vec – методика, яка представляє собою спосіб обробки природної мови (NLP). Вперше була опублікована в 2013 році [21]. Дана модель може оцінювати великі об’єми тексту для подальшого використання з метою утворення іншого тексту або ретельного аналізу векторного представлення слів моделі. Основними функціями моделі є пошук слова-синоніма та продовження речення в тексті. Як випливає з назви моделі, вона також здатна перетворювати слова у векторне представлення. В свої основі Word2Vec використовує нейронну мережу, яка здатна перетворювати слова у набір чисел, які є насправді векторами. Для даних векторів може бути застосована математична функція – косинус подібності – яка вкаже, наскільки два слова є подібними між собою. Ця властивість утворених векторизованих слів надає можливість використовувати модель у різних дослідженнях природних мов [13].

Під «капотом» дана модель реалізована як набір пласких двошарових найронних мереж. Ці моделі тренуються за допомогою великого набору тексту. В результаті даний текст допомагає утворити векторний простір, в якому кожне слово має відповідний вектор.

Word2Vec має декілька ключових параметрів, які в подальшому впливають на отриманий результат: алгоритм тренування, недовибирання, розмірність та вікно контексту.

Алгоритмом тренування може бути softmax, яка краще працює для рідких повторів слів, або negative sampling, яке краще працює для слів з великою кількістю входження. Також, дані алгоритми можуть бути використані в комплексі [1].

Недовибирання вказує поріг частоти входження слова, після якого слово перестає оброблюватися. Слова з великою частотою входження можуть надавати мало інформації.

Розмірність означає розмірність простору, в якому відображаються слова, зі збільшення розмірності – збільшується точність кінцевого результату.

Вікно контексту вказує, яка кількість слів, які оточуються поточне слово, буде впливати на векторне відображення поточного слова.

1.4.1 Skip-Gram

Skip-Gram – це один з двох алгоритмів, на основі яких може бути побудована модель Word2Vec. Головним використанням цього алгоритму є визначення контексту в якому, скоріш за все буде використане слово. Таким чином на основі одного слова можна визначити оточуючі його інші слова та сформувати таким чином цілий текст з використання навченої моделі.

Алгоритм Skip-Gram реалізований на основі найронної мережі з один прихованим шаром [1], як показано на рисунку 1.3

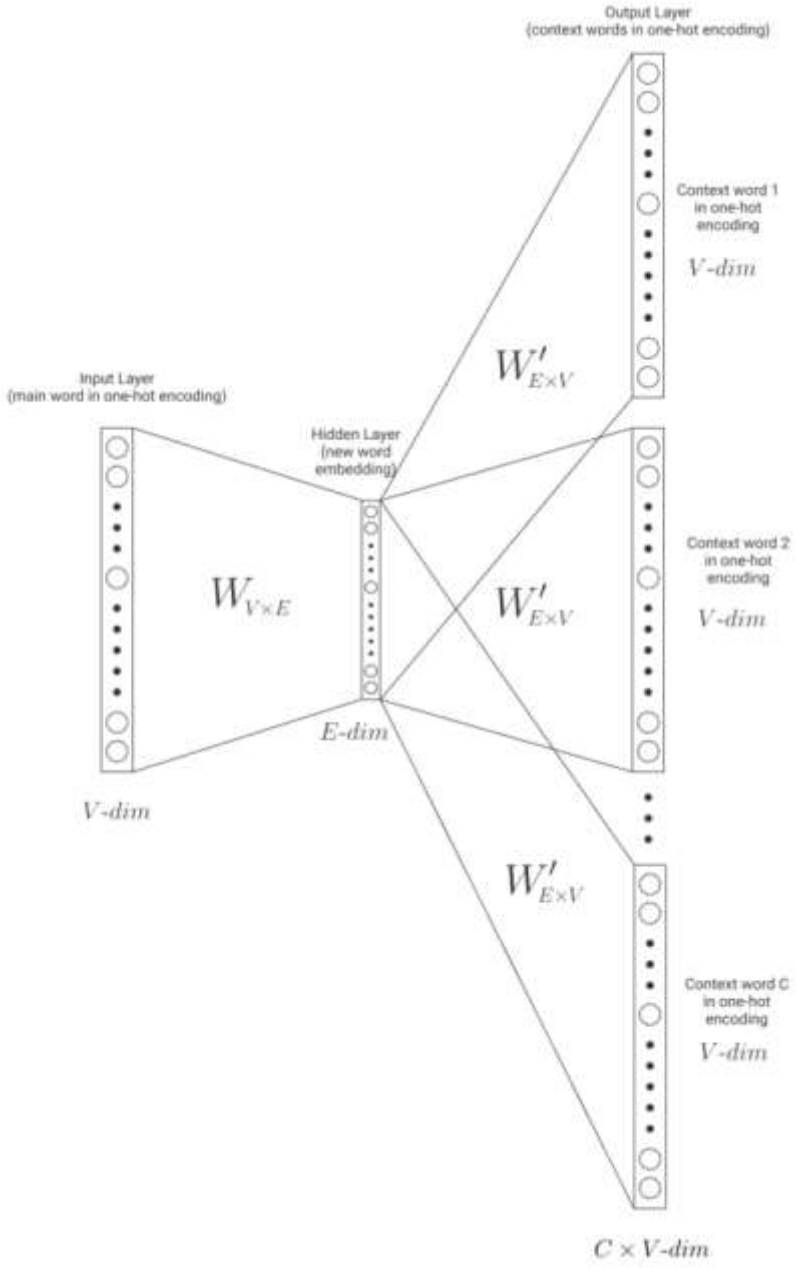


Рисунок 1.3 – Схема нейронної мережі Skip-Gram

Як видно з рисунку 1.3, нейронна мережа має вдідний шар, ромірністю V, де V це є розмір вихідного словарю навченої моделі, вхідними даними цього шару є one-hot кодування слова [24]. Далі вхідний шар перетворюється на прихований за допомогою вагової матриці W. І нарешті дані з прихованого шару перетворюються на вихідний шар за допомогою вагової матриці W`. Вихідними даними є передбачені слова контексту в кодуванні one-hot [1].

1.4.2 Continuous Bag of Words

Алгоритм CBOW або Continuous Bag of Words використовується для передбачення слова, яке буде вжито в наданому контексті (операція протилежна Skip-Gram). Модель нейронної мережі даного алгоритму представлена на рисунку 1.4.

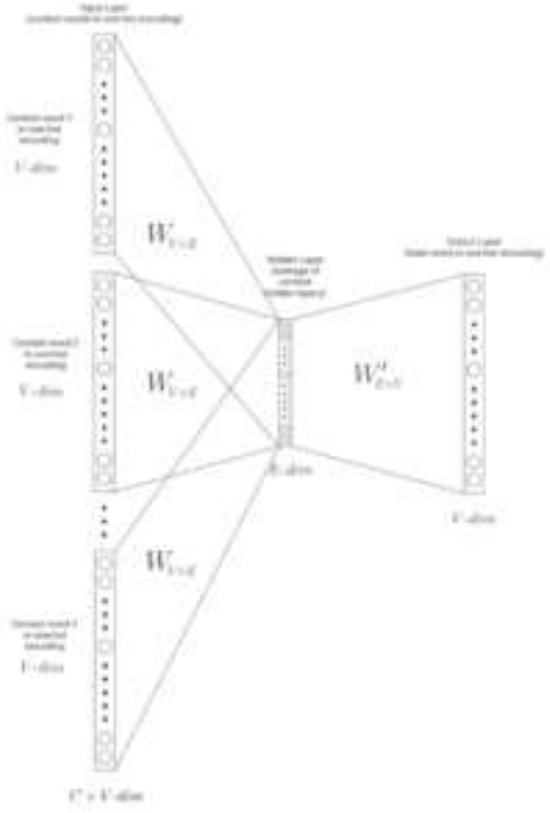


Рисунок 1.4 – Схема нейронної мережі CBOW

Дана нейрона мережа, представляє собою рівне відзеркалення нейронної мережі Skip-Gram.

1.5 Методи класифікації

1.5.1 Random Forest

Random Forest – алгоритм машинного навчання, який може бути використаним для класифікації, регресії та інших задач.

Алгоритм працює на основі побудованих під час навчання дерев прийняття рішень (decision trees), які можуть надати моду класів або усереднений прогноз.

Кожне дерево представляє собою комплекс рішень на основі вхідного датасету та параметрів, які надає сам метод Random Forest [2]. Спрощена модель внутрішньої реалізації моделі зображена на рисунку 1.5

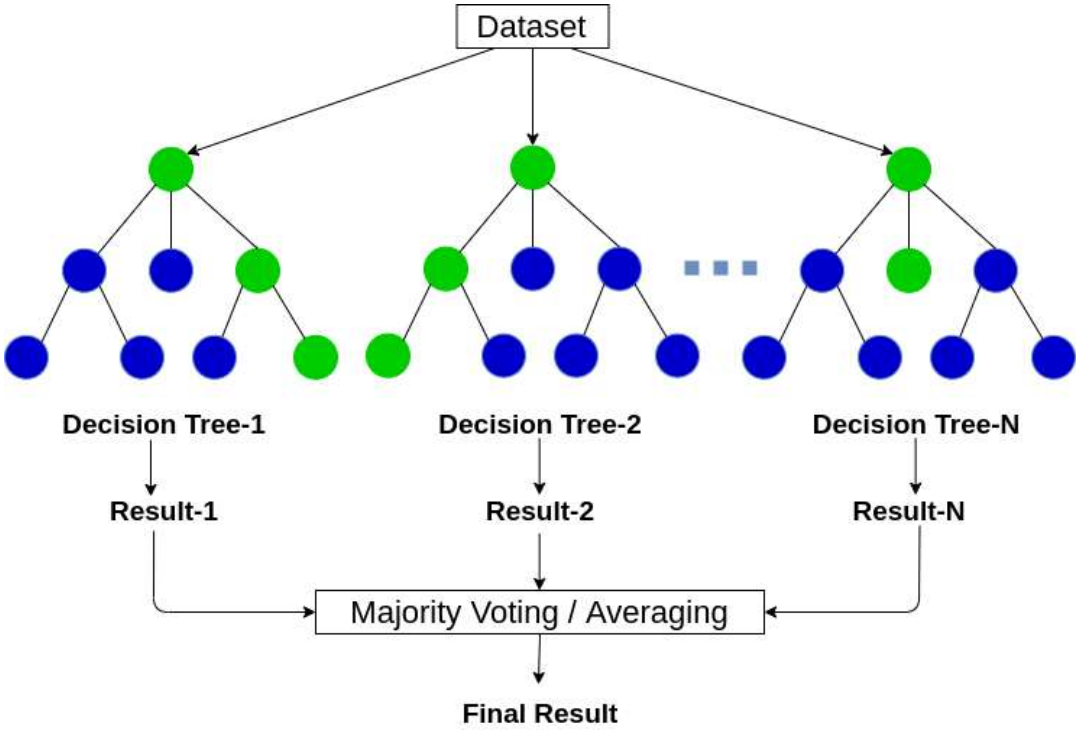


Рисунок 1.5 – Спрощена схема внутрішньої реалізації алгоритму Random Forest

Даний алгоритм може бути чудово застосованим для даних з великою кількістю ознак та класів. На вхід може приймати як дискретні так і безперервні ознаки, масштабування ознак не впливає на точність кінцевого результату. Також, підходить для оброблення великих масивів даних за рахунок того, що здатен вести навчання в декількох потоках.

Основним недоліком алгоритму є потенціальне перенавчання при деяких умовах, при навчанні на даних з великою кількістю неточностей та припущень.

1.5.2 Gradient boosting

Gradient boosting – алгоритм машинного навчання, який використовується для регресійного аналізу, задачі класифікації та інше. В основі використовує поєднання слабких моделей передбачення – зазвичай дерева прийняття рішень. Навчена модель, зазвичай, при однакових умовах, перевищує точність результатів алгоритму Random Forest, але абсолютно точно існує певна область задач та вихідних даних, в якому Random Forest може отримати більш точні результати [2].

Як і інші алгоритми «посилення», Gradient Boosting ітеративно покращує вихідний результат шляхом поєднання слабких алгоритмів машинного навчання [25]. Середня квадратична похибка є основним показником, на який орієнтується даний алгоритм. Шляхом її зменшення утворюється більш сильна модель, здатна до передбачення даних з більшою точністю. Спрощена візуалізація ітеративного покращення результату наведена на рисунку 1.6.

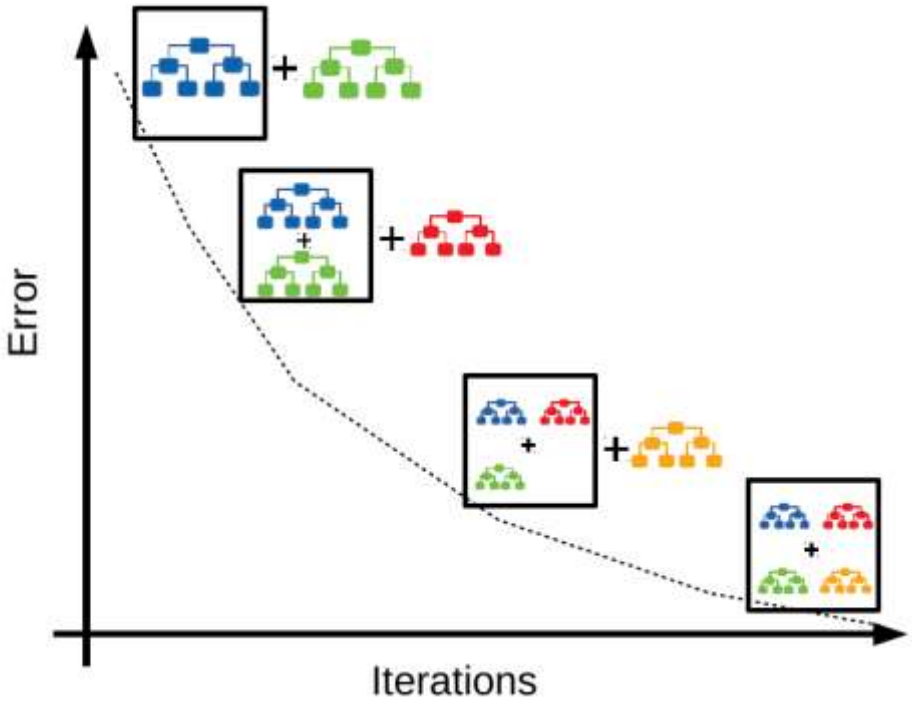


Рисунок 1.6 – Спрощена візуалізація ітеративного покращення моделі Gradient Boosting

1.6 Алгоритм стандартизації датасету Feature scaling

Feature scaling – це алгоритм, який використовуються для перед-обробки датасету з метою стандартизації вхідних даних, які потім будуть передані на вхід алгоритмам аналізу даних [2].

Вихідні дані рахуються як різниця між вхідними даними та середнього значення, поділене на стандартне відхилення. Де середнє значення та стандарте відхилення рахуються шляхом застосування стандартних формул на наданому тренувальному датасеті.

1.7 Наявні бібліотеки для реалізації обробки сигналу ЕКГ

1.7.1 Бібліотека Scikit Learn

Scikit Learn або sklearn – це бібліотека реалізована на основі мови Python, яка надає основні механізми для створення моделей, які в подальшому використовуються для передбачення даних. Також, дана бібліотека надає велику кількість додаткових інструментів та алгоритмів для перед-обробки, пост-обробки, збереження та перетворення даних [2].

Основними типами алгоритмів обробки даних в даній бібліотеці є зменшення розмірності моделі, обрання моделі, регресія, класифікація та кластерний аналіз. Також, дана бібліотека надає декілька базових набор даних для тестування та перевірки алгоритмів.

На рисунку 1.5 представлена схема модулів бібліотеки sklearn.

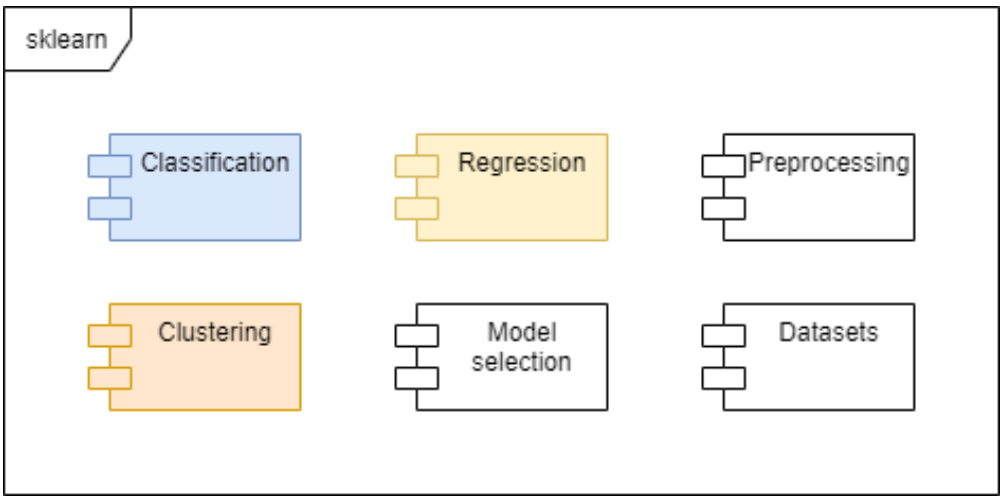


Рисунок 1.7 – Схема модулів бібліотеки Scikit Learn

1.7.2 Бібліотека Gensim

Gensim – це бібліотека з відкритим кодом, написана на Python. Дана бібліотека є осередком реалізації алгоритмів для представлення текстових документів у вигляді семантичних векторів, які можуть бути використані в алгоритмах аналізу даних, вхідними значеннями яких можуть бути виключно векторизовані дані [1].

Основні алгоритми реалізовані в Gensim мають під собою спільний фундамент функціоналу, який полягає у визначенні семантичної структури наданих текстових документів шляхом аналізування статистичних повторень схожих по суті схем текстових речень. Вхідними даними таких алгоритмів не обов'язково повинні бути тексти, які може зрозуміти людина, таким чином можна застосовувати всі ці методи для знаходження статистичних закономірностей будь-яких наборів слів, які представляються собою поєднання абсолютно вільних символів.

Принципи бібліотеки Gensim наголошують на тому, що вони підтримують більш практичну точку зору використання алгоритмів для вирішення реальних задач. Тобто набір алгоритмів більше сфокусований на інженерному застосуванні ніж на академічному.

1.7.3 Недоліки бібліотек при використанні у контексті обробки ЕКГ сигналу

Розглянуті бібліотеки мають ряд недоліків при використані програмістами з метою вирішення задач широкого кола. Рівень абстракції, який пропонується зазначеними в попередніх розділах модулями, не може влаштовувати програмістів які прагнуть отримати максимальний результат з мінімальними дослідженнями архітектури бібліотеки. Зокрема в даних бібліотеках відсутній механізм автоматичного кешування здобутих даних з метою скорочення часу повторної обробки датасетів, які зазвичай є надто великими. Python – це інтерпретована мова програмування зі строгою динамічною типізацією. Помилки в програмному забезпеченні, написаному на даній мові, зазвичай можна виявити лише по факту виконання (інтерпретування) конкретної строки коду, що призводить до моментальної втрати даних, які не були оголошені як глобальні. Такими даними зазвичай є навчені моделі, які були оголошені в межах конкретних функцій, і на навчання яких було витрачено значні ресурси часу та апаратного забезпечення. Втрата даних в такому випадку призводить до втрати зазначених ресурсів, що негативно впливає на ефективність розробки програмного забезпечення.

Запропонований рівень абстракції та запропонована архітектура в даних бібліотеках значно знижує швидкість написання коду, вимагаючи від програміста написання елементарних частин, які згодом накопичуються, збільшуючи об’єм коду, зменшуючи тим самим читабельність, а також, якщо програміст не має час на ретельну розробку архітектури власної програмної реалізації застосування бібліотек, то ще й впливає на testability, maintainability, reliability та інші характеристики програмного забезпечення, які в зрештою впливають на працездатність коду.

Також, реалізації зазначених бібліотек не надають інструментів роботи більш високого рівня у вузьконаправлених цілях, таких як сфера оброки сигналів електрокардіограми. Не вистачає функціоналу для швидкого написання коду для скачування, перед-обробки та збереження найбільш популярних датасетів електрокардіограм.

Зазначений список недоліків значно впливає на кінцеву оцінку отриманого програмного забезпечення, збільшуючи кількість людино-годин витрачених на його розробку.

1.8 Постановка комплексу завдань

Метою розробки є:

– покращення бібліотек для розробки програмного забезпечення для аналізу ЕКГ сигналу та моделі Word2Vec;

– використання розробленої бібліотеки для класифікації хвороб серця на основі датасету електрокардіограм.

Для виконання поставлених цілей необхідно виконати наступні задачі:

– проаналізувати наявні програмних засобів імпортування, обробки та збереження даних сигналів електрокардіограми;

– розробити програмну бібліотеку для полегшення розробки програмного забезпечення аналізу даних за допомогою моделі Word2Vec;

– застосувати розроблену бібліотеки для аналізу даних сигналу ЕКГ на основі методів Random Forest, Gradient Boosting, TextRank;

– проаналізувати ефективність математичного та програмного забезпечення.

Висновки по розділу

В даному розділі наведено теоретичні основи сигналів електрокардіограм, зокрема їх побудова та принципи перетворення у лінгвістичний вигляд.

Описано принцип роботи алгоритму векторизації лінгвістичних даних за допомогою моделі Word2Vec. Розглянуто алгоритми машинного навчання для задач класифікації та регресії – Random Forest та Gradient Boosting. Виконано огляд програмних бібліотек Scikit Learn та Gensim, які надають реалізацію зазначених алгоритмів машинного навчання.

Визначено основні недоліки бібліотек Scikit Learn та Gensim. Зроблено висновок щодо неефективності їх архітектури та рівню абстракції, з точки зору їх використання розробниками програмного забезпечення аналізу даних.

Визначено комплекс задач, які необхідно вирішити для покращення зазначених вище бібліотек.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Запропоновані розширення бібліотек

Бібліотека складається з декількох модулів, які забезпечують функціонал в окремих ключових етапах обробки вхідних даних, зокрема даних сигналів електрокардіограми. Модульність бібліотеки забезпечує можливість зосередження уваги програмістів на згрупованому функціоналі, який є необхідним на поточному кроці обробки інформації.

2.1.1 Модуль для підготовки датасетів сигналу ЕКГ

Одним з модулів розробленої бібліотеки є модуль для підготовки датасетів сигналу електрокардіограми. Даний модуль надає функціонал для завантаження найрозповсюдженіших датасетів: MIT-BIH AFIB Dataset та PhysioNet MIT-BIH [3]. Об’єм даних у вказаних датасетах забезпечую високу точність при машинному навчанні.

Даний модуль надає широкий інструментарій для обробки самих сигналів ЕКГ, зокрема можна виділити: зручна конвертація датасетів для забезпечення стандартизованого вигляду всього датасету та відсікання зайвих та невалідних серцебиттів у зчитаних сигналах електрокардіограми, таким чином зменшуючи потрібний об’єм пам’яті для зберігання датасетів, при цьому не втрачаючи ключової інформації, яка необхідна при рішенні задач за допомогою машинного навчання та алгоритмів аналізу даних.

Також, модуль надає функціонал для зручного збереження, перезавантаження, переформування даних датасетів.

Для швидкого застосування даних в алгоритмах машинного навчання, надається можливість розділити датасет на тренувальні та валідаційні вибірки з наступним кешування та можливістю переформування. Розділення датасету

відбувається за допомогою спеціального алгоритму, який забезпечую розподілення різних видів даних в рівних пропорціях як в тренувальній так і валідаційній частині, таким чином забезпечується максимальне покриття існуючих основних та крайових ситуацій в електрокардіограмі при навчанні моделі та її перевірки.

Всі ці функції надаються через зручну абстракцію, яка забезпечує вищу швидкість написання коду формування датасетів для подальшої обробки.

2.1.2 Система кешування даних алгоритмів машинного навчання

Кешування даних надається на всіх рівнях покращених алгоритмів, таким чином, програміст може на кожному етапі надійно та, що головне, швидко сформувати кеш даних після кожного етапу обробки інформації.

Механізм кешування також надає можливість переформування даних, таким чином, якщо етап було розроблено неправильно (що нерідко буває під час розробки зі застосуванням алгоритмів машинного навчання), то дані закешовані на ньому, можуть бути згенеровані знову.

Кешування даних, крім того забезпечує програміста інструментом відтворення експерименту. Оскільки деяка частина алгоритмів машинного навчання має визначений момент рандомізації кроків алгоритму, то деякі утворені дані можуть призвести до збою в роботі системи. Такі ситуації, найчастіше, дуже важко відтворити, особливо, якщо в алгоритм не було передано значення основи для генерації даних, яке також називається сідом (англ. seed). Неявне кешування надає можливість відтворення експерименту з оригінальними даними, що дає можливість програмісту виконати коригування алгоритму для забезпечення його роботи в деяких крайніх ситуаціях, які можуть в майбутньому ставатися час від часу.

Також, можна виділити, що кожне розширення основних моделей машинного навчання реалізує спільний інтерфейс, таким чином полегшує роботу програміста. Схема реалізації зображена на рисунку 2.1.

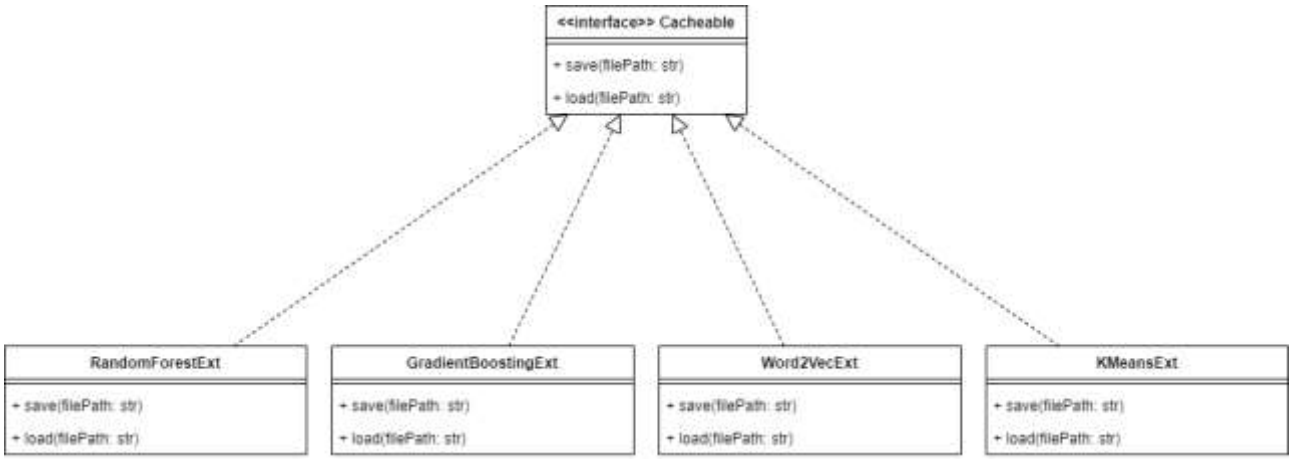


Рисунок 2.1 – UML-схема реалізації інтерфейсу Cacheable основними моделями

2.1.3 Підвищення рівня абстракції бібліотек машинного навчання для обробки ЕКГ сигналу

Бібліотеки sklearn та gensim виконані з таким рівнем абстракції [4], який, під час розробки програмного забезпечення зорієнтованого на аналізування сигналів електрокардіограми, потребує написання значної кількості коду, який є не настільки критичним для проведення аналізу та відволікає розробника від рішення ключової проблеми, яку планує розв’язати програміст після реалізації програмного забезпечення. Безумовно, під час розробки з використанням алгоритмів машинного навчання потребується введення конкретних параметрів, які б надали змогу контролювати поведінку алгоритмів та які б врешті решт вплинули на загальний результат всієї роботи - саме це є основним перешкодженням при піднятті рівня абстракції усієї системи комбінацій машинного навчання.

Рішенням цієї проблеми буде створення класів-розширень для вже існуючих алгоритмів. Розширення буде надавати додатковий більш абстрактний функціонал та буде реалізован як обгортка алгоритмів, які надають зазначені бібліотеки. Розроблені класи-розширення будуть надавати доступ до своїх “основ” (об’єктів з бібліотеки, для яких створюється розширення), таким чином, розробнику програмного забезпечення буде надаватися повний доступ до параметрів низького рівня, які скоріш за все будуть регулюватися для покращення результатів після повного застосування алгоритмів машинного навчання. Розроблений підхід також задовольняє дизайн-патерни об’єктно орієнтованого програмування, зокрема - composition over inheritance (композиція понад спадкування) [5].

Ще одним, безумовно важливою перевагою створення саме розширень функціоналу, а не його переписання (наприклад використання патерну proxy [5]), є те, що створені розширення буде легко підтримувати у випадках значних змін у базових бібліотеках, оскільки доповнення функціоналу залишаються лише доповненнями, і зміна сигнатури методів основи, які не використовуються у доповненнях, ніяк не впливає на реалізацію розширення. А оскільки функціонал, який використовується у доповненнях є відкритим інтерфейсом основи, який зазвичай помітно не змінюється у стабільних реалізаціях бібліотек, тоді й розширення ніяк не потрібно буде змінювати для підтримки нових версій. Таким чином, вихід нових версій бібліотек з критичними змінами будуть одразу підтримуватися.

Спрощена схема використання класу-розширення наведена на рисунку 2.2

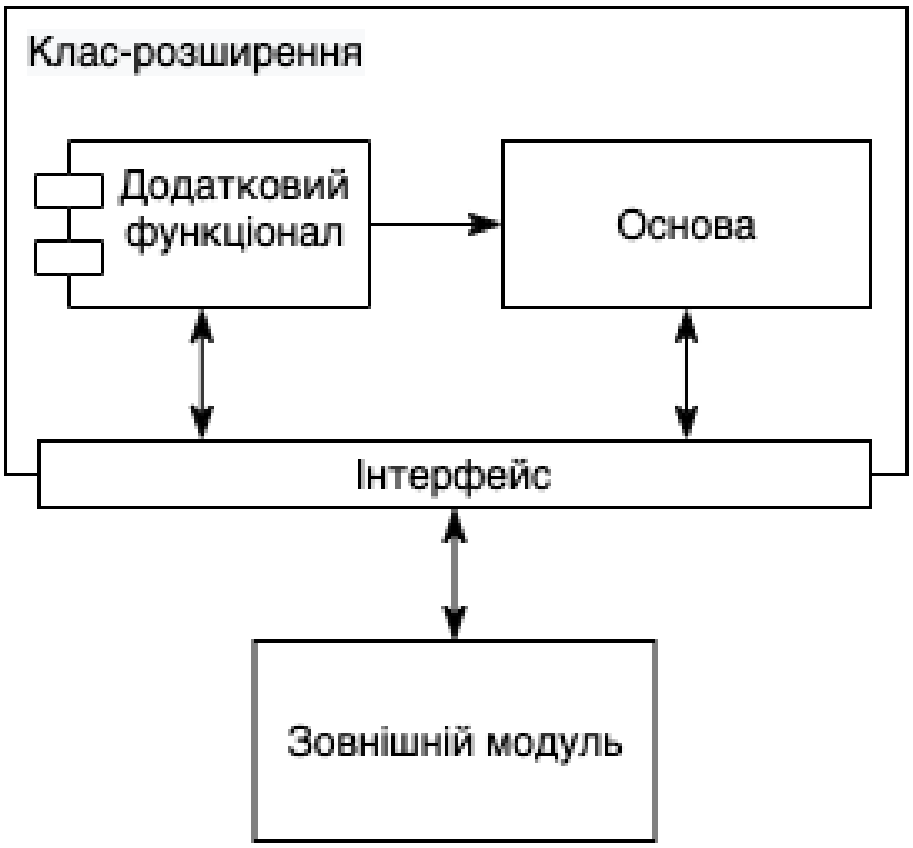


Рисунок 2.2 – Спрощена схема використання класу-розширення

2.2 Розробка класу-розширення для формування датасетів сигналів електрокардіограми

Даний клас-розширення буде надавати наступні функціональні можливості:

- завантаження та кешування неформатованих датасетів;
- стандартизація даних датасетів для більш точного машинного навчання;
- розбиття повної електрокардіограми на окремі серцебиття;
- переформатування даних з відсіканням невалідних серцебиттів;
- конвертація класу аритмії кожного серцебиття у відповідне числове значення;
- формування тренувальної та валідаційної вибірки згідно оцінки ступеня аритмічності кожної електрокардіограмі;

– кешування даних перед переформатуванням, після переформатування та після створення тренувальної та валідаційної вибірки;

– опціональне лімітування вибірки для тестових запусків.

Клас-розширення буде виконано у вигляді обгортки стандартного інструменту отримання датасетів - пакет waveform-database (WFDB).

Клас підтримує стандартні датасети електрокардіограм, наприклад: MIT-BIH Arrhythmia Database та MIT-BIH Atrial Fibrillation Database [3].

2.3 Розробка класу-розширення для моделі Word2Vec

Клас-розширення моделі Word2Vec [1] надає додатковий функціонал, корисний як для електрокардіограми так і для будь-яких інших вхідних даних.

Зокрема, надзвичайно корисним є розширений функціонал кешування, який одразу після підготовки та навчання моделі на основі вказаних параметрів, створює файл-кеш для готової моделі, яку можна явно та неявно використати у процесі написання програмного забезпечення для аналізу даних.

Також, спеціально для моделі Word2Vec було створено окремий функціонал по перетворенню повного масиву слів у відповідне векторне представлення з опущенням невалідних слів, для яких не було створено векторне перетворення всередині навченої моделі Word2Vec.

2.4 Розробка класу-розширення для моделей RandomForest та Gradient Boosting

Стандартні класи RandomForest та Gradient Boosting які надаються бібліотекою sklearn [2] мають схоже функціональне призначення, а саме параметризоване машинне навчання на основі вхідної вибірки вигляду відповідності x - y, проте дані алгоритми мають специфічні параметри притаманні виключно конкретній реалізації, тому об’єднати розширення для них не є можливим.

Для кожного з цих двох моделей машинного навчання були створені окремі розширення, які підтримують кешування та параметризацію алгоритмів за рахунок взаємодії з основою моделі.

2.5 Розробка додаткового функціоналу для аналізу сигналу ЕКГ

Окремий модуль аналізу сигналу електрокардіограми надає базові функції для отримання числового значення класу аритмії та оберненого перетворення з числового значення в клас аритмії, які визначено та описано в базі даних MIT.

Також, даний модуль надає функціонал для розбиття суцільного сигналу на окремі серцебиття та для виділення окремих складових частин QRS комплексу [10] кожного серцебиття. Виділення QRS комплексу може бути корисним під час проведення аналізу отриманих результатів та для формування основного блоку даних для навчання моделей.

Окремо всередині модуля описано клас BeatsToWordsConverter для конвертації сигналу електрокардіограми в лінгвістичний вигляд за допомогою алгоритму кластеризації K-Means [14]. Даний клас, також підтримує систему кешування та об’єднаної операції навчання моделі та кешування для більш спрощеного використання розробниками програмного забезпечення.

Інтерфейс класу BeatsToWordsConverter наведено на рисунку 2.3

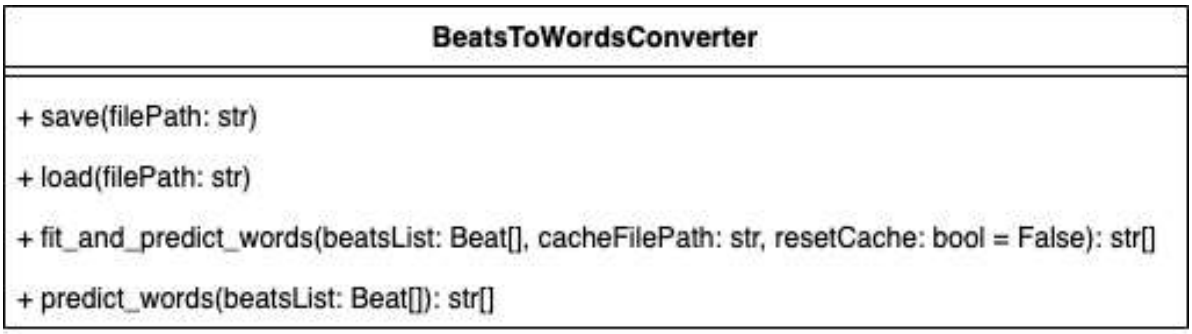


Рисунок 2.3 – Клас BeatsToWordsConverter в нотації UML

2.6 Застосування розробленої бібліотеки для аналізу ЕКГ сигналу

2.6.1 Задача класифікації аритмії на основі сигналу ЕКГ

Для рішення задачі класифікації аритмії на основі сигналу електрокардіограми, було розроблено алгоритм, який наведено на рисунку 2.4.

Рисунок 2.4 – Алгоритм класифікації сигналу ЕКГ

Розроблений алгоритм підтримує просту зміну моделі-класифікатора, для отримання та порівняння результатів при використанні різних алгоритмів машинного-навчання, зокрема Random Forest та Gradient Boosting [7].

Основними етапами рішення задачі класифікації стали:

- зчитування датасету, яке передбачає використання розробленого класу-розширення WFDB [3] з підтримкою основних датасетів електрокардіограми;
- форматування даних датасету для стандартизації та видалення шумів електрокардіограми;
- розбиття сигналу на окремі серцебиття засобами розробленої бібліотеки;
- параметризоване розбиття датасету на тренувальну та тестову вибірки за допомогою алгоритмів прописаних в середині підготовлених інструментів;
- виділення QRS комплексів [12] для розбиття суцільного серцебиття на частини, які піддаються кластеризації;
- навчання моделі K-Means з використання розробленого розширення;
- перетворення кластеризованих даних у лінгвістичне представлення, де кожному кластеру ставиться у відповідність певна літера;
- навчання розширеної моделі Word2Vec;
- перетворення сигналів ЕКГ, які на поточному етапі представляють собою речення, у векторні значення;
- навчання алгоритму-класифікатора тренувальними даними з використання розроблених класів-розширень для Random Forest та Gradient Boosting;
- класифікація валідаційної частини вибірки;
- отримання звіту з зазначеними похибками TF, FT, FF, TT для кожного класу [7].

2.6.2 Задача визначення взаємозв’язку між послідовними сигналами за допомогою Word2Vec

Word2Vec з використання алгоритму Skip Gram [1] може бути використана для пошуку контекстних слів, які оточують обране. Даний алгоритм можна застосувати для послідовності сигналів електрокардіограми для визначення, чи притаманний зв’язок між послідовними сигналами у контексті лінгвістичного представлення електрокардіограми.

Розроблений алгоритм представлено на рисунку 2.5.

Рисунок 2.5 – Алгоритм розрахунку зв’язків між послідовними сигналами ЕКГ

Алгоритм для отримання точності визначення наступного слова на основі попереднього представлено на рисунку 2.6.

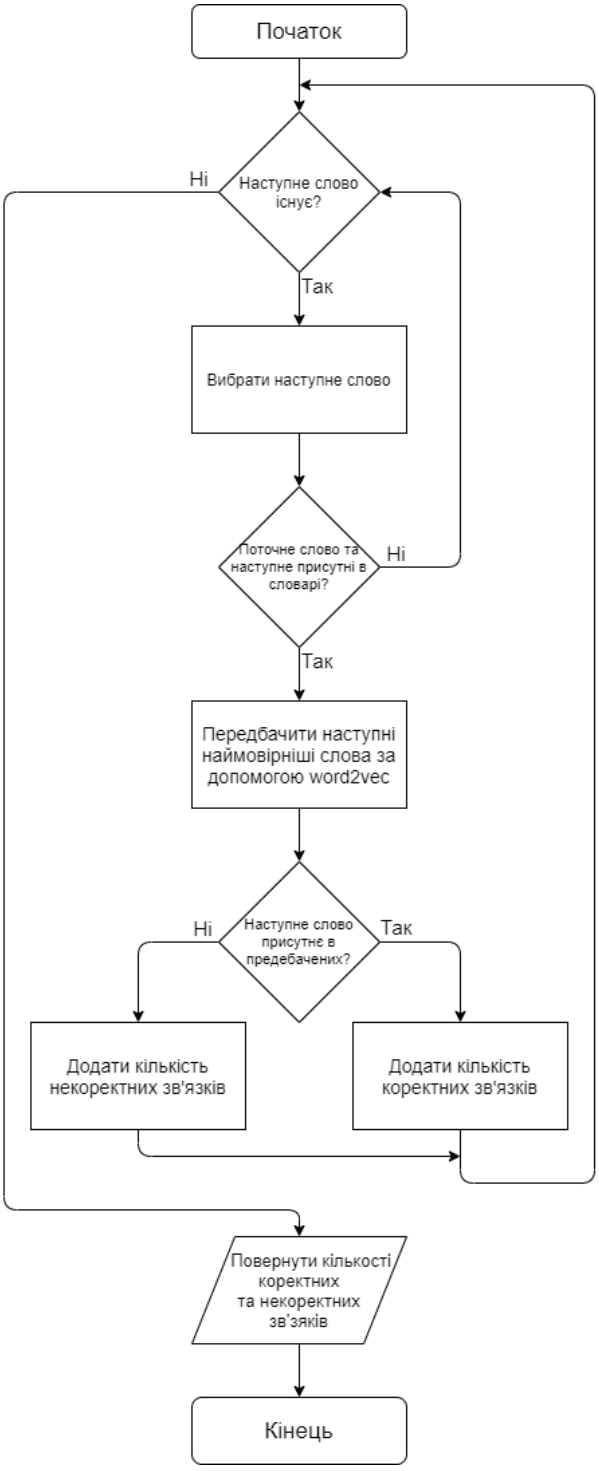


Рисунок 2.6 – Алгоритм отримання кількості передбачених та не передбачених послідовностей

Алгоритм представляє етапи формування датасету, створення word2vec моделі та перевірки кількості передбачених послідовностей слів. Причому враховуються лише ті слова, які входять в словниковий запас навченої моделі, оскільки нам потрібно визначити існування зв’язка між лінгвістичним представленням послідовних сигналів, а не якість алгоритму навчання.

Даний алгоритм можна реалізувати за допомогою підготовлених програмних засобів описаних в реалізованій бібліотеці, зокрема можна застосувати: клас для отримання стандартизованих датасетів та клас-розширення для моделі Word2Vec.

2.6.3 Аналіз застосування методу TextRank для лінгвістичного представлення сигналу ЕКГ

За рахунок представлення сигналу електрокардіограми у вигляді набору слів, можна використовувати алгоритми, які призначені для аналізу текстів, одним з таких алгоритмів є TextRank [7], який може визначити речення, які більше всього передають посил тексту.

Алгоритм застосування TextRank як і в попередніх задачах передбачає зчитування датасету методами розробленої бібліотеки та застосування Beats2words моделі для перетворення сигналу у лінгвістичний вигляд. Після отримання набору текстів, кожен з яких відноситься до конкретного датасету, можна визначити речення, які найбільш точно передають суть кожного датасету. Реченнями в даному випадку виступають сигнали ЕКГ, перетворені у буквене представлення.

Висновки по розділу

У даному розділі було описано запропоновані розширення для бібліотек машинного навчання Gensim, Sklearn та бібліотеки-сховища датасетів - WFDB. Було описано розроблені модулі для зручної взаємодії з датасетами та

формулювання алгоритму. Була описана система кешування даних алгоритмів машинного навчання. Також, було описано принцип підвищення рівня абстракції бібліотек зі збереженням можливості низькорівневого налаштування алгоритмів. Було описано основні розроблені класи-розширення.

Також, було описано декілька алгоритмів із можливістю застосування розроблених компонентів, зокрема: алгоритм для рішення задачі класифікації аритмії на основі електрокардіограми, дослідження для визначення зв’язків між послідовними сигналами ЕКГ в лінгвістичному представленні та аналіз даних представлених алгоритмом TextRank, який можна застосувати до окремих записів сигналу ЕКГ.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ЕКГ

3.1 Аналіз вимог до програмного забезпечення

Після проведеного дослідження існуючих програмних засобів для аналізу сигналу ЕКГ, а також алгоритмів-класифікаторів, було сформовано вимоги, якім потребує відповідати програмне забезпечення для усування недоліків існуючих рішень. Зокрема, можна виділити наступні функціональні вимоги:

- програмне забезпечення повинно надавати розширений функціонал для обробки сигналів ЕКГ;
- програмне забезпечення повинно мати більш абстрагований інструмент для завантаження, обробки та зберігання найбільш популярних стандартизованих датасетів сигналів електрокардіограми;
- програмне забезпечення повинно забезпечувати механізм зберігання проміжних результатів та моделей машинного навчання для швидкого відтворення досягнутих етапів обробки без потреби повторного виконання алгоритму машинного навчання;
- програмне забезпечення не повинно обмежувати низькорівневий доступ до моделей та їх параметрів, для надання можливості тонкого налаштування алгоритмів розробником, який буде використовувати бібліотеку.

Також, розроблене програмного забезпечення повинно відповідати характеристикам:

- ефективність – розширення розроблені в бібліотеці не повинні сильно впливати на швидкодію основних алгоритмів машинного навчання;
- надійність – алгоритми, які закладені в бібліотеці повинні надійно виконувати поставлену задачу з мінімальною кількістю виключних ситуацій;

– підтримуваність – бібліотеку можна легко підтримувати. Вихід нових мінорних версій базових бібліотек алгоритмів машинного навчання не повинні потребувати внесення значних змін в реалізацію створеної бібліотеки;

– інтегрованість – бібліотеки можна легко застосувати у вже розробленому програмному забезпеченні;

– простота та самодокументованість – для застосування елементів бібліотеки немає необхідності вивчати додатково значну документацію. Назви методів повинні прозоро відображати їх функціонал;

– компактність – бібліотека повинна бути легкою та не потребувати значних ресурсів жорсткого диску під час встановлення;

– читабельність – код розроблений за допомогою бібліотеки повинен легко розумітися.

3.2 Архітектура програмного забезпечення

Для реалізації поставлених функціональних та характеристичних вимог до програмного забезпечення, було створено відповідну архітектуру яка зображена на рисунку 3.1.

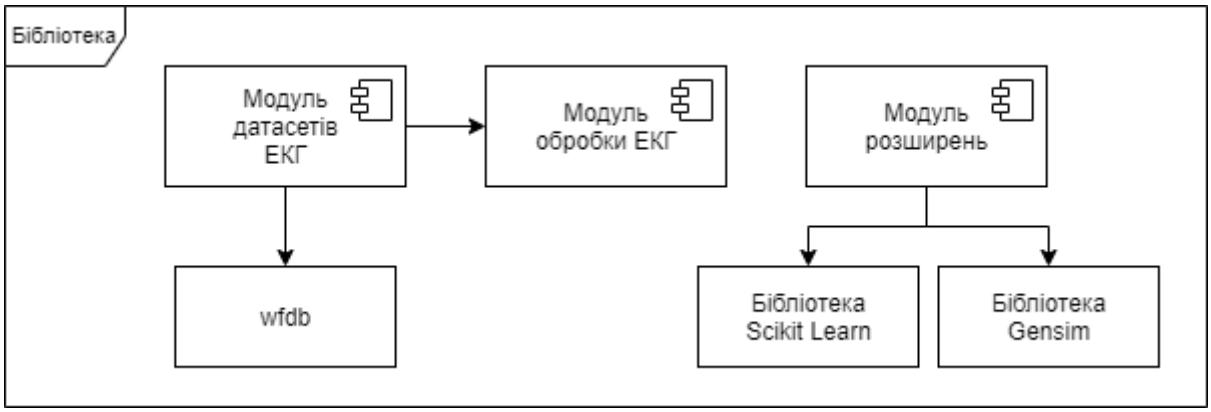


Рисунок 3.1 – Схема компонентів розробленої бібліотеки

Розроблена бібліотека складається з декількох модулів:

– модуль для завантаження та підготовки стандартизованих датасетів ЕКГ;

- модуль для стандартних операцій обробки сигналу електрокардіограми;
- модуль розширень для бібліотек машинного навчання.

3.2.1 Модуль наборів даних

Даний модуль надає можливості для завантаження та підготовки стандартних датасетів сигналу електрокардіограми. Зокрема надається доступ до двох найбільш популярних датасетів для рішення задач пов’язаних з аритмією сигналу електрокардіограми: MIT-BIH Arrhythmia Database та MIT-BIH Atrial Fibrillation Database.

Модуль використовує бібліотеку wfdb [3] для отримання доступу до репозиторію даних датасетів.

Модуль автоматично кешує дані отримані з репозиторію, та надає можливість перезавантаження даних у випадку, якщо цього потребує розробник.

Модуль містить два основних класи, схема яких приведена на рисунку 3.2.

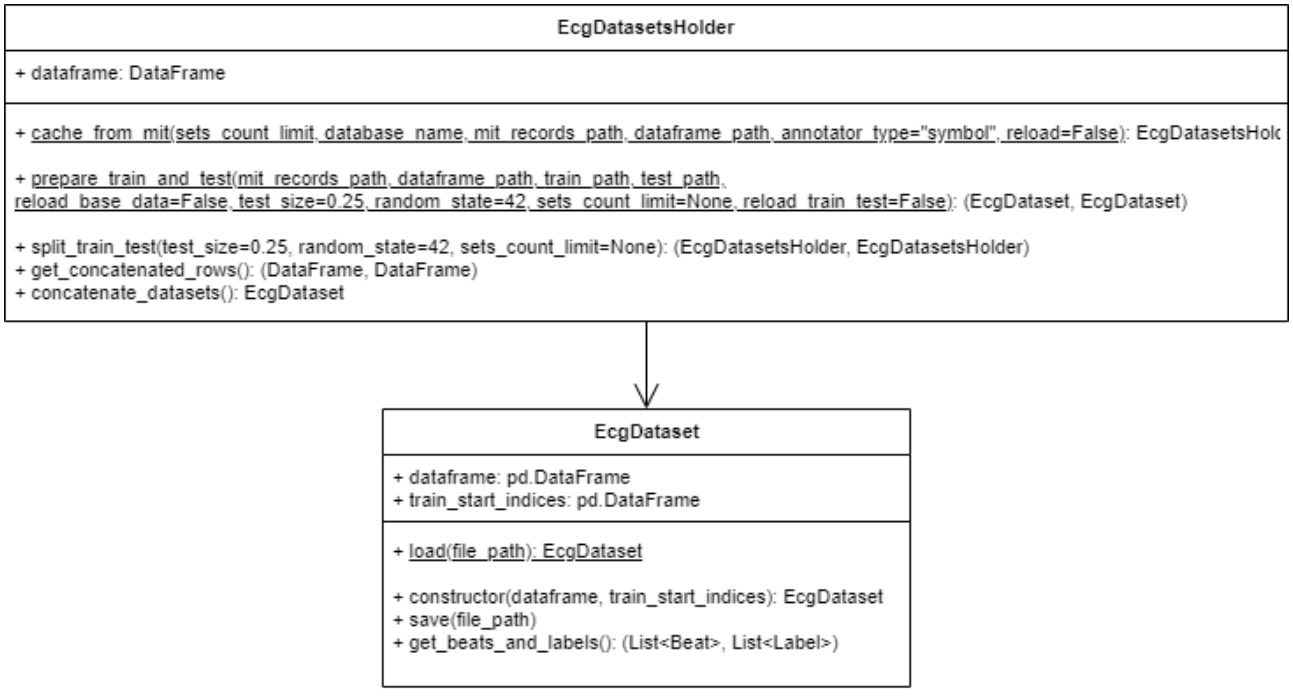


Рисунок 3.2 – UML-схема класів модулю датасетів

EcgDatasetsHolder представляє собою клас, який відповідає за завантаження набору датасетів з репозиторіїв та формування тренувальних та

валідаційних датасетів з можливістю обмеження вибірки. В середині об’єкти даного класу зберігають набір декількох датасетів, які потім об’єднуються в один набір, який зберігає EcgDataset.

EcgDataset представляє собою набір даних, які можна вже використовувати для тренування та валідації алгоритмів машинного навчання.

Специфікація функцій класу EcgDataset наведена в таблиці 3.1.

Таблиця 3.1 – Специфікація функцій класу EcgDataset

Назва	Вхідні дані	Вихідні дані	Призначення
load	file_path – шлях до файлу	EcgDataset	Завантаження датасету з файла
constructor	dataframe – датафрейм, який зберігає дані датасету; train_start_indices – початкові індекси речень	EcgDataset	Конструювання об’єкту
save	file_path – шлях до файлу		Зберегти датасет в файлі
get_beats_and_labels		(List<Beat>, List<Label>)	Отримати список серцебиттів та список позначень аритмії

Специфікація функцій класу EcgDatasetsHolder наведена у таблиці 3.2

Таблиця 3.2 – Специфікація функцій класу EcgDatasetsHolder

Назва	Вхідні дані	Вихідні дані	Призначення
cache_from_mit	sets_count_limit – лімітування кількості датасетів; database_name – назва бази даних wfdb; mit_records_path – шлях для збереження записів бази wfdb; dataframe_path – шлях для збереження відформатованого датафрейму; annotator_type="symbol" – тип анотації, яка використовується для позначення аритмії; reload=False – позначення, чи потрібно перезавантажити дані	EcgDatasetsHolder	Завантаження множини датасетів з бази даних wfdb
split_train_test	test_size=0.25 – розмір валідаційної вибірки; random_state=42 – сід для механізму рандомізації; sets_count_limit=None – лімітування базової вибірки	(EcgDatasetsHolder, EcgDatasetsHolder) – множини тренувальних та тестових датасетів	Розбити множину датасетів wfdb на тестову та тренувальну вибірки

Продовження таблиці 3.2

Назва	Вхідні дані	Вихідні дані	Призначення
prepare_train_and_test	mit_records_path – шлях для збереження записів бази wfdb; dataframe_path – шлях для збереження відформатованого датафрейму; train_path – шлях для збереження тренувальної вибірки; test_path – шлях для збереження тестової вибірки; reload_base_data=False – позначка, чи потрібно перезавантажити базовий датасет; test_size=0.25 – розмір тестовою вибірки; random_state=42 – сід для механізму рандомізації; sets_count_limit=None – лімітування базової вибірки; reload_train_test=False – позначка, чи потрібно переформувати тренувальну та валідаційну вибірки	(EcgDataset, EcgDataset) – тренувальний та валідаційний датасети	Завантажити датасети з бази даних wfdb та сформувати тренувальну та валідаційну вибірки

Продовження таблиці 3.2

Назва	Вхідні дані	Вихідні дані	Призначення
get_concatenated_rows		(DataFrame, DataFrame) – датафрейм сигналів ЕКГ та датафрейм індексів, які позначають межі датасетів	Об’єднати множину датасетів в один датафрейм
concatenate_datasets		EcgDataset – єдиний датасет сигналу ЕКГ	Об’єднати множину датасетів в один EcgDataset

3.2.2 Модуль розширень для бібліотек машинного навчання

Даний модуль надає розширення для алгоритмів машинного навчання, які надаються бібліотеками Scikit Learn [2] та Gensim [1]. Розширення реалізован таким чином, щоб не обмежувати свободу розробника у питаннях параметризації та тонкого налаштування алгоритмів.

Розширення надають більш абстрактний функціонал, який скорочує кількість коду, яку потрібно написати розробником для досягання певного результату, тим самим надаючи йому змогу зосередити свою увагу на

конкретних задачах машинного навчання, а не на дрібних діях по підготовці даних.

Зокрема надаються розширення для наступних алгоритмів машинного навчання: Word2Vec та KMeans [7].

UML-схема класу-розширення Word2Vec наведена на рисунку 3.3.

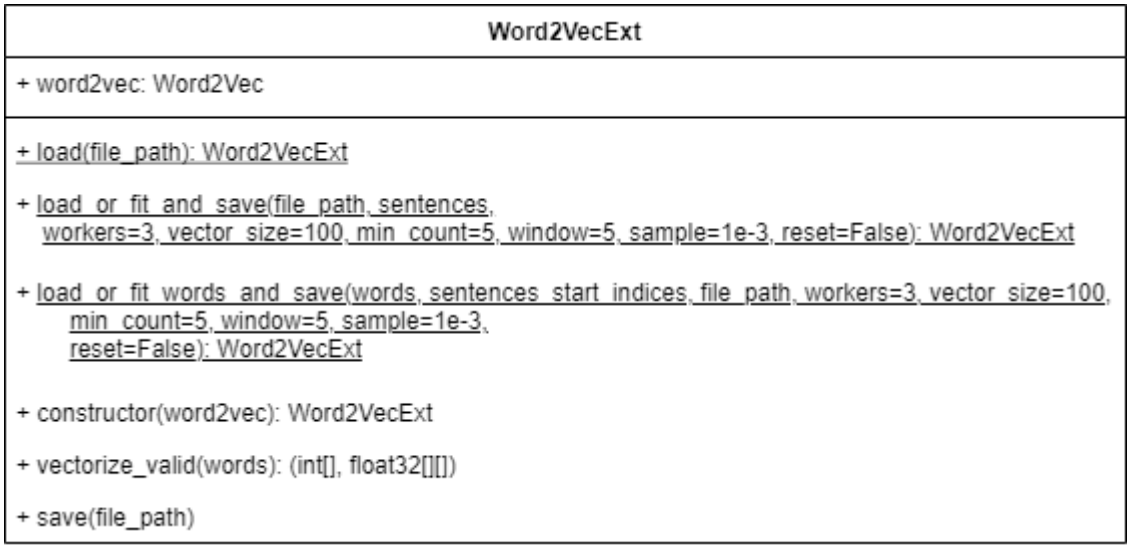


Рисунок 3.3 – UML-схема класу-розширення Word2VecExt

Клас-розширення Word2VecExt надає інтерфейс для швидкої реалізації навчання моделі та її застосування. Також, дане розширення підтримує механізм кешування.

Специфікація методів класу Word2VecExt наведена у таблиці 3.3

Таблиця 3.3 – Специфікація методів класу Word2VecExt

Назва	Вхідні дані	Вихідні дані	Призначення
load	file_path – шлях до файлу	Word2VecExt	Завантажити модель з файлу

Продовження таблиці 3.3

Назва	Вхідні дані	Вихідні дані	Призначення
load_or_fit_words_and_save	words – масив слів; sentences_start_indices – масив індексів, які позначають межу реченб; file_path – шлях до файлу; workers=3 – кількість обробників; vector_size=100 – розмір вектору; min_count=5 – мінімальна кількість повторень слів, для додання в словник; window=5 – кількість слів навколо поточного слова для обробки; sample=1e-3 – параметр навчання; reset=False – перестворити модель	Word2VecExt	Завантажити дані моделі з файлу, якщо він є. Якщо файл відсутній, то виконати навчання та зберегти модель у файл. Навчання моделі виконуються словами, межі речень між якими, вказані у параметрах
constructor	word2vec – модель Word2Vec з модуля Gensim	Word2VecExt	Конструктор класу

Продовження таблиці 3.3

Назва	Вхідні дані	Вихідні дані	Призначення
vectorize_valid	words – слова, які потрібно перетворити у векторизований вигляд	(int[], float32[][]) – масив індексів слів, які вдалося перетворити та самі векторизовані слова	Векторизовати слова, які присутні в словнику моделі
save	file_path – шлях до файлу		Зберегти модель у файл

3.2.3 Модуль обробки сигналів електрокардіограми

Даний модуль надає ряд стандартних операцій, які зазвичай потребуються під час роботи з датасетом сигналу електрокардіограми.

Також, в даному модулі присутній клас, який реалізує перетворення сигналів електрокардіограми у лінгвістичне представлення для наступної обробки засобами NLP. Даний клас називається Beats2Words. Він реалізован на основі алгоритму кластеризації K-Means. Інтерфейсні методи класу наведені на рисунку 3.4.

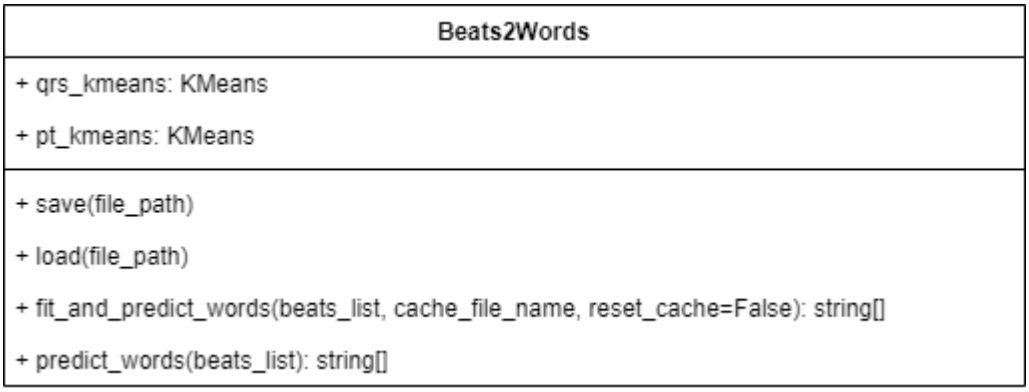


Рисунок 3.4 – UML-схема класу Bears2Words

Специфікація методів класу Bears2Words наведена у таблиці 3.4

Таблиця 3.4 – Специфікація методів класу Word2VecExt

Назва	Вхідні дані	Вихідні дані	Призначення
save	file_path – шлях до файлу		Зберегти модель у файл
load	file_path – шлях до файлу	Beats2Words	Завантажити модель з файла
fit_and_predict_words	beats_list – сигнали електрокардіограми; cache_file_name – шлях до файлу; reset_cache=False – перестворити модель	string[] – лінгвістичне представлення сигналів ЕКГ	Навчити модель та визначити лінгвістичне представлення сигналів ЕКГ. Зберегти модель у файл. Якщо файл існує, то завантажити з нього модель

Продовження таблиці 3.4

Назва	Вхідні дані	Вихідні дані	Призначення
predict_words	beats_list – список сигналів електрокардіограми	string[] – лінгвістичне представлення сигналів ЕКГ	Визначити лінгвістичне представлення сигналів ЕКГ

3.3 Схема інтеграції бібліотеки

Програмне забезпечення представляє собою бібліотеку на мові Python, яку можна легко інтегрувати у вже створені розробки та легко застосувати для створення нового проекту.

Схема інтеграції бібліотеки наведена на рисунку 3.5.

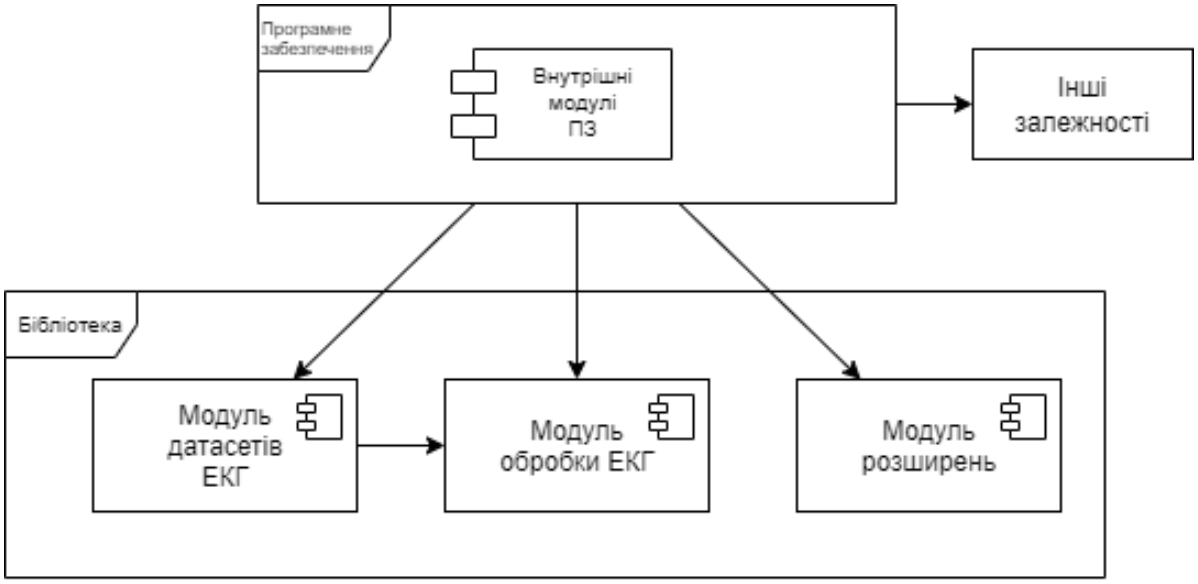


Рисунок 3.5 – Схема інтеграції бібліотеки в ПЗ

За рахунок модульності бібліотеки, розробник може вирішувати, яку частину функціоналу він хоче використовувати спираючись на конкретний модуль для рішення власних задач. Модулі надають доступ до стандартизованих типів даних, які можна отримати з класів-розширень та застосувати в інших

бібліотеках, таким чином дана розробка може бути легко інтегрована з іншими, які також підтримують стандартизоване представлення даних.

Висновки до розділу

В даному розділі дана характеристика розробленого програмного забезпечення, було наведено опис внутрішньої архітектури бібліотеки з вказання основного функціоналу та можливостей, які надаються даним продуктом. Наведена схема інтеграції розробленого ПЗ в проекти на мові Python, а також описана можливість легкої взаємодії внутрішніх модулів бібліотеки зі стороннім програмним забезпеченням, яке може використовувати розробник. Підтверджено ключові властивості програмного забезпечення, вимоги до якого було висунуто.

4 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ЗАСТОСУВАННЯ ПРОГРАМНОГО ТА МАТЕМАТИЧНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Мета та порядок досліджень

Для дослідження ефективності, точності та лаконічності розробленого програмного та математичного забезпечення, необхідно провести ряд експериментів:

- порівняння компактності коду програми при використанні бібліотеки, та без неї при вирішенні задач: підготовки датасету електрокардіограми, розбиття датасету на тренувальну та валідаційну вибірки, кешування даних проміжних етапів, загальна задача по класифікації аритмії на основі сигналу ЕКГ;
- розрахунок швидкодії алгоритму машинного навчання при використанні лінгвістичного представлення сигналу електрокардіограми;
- визначення точності кластеризованого представлення сигналу електрокардіограми при різних розмірах кластерів;
- застосування бібліотеки для задачі визначення наявності зв’язку між послідовними сигналами ЕКГ у лінгвістичному представленні;
- застосування бібліотеки для аналізу даних представлених методом TextRank при застосуванні до лінгвістичного представлення сигналу ЕКГ.

Для проведення досліджень використовувалося апаратне забезпечення з наступними характеристиками:

- Intel Core i5-6200U CPU 2.3-2.4 GHz;
- 12 GB RAM;
- Samsung 870 Evo-Series 1TB SATA III;
- AMD Radeon R5 M330.

Експерименти проводилися на операційній системі Windows 10 Корпоративна 2016.

4.2 Порівняння компактності коду шляхом підрахунку слів та символів

Для порівняння компактності коду будемо використовувати підрахунок кількості слів та символів для написання одного і того ж функціонально блоку з використанням розробленої бібліотеки, та без неї.

В таблиці 4.1 наведено порівняння кількості слів та символів при рішенні задачі підготовки датасету сигналу електрокардіограми та розбиття його на тренувальну та валідаційні вибірки. В таблиці 4.2 вказано порівняння реалізацій кешування даних.

Таблиця 4.1 – Підрахунок кількості слів та символів для рішення задачі підготовки датасету

Використане рішення	Кількість слів для рішення	Кількість символів для рішення
З використанням бібліотеки	37	545
Без використання бібліотеки	813	5577

Приклад застосування бібліотеки для підготовки датасету:

```
ecgdataset = ecgdatasetsholder.EcgDatasetsHolder.cache_from_mit(
    sets_count_limit=5,
    database_name="mitdb",
    mit_records_path=rsc_dir + "/mit_records",
    dataframe_path=rsc_dir + "/ecgdataset",
    annotator_type="symbol",
    reload=False
)

train, test = ecgdataset.split_train_test(test_size=0.25, random_state=42)

train_ready = train.concatenate_datasets()
test_ready = test.concatenate_datasets()
```

Таблиця 4.2 – Підрахунок кількості слів та символів для рішення задач кешування

Задача кешування, або навчання та кешування	З використанням бібліотеки		Без використання бібліотеки	
	Слів	Символів	Слів	Символів
Word2vec	37	336	169	1117
KMeans	20	237	121	736
WFDB Dataset	19	199	609	3932

Приклад застосування бібліотеки для задачі навчання та кешування Word2Vec:

```
num_features = 300
word2vecExtModel = \
    word2vecext.Word2VecExt.load_or_fit_words_and_save(train_words,

train_ready.train_start_indices["start_indices"].tolist(),
                                                       rsc_dir + "/word2vec",
                                                       vector_size=num_features,
                                                       reset=False)
train_data = word2vecExtModel.vectorize_valid_with_labels(train_words,
train_ready.dataframe["labels"].tolist())

validation_data = word2vecExtModel.vectorize_valid_with_labels(validation_words,
test_ready.dataframe["labels"].tolist())

(train_x, train_y), (validation_x, validation_y) = (train_data, validation_data)
```

4.3 Дослідження ефективності математичного забезпечення

4.3.1 Опис датасету

Для проведення дослідження математичного забезпечення було використано датасет MIT-BIH Arrhythmia Database, який містить сорок вісім півгодинних записів сигналу електрокардіограми амбулаторних пацієнтів. Даний датасет містить також позначки класу аритмії до кожного окремого

серцебиття. Загалом нараховується 18 класів різних видів аритмії, які були визначені в межах датасету.

4.3.2 Дослідження точності кластеризованого представлення ЕКГ сигналу

Формування кластеризованого представлення полягає у формуванні QRS комплексів з окремих серцебиттів сигналу електрокардіограми з наступною кластеризацією сформованих частин за допомогою двох моделей K-середніх.

Перша модель відповідає за кластеризацію інтервалів R-піків серцебиттів – найвищих точок в сигналі ЕКГ.

Друга модель відповідає за кластеризацію інтервалів PR, які знаходяться до відповідних інтервалів R-піку, та інтервалів ST, які знаходяться після відповідних інтервалів R-піку.

Таким чином весь сигнал ЕКГ підпадає під кластеризацію.

Після кластеризації, замінюємо відповідні сигнальні проміжки на індекси кластерів. В результаті отримаємо послідовність індексів, де кожні три індекси представляють собою одне серцебиття. Перший індекс – інтервал PR, другий індекс – інтервал R-піку, третій індекс – інтервал ST.

Для оберненого перетворення кластеризованої послідовності, необхідно замінити кожен індекс на значення центру відповідного кластеру. Таким чином отримаємо приближену до оригіналу копію сигналу електрокардіограми. Точність такого перетворення цілком залежить від кількості кластерів, які були вказані як параметри в обох моделях K-середніх.

Приклад оберненого перетворення наведено на рисунку 4.1. Помаранчева лінія – оригінальній сигнал, синя лінія – відтворений сигнал.

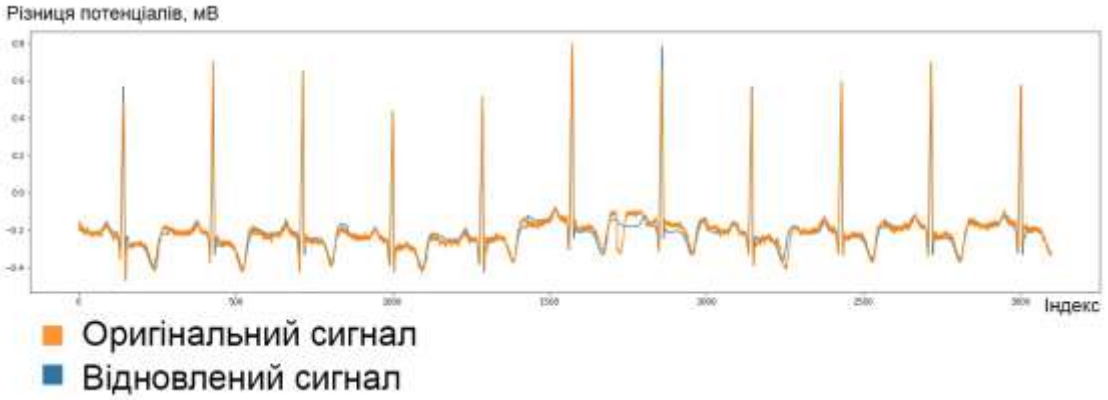


Рисунок 4.1 – Приклад відтвореного сигналу після проведення кластеризації

Для оцінки точності кластеризованого перетворення було використано середньоквадратичну похибку.

Графік зміни середньоквадратичної похибки в залежності від загальної кількості кластерів в обох моделях, наведено на рисунку 4.2. Конкретні значення наведені в таблиці 4.3.

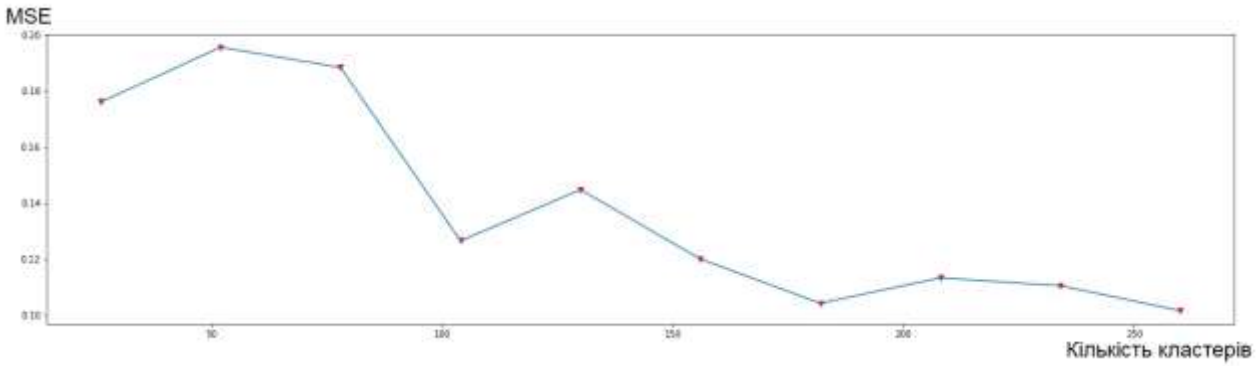


Рисунок 4.2 – Графік залежності середньоквадратичної похибки від загальної кількості кластерів

Таблиця 4.3 – Середньоквадратична похибка в залежності від кількості кластерів

Кількість кластерів у першій моделі	Кількість кластерів у другій моделі	Середньоквадратична похибка
6	20	0.173

Продовження таблиці 4.3

Кількість кластерів у першій моделі	Кількість кластерів у другій моделі	Середньоквадратична похибка
12	40	0.192
18	60	0.19
24	80	0.124
30	100	0.143
36	120	0.12
42	140	0.105
48	160	0.112
54	180	0.111
60	200	0.101

Як видно з графіку, похибка, зі збільшення кількості кластерів, зменшується. Так при використанні 60 кластерів для першої моделі та 200 для другої моделі, похибка становить 10%.

Графічне порівняння сигналу з найбільшою точністю наведено на рисунку 4.3.

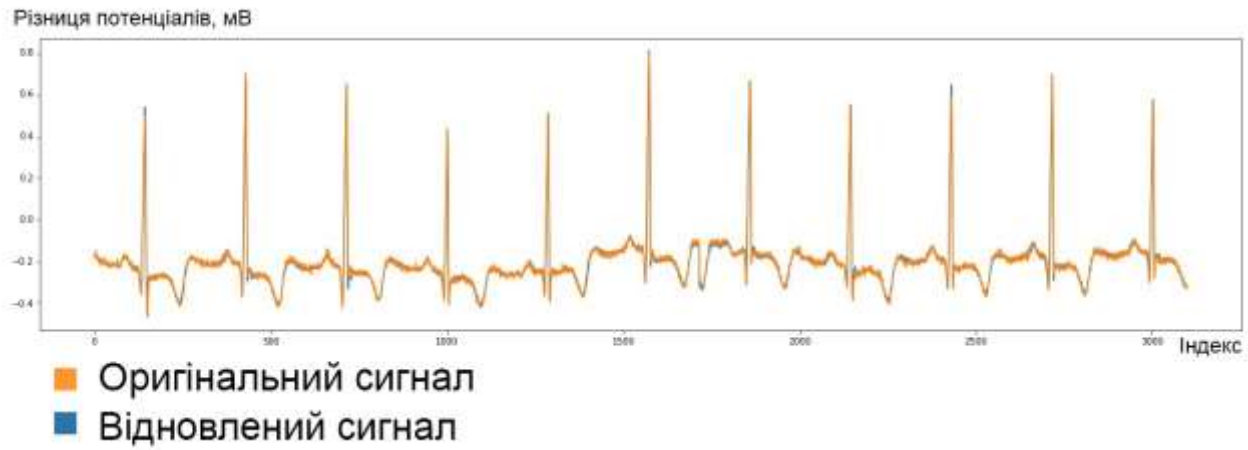


Рисунок 4.3 – Порівняння оригінального сигналу та сигналу відтвореного після кластеризації з використанням моделей на 60 та 200 кластерів

4.3.3 Порівняння швидкодії алгоритмів класифікації

При використання лінгвістичного представлення електрокардіограми з наступною векторизацією за допомогою Word2Vec, зменшується об’єм вхідних даних для алгоритмів машинного навчання, пришвидшуючи тим самим швидкість навчання моделей-класифікаторів. Розроблений алгоритм спроможний зменшити кількість вхідних даних в рази, шляхом початкової кластеризації, а потім векторизації за допомогою моделі Word2Vec.

Для визначення покращення швидкості, було проведено ряд експериментів з замірянням часу витраченого на навчання алгоритмів-класифікаторів. Результати наведені в таблиці 4.4.

Таблиця 4.4 – Швидкість навчання моделей при використанні та без використання моделі Word2Vec

Алгоритм	Початковий розмір датасету (тисяч серцебиттів)	Тривалість навчання (секунд)		F-міра	
		Без Word2Vec	З використанням Word2Vec	Без Word2Vec	З використанням Word2Vec
Random Forest	10.5	33,2447	27,796	0.95	0.96
Gradient Boosting		32,1669	27,877	0.96	0.98
Random Forest	13.2	495,244	45,063	0.95	0.97
Gradient Boosting		479,433	40,708	0.93	0.97

Як видно з таблиці, швидкість навчання дійсно зростає, при використанні векторизованого за допомогою моделі Word2Vec текстового представлення сигналу електрокардіограми. Особливо значні зміни помітні при збільшенні датасету.

4.3.4 Оцінка точності алгоритмів-класифікаторів основаних на лінгвістичних даних

За допомогою розробленого програмного забезпечення також можна оцінити точність алгоритмів-класифікаторів.

Для оцінки точності була використана метрика F-міри, яка оцінюється за допомогою влучності та повноти тесту. Влучність – кількість вірно визначених позитивних результатів (TT – істинно позитивні), розділена на кількість всіх оцінок – як правильно визначених так і неправильно (TT + FT, тобто істинно позитивні + хибно позитивні). Повнота – кількість правильно оцінених позитивних результатів (TT – істинно позитивні), розділена на загальну кількість випадків, які повинні були бути визначені як позитивні (TT + FF, тобто істинно позитивні + хибно негативні). Середнє гармонійне цих двох значень (повноти та влучності) і буде визначати F-міру.

Після проведення експериментів на датасетів розміром 10.5 тисяч серцебиттів, було отримано результати, які наведені в таблиці 4.5.

Таблиця 4.5 – Точність класифікації серцебиттів

Алгоритм	F-міра
Random Forest	96%
Gradient Boosting	98%

4.3.5 Застосування бібліотеки для дослідження наявності зв’язку між послідовними сигналами ЕКГ в лінгвістичному представленні

Розроблену бібліотеку також можна застосувати для проведення наукових досліджень у сфері електрокардіограм. Зокрема, можна перетворити сигнали електрокардіограми в лінгвістичне представлення, та провести дослідження з метою виявлення зв’язків між послідовними сигналами в такому виді за допомогою моделі Word2Vec із застосуванням алгоритму Skip Gram, який і надає можливість передбачення наступного слова на основі попереднього.

Таке дослідження може визначити наступні пріоритетні методи застосування лінгвістичного представлення сигналу електрокардіограми.

Розроблений код, при використанні бібліотеки значно скорочується, таким чином отримаємо наступне рішення:

```
def main():
    rsc_dir = "../example_rsc"

    ecgdataset = ecgdatasetsholder.EcgDatasetsHolder.cache_from_mit(
        sets_count_limit=10,
        database_name="mitdb",
        mit_records_path=rsc_dir + "/mit_records",
        dataframe_path=rsc_dir + "/ecgdataset_of_10_subsets",
        annotator_type="symbol",
        reload=False
    )

    train_ready = ecgdataset.concatenate_datasets()

    train_ready.save(rsc_dir + "/full_dataset")

    beats2wordsModel = beats2words.Beats2Words()
    train_words =
beats2wordsModel.fit_and_predict_words(train_ready.dataframe["beats"].tolist(),
                                                        rsc_dir +
"/beats2words_test_for_10_datasets",
                                                        reset_cache=True)

    num_features = 300
    word2vecExtModel = \
        word2vecext.Word2VecExt.load_or_fit_words_and_save(train_words,

train_ready.train_start_indices["start_indices"].tolist(),
                                                           rsc_dir + "/word2vec2",
                                                           vector_size=num_features,
                                                           sg=1,
```

```
                                                    sample=1e-3,
                                                    window=10,
                                                    alpha=0.01,
                                                    reset=True)

    calculate_word2vec_accuracy(word2vecExtModel.word2vec, train_words,
train_ready.train_start_indices["start_indices"].tolist())
```

Скорочена кількість дій, які необхідно виконати для підготовки даних, надає можливість зосередитися для самому дослідженні.

Код дослідження точності визначення наступного слова:

```
def calculate_word2vec_accuracy(c_word2vec: word2vecext.Word2Vec, c_validation_words,
validation_sentences_indices):
    true_cases = 0
    false_cases = 0
    index2word_set = set(c_word2vec.wv.index_to_key)

    for i in range(len(validation_sentences_indices)):
        if i == len(validation_sentences_indices) - 1:
            end_index = len(c_validation_words)
        else:
            end_index = validation_sentences_indices[i + 1]
        start_index = validation_sentences_indices[i]
        for j in range(start_index, end_index - 1):  # end_index - 1 to compare with
next word
            # Iterate over each word in sentences
            if not (c_validation_words[j] in index2word_set and c_validation_words[j +
1] in index2word_set):
                # Skip words pairs which are not in the dictionary
                continue
            word_to_vec_list =
c_word2vec.wv.most_similar(positive=[c_validation_words[j]])
            similar_words = map(lambda it: it[0], word_to_vec_list)

            if c_validation_words[j + 1] in similar_words:
                true_cases = true_cases + 1
            else:
                false_cases = false_cases + 1

    print("True_cases =", true_cases)
    print("False_cases =", false_cases)
    print("True_cases/total_cases:", true_cases / (true_cases + false_cases))
```

Результати проведеного дослідження представленні в таблиці 4.6

Таблиця 4.6 – Результати точності визначення наступного сигналу ЕКГ в лінгвістичному представленні за допомогою моделі Word2Vec

Кількість сигналів в датасеті	Кількість передбачених сигналів	Кількість не передбачених сигналів	Відношення передбачених до кількості оброблених
10600	4835	4611	0.5118568
21595	7743	13087	0.371723

Згідно наведеним результатам, у датасеті середнього розміру (на 11 тисяч сигналів) визначено 51% випадків, у яких наступне слово було в десятці передбачених за допомогою моделі Word2Vec. При збільшенні датасету збільшилася і кількість крайових випадків, які сильно вплинули на кінцевий результат, але відсоток у розмірі 37 теж може бути інтерпретований як отримання цікавих результатів.

Таким чином, можна визначити, що модель word2vec досить непогано себе показує при рішенні таких нестандартних задач, де вхідними даними є коротке лінгвістичне представлення сигналу ЕКГ. Також, можна сказати, що між сигналами у такому вигляді, згідно дослідження, присутні явні зв’язки, які можуть бути використані у подальших експериментах.

4.3.6 Застосування бібліотеки для дослідження ЕКГ методом TextRank

Засоби розроблені в бібліотеці, надають можливість простого отримання лінгвістичного представлення сигналу ЕКГ, тим самим надаючи можливість застосування методів, які орієнтовані на обробку текстових даних. Одним з таких методів є TextRank.

Код, для застосування методу виглядає наступним чином:

```
def main():
    rsc_dir = "../example_rsc"

    ecgdataset = ecgdatasetsholder.EcgDatasetsHolder.cache_from_mit(
        sets_count_limit=5,
        database_name="mitdb",
        mit_records_path=rsc_dir + "/mit_records",
        dataframe_path=rsc_dir + "/ecgdataset",
        annotator_type="symbol",
        reload=False
    )

    train_ready = ecgdataset.concatenate_datasets()

    train_ready.save(rsc_dir + "/full_dataset")

    beats2wordsModel = beats2words.Beats2Words()
    train_words =
beats2wordsModel.fit_and_predict_words(train_ready.dataframe["beats"].tolist(),
                                                        rsc_dir +
"/beats2words_test_for_5_datasets",
                                                        reset_cache=True)

    text_rank_test(train_words,
train_ready.train_start_indices["start_indices"].tolist(),
train_ready.dataframe["labels"].tolist())
```

Сам аналіз методом TextRank виглядає наступним чином:

```
def text_rank_test(c_validation_words, validation_sentences_indices, labels):
    for i in range(len(validation_sentences_indices)):
        if i == len(validation_sentences_indices) - 1:
            end_index = len(c_validation_words)
        else:
            end_index = validation_sentences_indices[i + 1]
        start_index = validation_sentences_indices[i]

        current_sentence_words = c_validation_words[start_index:end_index]
        current_sentence_labels = labels[start_index:end_index]
        current_sentence_labels_count = dict(Counter(current_sentence_labels))
        print("Text labels count: ", current_sentence_labels_count)

        current_sentence = '\n'.join(current_sentence_words)
        nlp = spacy.load("en_core_web_sm")
        nlp.add_pipe("textrank")
        doc = nlp(current_sentence)
        print("Text summary: ")
        phrase = doc._.phrases[0]
        word = phrase.text
        labels_of_word = [current_sentence_labels[i] for i, x in
enumerate(current_sentence_words) if x == word]
        estimated_labels_count = dict(Counter(labels_of_word))
        print("Word:", word, "; Related labels:", estimated_labels_count)
        print(phrase.rank, phrase.count)
```

Таким чином, використовуючи засоби бібліотеки, ми вкотре змогли зосередити увагу виключно на цілях дослідження.

Після запуску даного коду для датасету, який складається з 5 записів з загальною кількістю окремих серцебиттів – 11 тисяч, було отримано очікуваний результат – алгоритм TextRank виділив в кожному з текстів (записів) речення (окремі серцебиття) які найбільш притаманні даному датасету.

Результату для одного з записів:

Text labels count: {0: 2230, 1: 34}

Text summary:

Word: luj; Related labels: {0: 371, 1: 3}

{0: 2230, 1: 34} – це опис тексту (запису), розрахований до початку застосування методу TextRank. Цей опис вказує на те, що в записі присутні 2230 серцебиття без ознак аритмії та 34 серцебиття з наявною аритмією.

Перше речення (сигнал), який TextRank оцінює, як найкраще скорочення для тексту це слово «*luj*». Дане слово представляє 374 різних серцебиттів, три з яких мають ознаки аритмії.

Таким чином, TextRank обрав серцебиття, яке дійсно може описати весь датасет. Опис датасету є ні чим іншим, як відношення кількості серцебиттів з аритмією до кількості серцебиттів без ознак аритмії.

Такі ж самі результати були отримані для інших чотирьох записів.

Даний результат є очікуваним, оскільки вхідними даними алгоритму TextRank є кластеризовані значення всього датасету. Кластеризація допомогла отримати скорочені представлення для множини сигналів, а TextRank вже визначив з них ті, які частіше повторюються в межах тексту.

Висновки до розділу

Було проведено ряд експериментів для підтвердження характеристик розробленого програмного та математичного забезпечення.

Експеримент, проведений для перевірки компактності розробленої мови та її ефективного скорочення коду, показав зменшення розміру коду в більш ніж 5 разів, тим самим надаючи змогу розробнику зосередити свою увагу на більш абстрактних задачах машинного аналізу.

Експеримент аналізу точності кластеризованого представлення сигналу електрокардіограми визначив, що похибку становить 10% при використанні 200 кластерів для моделі інтервалів P-T, та 60 кластерів для моделі інтервалів піків. Було визначено збільшення швидкості навчання моделей-класифікаторів при використанні моделі Word2Vec. Розрахована швидкість зростає ще більше, при використанні більших датасетів, так, спостерігається зменшення часу навчання моделі в більш ніж 10 разів при застосування датасету на 13.2 тисячі серцебиттів.

Також, розроблену бібліотеку було застосовано для аналізу зв’язків між послідовними сигналами ЕКГ в лінгвістичному представленні. Даний експеримент допоміг вкотре оцінити ефективність написання коду із застосування бібліотеки, а також навів на наступні висновки: згідно дослідження, Word2Vec зміг встановити зв’язок між послідовними сигналами у 51% випадків для датасету на 11 тисяч сигналів та у 37% випадків для датасету на 22 тисячі сигналів.

Окремо було застосовано бібліотеку для аналізу лінгвістичного представлення датасетів за допомогою методу TextRank. Дане дослідження показало очікувані результати: метод TextRank обрав слова, які можуть описати характеристику кожного датасету окремо – відношення кількості серцебиттів без ознак аритмії до кількості серцебитті з присутньою аритмією в межах запису.

5 РОЗРОБЛЕННЯ МАРКЕТИНГОВОЇ ПРОГРАМИ СТАРТАП-ПРОЄКТУ

5.1 Опис ідеї проєкту

Таблиця 5.1 — Опис ідеї стартап-проєкту

<table>
<tr><th>Зміст ідеї</th><th>Напрямки застосування</th><th>Вигоди для користувача</th></tr>
<tr><td rowspan="2">Вихід на ринок ПЗ для інтелектуального аналізу електрокардіограм з використанням спеціалізованих баз знань.
Метою проекту є заключення партнерського договору з провідними виробниками подібних систем для отримання фінансування подальшої розробки, та впровадження розробленої системи в якості інтегрованого модуля для відповідного обладнання.</td><td>Впровадження програмного забезпечення в існуюче ПЗ для обробки ЕКГ</td><td rowspan="2">Підвищення швидкості реалізації програмного забезпечення для дослідження сигналу електрокардіограми</td></tr>
<tr><td>Застосування розробленої бібліотеки у нових проектах, спрямованих на аналіз сигналу електрокардіограм</td></tr>
</table>

Таблиця 5.2 — Опис ідеї стартап-проєкту

№	Техніко-економічні характери-тики ідеї	Продукція конкурентів		W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проєкт	Heart Rate Variability Analysis Software			
1	Обробка ЕКГ	Наявний	Наявний		N	
2	Інструменти для ефективного ведення досліджень	Наявний	Відсутній			S
3	Система кешування етапів	Наявний	Відсутній			S
4	Інструменти для аналізу лінгвістичного представлення ЕКГ	Наявний	Відсутній			S

5.2 Технологічний аудит ідеї проєкту

Таблиця 5.3 — Технологічна здійсненність ідеї проєкту

№	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Методи підготовки датасетів	Мова Python, бібіліотека WFDB	Наявна	Доступна
2	Метод переведення ЕКГ у лінгвістичний вигляд	Мова Python, бібліотека Sklearn	Наявна	Доступна
3	Система автоматичного кешування моделей	Мова Python, бібліотека Sklearn, Gensim	Наявна	Доступна
4	Розширення функціоналу моделей для аналізу ЕКГ	Мова Python, бібліотека Sklearn, Gensim	Наявна	Доступна
Обрана технологія реалізації ідеї проєкту: Python та популярні бібліотеки Gensim, sklearn, які є кращими серед наявних на ринку та доступними для членів команди.				

Висновок: технологічна реалізація продукту – можлива, вибрана технологія №1

5.3 Аналіз ринкових можливостей запуску стартап-проєкту

Таблиця 5.4 — Попередня характеристика потенційного ринку

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	1
2	Загальний обсяг продаж, грн./ум.од	Невідома
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Юридичні обмеження
5	Специфічні вимоги до стандартизації та сертифікації	Вимоги до роботи інформаційної системи у медичних закладах
6	Середня норма рентабельності в галузі або по ринку, %	Невідома

Висновок: враховуючи кількість головних гравців по ринку, зростаючу динаміку ринку, невелику кількість конкурентів та середню норму рентабельності можна зробити висновок, що на даний момент, ринок для входження стартап-продукту є привабливим.

Таблиця 5.5 — Характеристика потенційних клієнтів стартап-проєкту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці цільових груп клієнтів	Вимоги споживачів до товару
1	Швидка розробка програмного забезпечення для аналізу ЕКГ	Розробники ПЗ для аналізу ЕКГ	Відсутні	Можливість скорочення коду ПЗ для пришвидшення реалізації
2	Зменшення порогу входження в застосування інструментів аналізу ЕКГ	Вчені та дослідники	Відсутні	Можливість отримати результати із мінімальними витратами часу на дослідження концепцій бібліотеки

Таблиця 5.6 — Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренти	Наявність конкурентів котрі надають схожі рішення	Знизити ціну на надані послуги; Розробити унікальні характеристики продукту; Видача ліцензії на обслуговування

Продовження таблиці 5.6

№	Фактор	Зміст загрози	Можлива реакція компанії
2	Кошти на розробку та підтримку продукту	Закінчення грошей та недостатнє фінансування	Залучення більше інвесторів та заохочення майбутньою розробкою; Ітеративна розробка продукту для поетапного випуску продукту та відповіді користувача
3	Вихід аналогу	Вихід аналогу даного товару може призвести до знецінення та безідейності даного товару	Вихід на ринок за короткий час, але з неповним функціоналом, проте достатнім, щоб викликати інтерес у всіх цільових аудиторій; Проводити рекламні кампанії

Таблиця 5.7. — Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Новий продукт	Вихід на ринок, Зменшення монополії, Надання нових рішень у сфері	Розробка нових функцій; Доступ до ринку для нових продуктів; Відповідно до потреб користувачів/замовників надаються різні види ліцензій.

Продовження таблиці 5.7

№	Фактор	Зміст можливості	Можлива реакція компанії
2	Вихід аналогу	Надати продукт з певними характеристиками та можливостями що відсутні у компаній конкурентів	Ринок і користувачі аналізуються, щоб задовольнити їхні потреби та надати функції за нижчою ціною, ніж альтернативні продукти, у найкоротші терміни.
3	Зворотній зв’язок від користувачів	Можливість отримання необхідної інформації для вдосконалення продукту	Наявність вхідних даних і реакція на них команди розробників для задоволення потреб і побажань кінцевих користувачів бібліотеки.
4	Грошова винагорода за отримання доступу до програмного забезпечення	При достатньому попиту на систему можлива комерціалізація продукту на основі продажу екземплярів ПЗ задля отримання грошової винагороди для подальшого розвиту продукту та оплати заробітної плати працівникам	Точкова комерціалізація продукту; Залучення додаткових коштів для подальшого розвитку проекту.

Таблиця 5.8 — Ступеневий аналіз конкуренції на ринку

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1	Тип конкуренції: монополістична	Товар від кожної компанії на ринку, являється недосконалим замінником товару, реалізованого іншими фірмами; На ринку є умови для входу та виходу; Ціна корелює між суперниками;	Розробка продукту з функціями, які охоплюють сферу використання, яку не включають інші альтернативні продукти; Регулювання ціни відповідно до замінників; Різні види ліцензій.
2	Рівень конкурентної боротьби: світовий	Всі продукти замінники розроблялись інтернаціональними командами з різних куточків світу, продукти не належать до певної держави, а належать команді розробників	Вийти на ринок продуктів з функціями, які потрібні клієнтам; Налагодити маркетинг на основних інтернет-ресурсах, щоб охопити велику кількість потенційних користувачів; Надання бета-версію продукту.

Продовження таблиці 5.8

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
3	Галузева ознака: внутрішньогалузева	Даний тип продукту може використовуватися тільки у сфері розробки IT додатків \ продуктів	Забезпечити зручний та інтуїтивно зрозумілий інтерфейс; Підтримка відомих методів взаємодії із середовищем розробки; Надання документації та онлайн-підтримки.
4	Конкуренція за видами товарів: товарно-видова	Дана конкуренція – конкуренція між товарами одного виду.	Введення функцій, яких не вистачає в альтернативних продуктах; Спрощення інтерфейсу; надання підтримки.
5	Характер конкурентних переваг: цінова та не цінова	Цінові переваги – точкова комерціалізація; Не цінова – надання функціональності, що відсутня у товарах-замінниках.	Надавати платні ліцензії на ключові функції лише клієнтам із певним періодом підтримки, який обумовлений у відповідних ліцензіях; Введення унікальних функцій.

Продовження таблиці 5.8

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
6	За інтенсивністю: марочна	Наявність унікального знаку що відрізняє даний продукт від продуктів-замінників	Введення власної назви та власного товарного знаку.

Таблиця 5.9 — Аналіз конкуренції в галузі за М. Портером

Склад о вий аналіз у	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	HRV software	Невідомі	-	Контроль якості продукту	Наявність більш широкого функціоналу , зручнішого інтерфейсу та авторитет (якість), наявність необхідних сертифікацій

Продовження таблиці 5.9

Прямі конкуренти в галузі		Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Висновки	Досить інтенсивна конкурентна боротьба з вже закріпленими на ринку гравцями	Дані відсутні	-	Клієнти диктують умови роботи на ринку: зручний програмний інтерфейс, надійний, швидкий, точний та достовірний результат застосування бібліотеки для побудови моделей і ведення досліджень	Необхідно випускати ПЗ не гірше ніж у конкурентів та розширяти функціонал

Проаналізувавши можливості роботи на ринку з огляду на конкурентну ситуацію можна зробити висновок: оскільки кожний з існуючих продуктів не впливає у великій мірі на поточну ситуацію на ринку в цілому, кожний з існуючих продуктів має свою специфічну сферу використання та свої позитивні та негативні сторони щодо рішення певних типів задач, то робота та вихід на даний ринок є можливою і реалізованою задачею.

Для виходу на ринок продукт повинен мати функціонал що відсутній у продуктів-аналогів, повинен задовольняти потреби користувачів, мати

необхідний та достатній функціонал з конфігурування, підтримку зі сторони розробників та можливість розробки спеціального функціоналу за відповідною ліцензією.

Таблиця 5.10 — Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Зручність застосування	Використання сучасних патернів та архітектурних дизайн-принципів
2	Збільшення швидкості реалізації програмного забезпечення	Підвищення рівня абстракції при використанні бібліотек машинного навчання
3	Орієнтованість на кінцевого споживача	Продукт орієнтований на розробку та проведення досліджень сигналів ЕКГ вченими та програмістами
4	Розширений функціонал аналізу лінгвістичного представлення ЕКГ	Продукт надає інструменти для операцій необхідних при досліджені лінгвістичного представлення ЕКГ
5	Система відновлення даних	Бібліотека має вбудований механізм автоматичного кешування

Таблиця 5.11 — Порівняльний аналіз сильних та слабких сторін системи кешування мало змінних даних

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим						
			-3	-2	-1	0	+1	+2	+3
1	Зручність застосування	15		+					
2	Збільшення швидкості реалізації програмного забезпечення	20		+					
3	Орієнтованість на кінцевого споживача	13			+				
4	Розширений функціонал аналізу лінгвістичного представлення ЕКГ	18	+						
5	Система відновлення даних	19		+					

Таблиця 5.12 — SWOT аналіз стартап-проєкту

Сильні сторони (S): – зручність застосування; – збільшення швидкості реалізації програмного забезпечення; – орієнтованість на кінцевого споживача; – розширений функціонал аналізу лінгвістичного представлення ЕКГ; – система відновлення даних.	Слабкі сторони (W): – нерозуміння потреб ринку.
Можливості (O): – конкуренція; – нові методи для розробки ПЗ для дослідження ЕКГ.	Загрози (T): – конкуренти; – кошти на розробку та підтримку продукту; – вихід аналогу.

Таблиця 5.13 — Альтернативи ринкового впровадження стартап-проєкту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Безкоштовне надання певного функціоналу у користування споживачам на обмежений термін	Головний ресурс – люди, даний ресурс - наявний	3-4 місяці
2	Реклама	Залучення власних коштів для реклами товару	1-3 місяці
3	Написання статей та опис товару на відомих ресурсах	Головний ресурс – час, даний ресурс - наявний	1-2 місяці
4	Презентація товару на хакатонах й інших ІТ заходах	Ресурс – час та гроші для участі, наявні	1-3 місяці

5.4 Розроблення ринкової стратегії проєкту

Таблиця 5.14 — Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Приватні підприємства міського та міжнародного рівня, діяльність яких пов'язана з медичними дослідженнями	Висока	Високий	Сильна	Складно
2	Приватні клініки	Помірна	Помірний	Помірна	Середня
3	Науковці-програмісти	Помірна	Помірний	Помірна	Середня
4	Заклади медичного туризму	Помірна	Слабкий	Слабка	Просто
Які цільові групи обрано: 1,2,3					

Відповідно до проведеного аналізу можна зробити висновок, що підходящою цільовою групою для розповсюдження даного програмного продукту є приватні підприємства, які проводять дослідження ЕКГ, приватні клініки та нуковці програмісти. Відповідно до стратегії охоплення ринку збуту товару обрано стратегію масового маркетингу, оскільки для підприємств, клінік та індивідуальних науковців у цілому надається стандартизований продукт з можливістю розширення функціональності за домовленістю (відповідно до ліцензії).

Таблиця 5.15 — Визначення базової стратегії розвитку

Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Надання функціональності що відсутня у товарів-замінників, підтримка клієнтів	Проведення реклами, освітлення унікальної функціональності через інтернет ресурси та інші канали, контакт напряму з споживачами; формування лояльності і прихильності споживачів	Зниження ступеню замінності товару; Прихильність клієнтів; Відмінні властивості товару; Відмінні характеристики товару;	Стратегія диференціації

Таблиця 5.16 — Визначення базової стратегії конкурентної поведінки

Чи є проєкт «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, які?	Стратегія конкурентної поведінки
Ні, оскільки є товари-замінники, але дані товари замінники не мають деякого необхідного функціоналу та не мають ідентичної ефективності	Так, ціль компанії знайти нових споживачів та, частково, забрати існуючих у конкурентів задля задоволення потреб останніх	Компанія частково копіює характеристики товару конкурента, основна ціль компанії розробка нового унікального функціоналу, з підтримкою основного функціоналу конкурентів	Стратегія заняття конкурентної ніші

Таблиця 5.17 — Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту
1	Легкість розуміння, простота інтеграції в наявні рішення	Стратегія диференціації	Позиція на основі порівняння продукту з товарами конкурентів	Простота освоєння та застосування
2	Збільшення ефективності ведення досліджень	Стратегія диференціації	Висока якість архітектури розробленого продукту	Економія ресурсів потрібних для розробки
3	Невелика вартість	Спеціальні пропозиції	Період безкоштовного користування усіма функціями	Доступність

Відповідно до проведеного аналізу можна зробити висновок, що стартап-компанія вибирає як базову стратегію розвитку – стратегію диференціації, як базову стратегію конкурентної поведінки – стратегію заняття конкурентної ніші.

5.5 Розроблення маркетингової програми стартап-проєкту

Таблиця 5.18 — Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (наявні або такі, що потрібно створити)
1	Швидкість отримання реалізації	Зменшення кількості коду для реалізації	Пришвидшення розробки програмних продуктів, яке може спричинити більш ефективне дослідження ЕКГ та надання там тим самим конкуретноспроможності
2	Ефективність та простота застосування	Архітектура, яка задовольняє дизайн принципи розробки	Елегантність рішень популярних задач та надання інструментарію для вирішення власних
3	Стійка система збереження даних	Система кешування проміжних етапів	Надання можливості збереження даних задля можливості відновлення процесу навчання

Таблиця 5.19 — Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>1. Товар за задумом</td><td colspan="3">Програмна бібліотека для більш ефективного написання коду та проведення досліджень електрокардіограми</td></tr>
<tr><td rowspan="11">2. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх/Тл/Е/Ор</td></tr>
<tr><td>Якість</td><td>Нм</td><td>Тл</td></tr>
<tr><td>Ефективність</td><td>Нм</td><td>Тл</td></tr>
<tr><td>Зручність</td><td>М</td><td>Тх</td></tr>
<tr><td>Відмовостійкість</td><td>Нм</td><td>Е</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td colspan="3">Якість: надійність інструменту для питань аналізу електрокардіограми</td></tr>
<tr><td colspan="3">Пакування: відсутнє</td></tr>
<tr><td colspan="3">Марка: ECG Analysis Library</td></tr>
<tr><td rowspan="2">3. Товар із підкріпленням</td><td colspan="3">До продажу: наявна повна документація, акції на придбання декількох ліцензій, знижки для певних сегментів на покупку товару</td></tr>
<tr><td colspan="3">Після продажу: додаткова підтримка спеціалістів налаштування, підтримка з боку розробника</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності, патент на винахід</td></tr>
</table>

Таблиця 5.20 — Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
-	2000 грн необмежена ліцензія	10 000 грн / міс і вище	1000 грн необмежена ліцензія

Таблиця 5.21 — Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Орієнтація на безкоштовний функціонал	1. Забезпечення якості ПЗ 2. Пробний безкоштовний період	Середня	Безпосередня

Таблиця 5.22 — Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Орієнтація на якісний застосунок	GitHub, StackOverflow	Співвідношення ціна/якість	Якість продукту та потенціал використання	Аналізуй ефективно, не витрачай час дарма

Як результат було створено ринкову (маркетингову) програму, що включає в себе визначення ключових переваг концепції потенційного товару, опис моделі товару, визначення меж встановлення ціни, формування системи збуту та концепцію маркетингових комунікацій.

Висновки по розділу

В даному розділі описано стратегії та підходи з розроблення стартап-проєкту, визначено наявність попиту, динаміку та рентабельність роботи ринку, як висновок було вказано що існує можливість ринкової комерціалізації проєкту. Розглянувши потенційні групи клієнтів, бар’єри входження, стан конкуренції та конкурентоспроможність проєкту було встановлено що проєкт є перспективним. Розглянуто та вибрано альтернативу впровадження стартап-проєкту та доведено доцільність подальшої імплементації проєкту.

ВИСНОВКИ

В даній роботі була розроблена програмна бібліотека на мові Python для пришвидшення реалізації програмних продуктів орієнтованих на аналіз сигналів електрокардіограми на основі моделі word2vec з метою наукових досліджень або виводу результатів для полегшення роботи медичного персоналу.

Проведений аналіз існуючих бібліотек машинного навчання Gensim та Scikit Learn, а також бібліотеки для отримання даних датасетів з хмарного сховища, для виявлення недоліків, який показав, що дані програмні інструменти мають ряд недоліків, серед яких: занадто низький рівень абстракції, який не дає зосередитися розробнику на досліджені; відсутність механізму кешування даних проміжних етапів обробки, для можливості відновлення експериментів та уникнення повторної витрати часу; відсутність інструментів для перетворення сигналу ЕКГ в лінгвістичне представлення для обробки засобами NLP.

Спроектовано архітектурне рішення, яке б вирішувало недоліки бібліотек, додаючи при цьому новий функціонал для спеціалізованої обробки лінгвістичного представлення сигналу електрокардіограми засобами NPL. Розроблена архітектура підвищує рівень абстракції бібліотек при цьому не обмежуючи можливість тонкої параметризації алгоритмів машинного навчання.

Розроблена програмна бібліотека надає класи розширення, які мають вбудований механізм кешування даних, тим самим забезпечуючи автоматичне зберігання проміжних результатів роботи.

Було проведено ряд експериментів з дослідження ефективності розробленої програмної бібліотеки для оцінки характеристик. Проведені дослідження показали значне скорочення кількості коду, необхідного для реалізації базових етапів аналізу ЕКГ. Також, було окремо проведено дослідження ефективності вбудованого математичного забезпечення для лінгвістичного представлення сигналів електрокардіограми, яке показало

значний приріст швидкості виконання алгоритмів-класифікаторів у порівнянні з оригінальним (чисельним) представленням сигналів ЕКГ.

Продемонстровано застосування розробленої бібліотеки для проведення наукових досліджень із використанням засобів NLP. Так, було розроблено алгоритми аналізу лінгвістичного представлення за допомогою моделей Word2Vec, на базі Skip Gram, та TextRank.

Описаний спосіб інтеграції розробленого програмного засобу у нові та вже наявні рішення алгоритмів обробки сигналів електрокардіограми.

ПЕРЕЛІК ПОСИЛАНЬ

1) Gensim library [Електрон. ресурс]. – Режим доступу: https://radimrehurek.com/gensim/

2) Scikit Learn [Електрон. ресурс]. – Режим доступу: https://scikit-learn.org/

3) The WFDB Software Package [Електрон. ресурс]. – Режим доступу: https://archive.physionet.org/physiotools/wfdb.shtml

4) Architecture Patterns with Python by Percival, H. & Gregory, B. – W: «O'Reilly Media», 2020

5) Design patterns. The "Gang of Four": Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides – B.: Addison-Wesley Professional, 1994

6) Python Release Python 3.9.1: офіційний сайт: [Електрон. ресурс]. – Режим доступа: Python.org

7) Matplotlib: офіційний сайт: [Електрон. ресурс]. – Режим доступа: https://matplotlib.org

8) О. Й. Жарінов, В. О. Куць (2017) Основи електрокардіографії:[навч. посіб. для лікарів-слухачів закл. (ф-тів) післядиплом. освіти / Жарінов О. Й. та ін.] — Бібліогр.: с. 235—236

9) Baklan I., Mukha I., Oliinyk Y., Lishchuk K., Nedashkivsky E., Gavrilenko O. (2020) Anomalies Detection Approach in Electrocardiogram Analysis Using Linguistic Modeling. In: Hu Z., Petoukhov S., Dychka I., He M. (eds) Advances in Computer Science for Engineering and Education II. ICCSEEA 2019. Advances in Intelligent Systems and Computing, vol 938. Springer, Cham; pp 513-522, DOI - https://dx.doi.org/10.1007/978-3-030-16621-2_48; (Scopus)

10) J. Pan and W. J. Tompkins, “A real-time qrs detection algorithm,” IEEE Trans. Biomed. Eng, vol. 32, no. 3, pp. 230–236, 1985.

11) Engelse, W. A. H. and Zeelenberg, C. (1979). A single scan algorithm for QRSdetection and feature extraction. Computers in Cardiology, 6:37–42.

12) Hamilton, P. (2002). Open source ecg analysis. Computersin Cardiology

13) Журавлев Ю. И., Рязанов В. В., Сенько О. В. Распознавание. Математические методы. Программная система. Практические применения. — М.: Фазис, 2006.

14) Tryon, Robert C. (1939). Cluster Analysis: Correlation Profile and Orthometric (factor) Analysis for the Isolation of Unities in Mind and Personality. Edwards Brothers.

15) J. A. Lozano J. M. Pena and P. Larranaga, An empirical comparison of four initialization methods for the k-means algorithm," Pattern Reognition Letters, vol. 20, pp.1027-1040,1999.

16) М. Жамбю, Ієрархічний кластерний аналіз та відповідності. — М.: Фінанси і статистика, 1988. — 345 с.

17) M. Brand and K. Huang. A unifying theorem for spectral embedding and clustering. 9th International Conference on Artificial Intelligence and Statistics, 2002.

18) Abdi H., Williams L.J. (2010). Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics, 2: 433–459.75

19) Van der Maaten, L.J.P.; Hinton, G.E. (Nov 2008). Visualizing Data Using tSNE. Journal of Machine Learning Research 9: 2579–2605.

20) Liddy, E.D. 2001. Natural Language Processing. In Encyclopedia of Library and Information Science, 2nd Ed. NY. Marcel Decker, Inc. — P.1

21) История Компьютера Обработка естественного языка: офіційний сайт: [Електрон. ресурс]. – Режим доступа: http://chernykh.net/content/view/1105/1189/

22) Mikolov, T., Yih W., Zweig G. Linguistic regularities in continuous space word representations. // Proc of NAACL-HLT 2013. P. 746–751.

23) McGinnis W. Beyond one-hot: an exploration of categorical variables // Data science, technology, Atlanta. – 2015;

24) Pennington, J., Socher R., Manning C.D. Global Vectors for Word Representation. // Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing, P. 1532–1543.

25) Takala, P. Word Embeddings for Morphologically Rich Languages // Computational Intelligence and Machine Learning. Belgium. Bruges. 2016. P. 27–29.

26) Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space // Proc. of Workshop at ICLR. 2013. P. 1301-3781.

27) Ho, Tin Kam (1995). Random Decision Forests. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.

28) Sajad Mousavi, Fatemeh Afghah, Fatemeh Khadem, and U. Rajendra Acharya ECG Language Processing (ELP): New Technique to Analyze ECG Signals, 2020, pp.2-4

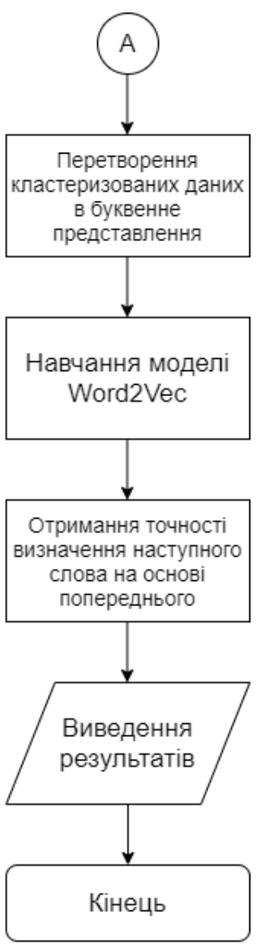
29) NumPy: офіційний сайт: [Електрон. ресурс]. – Режим доступа: numpy.org

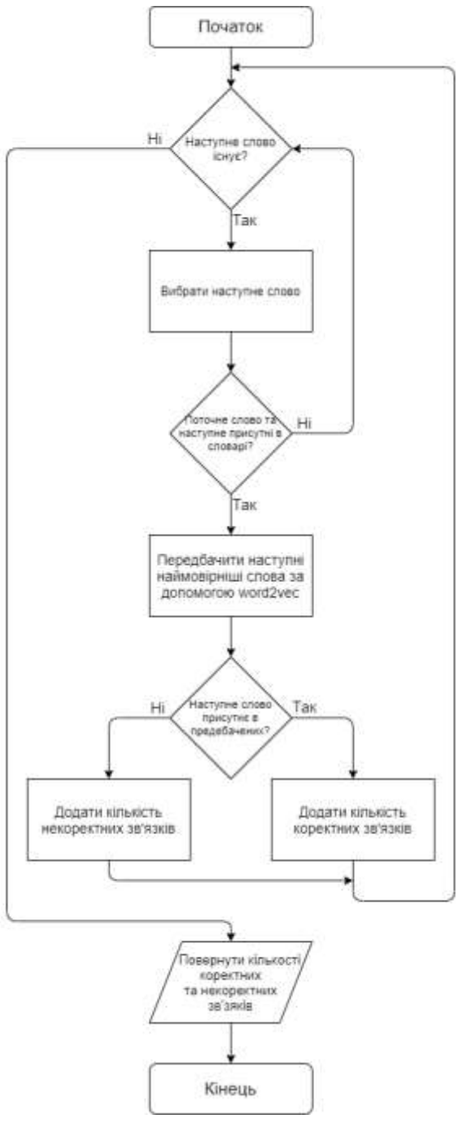
30) Baklan, I., Oliinyk, Y., Mukha, I., Lishchuk, K., Gavrilenko, O., Ocheretianyi, O., & Tsytsyliuk, A. Adaptive Multistage Method of Anomalies Detection in ECG Time Series. Proceedings of the 4th International Conference on Computational Linguistics and Intelligent Systems (COLINS 2020). Volume I: Main Conference Lviv, Ukraine, April 23-24, 2020. P. 670–679

31) Терещенко А.С., Олійник Ю.О. АНАЛІЗ ЕКГ СИГНАЛІВ ЗА ДОПОМОГОЮ МОДЕЛІ WORD2VEC. Інформаційні системи та технології управління: матеріали всеукр. наук.-практ. конф. молодих вчених та студентів, (м. Київ, 26-27 лист. 2020 р.). Київ, 2020. С. 77-80

ДОДАТОК А ГРАФІЧНІ МАТЕРІАЛИ

Алгоритми математичного забезпечення





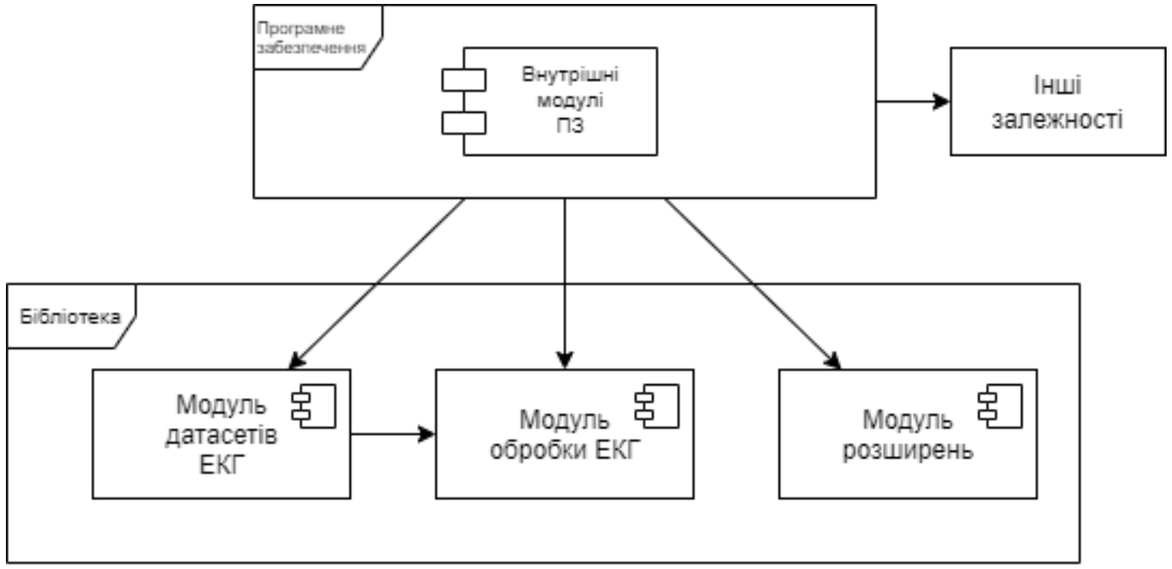
Демонстраційний плакат до магістерської дисертації

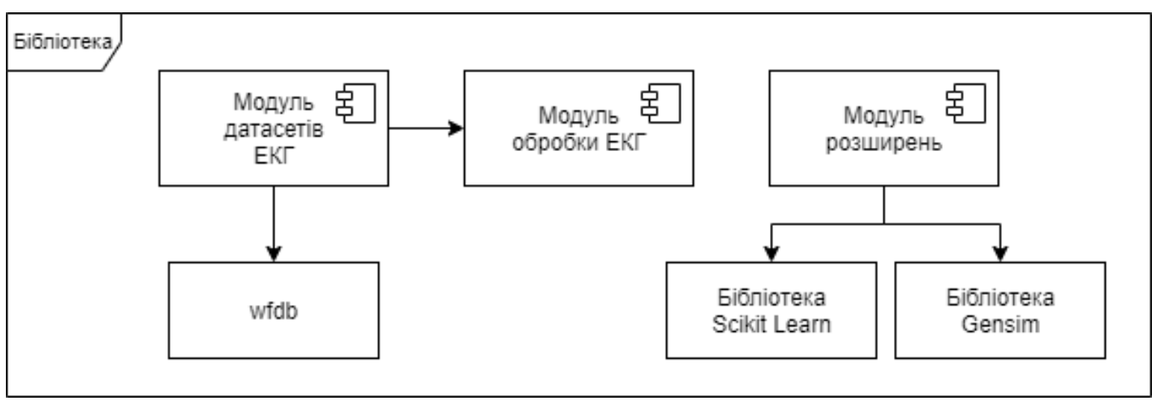
Алгоритми математичного забезпечення

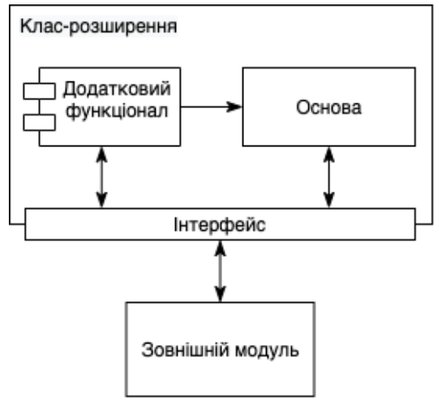
Виконав студент гр. ІП-02мп Язенок М. С.

Керівник Олійник Ю.О.

Архітектура програмного забезпечення







Демонстраційний плакат до магістерської дисертації

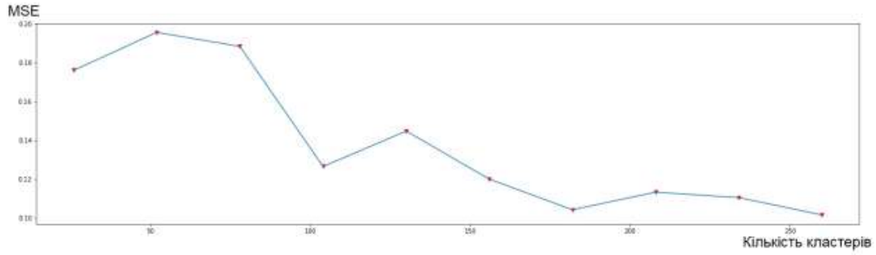
Архітектура програмного забезпечення

Виконав студент гр. ІП-02мп Язенок М. С.

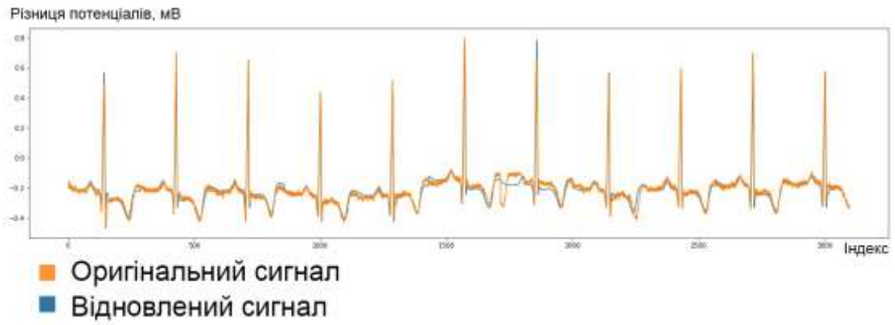
Керівник Олійник Ю.О.

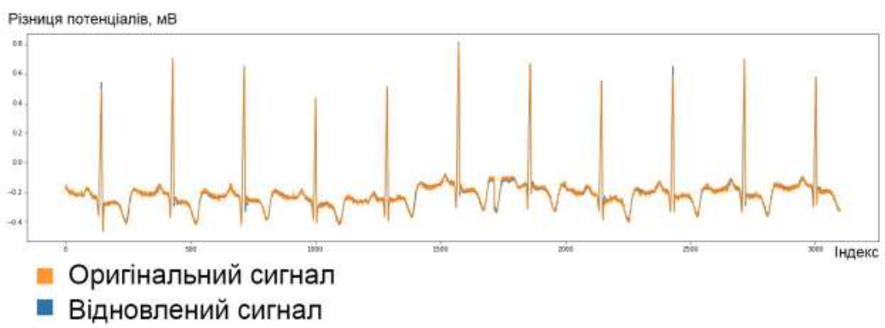
Дослідження ефективності програмного та математичного забезпечення

Кількість кластерів у першій моделі	Кількість кластерів у другій моделі	Середньоквадратична похибка
6	20	0.173
12	40	0.192
18	60	0.19
24	80	0.124
30	100	0.143
36	120	0.12
42	140	0.105
48	160	0.112
54	180	0.111
60	200	0.101



Задача кешування, або навчання та кешування	З використанням бібліотеки		Без використання бібліотеки	
	Слів	Символів	Слів	Символів
Word2vec	37	336	169	1117
KMeans	20	237	121	736
WFDB Dataset	19	199	609	3932





Алгоритм	Початковий розмір датасету (тисяч серцебиттів)	Тривалість навчання (секунд)		F-міра	
		Без Word2Vec	З використанням Word2Vec	Без Word2Vec	З використанням Word2Vec
Random Forest	10.5	33,2447	27,796	0.95	0.96
Gradient Boosting		32,1669	27,877	0.96	0.98
Random Forest	13.2	495,244	45,063	0.95	0.97
Gradient Boosting		479,433	40,708	0.93	0.97

Демонстраційний плакат до магістерської дисертації

Дослідження ефективності програмного та математичного забезпечення

Виконав студент гр. ІП-02мп Язенок М. С.

Керівник Олійник Ю.О.

ДОДАТОК Б ЛІСТИНГ КОДУ

```
import pickle
import os
import numpy as np
from sklearn.cluster import KMeans


class Beats2Words:
    qrs_kmeans = None
    pt_kmeans = None

    def save(self, file_name):
        pickle.dump(self.qrs_kmeans, open(file_name + "_qrs.pkl", "wb"))
        pickle.dump(self.pt_kmeans, open(file_name + "_pt.pkl", "wb"))

    def load(self, file_name):
        self.qrs_kmeans = pickle.load(open(file_name + "_qrs.pkl", "rb"))
        self.pt_kmeans = pickle.load(open(file_name + "_pt.pkl", "rb"))

    @staticmethod
    def _split_beats_in_p_qrt_t(beats_list):
        heart_beat_len = len(beats_list[0])
        retreat_value = heart_beat_len // 2
        c_p_waves = []
        c_qrs_waves = []
        c_t_waves = []
        for j in range(len(beats_list)):
            c_p_waves.append(beats_list[j][:retreat_value - 15])
            c_qrs_waves.append(beats_list[j][retreat_value - 15:retreat_value + 15])
            c_t_waves.append(beats_list[j][retreat_value + 15:])
        return c_p_waves, c_qrs_waves, c_t_waves

    # Convert PT cluster predictions to letters
    @staticmethod
    def _convert_pt_predictions_to_letters(predictions):
        alphabet_for_pt = {0: 'a', 1: 'b', 2: 'c', 3: 'd', 4: 'e', 5: 'f', 6: 'g', 7:
'h', 8: 'i',
                           9: 'j', 10: 'k', 11: 'l', 12: 'm', 13: 'n', 14: 'o', 15:
'p', 16: 'q', 17: 'r',
                           18: 's', 19: 't'}

        def get_symbol_pt(x):
            return alphabet_for_pt[x]

        vfunc = np.vectorize(get_symbol_pt)

        return vfunc(predictions)

    # Convert QRS clusters predictions to letters
    @staticmethod
    def _convert_qrs_predictions_to_letters(predictions):
        alphabet_for_qrs = {0: 'u', 1: 'v', 2: 'w', 3: 'x', 4: 'y', 5: 'z'}

        def get_symbol_qrs(x):
            return alphabet_for_qrs[x]

        vfunc_2 = np.vectorize(get_symbol_qrs)
```

```
        return vfunc_2(predictions)

    @staticmethod
    def _join_waves_letters_to_words(qrs_letters, pt_letters):
        words = []
        signal_half_len = len(qrs_letters)//2
        for i in range(len(qrs_letters)):
            word = ''
            word += pt_letters[i]
            word += qrs_letters[i]
            # T signal is encoded in second half
            word += pt_letters[i + signal_half_len]
            words.append(word)

        return words

    def fit_and_predict_words(self, beats_list, cache_file_name, reset_cache=False):

        if reset_cache or not (os.path.exists(cache_file_name + "_qrs.pkl")
                               and os.path.exists(cache_file_name + "_pt.pkl")):
            (p_waves, qrs, t_waves) = Beats2Words._split_beats_in_p_qrt_t(beats_list)
            print("Fitting QRS KMeans")
            self.qrs_kmeans = KMeans(init='k-means++', n_clusters=6, n_init=3)
            # self.qrs_kmeans = KMeans(init='k-means++', n_clusters=50, n_init=25,
max_iter=600)
            self.qrs_kmeans.fit(qrs)

            qrs_predicted = self.qrs_kmeans.predict(qrs)

            print("Fitting PT KMeans")
            c_pt_waves = np.array(p_waves + t_waves)
            self.pt_kmeans = KMeans(init='k-means++', n_clusters=20, n_init=10)
            # self.pt_kmeans = KMeans(init='k-means++', n_clusters=200, n_init=100,
max_iter=600)
            self.pt_kmeans.fit(c_pt_waves)

            print("Saving BeatsToWordsConverter to cache: ", cache_file_name)
            self.save(cache_file_name)

            pt_predicted = self.pt_kmeans.predict(c_pt_waves)

            qrs_letters =
Beats2Words._convert_qrs_predictions_to_letters(qrs_predicted)
            pt_letters = Beats2Words._convert_pt_predictions_to_letters(pt_predicted)
            return Beats2Words._join_waves_letters_to_words(qrs_letters, pt_letters)
        else:
            print("Using cached BeatsToWordsConverter from file: ", cache_file_name)
            self.load(cache_file_name)
            return self.predict_words(beats_list)

    def predict_words(self, beats_list):
        (p_waves, qrs, t_waves) = Beats2Words._split_beats_in_p_qrt_t(beats_list)
        c_pt_waves = np.array(p_waves + t_waves)
        qrs_predicted = self.qrs_kmeans.predict(qrs)
        pt_predicted = self.pt_kmeans.predict(c_pt_waves)

        qrs_letters = Beats2Words._convert_qrs_predictions_to_letters(qrs_predicted)
```

```
        pt_letters = Beats2Words._convert_pt_predictions_to_letters(pt_predicted)
        return Beats2Words._join_waves_letters_to_words(qrs_letters, pt_letters)
```

```
import pandas as pd


class EcgDataset:
    @staticmethod
    def load(file_path):
        df = pd.read_pickle(file_path)
        train_start_indices = pd.read_pickle(file_path + ".start_indices.pkl")
        return EcgDataset(df, train_start_indices)

    dataframe: pd.DataFrame
    train_start_indices: pd.DataFrame

    def __init__(self, dataframe, train_start_indices):
        self.dataframe = dataframe
        self.train_start_indices = train_start_indices

    def save(self, file_path):
        self.dataframe.to_pickle(file_path)
        self.dataframe.to_pickle(file_path + ".start_indices.pkl")

    def get_beats_and_labels(self):
        return self.dataframe["beats"].tolist(), self.dataframe["labels"].tolist()
```

```
import os
import wfdb
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split

from word2vec_ext.ecg_processing.ecgdataset import EcgDataset

_mit_abnormal_beats = [
    "L", "R", "B", "A", "a", "J", "S", "V",
    "r", "F", "e", "j", "n", "E", "/", "f", "Q", "?"
]

_mit_abnormal_aux = [
    "AFIB", "AFL", "J"
]


class EcgDatasetsHolder:

    @staticmethod
    def _classify_beat(symbol):
        if symbol in _mit_abnormal_beats:
            return 1
        elif symbol == "N" or symbol == ".":
```

```
            return 0

    @staticmethod
    def _classify_aux(aux):
        if aux in _mit_abnormal_aux:
            return 1
        elif aux == "N" or aux == ".":
            return 0

    @staticmethod
    def _multi_classify_aux(aux):
        try:
            return _mit_abnormal_aux.index(aux) + 1
        except ValueError:
            return 0

    @staticmethod
    def _get_sets_names(dir_path):
        from os import listdir
        from os.path import isfile, join
        onlyfiles = [f for f in listdir(dir_path) if isfile(join(dir_path, f))]

        def get_file_name(file):
            return file.split(".", 1)[0]

        data = [int(x) for x in list(set(list(map(get_file_name, onlyfiles))))]
        data.sort()
        return data

    @staticmethod
    def _combine_sets_beats_and_features(annotator_type, records_path, sets_numbers):
        """

        :param annotator_type: symbol/aux/aux_multi
        :param records_path:
        :param sets_numbers:
        :return:
        """
        sets_data = []
        for set_number in sets_numbers:
            print("Loading set: " + str(set_number))
            if os.path.exists(records_path + "/" + str(set_number) + ".dat"):
                beats, labels, normal_percentage =
EcgDatasetsHolder._load(annotator_type, records_path, set_number)
                if beats is not None:
                    sets_data.append({
                        "beats": beats,
                        "labels": labels,
                        "normal_percentage": normal_percentage
                    })
                else:
                    print("Skipping empty set #" + str(set_number))
            else:
                print("Set " + str(set_number) + " contains invalid data. Skipping")
        return sets_data

    @staticmethod
    def _get_sequence(signal, beat_loc, window_sec, fs):
        window_one_side = window_sec * fs
```

```
        beat_start = beat_loc - window_one_side
        beat_end = beat_loc + window_one_side
        if beat_end < signal.shape[0]:
            sequence = signal[beat_start:beat_end, 0]
            return sequence
        else:
            return np.array([])

    @staticmethod
    def _load(annotator_type, out_dir, record_number):
        record = wfdb.rdrecord(str(out_dir) + '/' + str(record_number))
        annotation = wfdb.rdann(str(out_dir) + '/' + str(record_number), "atr")
        atr_symbols = annotation.symbol
        atr_samples = annotation.sample
        atr_aux = [aux[1:] for aux in annotation.aux_note]
        fs = record.fs
        signal = record.p_signal

        labels = []
        valid_beats = []
        window_sec = 3
        aux_annotator_type = annotator_type == "aux"
        aux_multi_annotator_type = annotator_type == "aux_multi"
        for i, i_sample in enumerate(atr_samples):
            if aux_annotator_type:
                label = EcgDatasetsHolder._classify_aux(atr_aux[i])
            elif aux_multi_annotator_type:
                label = EcgDatasetsHolder._multi_classify_aux(atr_aux[i])
            else:
                label = EcgDatasetsHolder._classify_beat(atr_symbols[i])
            if label is not None:
                sequence = EcgDatasetsHolder._get_sequence(signal, i_sample,
window_sec, fs)
                if sequence.size > 0:
                    labels.append(label)
                    valid_beats.append(sequence)
        if len(labels) == 0:
            return None, None, None

        if aux_multi_annotator_type:
            def map_abnormal(beat_label):
                if beat_label > 0:
                    return 0
                else:
                    return 1

            normal_percentage = sum(list(map(map_abnormal, labels))) / len(labels)
        else:
            normal_percentage = sum(labels) / len(labels)

        assert len(valid_beats) == len(labels)
        return valid_beats, labels, normal_percentage

    @staticmethod
    def _download_dataset(database_name, out_dir, reload):
        """

        :param database_name: afdb or mitdb
        :param out_dir:
```

```
        :param reload:
        :return:
        """
        if not os.path.isdir(out_dir) or reload:
            print("Downloading dataset")
            wfdb.dl_database(database_name, out_dir)

    @staticmethod
    def _load_and_save_set(database_name, records_path, dataframe_path, annotator_type,
reload=False):
        """

        :param database_name: afdb or mitdb
        :param records_path:
        :param dataframe_path:
        :param annotator_type: symbol/aux/aux_multi
        :param reload:
        :return:
        """
        EcgDatasetsHolder._download_dataset(database_name, records_path, reload)
        if os.path.exists(dataframe_path) and not reload:
            return pd.read_pickle(dataframe_path)
        else:
            print("Reloading dataset")
            data_frame = pd.DataFrame(
                EcgDatasetsHolder._combine_sets_beats_and_features(annotator_type,
records_path,

EcgDatasetsHolder._get_sets_names(records_path)))
            data_frame.to_pickle(dataframe_path)
            return data_frame

    @staticmethod
    def cache_from_mit(sets_count_limit, database_name, mit_records_path,
dataframe_path, annotator_type="symbol",
                       reload=False):
        """

        :param sets_count_limit:
        :param database_name: afdb or mitdb
        :param mit_records_path:
        :param dataframe_path:
        :param annotator_type: symbol/aux/aux_multi
        :param reload:
        :return:
        """
        data_frame = EcgDatasetsHolder._load_and_save_set(database_name,
mit_records_path, dataframe_path, annotator_type,
                                                          reload)
        if sets_count_limit is not None:
            data_frame = data_frame[:sets_count_limit]
        return EcgDatasetsHolder(data_frame)

    @staticmethod
    def prepare_train_and_test(mit_records_path, dataframe_path,
                               train_path, test_path,
                               reload_base_data=False, test_size=0.25,
                               random_state=42, sets_count_limit=None,
reload_train_test=False):
```

```
        if reload_train_test or not os.path.exists(train_path) or not
os.path.exists(test_path):
            ecgdataset = EcgDatasetsHolder.cache_from_mit(sets_count_limit,
mit_records_path, dataframe_path, reload_base_data)

            train, test = ecgdataset.split_train_test(test_size=test_size,
random_state=random_state)

            train_ready = train.concatenate_datasets()
            test_ready = test.concatenate_datasets()

            train_ready.save(train_path)
            test_ready.save(test_path)
        else:
            return EcgDataset.load(train_path), EcgDataset.load(test_path)

    dataframe: pd.DataFrame

    def __init__(self, dataframe):
        self.dataframe = dataframe

    def split_train_test(self, test_size=0.25, random_state=42, sets_count_limit=None):
        data_frame = self.dataframe
        if sets_count_limit is not None:
            data_frame = data_frame[:sets_count_limit]
        print("Splitting dataset")
        bins = [0, 0.2, 0.6, 1.0]
        data_frame["bin"] = pd.cut(data_frame['normal_percentage'], bins=bins,
labels=False, include_lowest=True)
        train, validation = train_test_split(data_frame, test_size=test_size,
stratify=data_frame["bin"],
                                             random_state=random_state)

        return EcgDatasetsHolder(train), EcgDatasetsHolder(validation)

    def get_concatenated_rows(self):
        """
          :return: dataframe with connected rows
          """
        all_beats = []
        all_labels = []
        start_index = 0
        start_indices = []
        for i, row in self.dataframe.iterrows():
            start_indices.append(start_index)
            beats = row["beats"]
            all_beats.append(beats)
            all_labels.append(row["labels"])
            start_index = start_index + len(beats)

        beats = np.concatenate(all_beats).tolist()
        labels = np.concatenate(all_labels).tolist()
        return pd.DataFrame({
            "beats": beats,
            "labels": labels
        }), pd.DataFrame({"start_indices": start_indices})

    def concatenate_datasets(self):
```

```
        concated_df, train_start_indices = self.get_concatenated_rows()
        return EcgDataset(concated_df, train_start_indices)



import pickle
import uuid
import os
import wfdb
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import classification_report

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from ecgdetectors import Detectors
from gensim.models import Word2Vec
from datetime import datetime


class Word2VecExt:
    @staticmethod
    def load(file_path):
        word2vec = Word2Vec.load(file_path)
        return Word2VecExt(word2vec)

    @staticmethod
    def load_or_fit_and_save(file_path, sentences, workers=3, vector_size=100,
min_count=5, window=5, sample=1e-3,
                             reset=False):
        if not (os.path.exists(file_path) and reset):
            model = Word2Vec(sentences, workers=workers,
                             vector_size=vector_size, min_count=min_count,
                             window=window, sample=sample)
            model.save(file_path)
        else:
            model = Word2Vec.load(file_path)
        return Word2VecExt(model)

    @staticmethod
    def load_or_fit_words_and_save(words, sentences_start_indices, file_path,
workers=3, vector_size=100,
                                   min_count=5, window=5, sample=1e-3, sg=0, hs=0,
negative=5,
                                   alpha=0.025,
                                   reset=False):
        if not os.path.exists(file_path) or reset:
            print("TeachWord2Vec: Concatenating sentences")
            sentences = []
            for i in range(len(sentences_start_indices) - 1):
                start_index = sentences_start_indices[i]
                end_index = sentences_start_indices[i + 1]
                sentences.append(words[start_index:end_index])
            sentences.append(words[sentences_start_indices[-1]:])
```

```
            print("TeachWord2Vec: Start Word2Vec training")
            model = Word2Vec(sentences, workers=workers,
                             vector_size=vector_size, min_count=min_count,
                             window=window, sample=sample, sg=sg, hs=hs,
negative=negative, alpha=alpha)
            model.save(file_path)
        else:
            print("Using cached Word2Vec from file: ", file_path)
            model = Word2Vec.load(file_path)

        return Word2VecExt(model)

    word2vec: Word2Vec

    def __init__(self, word2vec=None):
        self.word2vec = word2vec

    def save(self, file_path):
        self.word2vec.save(file_path)

    def vectorize_valid(self, words):
        feature_vecs = np.zeros((len(words), self.word2vec.vector_size),
dtype='float32')

        index2word_set = set(self.word2vec.wv.index_to_key)

        indices = []
        valid_words_count = 0
        for i in range(len(words)):
            cur_word = words[i]
            if cur_word in index2word_set:
                feature_vecs[valid_words_count] = self.word2vec.wv[cur_word]
                valid_words_count = valid_words_count + 1
                indices.append(i)

        return indices, feature_vecs[:valid_words_count]

    def vectorize_valid_with_labels(self, words, labels):
        valid_beats_indices, features = self.vectorize_valid(words)

        valid_beats_labels = [labels[i] for i in valid_beats_indices]
        print("Valid labels count:")
        print(np.bincount(np.array(valid_beats_labels)))

        return features, valid_beats_labels
```

```
from word2vec_ext.ecg_processing import ecgdataset
from word2vec_ext.ecg_processing import ecgdatasetsholder
from word2vec_ext.ecg_processing import beats2words
from word2vec_ext.word2vec import word2vecext

import uuid
from sklearn.metrics import classification_report
from sklearn.ensemble import GradientBoostingClassifier
from datetime import datetime
```

```
if __name__ == '__main__':
    rsc_dir = "../example_rsc"
    test_start_time = datetime.now()

    ecgdataset = ecgdatasetsholder.EcgDatasetsHolder.cache_from_mit(
        sets_count_limit=5,
        database_name="mitdb",
        mit_records_path=rsc_dir + "/mit_records",
        dataframe_path=rsc_dir + "/ecgdataset",
        annotator_type="symbol",
        reload=False
    )

    train, test = ecgdataset.split_train_test(test_size=0.25, random_state=42)

    train_ready = train.concatenate_datasets()
    test_ready = test.concatenate_datasets()

    train_ready.save(rsc_dir + "/train_dataset")
    test_ready.save(rsc_dir + "/test_dataset")

    # All up code is equal to code below:
    # train_ready, test_ready = ecgdataset.prepare_train_and_test(rsc_dir +
"/mit_records", rsc_dir + "/ecgdataset",
    #                                                             rsc_dir +
"/train_dataset",
    #                                                             rsc_dir +
"/test_dataset", test_size=0.25, random_state=42)

    data_preparing_start_time = datetime.now()

    beats2wordsModel = beats2words.Beats2Words()
    train_words =
beats2wordsModel.fit_and_predict_words(train_ready.dataframe["beats"].tolist(),
                                                          rsc_dir + "/beats2words",
                                                          reset_cache=False)

    validation_words =
beats2wordsModel.predict_words(test_ready.dataframe["beats"].tolist())

    num_features = 300
    word2vecExtModel = \
        word2vecext.Word2VecExt.load_or_fit_words_and_save(train_words,

train_ready.train_start_indices["start_indices"].tolist(),
                                                           rsc_dir + "/word2vec",
                                                           vector_size=num_features,
                                                           reset=False)
    train_data = word2vecExtModel.vectorize_valid_with_labels(train_words,
train_ready.dataframe["labels"].tolist())

    validation_data = word2vecExtModel.vectorize_valid_with_labels(validation_words,
test_ready.dataframe["labels"].tolist())

    (train_x, train_y), (validation_x, validation_y) = (train_data, validation_data)

    data_preparing_time_spend = datetime.now() - data_preparing_start_time

    print("Training random forest")
```

```
    training_start_time = datetime.now()
    forest = GradientBoostingClassifier(n_estimators=500, learning_rate=0.001,
max_depth=7)
    forest = forest.fit(train_x, train_y)
    training_time_spend = datetime.now() - training_start_time

    print("Predicting using random forest")
    result = forest.predict(validation_x)

    print("Creating classification report")
    repord_id = uuid.uuid1()
    report = str(classification_report(validation_y, result))
    print("Report id: " + str(repord_id))
    print(report)
    f = open("results.txt", "a")

    now = datetime.now()
    f.write("=========================================\n")
    f.write("Report id: " + str(repord_id) + "\n")
    f.write("Report start date: " + test_start_time.strftime("%d.%m.%Y %H:%M:%S") +
"\n")
    f.write("Report end date: " + now.strftime("%d.%m.%Y %H:%M:%S") + "\n")
    f.write("Actual train size: " + str(len(train_x)) + "\n")
    f.write("Actual validation size: " + str(len(validation_x)) + "\n")
    f.write("Actual validation size: " + str(len(validation_x)) + "\n")
    f.write("Data preparing time millis: " +
str(data_preparing_time_spend.total_seconds() * 1000) + "\n")
    f.write("Training working time millis: " + str(training_time_spend.total_seconds()
* 1000) + "\n")
    f.write(report)
    f.write("\n=========================================")
    f.write("\n\n")
    f.close()
```

```
from word2vec_ext.ecg_processing import ecgdataset
from word2vec_ext.ecg_processing import ecgdatasetsholder
from word2vec_ext.ecg_processing import beats2words
from word2vec_ext.word2vec import word2vecext

import uuid
from sklearn.metrics import classification_report
from sklearn.ensemble import GradientBoostingClassifier
from datetime import datetime


def calculate_word2vec_accuracy(c_word2vec: word2vecext.Word2Vec, c_validation_words,
validation_sentences_indices):
    print(validation_sentences_indices)
    print(len(c_validation_words))
    true_cases = 0
    false_cases = 0
    index2word_set = set(c_word2vec.wv.index_to_key)

    for i in range(len(validation_sentences_indices)):
        if i == len(validation_sentences_indices) - 1:
            end_index = len(c_validation_words)
```

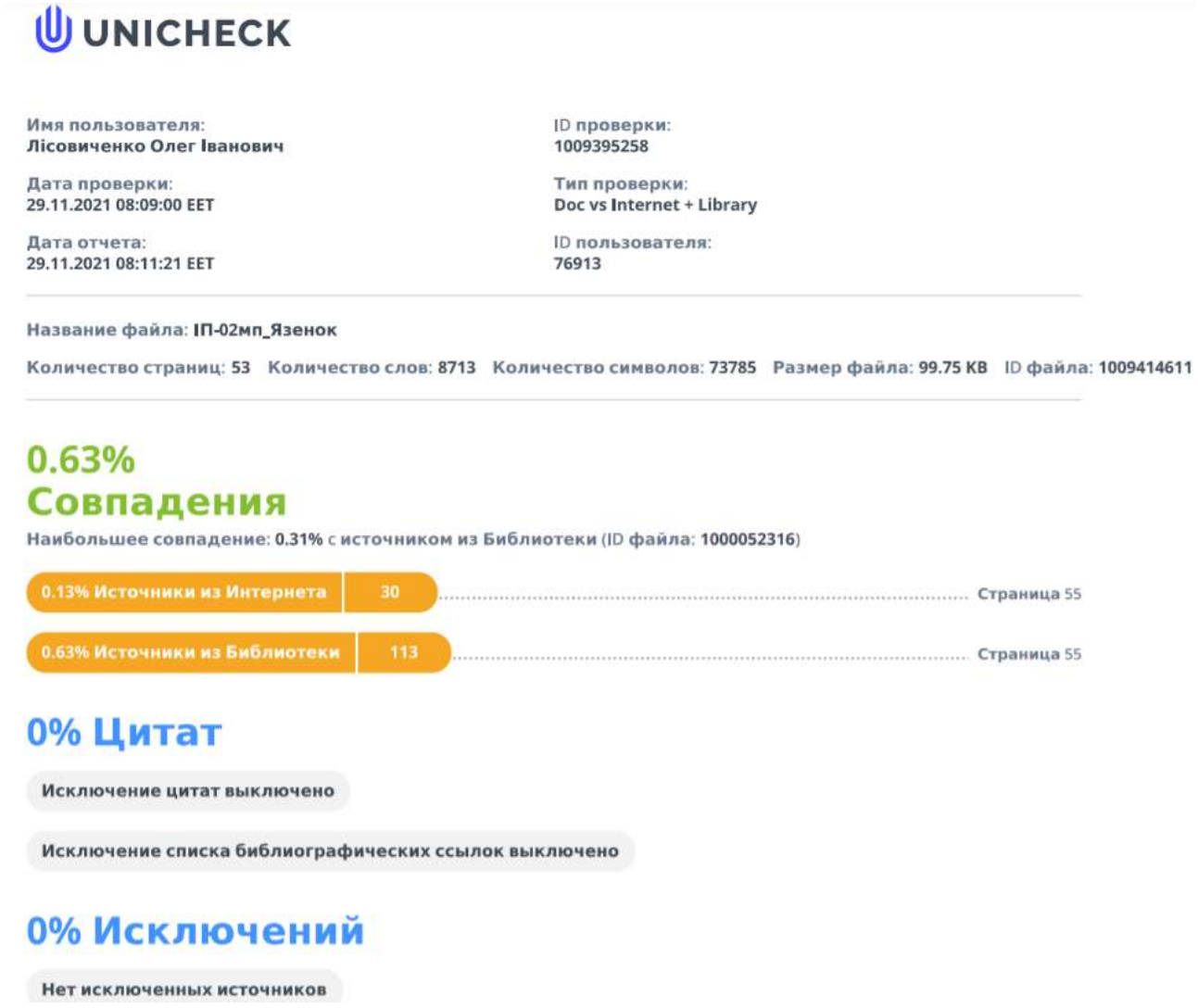
```
        else:
            end_index = validation_sentences_indices[i + 1]
        start_index = validation_sentences_indices[i]
        for j in range(start_index, end_index - 1):  # end_index - 1 to compare with
next word
            # Iterate over each word in sentences
            if not (c_validation_words[j] in index2word_set and c_validation_words[j +
1] in index2word_set):
                # Skip words pairs which are not in the dictionary
                continue
            word_to_vec_list =
c_word2vec.wv.most_similar(positive=[c_validation_words[j]])
            similar_words = map(lambda it: it[0], word_to_vec_list)

            if c_validation_words[j + 1] in similar_words:
                true_cases = true_cases + 1
            else:
                false_cases = false_cases + 1

    print("True_cases =", true_cases)
    print("False_cases =", false_cases)
    print("True_cases/total_cases:", true_cases / (true_cases + false_cases))


def main():
    rsc_dir = "../example_rsc"

    ecgdataset = ecgdatasetsholder.EcgDatasetsHolder.cache_from_mit(
        sets_count_limit=5,
        database_name="mitdb",
        mit_records_path=rsc_dir + "/mit_records",
        dataframe_path=rsc_dir + "/ecgdataset",
        annotator_type="symbol",
        reload=False
    )

    train_ready = ecgdataset.concatenate_datasets()

    train_ready.save(rsc_dir + "/full_dataset")

    beats2wordsModel = beats2words.Beats2Words()
    train_words =
beats2wordsModel.fit_and_predict_words(train_ready.dataframe["beats"].tolist(),
                                                        rsc_dir +
"/beats2words_test_for_5_datasets",
                                                        reset_cache=True)

    num_features = 300
    word2vecExtModel = \
        word2vecext.Word2VecExt.load_or_fit_words_and_save(train_words,

train_ready.train_start_indices["start_indices"].tolist(),
                                                           rsc_dir +
"/word2vec_5_dataset_test",
                                                           vector_size=num_features,
                                                           sg=1,
                                                           sample=1e-3,
                                                           window=10,
                                                           alpha=0.01,
```

```
                                                        reset=True)

    calculate_word2vec_accuracy(word2vecExtModel.word2vec, train_words,
train_ready.train_start_indices["start_indices"].tolist())
    return
    # noinspection DuplicatedCode
    train_data = word2vecExtModel.vectorize_valid_with_labels(train_words,
train_ready.dataframe["labels"].tolist())

    validation_data = word2vecExtModel.vectorize_valid_with_labels(validation_words,
test_ready.dataframe["labels"].tolist())

    (train_x, train_y), (validation_x, validation_y) = (train_data, validation_data)

    data_preparing_time_spend = datetime.now() - data_preparing_start_time

    print("Training random forest")
    training_start_time = datetime.now()
    forest = GradientBoostingClassifier(n_estimators=500, learning_rate=0.001,
max_depth=7)
    forest = forest.fit(train_x, train_y)
    training_time_spend = datetime.now() - training_start_time

    print("Predicting using random forest")
    result = forest.predict(validation_x)

    print("Creating classification report")
    repord_id = uuid.uuid1()
    report = str(classification_report(validation_y, result))
    print("Report id: " + str(repord_id))
    print(report)
    f = open("results.txt", "a")

    now = datetime.now()
    f.write("=========================================\n")
    f.write("Report id: " + str(repord_id) + "\n")
    f.write("Report start date: " + test_start_time.strftime("%d.%m.%Y %H:%M:%S") +
"\n")
    f.write("Report end date: " + now.strftime("%d.%m.%Y %H:%M:%S") + "\n")
    f.write("Actual train size: " + str(len(train_x)) + "\n")
    f.write("Actual validation size: " + str(len(validation_x)) + "\n")
    f.write("Actual validation size: " + str(len(validation_x)) + "\n")
    f.write("Data preparing time millis: " +
str(data_preparing_time_spend.total_seconds() * 1000) + "\n")
    f.write("Training working time millis: " + str(training_time_spend.total_seconds()
* 1000) + "\n")
    f.write(report)
    f.write("\n=========================================")
    f.write("\n\n")
    f.close()


if __name__ == '__main__':
    main()
```

```
from word2vec_ext.ecg_processing import ecgdataset
from word2vec_ext.ecg_processing import ecgdatasetsholder
from word2vec_ext.ecg_processing import beats2words
from word2vec_ext.word2vec import word2vecext
import spacy
import pytextrank
from collections import Counter
import uuid
from sklearn.metrics import classification_report
from sklearn.ensemble import GradientBoostingClassifier
from datetime import datetime


def text_rank_test(c_validation_words, validation_sentences_indices, labels):
    print(validation_sentences_indices)
    print(len(c_validation_words))
    print()
    print()

    for i in range(len(validation_sentences_indices)):
        if i == len(validation_sentences_indices) - 1:
            end_index = len(c_validation_words)
        else:
            end_index = validation_sentences_indices[i + 1]
        start_index = validation_sentences_indices[i]

        current_sentence_words = c_validation_words[start_index:end_index]
        current_sentence_labels = labels[start_index:end_index]
        current_sentence_labels_count = dict(Counter(current_sentence_labels))
        print("Text labels count: ", current_sentence_labels_count)

        current_sentence = '\n'.join(current_sentence_words)
        nlp = spacy.load("en_core_web_sm")
        nlp.add_pipe("textrank")
        doc = nlp(current_sentence)
        print("Text summary: ")
        phrase = doc._.phrases[0]
        word = phrase.text
        labels_of_word = [current_sentence_labels[i] for i, x in
enumerate(current_sentence_words) if x == word]
        estimated_labels_count = dict(Counter(labels_of_word))
        print("Word:", word, "; Related labels:", estimated_labels_count)
        print(phrase.rank, phrase.count)
        print()
        print()

        # Code below make no sense
        # print(current_sentence_labels_count)
        # real_relation = current_sentence_labels_count.get(0, 0) /
(current_sentence_labels_count.get(1, 0) + current_sentence_labels_count.get(0, 0))
        # estimated_relation = estimated_labels_count.get(0, 0) /
(estimated_labels_count.get(1, 0) + estimated_labels_count.get(0, 0))
        # print("Real relation:", real_relation, "Estimated relation:",
estimated_relation)
        # print("Error:", (estimated_relation - real_relation) / real_relation)
        # print()
        # print()
```

```
def main():
    rsc_dir = "../example_rsc"

    ecgdataset = ecgdatasetsholder.EcgDatasetsHolder.cache_from_mit(
        sets_count_limit=5,
        database_name="mitdb",
        mit_records_path=rsc_dir + "/mit_records",
        dataframe_path=rsc_dir + "/ecgdataset",
        annotator_type="symbol",
        reload=False
    )

    train_ready = ecgdataset.concatenate_datasets()

    train_ready.save(rsc_dir + "/full_dataset")

    beats2wordsModel = beats2words.Beats2Words()
    train_words =
beats2wordsModel.fit_and_predict_words(train_ready.dataframe["beats"].tolist(),
                                                        rsc_dir +
"/beats2words_test_for_5_datasets",
                                                        reset_cache=True)

    text_rank_test(train_words,
train_ready.train_start_indices["start_indices"].tolist(),
train_ready.dataframe["labels"].tolist())


if __name__ == '__main__':
    main()
```

ДОДАТОК В РЕЗУЛЬТАТИ ПЕРЕВІРКИ РОБОТИ НА СПІВПАДІННЯ



UNICHECK

Имя пользователя:
Лісовиченко Олег Іванович

ID проверки:
1009395258

Дата проверки:
29.11.2021 08:09:00 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
29.11.2021 08:11:21 EET

ID пользователя:
76913

Название файла: ІП-02мп_Язенок

Количество страниц: 53 Количество слов: 8713 Количество символов: 73785 Размер файла: 99.75 KB ID файла: 1009414611

0.63%
Совпадения

Наибольшее совпадение: 0.31% с источником из Библиотеки (ID файла: 1000052316)

0.13% Источники из Интернета 30 ..... Страница 55

0.63% Источники из Библиотеки 113 ..... Страница 55

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников