Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Системотехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Другий (магістерський)
(рівень вищої освіти)

Розробка та дослідження компонентів системи підтримки прийняття рішень для діагностики інфаркту міокарда
(тема)

Виконав: здобувач групи ІТПм-21-1
спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)
освітньої програми Інформаційні технології проектування
(повна назва освітньої програми)
Брехер Д. К.
(прізвище, ініціали)
Керівник проф. кафедри СТ Нечипоренко А.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри системотехніки _____ Гребеннік І. В.
(підпис) (прізвище, ініціали)

2022р.

Я як студент ХНУРЕ розумію і підтримую політику закладу із академічної доброчесності. Я не надавала і не одержувала недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

14.12.2022 Брехер Д. К.

Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.

Кваліфікаційна робота виконана у відповідності до стандартів, що діють в Україні.

Попередній захист проведено 14 грудня 2022 р.

Керівник кваліфікаційної роботи проф. Нечипоренко А.С.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Системотехніки

Освітньо-кваліфікаційний рівень Другий (магістерський)

Спеціальність 122 Комп'ютерні науки та інформаційні технології

(код і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри СТ Гребеннік І. В.
(підпис)
« » 2022 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЕКТ)

студентові Брехеру Денису Костянтиновичу
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) Розробка та дослідження компонентів системи підтримки прийняття рішень для діагностики інфаркту міокарда

затверджена наказом по університету від « » 2022 р. №

2. Термін подання студентом роботи (проекту) 20.12.2022

3. Вихідні дані до роботи (проекту): *Розробити компоненти системи підтримки прийняття рішень для діагностування інфаркту міокарду. Функції системи повинні забезпечувати: введення даних про пацієнта, виявлення інфаркту міокарда в пацієнта. Перелік програмних засобів, що були використані: ОС Microsoft Windows v.10, інтегроване середовище розробки PyCharm 2022.3.*

4. Зміст пояснювальної записки (перелік питань, що потрібно розробити)

4.1 Вступ 4.2 Аналіз предметної області 4.2.1 Проблеми виявлення інфаркту міокарда 4.2.2 Огляд та аналіз характеристик СППР для діагностування ІМ 4.2.3 Аналіз існуючих рішень 4.2.4 Постановка завдання 4.3 Розробка компонентів СППЛР 4.3.1 Визначення функціональних вимог до СППЛР для виявлення ІМ 4.3.2 Моделювання даних 4.3.3 Діаграма прецедентів СППЛР. 4.3.4 Визначення типу СПЛЛР. 4.3.4 Архітектурне проектування СППЛР. 4.3 Розробка моделі класифікації для виявлення ІМ. 4.3.1 Розробка декількох моделей класифікації для виявлення ІМ. 4.3.2 Результати порівняння моделей класифікації. 4.4 Практична реалізація розроблених компонентів СППЛР. 4.4.1 Обґрунтування вибору мови програмування. 4.4.2 Побудова та навчання нейронної мережі. 4.4.3. Реалізація веб-додатку СППЛР

5. Консультанти розділів роботи (проекту)

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
	проф. Нечипоренко А. С.		
	проф. Нечипоренко А. С.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів роботи	Примітка
1	Отримання завдання атестаційної роботи	01.10.22	
2	Аналіз завдання, літератури та аналогів з теми атестаційної роботи	02.10 – 17.10.22	
3	Вибір засобів для розробки технічних вимог до програми	18.10 – 28.10.22	
4	Структурне проектування	29.10 – 15.11.22	
5	Вибір середовища розробки програми	16.11 – 22.11 22	
6	Розробка програми	23.11 – 12.12.22	
7	Тестування програми	12.12 – 13.12.22	
8	Розробка «Посібника користувача»	14.12 – 15.12.22	
9	Оформлення пояснювальної записки та програмної документації	01.10 – 15.12.22	
10	Оформлення графічної частини та презентаційних матеріалів комп’ютерного захисту	за 5 днів	
11	Представлення на рецензування	за 3 дні	
12	Представлення атестаційної роботи в ДЕК	за 2 дні	

Студент _____ Брехер Д. К.
(підпис)

Керівник роботи _____ проф., д.т.н. Нечипоренко А.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра: 80 стор., 25 джерел, 5 фрагменти коду, 19 рисунків, 4 таблиці, 4 формули.

СИСТЕМИ ПІДТРИМКИ ПРЙНЯТТЯ РІШЕНЬ, ІНФАРКТ МІОКАРДУ, ШТУЧНИЙ ІНТЕЛЕКТ, WEB-ЗАСТОСУНОК, ІНТЕРФЕЙС, PYTHON.

Розробка та дослідження систем підтримки прийняття рішень для діагностування інфаркту міокарда.

Об’єкт дослідження – процес розробки та дослідження системи підтримки прийняття рішень для діагностування інфаркту міокарда.

Предмет дослідження – методи розробки та дослідження систем підтримки прийняття рішень для діагностування інфаркту міокарда.

Мета роботи – розробити та дослідити компоненти системи підтримки прийняття рішень для діагностування інфаркту міокарда, а саме модуль прийняття рішень та інтерфейс користувача.

Методи дослідження – системний підхід, методи структурного аналізу та теорії систем, методи машинного навчання для класифікації медичних даних. Мова програмування – Python.

Завдання, які вирішуються – розробка та дослідження системи підтримки прийняття рішень для діагностування інфаркту міокарда.

.

ABSTRACT

Certification work: 80 pages, 19 pictures, 4 tab., 5 add., 25 sources, 4 formulas.

DECISION SYPPORT SYSTEMS, MYOCARDIAL INFRACTION, ARTIFICIAL INTELLIGENCE, WEB-APPLICATION, INTERFACE, PYTHON.

The object is the development and research of decision support system for diagnosing myocardial infarction.

The subject of the research are methods of development and research of decision support systems for diagnosing myocardial infarction.

The purpose of the thesis is the development and research of components of decision support systems for diagnosing myocardial infarction.

Research methods – – system approach, methods of structural analysis and systems theory, machine learning methods for the classification of medical data. The programming language is Python.

The tasks to be solved are the development and research of decision support systems for diagnosing myocardial infarction.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧОК, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

СППР – система підтримки прийняття рішень;

СППЛР – система підтримки прийняття лікарських рішень;

ІМ – інфаркт міокарду;

MI (eng.) – myocardial infraction;

ЕКГ – електрокардіограма;

ПР – прийняття рішень;

МН – машине навчання;

БД – база даних;

СУБД – система управління базою даних;

АІС – автоматизована інформаційна система.

ВСТУП

Інженери в галузі інформаційних технологій дуже активно працюють над створенням та вдосконаленням комп'ютерних систем, які б допомагали людству швидше і якісніше вирішувати різного типу проблеми та задачі.

Один з прикладів таких комп'ютерних систем – це системи підтримки рішень. Їх існує велика кількість, в залежності від предметної області та конкретної задачі, для розв'язання якої вони використовуються.

Галузі медицини завжди приділяється багато уваги та зусиль, тому що подовження тривалості життя та підвищення якості здоров'я – це чи не найголовніші потреби нас, як біологічного виду.

Об'єктом кваліфікаційної роботи є процес виявлення інфаркту міокарда. Предметом кваліфікаційної роботи є методи класифікації даних для виявлення інфаркту міокарду та методи теорії систем для розробки компонентів СППР.

Галузь та специфіка майбутньої системи вказують на дуже високу актуальність роботи.

На даний момент вже існують подібні системи саме цього типу. Але будь-яку систему можна вдосконалити за рахунок підвищення точності в прогнозуванні за рахунок аналізу вхідних даних. Це і стало підставою для вибору теми кваліфікаційної роботи.

Наукова новизна одержаних результатів полягає в тому, що за рахунок запропонованих удосконалень нам вдалося зробити процес аналізу даних в системі більш ефективним.

Ми можемо наглядно переконатись в тому, що система підтримки прийняття рішень стали краще виконувати свою функцію. Таким чином ми з більшою вірогідністю можемо діагностувати інфаркт міокарду на основі вхідних даних. Це – що стосується практичної користі, яку ми отримали від результатів кваліфікаційної роботи.

Тож, для виконання кваліфікаційної роботи необхідно виконати на ступні задачі:

- провести аналіз стану питання виявлення ІМ та існуючих рішень;
- виявити недоліки існуючих СППР та врахувати їх під час розробки;
- розробити СППР, а саме модулі прийняття рішень та інтерфейс користувача;
- програмно реалізувати метод класифікації та інтерфейс користувача.

Створена система підтримки прийняття рішень може бути використана, як повноцінний інструмент для виявлення інфаркту міокарда.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблеми виявлення інфаркту міокарда

1.1.1 Загальний аналіз медичної складової

Інфаркт міокарда — крайній ступінь ішемічної хвороби серця, який характеризується розвитком ішемічного некрозу ділянки міокарда, що виник внаслідок недостатності кровопостачання у цій ділянці.

Гострий інфаркт міокарда визначають, користуючись клінічними, електрокардіографічними, біохімічними та патоморфологічними характеристиками. Визнано, що термін «гострий інфаркт міокарда» відображає смерть кардіоміоцитів, спричинену тривалою ішемією.

Для кращого розуміння наступного опису приводиться схематичне зображення циклу серцевого биття з визначенням його елемнтів:

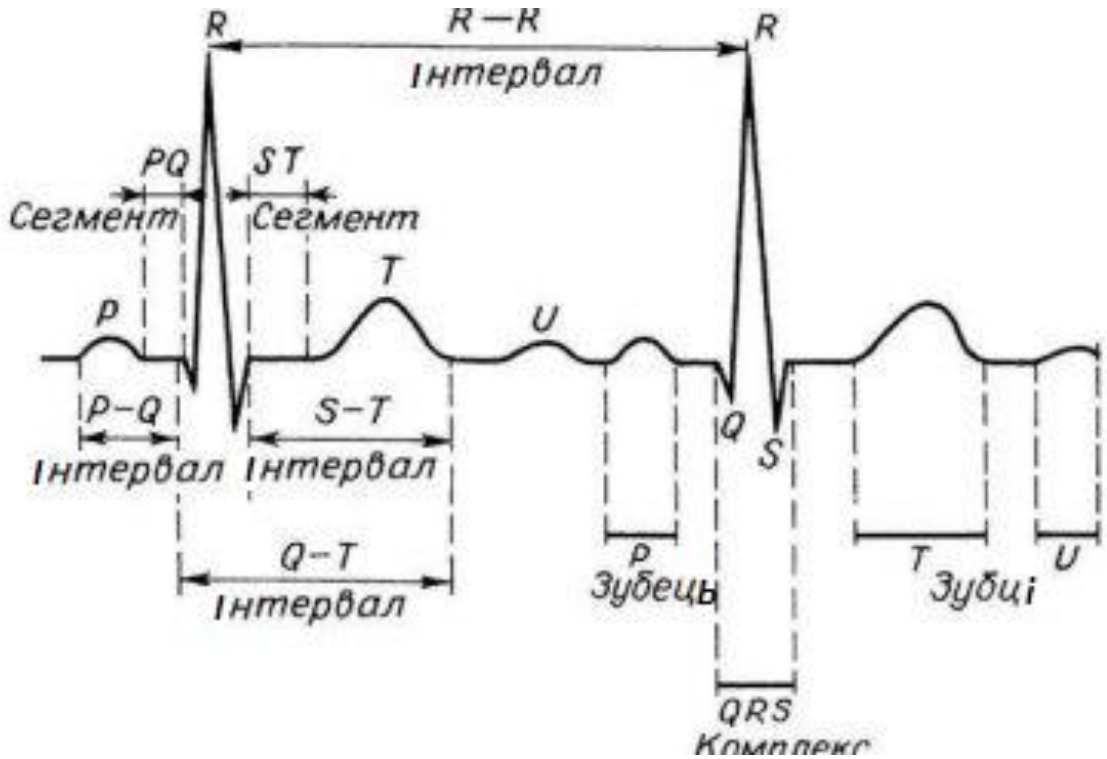

Рисунок 1.1 – Схематичне зображення циклу серцевого биття

На ЕКГ можна виявити ознаки ішемії міокарда – зміни ST та T, а також ознаки некрозу міокарда, зокрема конфігурації комплексу QRS.

Робоче визначення гострого прогресуючого ІМ (з елевацією сегмента ST) сформульовано наступним чином: пацієнти з наявністю больового синдрому (біль за грудиною, яка може іррадіювати в руки, під ліву лопатку, нижню щелепу, шию, спину, триває більше 20 хв, не купує нітрогліцерином), підвищенням сегмента ST >0,2 мВ у двох або більше суміжних прекардіальних відведеннях і >0,1 мВ в одному або більше дистантних відведеннях.

Формулювати діагноз слід у такій послідовності: причина розвитку гострого ІМ (наприклад, ІХС – ішемічна хвороба серця); раптова коронарна смерть із пожвавленням; гострий ІМ (з відповідними уточненнями); ускладнення ІМ (з відповідними уточненнями); наявність різних форм кардіосклерозу (при постінфарктному кардіосклерозі по можливості вказувати дату, глибину та локалізацію всіх попередніх ІМ).

1.1.2 Класифікація типів ІМ

Відповідно до МКХ-10 (міжнародної класифікації хвороб десятого перегляду) серед різновидів гострого ІМ виділяють:

- гострий ІМ з наявністю патологічного зубця Q;
- гострий ІМ без патологічного зубця Q;
- гострий ІМ (неуточнений – у разі ускладненої діагностики);
- рецидивуючий/ повторний ІМ;
- гостру коронарну недостатність (проміжний).

Патогенетично гострий ІМ із зубцем Q є етапом розвитку гострого ІМ, коли обсяг ураженого (некротизованого) міокарда вже значний, причому за амплітудою та тривалістю зубця Q можна побічно судити про глибину ураження міокарда, а за кількістю відведень з наявністю патологічного зубця Q – про його поширеність . Найчастіше гострий ІМ із зубцем Q діагностують при переході гострого ІМ із найгострішої фази в гостру і потім у підгостру. При пізній установці діагнозу у випадках, слід пам'ятати, що самі собою зубці Q можуть бути ознакою перенесеного раніше інфаркту.

Гострий ІМ без патологічного зубця Q відповідає поняттю «дрібноосередковий ІМ» і має на увазі ГКС, що завершився формуванням вогнища ураження (некрозу) міокарда, але все ж таки недостатньо великого (по глибині), щоб привести до формування патологічних зубців Q на ЕКГ. Як приклад наводять випадки, коли стійкі зміни на ЕКГ у вигляді негативних зубців Т відзначені у всіх грудних відведеннях, і при відповідній клініці та високому рівні ферментемії вогнище ураження можна розцінювати як поширене інтрамуральне, тоді як відсутність патологічних зубців Q відносить його до гострого ІМ без зубця Q».

Рецидивуючий та повторний ІМ відносять відповідно до тих випадків, коли після першого перенесеного гострого ІМ формується другий та більше. Причому терміни розвитку рецидиву гострого ІМ – від 3 до 28 діб з моменту розвитку вихідного інфаркту, а після закінчення цього терміну слід говорити про повторний ІМ. Якщо ЕКГ-діагностика розмірів та локалізації вогнища ураження утруднена, у діагнозі її вказувати не обов'язково.

Гостра коронарна недостатність відповідає міжнародному терміну «гострий коронарний синдром» і використовується як проміжний діагноз в ранні терміни захворювання. Постановка такого діагнозу заснована на виявленні елевації або депресії сегмента ST у поєднанні з тривалим (понад 20 хвилин) ангінозним болем.

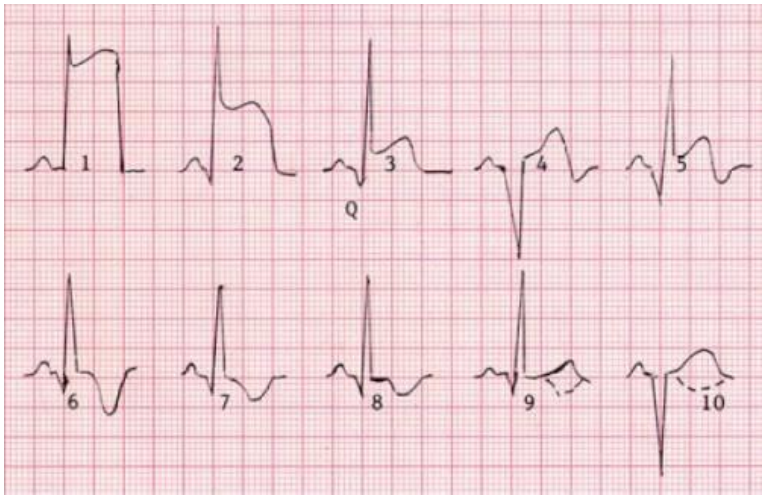

Рисунок 1.2 – Прояви гострого ІМ із зубцем Q на ЕКГ

Гострий інфаркт при типовому перебігу проходить 3 періоди розвитку. Кожен має свої прояви на ЕКГ:

– 1 та 2 вказують на гострий період;
– 3-9 ілюструє утворення зона некрозу та поступове рубцювання;
– 9 відображає повне відновлення;
– 10 констатує сформований рубець.

Також можна дізнатися, наскільки велика зона пошкодження серця. Про поширеність інфаркту судять щодо виявлення змін у відведеннях:

– дрібновогнищевий інфаркт проявляється лише негативним «коронарним» Т та зміщення інтервалу ST, патології R та Q не спостерігається;
– поширений інфаркт викликає зміни у всіх відведеннях.

Раніше до несприятливих факторів, що посилюють перебіг гострого ІМ, крім розмірів та локалізації інфаркту, традиційно відносили літній вік, жіночу стать, наявність супутнього цукрового діабету, інші соціальні, спадкові фактори та супутні захворювання. Тепер із появою нових медичних технологій структура чинників ризику змінилася: значний внесок як і ранній, і віддалений прогноз в хворих, які перенесли гострий ІМ, вносять терапія гострої фази захворювання, терміни звернення по медичну допомогу.

Таблиця 1.2 – Предиктор смерті після гострого ІМ, 5-річне спостереження

	Так, n (%)	Ні, n (%)	p
Чоловіча стать	196 (30,9)	161 (38,2)	<0,05
Тромболізис	54 (24,1)	219 (40,2)	<0,001
Куріння	147 (30,1)	115 (39,1)	<0,05
Сімейний анамнез ІХС	61 (21,5)	185 (39,9)	<0,001
ЛШН	184 (51,0)	110 (23,8)	<0,001
Реінфаркт	26 (60,0)	132 (17,0)	<0,001
Шлуночкова аритмія	64 (43,8)	229 (33,9)	<0,001

За результатами спостереження, проведеного Пітером МакГоверном у 1996 році випадків лікування гострого ІМ у реальній клінічній практиці сформовано список предикторів смерті після гострого ІМ протягом 5-річного спостереження (таб. 1.1).

Де n – кількість пацієнтів, що померли в стаціонарі, з наявністю або без наявності ознаки; % – відсоток пацієнтів, які померли в стаціонарі, серед пацієнтів з наявністю та без наявності ознаки; ЛШН – лівошлуночкова недостатність.

За даними таблиці до незалежних предикторів ранньої смерті після перенесеного гострого ІМ можна віднести наявність гострої лівошлуночкової недостатності, розвиток шлуночкових аритмій та періоду рецидиву.

На підставі отриманих даних побудовано як загальну криву виживання після перенесеного гострого ІМ (рис. 1.3).

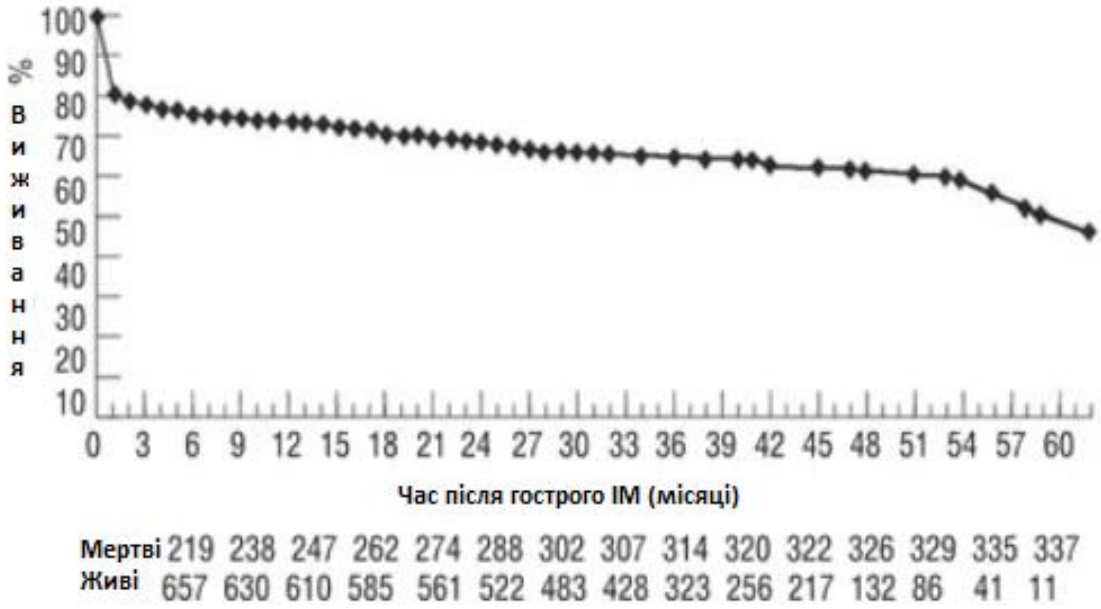

Рисунок 1.3 – Крива виживання після перенесеного гострого інфаркту міокарда

1.2 Огляд та аналіз характеристик СППР для діагностування ІМ

1.2.1 Загальний аналіз комп’ютерної складової

Система підтримки прийняття рішень (СППР) – автоматизована комп'ютерна система, метою якої є допомога людям, які приймають рішення у складних умовах для повного та об'єктивного аналізу предметної діяльності. Це означає, що вона видає інформацію, ґрунтуючись на вхідних даних, що допомагає людям швидко та точно оцінити ситуацію та ухвалити рішення. СППР виникли внаслідок злиття управлінських інформаційних систем та систем управління базами даних.

Для аналізу та вироблення пропозицій у СППР використовуються різні методи.

Це можуть бути:

- інформаційний пошук;
- інтелектуальний аналіз даних;
- пошук знань у базах даних;
- міркування на основі прецедентів;
- імітаційне моделювання;
- еволюційні обчислення
- генетичні алгоритми;
- нейронні мережі;
- ситуаційний аналіз.

Деякі з цих методів були розроблені у рамках штучного інтелекту. Якщо в основі роботи системи підтримки прийняття рішень лежать методи штучного інтелекту, то говорять про інтелектуалізовану систему підтримки прийняття рішень чи ІСППР.

Близькі до СППР класи систем – це експертні системи та автоматизовані системи управління.

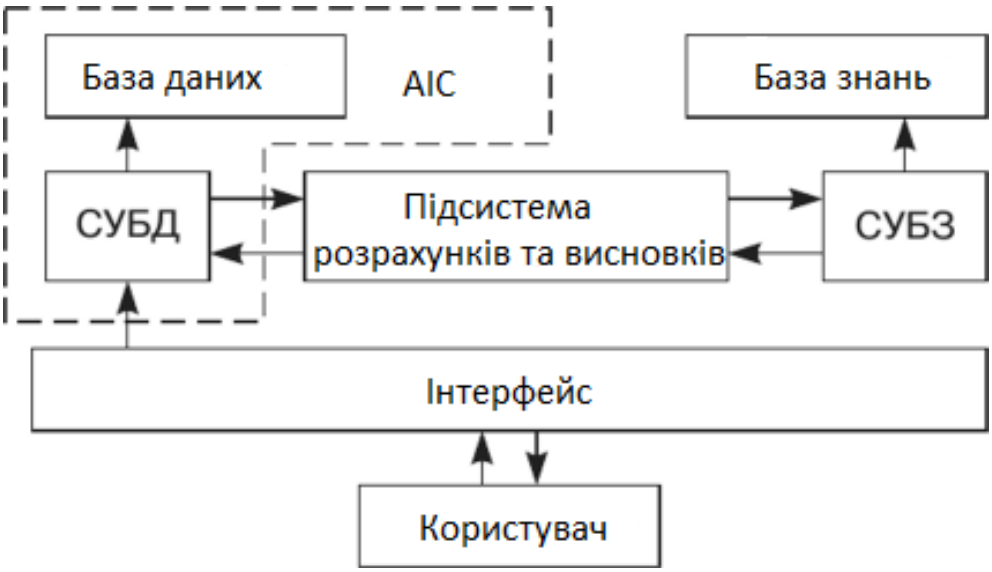

Рисунок 1.4 – Загальна структурна схема систем підтримки прийняття рішень

На загальній структурній схемі СППР видно, що вона складається з таких частин:

- база даних та система управління базою даних, які умовно об’єднуються в блок під назвою АІС (автоматизована інформаційна система);
- база знань;
- система управління базою знань;
- підсистема розрахунків та висновків;
- інтерфейс користувача;
- користувач.

При цьому не всі елементи пов’язані між собою. Центральними блоками системи виступають автоматизована інформаційна система та підсистема розрахунків та висновків. АІС виконує роль буферу даних.

Висновки, інтерпретовані вхідні дані та проміжні данні зберігаються в базі даних за допомогою системи управління базою даних.

Інтерфейс користувача інформацію саме з БД через СУБД. Підсистема розрахунків та висновків містить логіку обробки даних та формування висновків. Специфіка цієї низькорівневої логіки залежить від типу СППР. Також підсистема розрахунків та висновків взаємодіє з базою знать через систему управління базою знань, звідки отримує вхідні дані.

1.2.2 СППЛР як підвид СППР

За призначенням системи підтримки прийняття рішень варто класифікувати в залежності від галузі, в якій конкретна система використовуються. СППР, що створенні для розв’язанні будь-якого роду медичних задач називаються системами підтримки лікарняних рішень (СППЛР) та є окремим підвидом.

Система підтримки прийняття лікарських рішень – медична інформаційна система, призначена для допомоги лікарям та іншим медичним фахівцям у роботі із завданнями, пов'язаними з прийняттям клінічних рішень.

Розробка та впровадження СППЛР у практику належать до найголовніших напрямків розвитку інформаційних систем в області медицині.

У своїй практичній діяльності лікар стикається з великою кількістю завдань, які потребують швидкого та точного реагування. При постановці діагнозу та лікуванні необхідно врахувати:

– дані огляду;
– індивідуальні особливості пацієнта;
– результати лабораторних та інструментальних методів дослідження.

Рутинні операції — оформлення медичної документації, моніторинг стану пацієнтів, контроль за дотриманням призначень створюють додаткове навантаження на фахівців.

Ситуація ускладняється появою нових регламентів надання медичної допомоги: клінічних рекомендацій, стандартів та протоколів. \

При цьому важливо забезпечити своєчасність та безпеку клінічних заходів: вчасно виявити захворювання та розпочати правильне лікування. Часткова автоматизація лікувально-діагностичного процесу та інформаційна підтримка фахівця – напрямки, які покликані знизити навантаження на лікаря.

Помічником лікаря під час вирішення клінічних завдань може бути штучний інтелект, реалізований у СППЛР.

1.2.3 Огляд основних функцій СППЛР

Серед основних функцій, як виконують системі підтримки прийняття лікарських рішень відділяються наступні:

- довідково-інформаційна підтримка;
- допомога в оформленні медичної документації;
- визначення ступеня та тяжкості захворювання;
- генерація тривожних сигналів;
- оптимізація лікування;
- підтримка у діагностиці.

СППЛР, яка виконує функцію довідково-інформаційна підтримки забезпечує спеціаліста актуальними клінічними рекомендаціями та протоколами. Відомості про лікарські препарати допомагають уточнити аспекти терапії.

СППЛР, яка виконує функцію допомоги в оформленні медичної документації проводить сортування та облік електронних медичних карток. Хвороба пацієнта кодується за міжнародною класифікацією хвороб 10-го перегляду (МКХ-10). При постановці діагнозу лікар обирає відповідний код згідно з документацією.

СППЛР, яка виконує функцію визначення ступеня та тяжкості захворювання оцінює стан пацієнта, використовуючи класифікацію у межах машинного навчання. На виході лікар отримує висновок про групу ризику.

СППЛР, яка виконує функцію визначення ступеня та тяжкості захворювання здатна виявити приховані закономірності, які може помітити лікар. Алгоритми попереджають про можливі ускладнення, наприклад: підвищення рівня глюкози при цукровому діабеті, інфекції в післяопераційний період, декомпенсацію захворювання.

СППЛР, яка виконує функцію оптимізація лікування допомагає підібрати правильне лікування: призначити оптимальне дозування ліків, спрогнозувати тривалість перебування у стаціонарі провести моніторинг.

СППЛР, яка виконує функцію може робити висновки на основі вхідних даних: симптомів та скарг користувача. Система видає діагноз, який є орієнтиром для лікаря. Інший напрямок – діагностична візуалізація. СППВР розпізнає медичні зображення та виділяє підозрілі області.

Враховуючи предмет дослідження, сфокусуємося на останньому класі функцій – підтримка у діагностиці.

1.2.4 Огляд існуючих СППЛР на основі ШІ

Діагностика за допомогою СППЛР заснована на органічному поєднанні професійних навичок лікарів та технічних можливостей ШІ. Висновок штучного інтелеку має імовірнісний характер і є орієнтиром для клініциста. Система автоматизує рутинні процеси у медицині.

За підтримки Google розроблено алгоритм автоматичного виявлення діабетичної ретинопатії – ураження сітківки ока при цукровому діабеті. Як навчальні дані використовувалося 128 175 зображень сітківки.

Набори для перевірки клінічних умов включали 9963 і 1748 зображень відповідно. Офтальмологи оцінювали якість зображення та наявність на ньому патології зору. Їхні висновки порівнювалися з рішеннями нейромережі. За даними дослідників, чутливість для наборів склала 97,5% та 96,1%.

Рисунок 1.5 – Приклад вхідних даних для алгоритму виявлення діабетичної ретинопатії від Google

Сервіс Care Mentor AI використовує ШІ для підвищення точності висновків лікаря-рентгенолога. Експертна система аналізує результати променевих досліджень, щоб виявити ознаки захворювань органів грудної клітки.

Сервіс Care Mentor AI дозволяє:

– виявити ураження легеневої тканини, наявність стороннього тіла або порожнини з рівнем рідини;
– визначити зони патологічних змін;
– розрахувати розміри підозрілих утворень;
– сформувати підсумковий висновок.

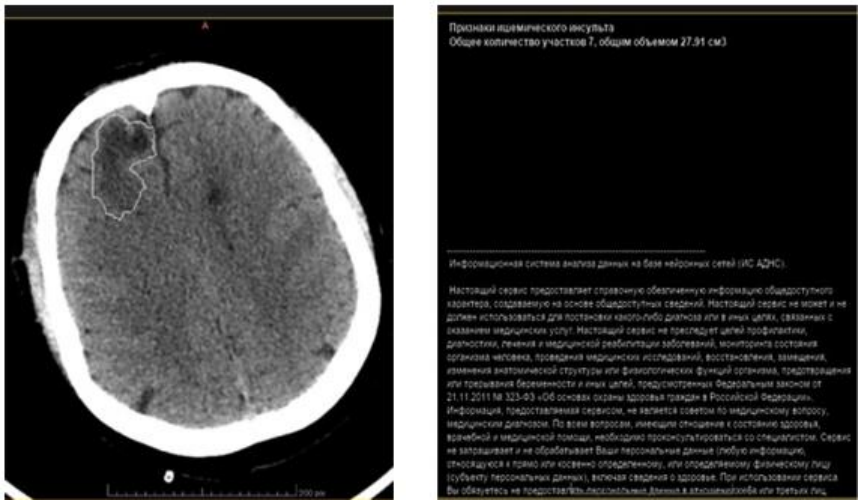

Рисунок 1.6 – Приклад роботи сервісу CT Stroke

Сервіс CT Stroke проводить на томограмах пошук областей із гострим порушенням мозкового кровообігу. Його основні функції:

– розмітка області та розміру ураження;
– автоматична розшифровка дослідження.

Приклад роботи сервісу CT Stroke зображено на рисунку 1.7.

Сервіс CT Lungs аналіз знімків органів грудної клітини (ОГК) спрямований на пошук ознак вірусної пневмонії, у тому числі природи COVID-19 та онкологічних захворювань. Сервіс виділяє уражені зони та мінімальні вузликові новоутворення у легеневій тканині.

Основні функції цього сервісу:

- розмітка області та ступеня ураження;
- виявлення наявності вірусної пневмонії;
- виявлення онкологічних змін.

Приклад роботи сервісу CT Lungs зображено на рисунку 1.8.

Рисунок 1.7 – Приклад роботи сервісу CT Lungs

1.2.5 Проблеми сучасних СППЛР

Багато медичних закладів та IT-компаній зробили значні зусилля з розробки адекватно функціонуючих СППЛР, що підтримують всі аспекти клінічних завдань. Однак, з урахуванням складності клінічного процесу та роботи медичного персоналу в умовах дефіциту часу, установа, що впроваджує у свою практику СППЛР, має зробити цю систему природною та невід'ємною частиною клінічного робочого процесу.

СППЛР, зосереджені на вирішенні діагностичних завдань, продемонстрували свою ефективність, однак, із значними обмеженнями щодо адаптації та сфери застосування. У 2011 році в госпіталі Університету Лідса запрацювала Система абдомінального болю Лідса. За даними аналізу, вона дозволяла встановити правильний діагноз у 87,8 % випадків, тоді як відсоток вірних діагнозів, поставлених лікарями, становив 79,6 %. Незважаючи на велику кількість спроб розробки та використання даних систем медичними установами, більшість СППЛР досі не набули широкого поширення.

Однією з основних труднощів на шляху впровадження систем історично є їхня інтеграція в робочий процес. До останнього часу існувала тенденція до зосередження лише на функціональному ядрі прийняття рішень у рамках СППЛР, що призводило до неефективного планування фактичного застосування препарату лікарями на робочому місці.

Найчастіше СППЛР являли собою окремі додатки, що вимагають переривання роботи лікаря в системі, що вже використовується, перемикання на СППЛР, введення необхідних даних (навіть якщо вони вже були раніше введені в іншу систему) і аналізу отриманих результатів. Додаткові дії порушують нормальний робочий процес та забирають у лікаря цінний час.

СППЛР у деяких галузях стикаються зі значними технічними проблемами. Біологічні системи дуже складні і клінічне рішення може вимагати аналізу величезного обсягу потенційно релевантних даних. Наприклад, електронна система, що працює на основі принципів доказової медицини, при формулюванні рекомендацій щодо плану лікування пацієнта потенційно може враховувати симптоми, медичний анамнез, сімейний анамнез та генетику пацієнта.

Ще одним негативним моментом, що стосується багатьох СППЛР, є велика кількість повідомлень, що генеруються ними. Коли системи видають велику кількість попереджень (особливо не передбачають подальших дій), це набридає лікарям і іноді призводить до того, що вони перестають приділяти попередженням достатньо уваги; останнє, своєю чергою, може потенційно призвести до ігнорування критично важливих повідомлень.

Підсумовуючи, можна виділити основні проблеми сучасних систем підтримки прийняття лікарських рішень:

- складнощі для лікарів в освоєнні;
- необхідність витрачати велику кількість часу для взаємодії з системою;
- складнощі в інтегруванні СППЛР в робочий процес;
- необхідність в постійному супроводженні та актуалізації системи.

Всі ці проблеми будуть враховані під час постановки завдання, розробки технічних вимог до майбутньої системи, а також реалізації самої СППЛР.

1.3 Аналіз існуючих рішень

У даному підрозділі проведемо аналіз існуючих рішень щодо діагностики інфаркту міокарда. Для цього необхідно розглянути:

- підходи, що використовуються для діагностування ІМ;
- системи підтримки прийняття рішень, що використовують алгоритми на основі підходів в своїй роботі.

Після цього ми матимемо технічну базу від якої можна буде відштовхуватись в пошуку недоліків і найголовніше – варіантів їхнього вдосконалення.

Головна проблема, з якою ми стикаємося на цьому етапі полягає в тому, що подібні СППЛР – це дуже специфічні системи. Вони розроблюються під замовлення певних медичних закладів і коштують великих грошей. Саме тому мало інформації реально знайти у відкритому доступі про ці системи і особливо про їхні методи реалізації.

На початку березня 2019 року компанії IBM і AstraZeneca представили нейромережу, яка може передбачити серцевий приступ. Результати роботи нової технології описані в опублікованій основі статті «Кластеризація результатів пацієнтів з гострим коронарним синдромом при використанні багатозадачної нейронної мережі» [1].

Команда дослідників зібрала дані про вік, анамнезу життя та захворювання, шкідливим привичкам, а також результати лабораторних досліджень, інформацію про проведення лікування та майже 40 інших показників серед 26 986 пацієнтів. Всі дані були завантажені у нейромережу, яка повинна була виявити, чи мав пацієнта у минулому серцево-судинне захворювання, а також чи отримував він антитромбоцитарні препарати, бета-блокатори та статини – препарати, що знижують прояви коронарної недостатності та для запобігання інфаркту міокарда.

Проте їхня робота демонструє, що кластерний аналіз на основі ШІ хоча і є перспективним підходом для класифікації пацієнтів з ІМ проте володіє низьким рівнем інтерпретації результатів.

У квітні 2017 року вчені з Університету Ноттінгема представили технологію штучного інтелекту, здатну передбачати настання серцевого нападу. Розробники стверджують, що точність прогнозування вища, ніж у лікарів [2].

У ході дослідження було зроблено порівняльний аналіз ефективності рекомендацій медиків із результатами чотирьох програмних засобів на основі алгоритмів машинного навчання. Вчені мали на меті знайти закономірності в записах понад 378 тис. пацієнтів.

Як бачимо точність не є основним недоліком вже існуючих СППЛР для дігностування ІМ. Зазвичай вони частіше ставлять правильний діагноз ніж лікарі. Хоча ми всеж таки можемо спробувати покращити і цю характеристику. Для цього нам також необхідно буде збільшити кількість даних для навчання моделі.

Однією із проблем є так звана комплексніть подібних систем, яка заважає її інтеграції в реальний процес. Щоб виправити це, система має бути максимально простою, наскільки це можливо.

1.4 Постановка завдання

Завданням кваліфікаційної роботи є розробка та дослідження компонентів системи підтримки прийняття рішення для діагностики інфаркту міокарда.

Користувач, маючі вхідні данні, що надаються пристроєм ЕКГ, повинен ввести деякі відомості про пацієнта та отримати від системи відповідь на питання чи є в пацієнта інфаркт міокарда, чи ні.

Система має містити блок прийняття рішень з використанням методів машинного навчання. Попередньо система має пройти навчання на основі великої кількості клінічних даних.

Точність класифікації СППЛР має бути вище 90%. При цьому система має бути максимально простою для користувача, та давати відповідь на основі клінічних даних пацієнта.

Таким чином, для розробки компонентів СППЛР необхідно виконати такі завдання:

- провести аналіз характеристик існуючих СППЛР;
- визначити функціональні вимоги до СППЛР для діагностики ІМ;
- розробити модуль прийняття рішень на основі методів машинного навчання;
- програмно реалізувати модель класифікації даних
- реалізувати інтерфейс користувача у вигляді веб-додатка.

2 РОЗРОБКА КОМПОНЕНТІВ СППЛР

2.1 Визначення функціональних вимог до СППЛР для виявлення ІМ

Для об'єктивної формалізації системи і визначення її кордонів необхідно скласти концептуальну модель даних. Дана модель визначає основні поняття і потоки інформації. Для побудови була використана методологія IDEF0, в рамках якої модель представлена у вигляді набору модулів, підпорядкованих один одному. У лівому верхньому кутку знаходиться більш важлива функція, далі розташовуються модулю за ступенем важливості, в яких відображаються вхідні, вихідні дані, механізми і способи управління модулями, їх логічна послідовність.

Контекстна діаграма показує головний бізнес-процес і пов'язані з нею потоки даних, механізми та інструкції.

Діаграма складається з головного функціонального блоку «Виявлення інфаркту міокарда» (рисунок 2.1).

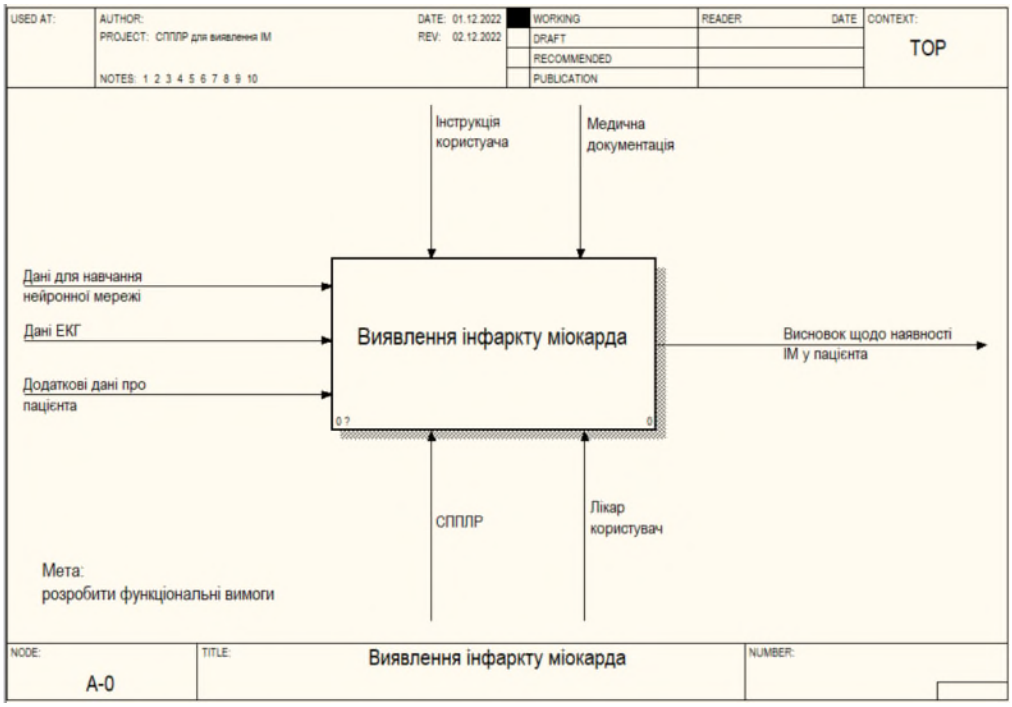

Рисунок 2.1 – Діаграма A-0 Контекстна діаграма

Стрілки поділяються на чотири види: вхідні дані, вихідні дані, дані для управління, механізми.

Зліва від блоку відображені вхідні дані. Праворуч від блоку відображені вихідні дані. Вгорі блоку відображені дані для управління. Дані управління включають в себе інструкція користувача, медична документація. Внизу блоку відображені механізми, за допомогою яких відбувається перетворення вхідних даних у вихідні дані. До механізмів відносяться: лікар користувач та СППЛР.

Діаграма першого рівня IDEF0 «Виявлення інфаркту міокарда» (рисунок 2.2) є декомпозицією контекстної діаграми IDEF0 і складається з 4 функціональних блоків:

– навчання нейронної мережі штучного інтелекту;
– надання інформації про пацієнта;
– класифікація випадка;
– верифікація висновка СППЛР лікарем.

Функціональний блок «надання інформації про пацієнта» на виході має готові дані для аналізу, які передаються в блок «класифікація випадка». Звідти висновок йде на вихід, а також в блок «верифікація висновка».

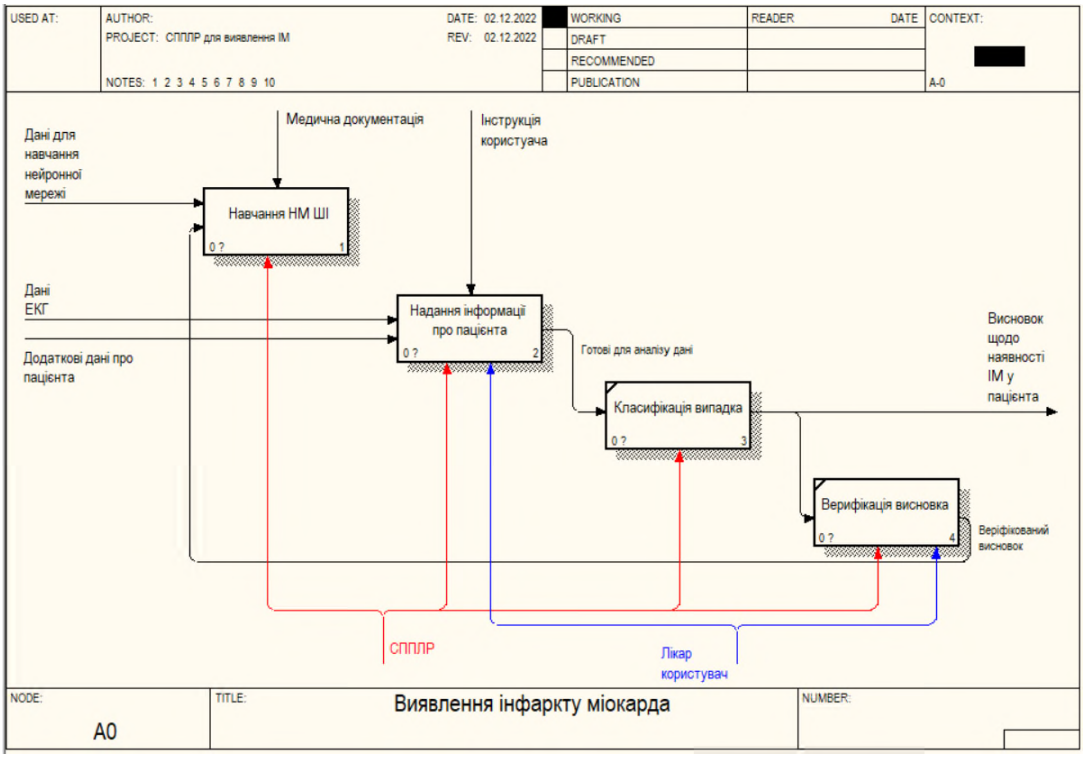

Рисунок 2.2 – Декомпозиція контекстної діаграми А0

Діаграма другого рівня «Навчання НМ ШІ» (рисунок 2.3) складається з 4 функціональних рівнів:

– вибір одиниці даних;
– пропуск сигналу через мережу;
– розрахунок помилки;
– перерахунок ваг.

Блок «вибір одиниці даних» бере один з записів в даних, які є на вході, а саме: дані для навчання нейронної мережі та верифіковані лікарем висновки, видаючи на виході окремий блок даних. Це блок пропускаються через мережу в «пропуск сигналу через мережу». На виході отримуємо висновок, який шляхом розрахунку помилки проходить перевірку. Якщо рівень помилки допустимий, висновок йде на виді, якщо ні – використовуються для перерахунку ваг нейронної мережі штучного інтелекту, яка використовуються в системі підтримки прийняття рішень для діагностування інфаркту міокарда.

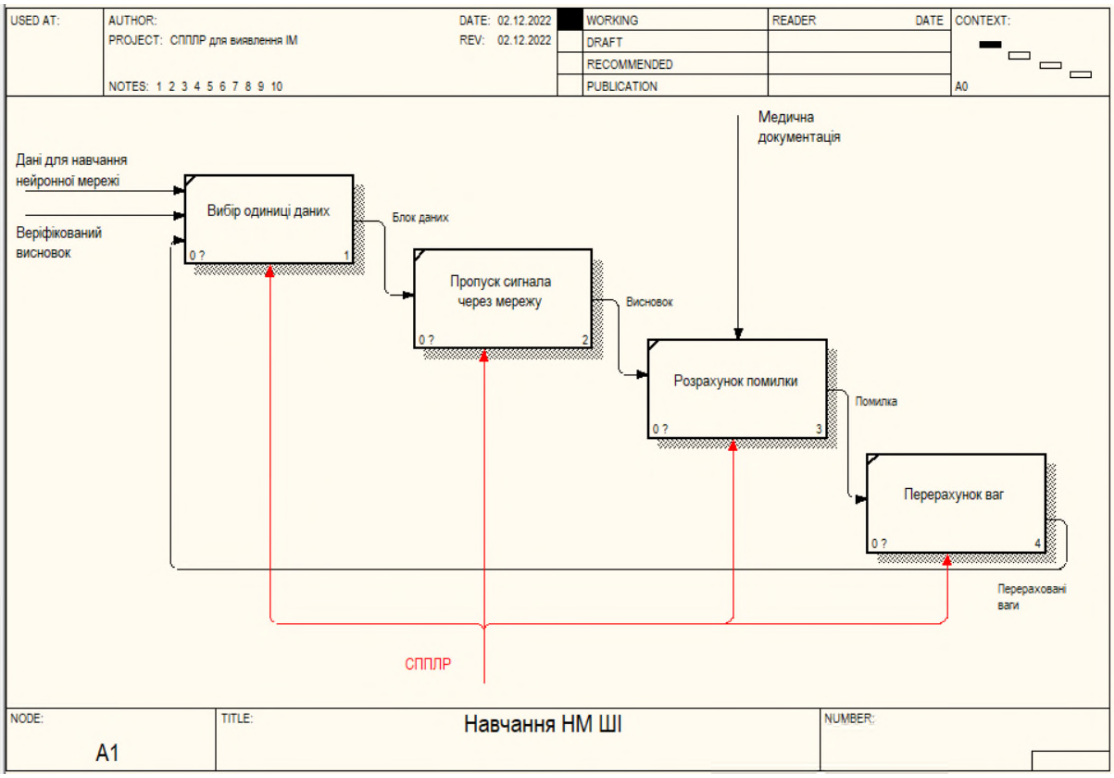

Рисунок 2.3 – Декомпозиція «Навчання НМ ШІ» А1

Перераховані ваги направляються на вхід для повторення циклу навчання нейронної мережі.

Діаграма другого рівня «Надання інформації про пацієнта» (рисунок 2.4) складається з 2 функціональних рівнів:

– введення даних про пацієнта в систему;
– валідація даних.

Після введення лікарем даних про пацієнта в систему вони об’єднуються з даними отриманими від апарату ЕКГ та проходять валідацію. Під час цього процесу дані перевіряються на коректність та очищуються.

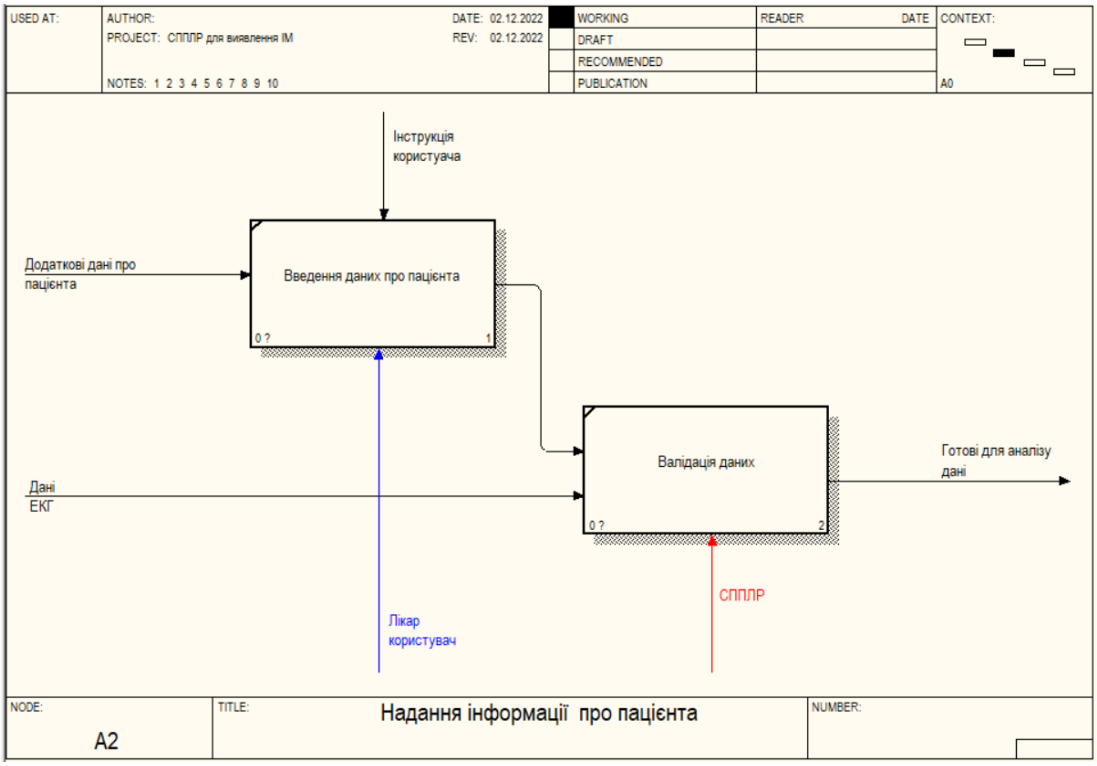

Рисунок 2.4 – Декомпозиція «Надання інформації про пацієнта» A1

Звідси готові для аналізу дані потравляють в блок класифікації випадку. На виході маємо висновок, який отримує лікар користувач, щодо наявності інфаркту міокарда в пацієнта.

Також це висновок має спочатку пройти верифікацію лікаря. А потім відправитись до інших даних для навчання для підвищення ефективності роботи системи підтримки прийняття рішень для виявлення інфаркту міокарда.

2.2 Опис вихідних даних для блоку прийняття рішень

Основою блоку прийняття рішень системи підтримки прийняття рішень що розроблюється є штучна нейронна мережа. Вона обробляє вхідні дані з метою виявлення в пацієнта інфаркт міокарда. Запорукою ефективності роботи НМ є попередньо оброблений набір даних для навчання моделі.

Електрокардіографія (ЕКГ) є основним діагностичним інструментом для оцінки стану серцевої функції пацієнта. СППЛР які містять функціонал для автоматичної інтерпретації ЕКГ обіцяють значне полегшення для медичного персоналу. Однак розробка таких систем вимагає великих навчальних наборів даних і чітких процедур тестування.

Необроблені дані сигналу ЕКГ повинні бути збережені у форматі .dat. Для всіх сигналів ми надаємо стандартний набір з 12 відведень (I, II, III, AVL, AVR, AVF, V1, ..., V6) з електродами порівняння на правій руці.

Також наш датасет має містити відповідні метадані в файлі типу .csv (таблиця 2.1).

Таблиця 2.1– Метадані

Поле	Тип даних	Опис
ecg_id	Integer	Ідентифікатор кардіограми
patient_id	Integer	Ідентифікатор пацієнта
age	Integer	Вік
sex	String	Стать
height	Float	Зріст
weight	Float	Вага
device	String	Пристрій запису
recording_date	Date	Дата запису
report	String	Звіт
scp_codes	String	Стандартний протокол зв'язку ЕКГ

heart_axis	Float	Вісь серця
validated_by	String	Ім’я того, хто перевірив
filename	String	Назва файла з даними про сигнал ЕКГ

Метадані містять 11 полів та можуть бути розділені на наступні декілька типів:

– ідентифікатори: ecg_id, patient_id, filename;

– загальні метадані: age, sex, height, weight, device, recording і т.д.;

– виписки ЕКГ: scp_codes, heart_axis.

На всяк випадок, бажано, щоб сигнал ЕКГ дублювався на дані з частотою дискретизації 100 Гц (records100), та на дані з частотою дискретизації 500 Гц (records500).

Загальна структура датасету, який буде використано для навчання НИ ШІ зображена на рисунках 2.5 та 2.6.

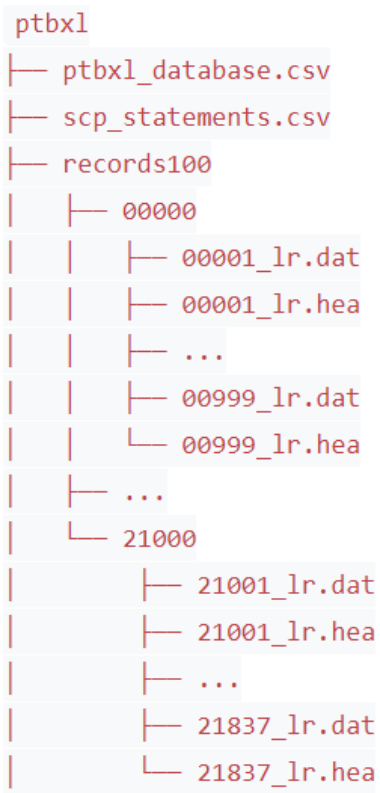

Рисунок 2.5 – Структура датасету

Рисунок 2.6 – Структура датасету

2.3 Діаграма прецедентів (Use Case Diagram) СППЛР

UML — уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Ми використали їх для зручності проектування системи. UML є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML — це мовою широкого профілю, а також відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи. Мета UML — визначення, візуалізація, проектування й документування програмних систем. UML — це не мова програмування, але можлива генерація коду.

Описання проводиться с точки зору чинного особи, групу дій в системі, які призводять до конкретного результату. Варіанти використання є описами типових взаємодій між користувачами системи і самою системою. Вони відображають зовнішній інтерфейс системи і вказують форму того, що система повинна зробити (саме що, а не як).

Діаграми використовується для опису взаємини і залежності між групами варіантів використання і дійових осіб, які беруть участь в процесі. Діаграми варіантів використання не призначені для відображення проекту і не можуть описувати внутрішній устрій системи, вони призначені для спрощення взаємодії з майбутніми користувачами системи, з клієнтами, і для визначення необхідних характеристик системи. Тобто, діаграми варіантів використання говорять про те, що система повинна робити, без вказівок методів.

Як було виявлено під час аналізу, для максимально ефективного впровадження системи в робочий процес діагностування інфаркту міокарда, вона має бути максимально простою та виконувати тільки основну функцію – виявлення ІМ в пацієнта виходячи з показників ЕКГ та додаткової інформації про нього, введеної лікарем користувачем.

Нижче (рисунок 2.7) схематично зображений функціонал, який буде доступний в системі підтримки прийняття рішень для виявлення інфаркту міокарда.

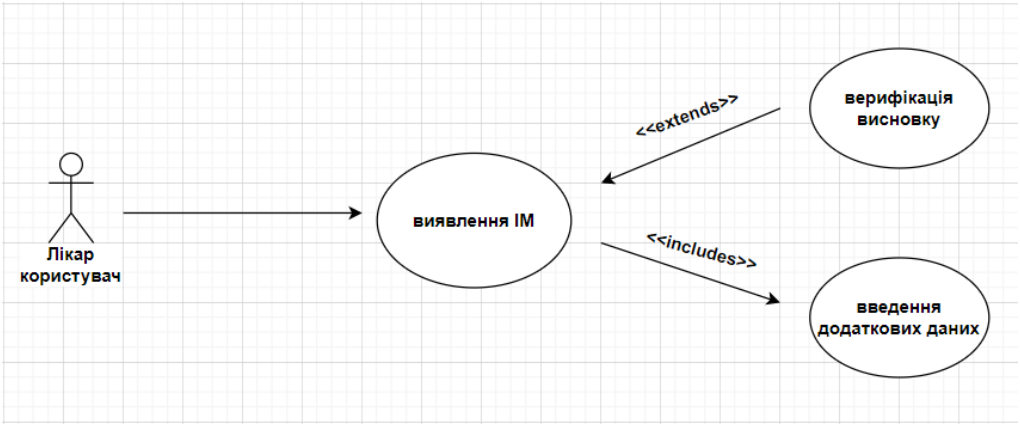

Рисунок 2.7 – Діаграма прецедентів СППЛР

Тож, в нашій системі буде тільки один тип користувача. Йому буде доступні таки прецеденти:

– виявлення інфаркту міокарда;
– верифікація висновку;
– введення додаткової інформації про пацієнта.

Важливо, що введення додаткової інформації про пацієнта є невід’ємною частиною виявлення інфаркту міокарда. Тоді як верифікація висновку – додаткова опціональна дія.

2.4 Визначення типу СПЛЛР

За типом функціоналу, закладеного в системи підтримки прийняття лікарських рішень вони поділяються на дві групи:

– СППЛР на основі баз знань;

– СППЛР не на основі баз знань;

Серед усіх СППВР більшість – це системи на основі баз знань. Вони складаються із трьох частин:

– інформаційної бази;

– механізму логічних висновків;

– та механізму комунікації.

Інформаційна база містить правила та зв'язки даних мета аналізу, які найчастіше набувають форми правил ЯКЩО-ТО. Якщо це система визначення лікарських взаємодій, правило може мати такий вигляд: ЯКЩО призначений препарат «X» І призначений препарат «Y», ТО попередити користувача. При використанні іншого інтерфейсу досвідчений користувач може редагувати інформаційну базу підтримки її актуальності з урахуванням появи нових лікарських засобів. Механізм логічних висновків поєднує правила з інформаційної бази з даними пацієнта. Механізм комунікації дозволяє системі представити результати користувачеві та забезпечує введення даних у систему.

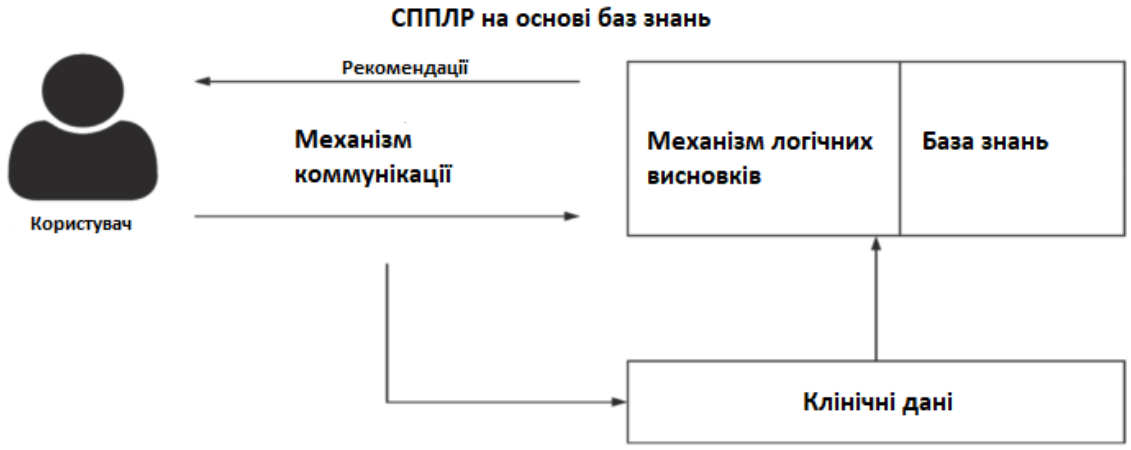

Рисунок 2.8 – Загальна структурна схема СППЛР на основі баз знань

СППЛР, які не використовують наукові медичні знання, експлуатують форму штучного інтелекту, відому під назвою «машинне навчання», яка дозволяє комп'ютерним системам навчатися на основі отриманого досвіду.

Також подібні системи можуть встановлювати закономірності в межах масиву клінічних даних. Зазначене усуває необхідність написання правил та експертного введення.

Однак, оскільки системи, засновані на «машинному навчанні», не завжди володіють достатнім рівнем інтерпретації отриманих результатів, деякі клініцисти не використовують їх безпосередньо для постановки діагнозів через невпевненість у точності та достовірності результатів.

Тим не менш, такі системи можуть бути корисними для використання в постдіагностичному періоді, розкриваючи перед лікарями певні закономірності для більш глибокого аналізу.

Виділяють три види систем підтримки прийняття лікарських рішень не на основі баз знань:

– машини опорних векторів;
– штучні нейронні мережі;
– генетичні алгоритми.

Рисунок 2.9 – Загальна структурна схема СППЛР не на основі баз знань

Штучна нейронна мережа використовує вузли та зважені зв'язки між ними для аналізу закономірностей у масиві даних пацієнтів з метою встановлення асоціацій між симптомами та діагнозами.

Генетичні алгоритми ґрунтуються на спрощених еволюційних процесах з використанням спрямованого відбору для досягнення оптимальних результатів роботи СППЛР. Алгоритми відбору оцінюють компоненти випадкових наборів розв'язання проблеми. Рішення, що потрапляють вгору списку, рекомбінуються і видозмінюються, після чого процес повторюється. Це відбувається знову і знову доти, доки виявляється потрібне рішення.

Генетичні алгоритми функціонально подібні до нейронних мереж у тому плані, що також є «темними конячками», які намагаються витягти інформацію з масиву даних пацієнтів. Мережі на основі ШІ часто зосереджуються на обмеженому наборі симптомів (наприклад, на симптомах одного захворювання), на відміну від таких, що базуються на інформації та дозволяють діагностувати різні захворювання.

Для майбутньої системи була обрана архітектура не на основі баз знань. В якості механізму пошуку відповіді буде використана нейрона мережа штучного інтелекту.

2.5 Проектування архітектури СППЛР

Взагалі, немає загальноприйнятого терміну «архітектура програмного забезпечення». Проте коли справа стосується практики, то для більшості розробників і так зрозуміло який код є хорошим, а який поганим. Хороша архітектура – це архітектура, що робить процес розробки та супроводу програми більш простим та ефективним. Програму з гарною архітектурою легше розширювати та змінювати, а також тестувати, налагоджувати та розуміти.

Система повинна вирішувати поставлені завдання та добре виконувати свої функції. Сюди можна віднести такі характеристики, як надійність, безпека, продуктивність, здатність справлятися зі збільшенням навантаження (масштабованість) тощо.

Будь-який додаток доводиться змінювати з часом, тому що змінюються вимоги, додаються нові. Чим швидше і зручніше можна ввести зміни до існуючого функціоналу, чим менше проблем і помилок це викличе – тим краще. Тому в процесі розробки намагайтеся оцінювати те, що виходить, щодо того, як вам це потім, можливо, доведеться змінювати. Зміна одного фрагмента системи має впливати на її інші фрагменти. Гарна архітектура дозволяє відкладати прийняття ключових рішень і мінімізує ціну помилок.

Можливість додавати до системи нові сутності та функції, не порушуючи її основної структури. На початковому етапі в систему має сенс закладати лише основний та найнеобхідніший функціонал (принцип YAGNI – you ain't gonna need it, «Вам це не знадобиться») Але при цьому архітектура повинна дозволяти легко нарощувати додатковий функціонал у міру потреби. Причому так, щоб внесення найімовірніших змін вимагало найменших зусиль.

Вимога, щоб архітектура системи мала гнучкість і розширюваність (тобто була здатна до змін та еволюції) є настільки важливою, що вона навіть сформульована у вигляді окремого принципу – «Принципу відкритості/закритості» (Open-Closed Principle – другий із п'яти принципів SOLID): Програмні сутності (класи, модулі, функції тощо) повинні бути відкритими для розширення, але закритими для модифікації.

Інші принципи SOLID зазначені в таблиці 2.2.

Таблиця 2.2– Принципи SOLID

Ініціал	Принцип
S	Принцип єдиної відповідальності (Single Responsibility Principle)
O	Принцип відкритості-закритості (Open closed Principle)
L	Принцип підстановки Барбари Лисков (Liskov substitution Principle)
I	Принцип поділу інтерфейсів (Interface Segregation Principle)
D	Принцип інверсії залежностей (Dependency Inversion Principle)

Можливість скоротити термін розробки рахунок додавання до проекту нових людей. Архітектура повинна дозволяти розпаралелити процес розробки, щоб багато людей могли працювати над програмою одночасно.

Код, який легше тестувати, міститиме менше помилок та надійніше працюватиме. Але тести не лише покращують якість коду. Багато розробників приходять до висновку, що вимога «хорошої тестованості» є також спрямовуючою силою, що автоматично веде до гарного дизайну.

Систему бажано проектувати так, щоб її фрагменти можна було використовувати повторно в інших системах.

Над програмою, як правило, працює багато людей – одні йдуть, приходять нові, яким доводиться супроводжувати програму. Система має бути добре структурованою, не містити дублювання, мати добре оформлений код та бажано документацію.

Таким чином, можна сформулювати список критеріїв, яким буде відповідати майбутня система підтримки прийняття рішень для виявлення ІМ:

- ефективність системи;
- гнучкість системи;
- розширюваність системи;
- масштабованість процесу розробки;
- можливість ретельно протестувати;
- можливість повторного використання;
- добре структурований та зрозумілий код/супроводжуваність.

3 РОЗРОБКА МОДЕЛІ КЛАСИФІКАЦІЇ ДЛЯ ВИЯВЛЕННЯ ІМ

3.1 Розробка моделей класифікації для виявлення ІМ

Як основу блоку прийняття рішень СППЛР, досліджено чотири моделі на основі методів машинного навчання для класифікації пацієнтів з ІМ. Моделі машинного навчання базуються на класифікаторі k-найближчих сусідів, радіальній базисній функції, дереві рішень і випадковому лісі.

3.1.1 K-Nearest Neighbor Classifier

Принцип, що лежить в основі методу k-nearest neighbor classifier, полягає в тому, щоб знайти заздалегідь визначену кількість навчальних вибірок, найближчих за відстанню до нової точки, і надати значення для даних. Незважаючи на свою простоту, метод k-nearest neighbor classifier досяг успіху в багатьох проблемах класифікації та регресії, включаючи медичну область. Будучи непараметричним методом, він часто є успішним у ситуаціях класифікації, коли межа рішення нечітка.

Евклідова відстань є загальновживаною метрикою відстані для безперервних змінних. Для дискретних змінних, таких як класифікація тексту, ви можете використовувати іншу метрику, таку як метрика перекриття (або відстань Хеммінга). Крім того, k-nearest neighbor classifier використовується із такими коефіцієнтами кореляції, як Пірсона та Спірмена. Часто точність класифікації можна значно підвищити, якщо вимірювати відстань за допомогою спеціалізованих алгоритмів, таких як аналіз компонентів високої маржі, найближчого сусіда або сусідства.

Недоліком первинної класифікації більшості голосів є спотворення розподілу класів. Частіші класові приклади, як правило, домінують у передбаченні нового прикладу, оскільки вони, як правило, поширюються серед найближчих сусідів через їхню велику кількість.

Одним із способів подолання цієї проблеми є зважування класифікації з урахуванням відстані від контрольної точки до кожного з її найближчих сусідів.

Клас (або значення в задачах регресії) кожної з k найближчих точок множиться на вагу, пропорційну зворотній відстані від цієї точки до контрольної точки. Ще один спосіб подолати перекіс – абстрагувати представлення даних.

Для класифікації об’єктів досліджуваної вибірки необхідно послідовно виконати такі дії:

– розрахувати відстань до кожного з об'єктів навчальної вибірки;
– вибрати об'єкт навчальної вибірки, відстань до якого мінімальна;
– визначити клас, який найчастіше зустрічається серед k найближчих сусідів.

Евклідова відстань у багатовимірному просторі ознак обчислюється наступним чином:

(3.1)

$$d_{ab} = \sqrt{\sum_{i=1}^{n} (x_{ai} - x_{bi})^2}$$

Де a і b – точки в n-вимірному просторі, i – порядковий номер ознаки, x_{ai} та x_{bi} – координати точок a і b за ознакою i.

Клас з найбільшою кількістю голосів призначається новому елементу:

(3.2)

$$y_a(a, X, k) = argmax_{y \in Y} \sum_{i=1}^{k} (y_a^i = y)$$

Де a – новий елемент, X – навчальна вибірка, y – це клас, Y – набір класів, y_a^i – це клас i-го сусіда, k – кількість сусідів.

3.1.2 Radial Basis Function

У машинному навчанні радіальна базисна функція використовується в різних алгоритмах навчання ядра. Зокрема, він зазвичай використовується для класифікації опорних векторних машин. Ядро радіальної базисної функції на двох вибірках x і x′, представлених як вектори ознак у деякому вхідному просторі, визначається як:

(3.3)

$$K(x, x') = \exp(\frac{-\|x - x'\|^2}{2\sigma^2})$$

Де $\|x-x'\|_2$ можна визначити як квадрат евклідової відстані між двома векторами ознак, σ – вільний параметр.

Еквівалентне визначення включає параметр $\gamma=12\sigma2$:

(3.3)

$$K(x, x') = \exp(-\gamma\|x - x'\|^2)$$

Оскільки значення ядра RBF зменшується з відстанню і коливається від нуля (на межі) до одиниці (коли x = x′), воно має готову інтерпретацію як міра подібності. Простір функцій ядра має нескінченну кількість вимірів; при σ = 1 зростає:

$$exp\left(\frac{-1}{2}\|x - x'\|^2\right) = exp\left(\frac{2}{2}x^T x' - \frac{1}{2}\|x\|^2 - \frac{1}{2}\|x'\|^2\right) =$$
$$exp\left(x^T x'\right) exp\left(\frac{-1}{2}\|x\|^2\right) exp\left(\frac{-1}{2}\|x'\|^2\right) =$$
$$\sum_{j=0}^{\infty} \frac{\left(x^T x'\right)}{j!} exp\left(\frac{-1}{2}\|x\|^2\right) exp\left(\frac{-1}{2}\|x'\|^2\right) =$$
$$\sum_{j=0}^{\infty} \sum_{\sum n_i = j} \frac{\left(x^T x'\right)}{j!} exp\left(\frac{-1}{2}\|x\|^2\right) exp\left(\frac{-1}{2}\|x'\|^2\right)$$

Рисунок 3.1 – Обчислення простору функцій ядра

3.1.3 Decision Tree

Decision Tree – це непараметрична методика навчання під наглядом, яка використовується для класифікації та регресії. Мета полягає в тому, щоб створити модель, яка передбачає значення цільової змінної шляхом вивчення простих правил прийняття рішень, отриманих на основі характеристик даних. Дерево можна розглядати як постійне наближення на певну відстань.

Дерева рішень навчаються на даних для апроксимації синусоїди за допомогою набору правил прийняття рішень if-then-else. Чим глибше дерево, тим складніші правила прийняття рішень і краща модель. Переваги дерев рішень:

- легко зрозуміти та інтерпретувати;
- дерева можна візуалізувати;
- вимагає невеликої підготовки даних;
- вага використання дерева – це логарифмічне число точок даних, які використовуються для навчання дерева.

До недоліків дерев рішень належать:

- навчені моделі рішень можуть створювати дуже складні дерева;
- дерева рішень можуть бути нестабільними, оскільки незначні зміни в даних можуть призвести до зовсім іншого дерева.

Щоб уникнути першої проблеми, потрібні такі механізми, як обрізка, встановлення мінімальної кількості зразків, необхідних у листовому вузлі, або встановлення максимальної глибини дерева Використання ансамблевих дерев рішень пом’якшує другу проблему.

Проблема вивчення оптимального дерева рішень є NP-повною в кількох аспектах оптимуму, навіть для простих концепцій. Таким чином, практичні алгоритми навчання дерева рішень базуються на евристичних алгоритмах, таких як жадібний алгоритм, де локально оптимальні рішення приймаються на кожному вузлі. Такі алгоритми не можуть гарантувати повернення глобально оптимального дерева рішень. Це можна пом’якшити, навчивши кілька дерев в ансамблі, де функції та зразки відбираються випадковим чином із заміною.

3.1.4 Random Forest

Random Forest – це тип керованого алгоритму машинного навчання на основі ансамблевого навчання. Ансамблеве навчання – це тип навчання, у якому ви комбінуєте різні типи алгоритмів або той самий алгоритм кілька разів, щоб сформувати потужнішу модель прогнозування. Алгоритм випадкового лісу поєднує кілька алгоритмів одного типу, тобто кілька дерев рішень, у результаті чого утворюється ліс дерев, звідси й назва Випадковий ліс. Алгоритм випадкового лісу можна використовувати як для задач регресії, так і для класифікації.

Ці два джерела випадковості спрямовані на зменшення дисперсії оцінки лісу. Окремі дерева рішень зазвичай демонструють високу дисперсію та мають тенденцію до переповнення. Введена випадковість у лісах дає дерева рішень із дещо ізольованими помилками передбачення. Взявши середнє значення цих прогнозів, можна уникнути деяких помилок. Випадкові ліси досягають зменшення дисперсії шляхом комбінування різноманітних дерев, іноді ціною незначного збільшення зміщення. На практиці зменшення дисперсії часто є значним, що дає загальну кращу модель.

Алгоритм випадкового лісу не є упередженим, оскільки існує кілька дерев, і кожне дерево навчається з підмножини даних. Алгоритм випадкового лісу покладається на силу «натовпу», загальне зміщення алгоритму зменшується.

Алгоритм є стабільним. Навіть якщо в набір даних буде введено нову точку даних, це істотно не вплине на загальний алгоритм, оскільки нові дані можуть вплинути на одне дерево. Проте вразити всі дерева складно.

Також варто зазначити, що алгоритм випадкового лісу добре працює, якщо вибірка містить як категоричні, так і числові ознаки. Алгоритм випадкового лісу також добре працює, коли дані відсутні або не були добре масштабовані.

Основним недоліком випадкового лісу є його складність. Модель потребує набагато більше обчислювальних ресурсів завдяки великій кількості об’єднаних дерев рішень. Через їхню складність вони потребують набагато більше часу для навчання інших подібних алгоритмів.

3.2 Результати порівняння моделей класифікації для виявлення ІМ

Після попереднього аналізу та підготовки даних ми розділили наші дані на дані навчання та перевірки. Ми хочемо надати моделі якомога більше навчальних даних. Однак ми також хочемо переконатися, що маємо достатньо даних для тестування моделі. Оскільки кількість рядків у наборі даних збільшується, ми можемо надавати більше даних для навчального набору. Інший важливий параметр – змішування даних.

В рамках дослідженні набір було розподілено у співвідношенні 80% до 20% для даних навчання та перевірки. Для тестування з цим набором даних було визначено набір класифікаторів регресії та моделей машинного навчання. Для перевірки достовірності результату були проведені тести статистичної значущості. Для цього ми оцінили модель 10 разів і отримали середні значення точності та середньоквадратичної помилки.

Результати розроблених моделей машинного навчання наведено в таблиці 3.1.

Таблиця 3.1– Результати розроблених моделей

Модель	Точність	Середньоквадратична помилка
K-nearest neighbors classifier	71.105%	0.289
Radial basis function	75.408%	0.245
Decision tree	89.867%	0.109
Random forest	97.774%	0.022

Класифікатор Random Forest навчається за допомогою агрегації навантажень, де кожне нове дерево вибирається із вибірки спостережень за навантаженнями. Out of bag – це середня помилка для кожного обчисленого з використанням передбачень дерева, які не містяться у вибірці навантаження.

Це дозволяє адаптувати та перевіряти побудовану модель випадкового лісу під час навчання.

Основними параметрами, які потрібно налаштувати під час використання цих методів, є n_estimators і max_features. Перший – це кількість дерев у лісі. Чим більше, тим краще, але і більше часу піде на розрахунок. Крім того, результати більше не будуть суттєво покращуватися при критичній кількості дерев. Останнє — це розмір випадкових підмножин ознак, які слід враховувати при розділенні вузла. Чим менше, тим суттєвіше зменшення дисперсії, але також значніше збільшення зміщення. Емпірично «правильні» значення за замовчуванням: max_features = None (завжди враховуються всі ознаки замість випадкової підмножини) для проблем регресії та max_features = «sqrt» (з використанням випадкової підмножини розміром sqrt(n_features)) для проблем класифікації (де n_features — це кількість ознак у даних). Хороші результати часто досягаються, якщо встановити max_depth = None у поєднанні з min_samples_split = 2 (тобто, коли дерева повністю розвинуті). Однак ці значення зазвичай не є оптимальними і можуть призвести до того, що моделі споживають багато оперативної пам’яті.

Точність і абсолютна похибка даних перевіряються за даними перевірки. Побудова та навчання моделі проводилися кілька разів, і точність підтримувалася на рівні 99,629 %, що на 2 % краще, ніж при першій настройці моделі. Навіть при змішуванні даних на етапах підготовки даних вони ніяк не впливають на результат. Отриману стабільність можна пояснити властивостями деревовидної структури алгоритму.

Точність оптимізованої моделі становить 99,629%, а середня абсолютна похибка становить 0,0037.

Таким чином, за результатами експерементальних досліджень для блоку прийняття рішень обрано модель Random Forest.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ РОЗРОБЛЕНИХ КОМПОНЕНТІВ СППЛР

4.1 Обґрунтування вибору мови програмування

Для реалізації системи підтримки прийняття рішень в процесі діагностування інфаркту міокарда необхідно написати back-end – серверну сторону системи, яка відповідає за все те, що насправді відбувається, але ви не бачите цього у себе на екранах та front-end – презентаційну частину системи, її інтерфейс користувача і пов'язані з ним компоненти.

Back-end – це серверна сторона веб-сайту. Він зберігає та впорядковує дані, а також гарантує, що все на клієнтській стороні веб-сайту працює нормально. Це частина веб-сайту, яку ви не можете бачити та взаємодіяти з нею. Це частина програмного забезпечення, яка не контактує безпосередньо з користувачами. Користувачі опосередковано отримують доступ до частин і характеристик, розроблених дизайнерами серверної частини, через зовнішню програму. Такі види діяльності, як написання API, створення бібліотек і робота з системними компонентами без інтерфейсів користувача або навіть систем наукового програмування, також включені в back-end.

Мови, які використовуються для написання front-end частини:

– PHP;
– Java;
– Python;
– Node.js.

PHP – це серверна мова сценаріїв, розроблена спеціально для веб-розробки. Оскільки код PHP виконується на стороні сервера, він називається мовою сценаріїв на стороні сервера.

Java є однією з найпопулярніших і широко використовуваних мов і платформ програмування. Він дуже масштабований.

Python — це мова програмування, яка дає змогу швидко працювати та ефективніше інтегрувати системи.

Node.js — це кросплатформне середовище виконання з відкритим кодом для виконання коду JavaScript поза браузером.

Зазначимо, що NodeJS не є фреймворком і не є мовою програмування. Більшість людей плутають і розуміють, що це фреймворк або мова програмування. Ми часто використовуємо Node.js для створення внутрішніх служб, таких як API, наприклад Web App або Mobile App.

Частина веб-сайту, з якою користувач безпосередньо взаємодіє, називається інтерфейсом. Його також називають «клієнтською стороною програми». Він включає в себе все, що користувачі відчувають безпосередньо: кольори та стилі тексту, зображення, графіки та таблиці, кнопки, кольори та меню навігації. Чуйність і продуктивність є двома головними цілями front-end. Розробник повинен переконатися, що сайт адаптивний, тобто він правильно відображається на пристроях будь-якого розміру, жодна частина веб-сайту не повинна працювати ненормально, незалежно від розміру екрана.

Мови, які використовуються для написання front-end частини:

– HTML;
– CSS;
– JavsScript;

HTML означає мову розмітки гіпертексту. Він використовується для розробки зовнішньої частини веб-сторінок за допомогою мови розмітки. HTML – це комбінація гіпертексту та мови розмітки.

Каскадні таблиці стилів, які люблять називати CSS, – це просто розроблена мова, призначена для спрощення процесу створення веб-сторінок презентабельним. CSS дозволяє застосовувати стилі до веб-сторінок.

JavaScript – це відома мова сценаріїв, яка використовується для створення магії на сайтах, щоб зробити сайт інтерактивним для користувача. Він використовується для покращення функціональності веб-сайту для запуску класних ігор і веб-програмного забезпечення.

Існує багато інших мов, за допомогою яких можна розробляти інтерфейс, залежно від фреймворку, наприклад, Flutter використовує Dart, React використовує JavaScript, а Django використовує Python, і багато іншого.

Ці дві частини мають бути сумісними. Тож, має сенс обрати перевірні часом комбінації технологій.

При виборі мови для реалізації back-end треба враховати специфіку задач, які має виконувати система:

– робота з великими об’ємами даних;
– наявність реалізації нейронної мережі штучного інтелекту, яку треба обучити на тестовому датасеті.

Для цього ідеально підходить мова програмування Python за рахунок великої кількості інструментів (бібліотек) для роботи з великими об’ємами даних та нейронними мережами.

Python – це універсальний сучасний ЯП високого рівня, до переваг якого відносять високу продуктивність програмних рішень і структурований код, що добре читається. Синтаксис Пітона максимально полегшений, що дозволяє вивчити його порівняно короткий час. Ядро має дуже зручну структуру, а широкий перелік вбудованих бібліотек дозволяє застосовувати значний набір корисних функцій та можливостей. ЯП може використовуватися для написання прикладних програм, а також розробки WEB-сервісів.

Python може підтримувати широкий перелік стилів розробки додатків, у тому числі дуже зручний для роботи з ОВП та функціонального програмування.

Один із найпопулярніших інтерпретаторів мови – CPython, написаний на C. Поширюється це середовище розробки безкоштовно за вільною ліцензією. Інтерпретатор підтримує більшість популярних платформ.

Python активно розвивається. Приблизно раз на 2 роки виходять оновлення. Важливою особливістю мови є відсутність таких стандартів кодування як ANSI, ISO та деяких інших, вони працюють завдяки інтерпретатору.

Python має чітко структуроване семантичне ядро та простий синтаксис. Все, що пишеться цією мовою, завжди легко читається. У разі потреби передати аргументи мова використовує функцію call-by-sharing.

Набір операторів у мові цілком стандартний.

Зручна особливість синтаксису – це форматування тексту коду за допомогою розбивки їх на блоки за допомогою відступів, які створюють натисканням «Space» та «Tab». У синтаксисі відсутні фігурні або операторні дужки, що позначають початок та кінець блоку.

4.2 Побудова та навчання нейронної мережі

Розглянемо, як за допомогою пакетів NumPy , Scikit-learn, Matpotlib реалізовується нейронна мережа.

Нейронні мережі добре справляються з вивченням тенденцій як і великих, і у малих датасетах. Тим не менш, фахівці за даними повинні мати на увазі небезпеку можливого перенавчання, яке частіше зустрічається в проектах з невеликими наборами даних. Перенавчання відбувається, коли алгоритм занадто довго навчається на датасеті, в результаті чого модель просто запам'ятовує представлені дані, даючи хороші результати безпосередньо на виборці, що використовується. При цьому вона суттєво гірше узагальнюється на нові дані, адже саме це нам від неї і потрібно.

Щоб гарантувати оцінку моделі з позиції її можливості прогнозувати саме нові точки даних, прийнято розділяти датасети на навчальну та контрольну та тестові вибірки (фрагмент коду 3.1).

Фрагмент коду 4.1– Створення навчальної та тестової вибірок

```
input_train = np.array([[0, 1, 0],
                        [0, 1, 1],
                               [0, 0, 0],
                                      [10, 0, 0],
                                             [10, 1, 1],
                                                    [10,   0,   1]])
output_train   =   np.array([[0],   [0],   [0],   [1],   [1],   [1]])
input_pred = np.array([1, 1, 0])
     input_test = np.array([[1, 1, 1],
                        [10, 0, 1],
                               [0, 1, 10],
                                      [10, 1, 10],
                                             [0, 0, 0],
                                                    [0,   1,   1]])
output_test = np.array([[0], [1], [0], [1], [0], [0]])
```

Фрагмент коду 4.2 – Створення класу НМ

```
class NeuralNetwork():
  def __init__(self, ):
        self.inputSize = 3
        self.outputSize = 1
        self.hiddenSize = 3
        self.W1 = np.random.rand(self.inputSize,self.hiddenSize)
        self.W2 =np.random.rand(self.hiddenSize,self.outputSize)
        self.error_list = []
        self.limit = 0.5
        self.true_positives = 0
        self.false_positives = 0
        self.true_negatives = 0
        self.false_negatives = 0

  def forward(self, X):
        self.z = np.matmul(X, self.W1)
        self.z2 = self.sigmoid(self.z)
        self.z3 = np.matmul(self.z2, self.W2)
        o = self.sigmoid(self.z3)
  return o

  def sigmoid(self, s):
  return 1 / (1 + np.exp(-s))

  def sigmoidPrime(self, s):
  return s * (1 - s)

  def backward(self, X, y, o):
        self.o_error = y - o
        self.o_delta = self.o_error * self.sigmoidPrime(o)
        self.z2_error = np.matmul(self.o_delta,
        np.matrix.transpose(self.W2))
        self.z2_delta = self.z2_error*self.sigmoidPrime(self.z2)
        self.W1+=np.matmul(np.matrix.transpose(X),self.z2_delta)
        self.W2 += np.matmul(np.matrix.transpose(self.z2),
        self.o_delta)

  def train(self, X, y, epochs):
        for epoch in range(epochs):
        o = self.forward(X)
        self.backward(X, y, o)
        self.error_list.append(np.abs(self.o_error).mean())

  def predict(self, x_predicted):
  return self.forward(x_predicted).item()

  def predict(self, x_predicted):
  return self.forward(x_predicted).item()

  def view_error_development(self):
        plt.plot(range(len(self.error_list)), self.error_list)
        plt.title('Mean Sum Squared Loss')
        plt.xlabel('Epoch')
        plt.ylabel('Loss')
```

Під час реалізації замість умовних нулів та одиниць будуть використані вхідні медичні дані про пацієнта, а також дані ЕКГ.

Далі нам треба створити клас нейронної мережі. Один із найпростіших способів познайомитися з усіма елементами нейронної мережі – створити відповідний клас (фрагмент коду 3.2). Він повинен включати всі змінні та функції, які потрібні для належної роботи нейронної мережі.

На рисунку рисунку 3.1 зображено схематичне представлення штучної нейронної мережі, яка складається з трьох вхідних вузлів.

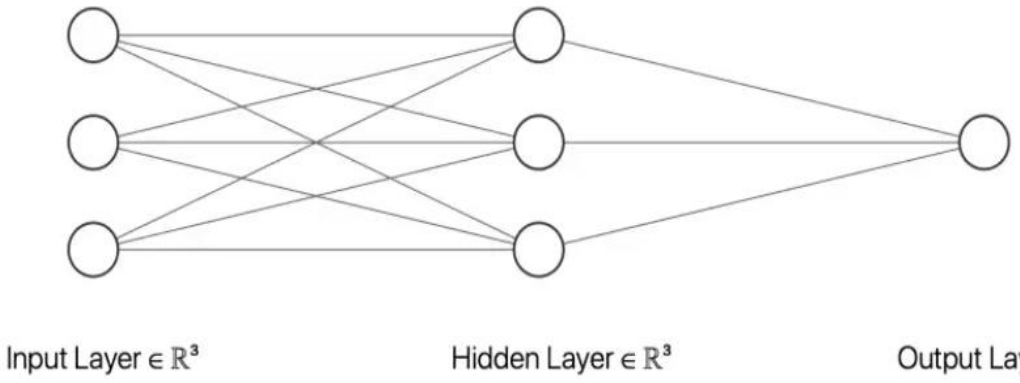

Рисунок 4.1 – Схематичне зображення штучної нейронної мережі

Функція _init_ ініціалізує змінні, що описують розмір нейронної мережі. inputSize - це кількість вхідних вузлів, яка повинна дорівнювати кількості ознак у вхідних даних. outputSize дорівнює числу вихідних вузлів, а hiddenSize вказує їх кількість у прихованому шарі. Крім того, між вузлами мережі будуть присутні ваги, що підлаштовуються в процесі навчання.

На додаток до змінних, що описують розмір нейронної мережі та її ваги, я створив кілька змінних, які ініціалізуються під час створення об'єкта NeuralNetwork, який буде використаний для оцінки ефективності мережі. error_list буде містити середню абсолютну помилку (MAE) для кожної епохи, а її поріг буде вказувати межу, визначальну чи повинен класифікуватися вектор як містить або не містить на початку елемент 10. Потім йдуть змінні, які будуть служити для зберігання кількості вірних позитивних і помилкових позитивних , а також вірних негативних та хибних негативних результатів.

Мета функції forward у прямому проході через усі шари нейронної мережі та прогнозування виходу для кожної епохи. Після цього на основі різниці між спрогнозованим виходом та фактичними даними у процесі зворотного розповсюдження відбувається оновлення ваг.

Для обчислення значень вузлів кожного шару виконується операція матричного множення значень вузлів попереднього шару відповідні ваги, після чого застосовується нелінійна функція активації для розширення ймовірностей кінцевої вихідної функції. У цьому прикладі я вибрав як функцію активації сигмоїду, але є й інші альтернативи.

Зворотне поширення помилки – це процес оновлення ваг вузлів нейронної мережі, що визначає їхню важливість. У функції backward підсумкова помилка вихідного шару обчислюється як різницю між спрогнозованим виходом, отриманим під час прямого поширення, і фактичним виходом. Потім ця помилка множиться на сигмоїду для виконання градієнтного спуску, після чого весь процес повторюється. На завершення ваги між шарами оновлюються.

У процесі навчання алгоритм виконує прямий та зворотний прохід, оновлюючи ваги стільки разів, скільки буде пройдено епох. Це необхідно, щоб у результаті отримати найбільш точні значення.

Після тонкого настроювання ваг алгоритм готовий прогнозувати вихід нових точок даних. За це відповідає функція predict. Це виконується однією ітерацією прямого проходу.

Щоб випробувати наш клас нейронної мережі, ми почнемо з ініціалізації об'єкта типу NeuralNetwork. Після цього мережа протягом 200 епох навчається на навчальній вибірці для тонкого налаштування ваг. Потім підсумкова модель тестується контрольному векторі. Після цього графічно відображається зміна помилки і модель оцінюється на контрольній вибірці.

Для того, щоб навчити нейронну мережу на медичних даних для початку треба їх прочитати з архіву (фрагмент коду 3.3).

Фрагмент коду 4.3 – Читання даних з архіву

```
def load_raw_data(df, sampling_rate, path):
    if sampling_rate == 100:
        data = [wfdb.rdsamp(path+f) for f in df.filename_lr]
    else:
        data = [wfdb.rdsamp(path+f) for f in df.filename_hr]
    data = np.array([signal for signal, meta in data])
    return data

path = 'path/to/ptbxl/'
sampling_rate=100

# load and convert annotation data
Y = pd.read_csv(path+'ptbxl_database.csv', index_col='ecg_id')
Y.scp_codes = Y.scp_codes.apply(lambda x: ast.literal_eval(x))

# Load raw signal data
X = load_raw_data(Y, sampling_rate, path)

# Load scp_statements.csv for diagnostic aggregation
agg_df = pd.read_csv(path+'scp_statements.csv', index_col=0)
agg_df = agg_df[agg_df.diagnostic == 1]
```

Потім необхідно агрегувати дані та розділити датасет на дані для навчання та для тестування (фрагмент коду 4.4).

Фрагмент коду 4.4 – Агрегування даних

```
def aggregate_diagnostic(y_dic):
    tmp = []
    for key in y_dic.keys():
        if key in agg_df.index:
            tmp.append(agg_df.loc[key].diagnostic_class)
    return list(set(tmp))

# Apply diagnostic superclass
Y['diagnostic_superclass'] = Y.scp_codes.apply(aggregate_diagnostic)

# Split data into train and test
test_fold = 10

# Train
X_train = X[np.where(Y.strat_fold != test_fold)]
y_train = Y[(Y.strat_fold != test_fold)].diagnostic_superclass

# Test
X_test = X[np.where(Y.strat_fold == test_fold)]
y_test = Y[Y.strat_fold == test_fold].diagnostic_superclass
```

4.3 Реалізація веб-додатку СППЛР

Реалізований блок логічних ріщень необхідно огорнути в веб-додаток, яким зможе користуватись лікар. Для цього нам знадобляться бібліотеки streamlit та pandas.

Інтерфейс умовно поділяється на дві частини:

– бокова панель для введення додаткових вхідних даних;
– основна сторінка з дублюванням вхідних даних та висновком.

Дві основні функції реалізовного інтерфейсу інферфейсу:

– прийняття додаткових вхідних даних;
– відображення результату роботи блоку логічних висновків.

- Додаткові вхідні дані при цьому складаються з таких пунктів:

– вік пацієнта (Age);
– стать (Sex);
– чи курить пацієнт (Smoking);
– чи є цей випадок рецидивом (Recurrence).

Зовнішній вигляд реалізованого веб-застосунку зображено на рисунку 5.2.

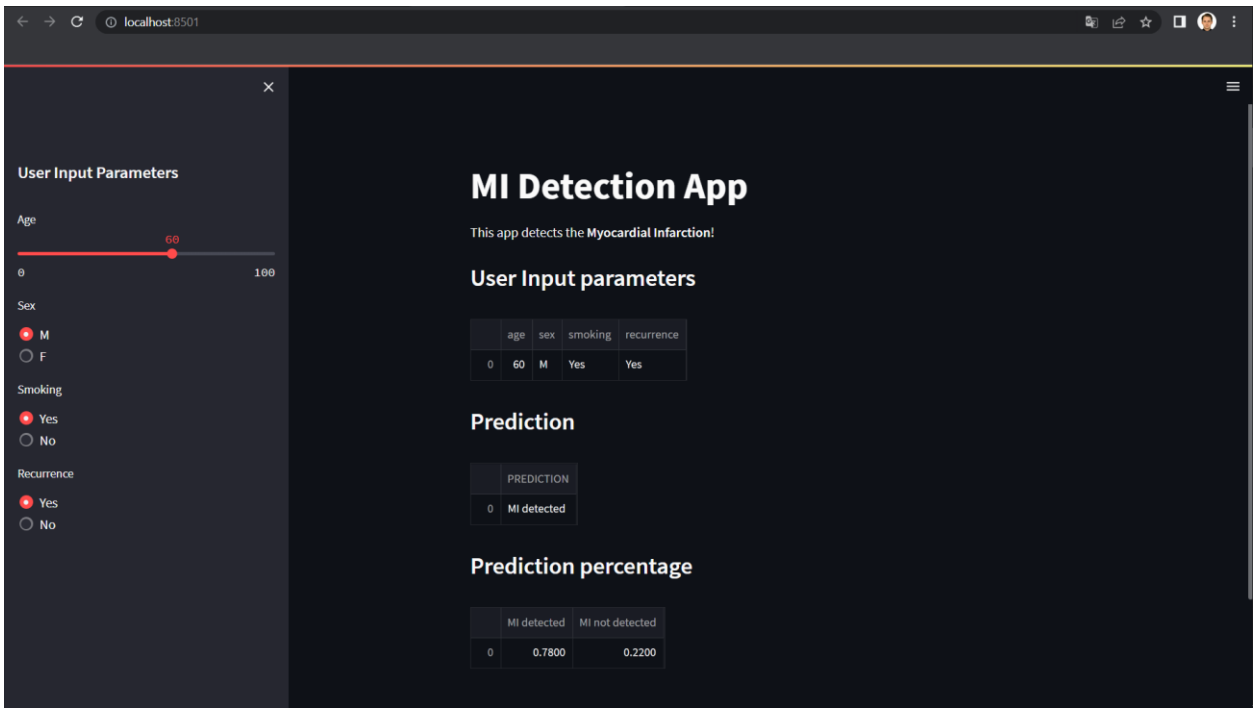

Рисунок 4.2 – Інтерфейс веб-застосунку

Код реалізації інтерфейсу веб-застосунку наведено в фрагменті коду 3.5.

Фрагмент коду 4.5 – Реалізація інтерфейсу

```
import streamlit as st
import pandas as pd


st.write("""
# MI Detection App
This app detects the **Myocardial Infarction**!
""")

st.sidebar.header('User Input Parameters')

def user_input_features():
    age = st.sidebar.slider('Age', 0, 100, 60)
    sex = st.sidebar.radio('Sex', ['M', 'F'])
    smoking = st.sidebar.radio('Smoking', ['Yes', 'No'])
    recurrence = st.sidebar.radio('Recurrence', ['Yes', 'No'])
    signal = load_raw_data(annotation_data, sampling_rate, path)
    data = {'age': age,
            'sex': sex,
            'smoking': smoking,
            'recurrence': recurrence,
            'signal': signal}
    features = pd.DataFrame(data, index=[0])
    return features

df = user_input_features()

st.subheader('User Input parameters')
st.write(df)

nn = NeuralNetwork()
nn.fit(signal, annotation_data)

prediction = nn.predict(df)
prediction_proba = nn.predict_proba(df)

st.subheader('Class labels and their corresponding index number')
st.write(annotation_data.target_names)

st.subheader('Prediction')
st.write(annotation_data.target_names[prediction])
```

За допомогою фреймворку Streamlit можна легко створювати інтерактивні веб-застосунки. Адже Streamlit спочатку влаштований виконувати найскладнішу роботу зі створення та компонування веб-елементів, дозволяючи нам займатися лише даними. Ця бібліотека містить широкий спектр функцій, що підтримує фреймворк, які так необхідні фахівцям в роботі з даними.

Pandas – це програмна бібліотека, написана мовою Python для обробки та аналізу даних. Робота pandas з даними будується поверх бібліотеки NumPy, що є інструментом нижчого рівня. Надає спеціальні структури даних та операції для маніпулювання числовими таблицями та тимчасовими рядами. Назва бібліотеки походить від терміну «панельні дані», який використовується для опису багатовимірних структурованих наборів інформації.

ВИСНОВКИ

За результатами аналізу медичної та комп'ютерної складових предметної області визначені основні функції, що вимагають автоматизації заради функціювання системи підтримки прийняття рішень для діагностування інфаркту міокарда.

У роботі було проведено аналіз існуючих рішень, виділені основні недоліки, які були враховані на етапі розробки системи підтримки прийняття лікарських рішень. Результатом аналітичних робіт – є постановка завдання на проектування та реалізацію системи підтримки прийняття рішень для діагностування інфаркту міокарда.

Розроблені вимоги до системи та її інтерфейсу. Для цього були побудовані діаграма прецедентів, контекстна діаграма та обрана архітектура системи. Також було проведено попередню обробку та підготовку даних для моделей класифікації.

З урахуванням визначених вимог були здійснені такі роботи для реалізації системи підтримки прийняття рішень для діагностування інфаркту міокарда:

- проведено аналіз характеристик існуючих СППЛР;
- визначено функціональні вимоги до СППЛР для діагностики ІМ;
- розробити модуль прийняття рішень на основі методів машинного навчання;
- програмно реалізовано модель класифікації даних
- реалізовано інтерфейс користувача у вигляді веб-додатка.

Отже, усі завдання кваліфікаційної роботи було виконано, а мета досягнена.

Розроблена інформаційна система підтримки прийняття рішень для діагностування інфаркту міокарда готова до експлуатування.

•

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. AstraZeneca and IBM collaborate in order to strengthen digital health expertise: https://nordiclifescience.org/astrazeneca-and-ibm-collaborate-in-order-to-strengthen-digital-health-expertise/

2. Штучний інтелект навчили передбачати ризик серцевого нападу: https://www.ukrinform.ua/rubric-technology/3624230-stucnij-intelekt-navcili-peredbacati-rizik-sercevogo-napadu-j-insultu-za-rentgenom.html

3. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. Чинний від 2017-07-01. – Київ: ДП «УкрНДНЦ», 2016. – 26 с.

4. Лутц, Марк. Вивчаємо Python. Издательский дом «Вильямс», 2009. – 224 с.

5. Фаулер М. UML. Основы, 3е издание. СимволПлюс, 2004. – 192 с., ил.

6. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання: (ГОСТ 7.1–2003, IDT): ДСТУ ГОСТ 7.1:2006. – Чинний з 2007-07-01. – К.: Держспоживстандарт України, 2007. – III, 47 с.

7. ISO/IEC 15288:2002 «System engineering – System life cycle processes» – URL : https://www.iso.org/ru/standard/43564.html

8. Ishaque, S.; Khan, N.; Krishnan, S. Trends in Heart-Rate Variability Signal Analysis. Front. Digit. Health 2021, 3, 639444.

9. Hodgart, E.; Macfarlane, P.W. 10 second heart rate variability. In Proceedings of the Computers in Cardiology 2004, Chicago, IL, USA, 19–22 September 2004; pp. 217–220

10. Sahu, S.K.; Mishra, B.; Thakur, R.S.; Sahu, N. Normalized hamming k-nearest neighbor (NHK-nn) classifier for document classifi-cation and numerical result analysis. *Glob. J. Pure Appl. Math.* 2017, *13*, 4837–4850.

11. Rovetta, A. Raiders of the Lost Correlation: A Guide on Using Pearson and Spearman Coefficients to Detect Hidden Correlations in Medical Sciences. *Cureus* 2020, *12*, e11794

12. Alexandridis, A.; Chondrodima, E. A medical diagnostic tool based on radial basis function classifiers and evolutionary simulated annealing. *J. Biomed. Inform.* 2014, *49*, 61–72.

13. Shashua, A. Introduction to machine learning: Class notes 67577. *arXiv* 2009, arXiv:0904.3664.

14. Dudkina, T.; Meniailov, I.; Bazilevych, K.; Krivtsov, S.; Tkachenko, A. Classification and prediction of diabetes disease using decision tree method. In Proceedings of the CEUR Workshop Proceedings 2021, Symposium on Information Technologies & Applied Sciences (IT&AS 2021), Bratislava, Slovakia, 5 March 2021; Volume 2824, pp. 163–172.

15. Sokoliuk, A.; Kondratenko, G.; Sidenko, I.; Kondratenko, Y.; Khomchenko, Y.; Atamanyuk, I. Machine Learning Algorithms for Binary Classification of Liver Disease. In Proceedings of the 2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, Ukraine, 6–9 October 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 417–421

16. Wagner, P.; Strodthoff, N.; Bousseljot, R.; Samek, W.; Schaeffter, T. *PTB-XL*, version 1.0.1. PTB-XL, A Large Publicly Available Electrocardiography Dataset. PhysioNet: Bristol, UK, 2020.

17. Goldberger, A.; Amaral, L.; Glass, L.; Hausdorff, J.; Ivanov, P.C.; Mark, R.; Mietus, J.E.; Moody, G.B.; Peng, C.K.; Stanley, H.E. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation* 2020, *101*, e215–e220.

18. Wagner, P.; Strodthoff, N.; Bousseljot, R.-D.; Kreiseler, D.; Lunze, F.I.; Samek, W.; Schaeffter, T. PTB-XL, a large publicly available electrocardiography dataset. *Sci. Data* **2020**, *7*, 154.

19. Smith, S.J.M. EEG in the diagnosis, classification, and management of patients with epilepsy. *J. Neurol. Neurosurg. Psychiatry* 2005, *76* (Suppl. 2), ii2–ii7.

20. Malinova, V.; von Eckardstein, K.; Mielke, D.; Rohde, V. Diagnostic yield of fluorescence-assisted frame-based stereotactic biopsies of intracerebral lesions in comparison with frozen-section analysis. *J. Neuro-Oncology* 2020, *149*, 315–323.

21. Ghumbre, S.; Patil, C.; Ghatol, A. Heart disease diagnosis using support vector machine. In Proceedings of the International Conference on Computer Science and Information Technology 2011, Penang, Malaysia, 22–24 February 2011; pp. 84–88.

22. Park, J.; An, J.; Kim, J.; Jung, S.; Gil, Y.; Jang, Y.; Lee, K.; Oh, I.-Y. Study on the use of standard 12-lead ECG data for rhythm-type ECG classification problems. *Comput. Methods Programs Biomed.* **2021**, *214*, 106521.

23. Lu, L.; Liu, M.; Sun, R.R.; Zheng, Y.; Zhang, P. Myocardial Infarction: Symptoms and Treatments. *Cell Biophys.* **2015**, *72*, 865–867.

24. Chartrain, A.G.; Kellner, C.P.; Mocco, J. Pre-hospital detection of acute ischemic stroke secondary to emergent large vessel occlusion: Lessons learned from electrocardiogram and acute myocardial infraction. *J. NeuroInterv. Surg.* 2018, *10*, 549–553.

25. Закон України «Про захист персональних даних» – URL: https://zakon.rada.gov.ua/laws/show/2297-17#Text.