Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

ПРОГРАМНИЙ МОДУЛЬ ДІАГНОСТУВАННЯ ЗАХВОРЮВАНЬ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

Виконала: студентка 4 курсу, групи 3КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Коновалова М. С.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф.КН
Паночишин Ю.М.
(прізвище та ініціали)
«04» червня 2025 р.

Рецензент: к.т.н., проф. каф. КСУ
Биков М. М.
(прізвище та ініціали)
«05» червня 2025 р.

Допущено до захисту
Зав. кафедри Яровий А.А.
«06» червня 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо - професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«05» травня 2025 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Коноваловій Марті Сергіївні

1. Тема роботи: Програмний модуль діагностування захворювань на основі нейронної мережі.
керівник роботи: Паночишин Юрій Миколайович, к.т.н., доц.
затверджені наказом вищого навчального закладу "20" березня 2025 року №97

2. Термін подання студентом роботи 30 травня 2025 р.

3. Вихідні дані до роботи :
кількість симптомів для визначення захворювання – не менше 5, вхідні дані вводяться користувачем у вікні програми, вхідні дані - це симптоми вибрані з випадаючого вікна, кількість передбачуваних захворень – не менше 30.

4. Зміст текстової частини (перелік питань, які потрібно розробити):
Вступ, постановка задачі, аналіз предметної області визначення захворювань, проектування основного компонента програми визначення захворювань, програмна реалізація модуля визначення захворювань, аналіз результатів тестування програми визначення захворювань, висновки, перелік використаних джерел, додатки.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
схема алгоритму реалізації модуля,
повторювана LSTM структура,
робочі вікна програми,
порівняння результатів роботи розробленого програмного модуля та програми-аналога.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Паночишин Ю.М., к.т.н., доц. каф. КН		

7. Дата видачі завдання „05" січня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	05.05.25	07.05.25	
2	Обгрунтування методу розв'язання задачі, проектування програмного модуля діагностування захворювань на основі нейронної мережі	08.05.25	11.05.25	
3	Розробка алгоритму роботи програмного модуля	12.05.25	15.05.25	
4	Програмна реалізація модуля діагностування захворювань на основі нейронної мережі	16.05.25	26.05.25	
5	Аналіз результатів тестування модуля діагностування захворювань на основі нейронної мережі	30.05.25	01.06.25	
6	Оформлення матеріалів до захисту БКР	02.06.25	04.06.25	

Студентка _____ **Коновалова М. С.**
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ **Паночишин Ю.М.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Коновалова М. С. Програмний модуль діагностування захворювань на основі нейронної мережі. Бакалаврська кваліфікаційна робота скаладається з 67 сторінок формату А4, на яких є 23 рисунки, 5 таблиць, список використаних джерел містить 19 найменувань.

Метою роботи є підвищення достовірності діагностування захворювань на основі медичних даних та симптомів.

Розроблений програмний модуль призначений для діагностування захворювань на основі симптомів з використанням нейронної мережі LSTM. Модуль розроблено на мові програмування Python у середовищі розробки PyCharm та спеціалізованих бібліотек TensorFlow, Keras, Sklearn, Numpy, Pandas, Matplotlib. Набір даних симптомів захворювань для навчання та тестування модуля містить 4428 навчальних прикладів та 492 тестових приклади. Розроблений програмний модуль має достовірність діагностування захворювання 89%, а програма-аналог майже 82%. Тобто достовірність діагностування захворювань підвищена на 7%.

Ключові слова: діагностування захворювань, симптоми, нейронна мережа LSTM, достовірність діагностування.

ABSTRACT

Konovalova M. S. Software module for diagnosing diseases based on a neural network. The bachelor thesis consists of 67 pages of A4 format, on which there are 23 figures, 5 tables, the list of used sources contains 19 names.

The aim of the work is to increase the reliability of diagnosing diseases based on medical data and symptoms.

The developed software module is designed for diagnosing diseases based on symptoms using the LSTM neural network. The module is developed in the Python programming language in the PyCharm development environment and specialized libraries TensorFlow, Keras, Sklearn, Numpy, Pandas, Matplotlib. The data set of disease symptoms for training and testing the module contains 4428 training examples and 492 test examples. The developed software module has a reliability of diagnosing the disease of 89%, and the analog program has almost 82%. That is, the reliability of diagnosing diseases is increased by 7%.

Keywords: diagnosing diseases, symptoms, LSTM neural network, diagnostic reliability.

ЗМІСТ

ВСТУП

Актуальність досліджень. Сфера здоров'я завжди була однією з найбільш важливих та актуальних сфер у суспільстві. Здоров'я та добробут людей є ключовими аспектами розвитку будь-якої нації, і важливо забезпечити доступ до високоякісної медичної допомоги. У світі, де велика кількість захворювань може стати загрозою для життя та здоров'я людей, науковий прогрес та сучасні технології надають можливість розвивати нові підходи до діагностики та передбачення захворювань.

Медичні дані та симптоми є важливими джерелами інформації для зрозуміння стану здоров'я пацієнтів і можуть бути ключовими показниками для вчасної діагностики та надання відповідної медичної допомоги. Проте, традиційні методи аналізу медичних даних можуть бути обмеженими у точності та швидкості, що часто є критичними факторами у лікуванні.

Саме тому використання передових методів штучного інтелекту та машинного навчання для розробки інтелектуальних систем діагностики та діагностування захворювань є актуальним та обіцяючим напрямком досліджень.

Отримані результати суттєво покращать діагностування захворювань, сприяючи ранньому виявленню та наданню ефективного лікування. Це може позитивно вплинути на якість медичної допомоги та підвищити шанси на одужання для пацієнтів із різноманітними медичними проблемами.

У цій роботі ставиться завдання побудови програмного модуля діагностування захворювань на основі моделі нейронної мережі за симптомами, яка буде мати підвищенну достовірність діагностування захворювань.

Метою дослідження є підвищення достовірності діагностування захворювань на основі медичних даних та симптомів.

Об’єктом дослідження є процес комп’ютеризованого діагностування захворювання на основі симптомів.

Предметом дослідження є алгоритми та програмні засоби для діагностування захворювань за симптомами на основі нейронних мереж та достовірність їх роботи.

Задачі дослідження:

1. Проаналізувати відомі методи діагностування захворювань та обрати напрямок досліджень;
2. Обгрунтувати вибір архітектури нейромережі;
3. Розробити структуру нейронної мережі для діагностування захворювань;
4. Розробити алгоритм роботи програмного модуля діагностування захворювань на основі нейронної мережі;
5. Здійснити програмну реалізацію модуля діагностування захворювань на основі нейронної мережі;
6. Провести тестування програмного модуля діагностування захворювань на основі нейронної мережі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ДІАГНОСТУВАННЯ ЗАХВОРЮВАНЬ

1.1 Постановка задачі

Для кращого розуміння процесу діагностування захворювань за симптомами, розглянемо характеристики вхідних та вихідних даних, а також вимоги до їх обробки:

Почнемо з аналізу атрибутів датасету:

Симптоми (symptoms) – це конкретні ознаки або прояви захворювань у пацієнтів.

Симптоми будуть вибиратись користувачем з певного готового списку, в кількості п’яти.

Хвороба (prognosis) – в датасеті відображає діагноз або прогноз захворювання, який ставлять лікарі пацієнтам на підставі їх симптомів та інших характеристик.

Кожен рядок у датасеті містить інформацію про симптоми, які спостерігаються у пацієнта, а також прогнозовану хворобу, яка може бути результатом аналізу цих симптомів.

У датасеті ця характеристика може представляти собою текстове значення, що вказує на конкретну хворобу або діагноз, наприклад "Грибкова інфекція", "Грип", "Ангіна" тощо.

Тепер перейдемо до загальних вимог до даних.

Формат файлів. Вхідні дані можуть бути у форматі CSV або JSON для зручності обробки та аналізу.

Розмір даних. Дані повинні бути представлені в достатньому обсязі, щоб забезпечити відповідний аналіз та навчання моделі.

Якість даних. Дані мають бути чистими, без відсутніх значень або помилок, щоб уникнути спотворень у діагностуванні.

Структура даних. Дані мають бути структурованими і організованими у вигляді таблиці для зручності обробки.

Тип даних. Вхідні дані можуть включати як числові, так і категоріальні дані, що представлені у відповідному форматі.

Формат даних. Дані можуть бути представлені у вигляді числових значень, текстових описів або категоріальних міток.

Виконувані функції:

Попередня обробка даних. Включає очищення даних від шуму, кодування категоріальних даних та нормалізацію числових значень.

Побудова моделі. Розробка та тренування моделей машинного навчання, таких як класифікатори чи нейронні мережі, для діагностування захворювань на основі симптомів.

Оцінка моделі. Визначення точності та ефективності моделі за допомогою метрик, таких як точність, повнота та F-міра.

Використання моделі: Використання навченої моделі для діагностування захворювань на нових даних.

Також слід враховувати:

Ресурси обчислювальної системи. Модель може вимагати певних обчислювальних ресурсів для навчання та використання.

Час навчання моделі. Час, необхідний для навчання моделі, може бути значним і може залежати від обсягу даних та складності моделі.

Гіперпараметри моделі. Налаштування параметрів моделі може впливати на її ефективність та точність діагностування.

Метрики ефективності. Досяжні метрики ефективності можуть бути використані для оцінки якості діагностування та порівняння різних моделей.

При врахуванні цих вимог, метою є розробка моделі, яка з точністю не менше 50% може прогнозувати захворювання на основі симптомів.

1.2 Огляд відомих методів розв'язання задачі діагностування захворювань

Задача визначення хвороби на основі симптомів і інших клінічних даних є важливим завданням у медичній діагностиці та лікуванні. Для розв'язання цієї задачі використовуються різні методи, моделі та підходи. Ось огляд деяких з них:

Методи машинного навчання:

Логістична регресія — є одним з найпростіших методів (рисунок 1.1) класифікації у машинному навчанні. Вона використовує лінійну комбінацію вхідних змінних для передбачення категорії захворювання. Цей метод особливо ефективний у випадках, коли існує лінійна залежність між вхідними змінними та вихідними класами.

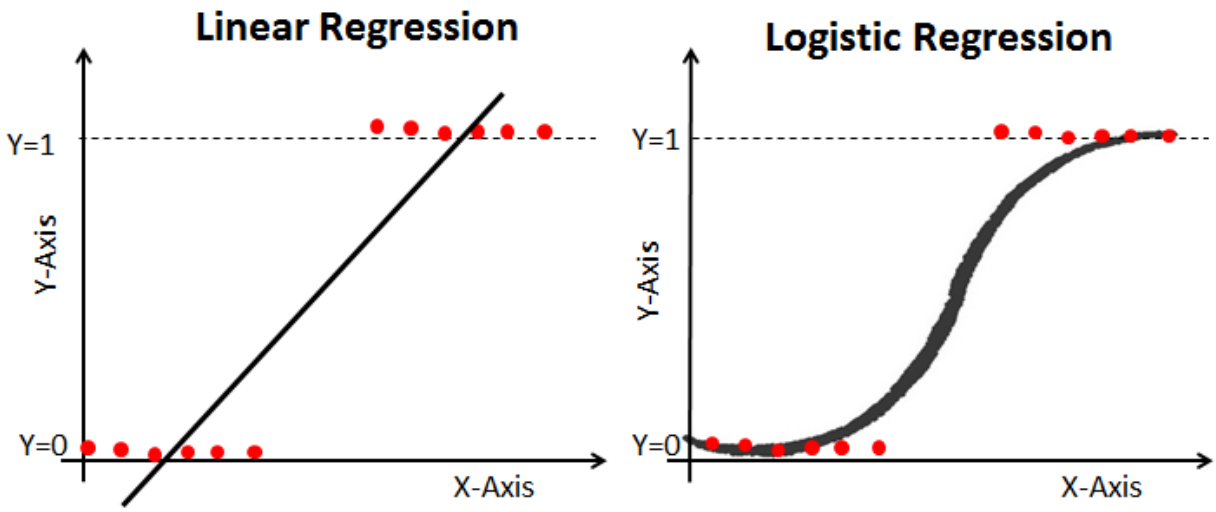

Рисунок 1.1 — Логістична регресія

Дерева рішень — є іншим популярним методом класифікації, який базується на створенні дерева рішень (рисунок 1.2). Це дерево розділяє дані на різні класи, основуючись на значеннях характеристик. Під час класифікації,

нове спостереження проходить вузли дерева від кореня до листя, і його клас визначається значенням, яке знаходиться в листях.

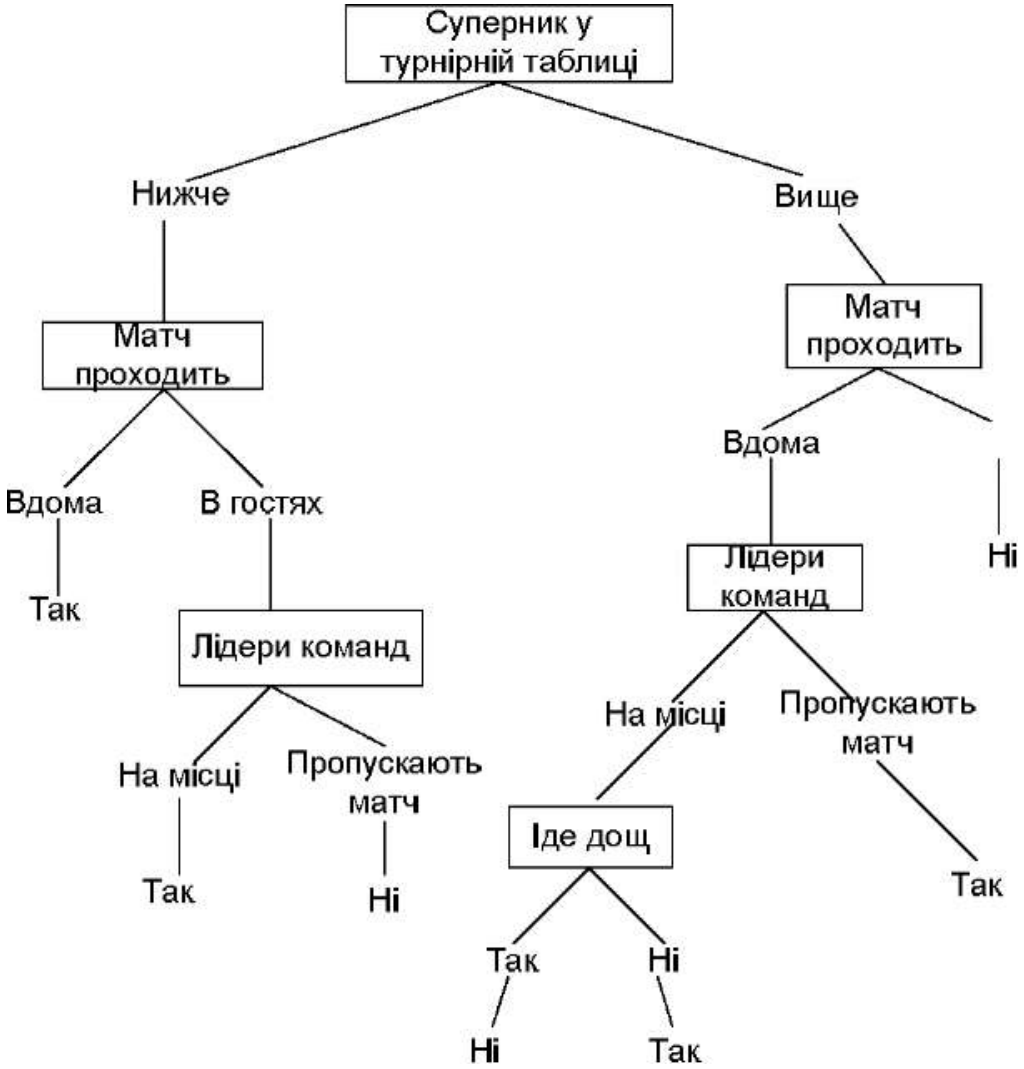

Рисунок 1.2 — Дерева рішень

Випадковий ліс (Random Forest) — є ансамбльним методом, який використовує кілька дерев рішень для класифікації (рисунок 1.3). Він покращує точність шляхом об'єднання результатів багатьох дерев. Кожне дерево випадкового лісу навчається на випадковій підмножині даних, що дозволяє зменшити перенавчання і покращити узагальнювальну здатність моделі [1].

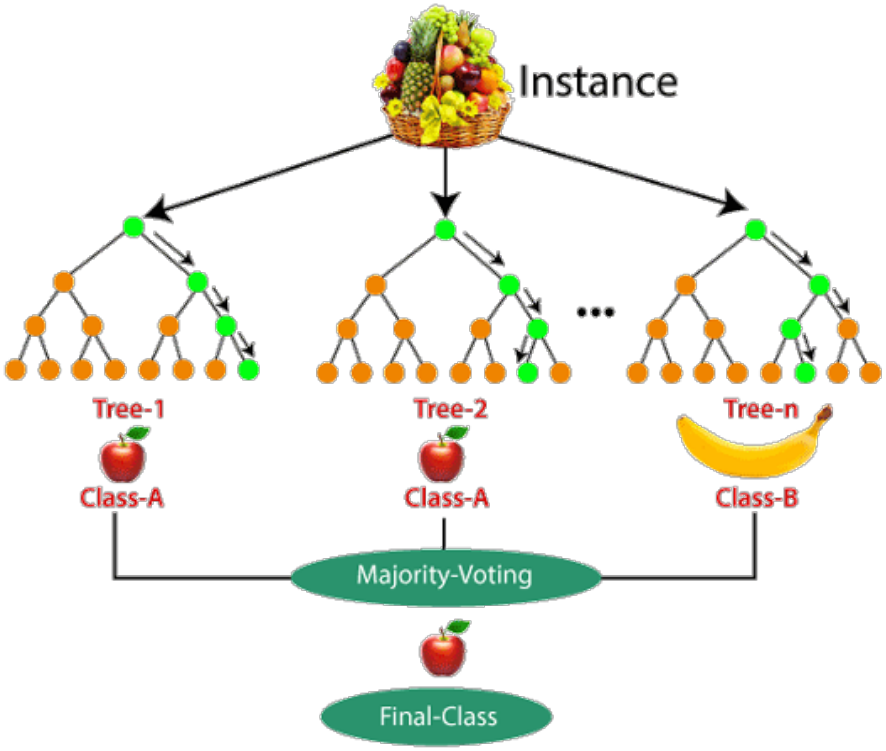

Рисунок 1.3 — Принцип роботи Random Forest

Метод опорних векторів (Support Vector Machines, SVM) — шукає оптимальну розділяючу границю між класами, що максимізує відстань між ними. Цей метод особливо ефективний у випадках, коли класи не лінійно роздільні, оскільки він може використовувати ядрові функції для перетворення даних у вищорозмірний простір, де вони можуть бути лінійно розділені [2].

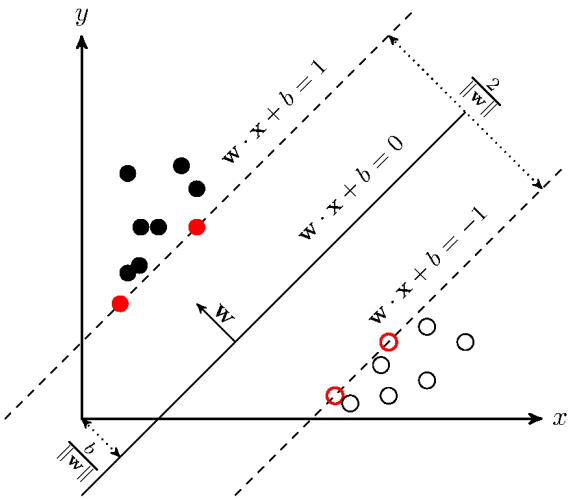

Рисунок 1.4 — Метод опорних векторів

Глибоке навчання:

Згорткові нейронні мережі (Convolutional Neural Networks, CNN) — це потужний тип нейронних мереж, які зазвичай використовуються для обробки та аналізу зображень. Ці мережі мають здатність автоматично виявляти властивості та ознаки в зображеннях за допомогою шарів згортки та підсумування [2].

У контексті медичних даних, CNN можуть бути використані для аналізу різних видів зображень, таких як рентгенівські знімки, зображення МРТ, комп'ютерні томографії (КТ) та інші. Вони можуть автоматично визначати патологічні зміни, структури органів та інші важливі деталі на медичних зображеннях.

Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) — це клас нейронних мереж, які призначені для роботи з послідовними даними, де кожен вхід може бути пов'язаним з попередніми входами. Цей тип мережі добре підходить для обробки послідовних даних, таких як часові ряди симптомів або медичні звіти, де зв'язок між даними в часі є важливим [2].

Експертні системи. Системи, що базуються на правилах — це програмні системи, які використовують набір правил, розроблених експертами у відповідній галузі, для прийняття рішень. У медичному контексті такі системи використовуються для діагностики захворювань та прийняття рішень щодо лікування.

Ці експертні системи роблять припущення та рекомендації, використовуючи набір правил, які визначають логіку та критерії прийняття рішень. Наприклад, правила можуть включати специфічні симптоми, результати лабораторних аналізів та інші клінічні даними, що вказують на певне захворювання або стан пацієнта [1].

Системи на основі байєсівських мереж — це клас систем штучного інтелекту, які використовують теорію ймовірностей для моделювання відносин між симптомами і хворобами. Ці системи оперують на основі

теореми Байєса, яка дозволяє оновлювати вірогідності подій на основі нових даних чи доказів.

У медичному контексті, системи на основі байєсівських мереж можуть бути використані для діагностики хвороб на основі симптомів, які виявлені у пацієнтів. Основна ідея полягає у тому, щоб визначити ймовірності різних хвороб, враховуючи наявність певних симптомів.

Гібридні системи:

Комбінація методів машинного навчання і експертних систем є інноваційним підходом до діагностики хвороб, який поєднує переваги обох підходів для отримання кращих результатів.

Експертні системи базуються на знаннях експертів у галузі медицини та використовують набір правил для прийняття рішень щодо діагностики. Ці системи зазвичай добре працюють для задач, де правила можуть бути чітко визначені та визнані експертами.

З іншого боку, методи машинного навчання можуть виявити складні зв'язки в даних та навчитися приймати рішення на основі цих зв'язків, навіть якщо вони неочевидні для людини. Машинне навчання може адаптуватися до нових даних і покращувати свої прогнози з часом.

Комбінування цих двох підходів дозволяє створити систему, яка поєднує експертні знання з можливостями машинного навчання. Наприклад, можна використовувати експертні системи для формулювання правил на основі клінічних даних та медичних знань, а потім використовувати методи машинного навчання для підтримки прийняття рішень і підтвердження діагнозу за допомогою аналізу великих обсягів даних.

Біомедичне моделювання і обробка сигналів:

Методи обробки сигналів і зображень відіграють важливу роль у медичній діагностиці та аналізі різних медичних даних. Вони дозволяють аналізувати та інтерпретувати різні типи медичних образів, такі як

електрокардіограми, результати магнітно-резонансної томографії (МРТ), рентгенівські знімки та інші.

Із застосуванням методів обробки сигналів можна виявити та аналізувати різноманітні патологічні зміни та аномалії в сигналах, що отримані з медичних приладів. Наприклад, електрокардіограми можуть бути оброблені для виявлення аритмій, а результати МРТ — для визначення розподілу тканин та виявлення ознак хвороб.

Зображення, отримані за допомогою медичних обладнань, також піддаються обробці за допомогою методів обробки зображень. Це може включати виявлення та аналіз різних структур та ознак, визначення розмірів утворень, а також класифікацію хвороб на основі зображень.

З огляду на різноманітність методів машинного навчання та підходів у медичній діагностиці, вибір рекурентних нейронних мереж для вирішення задачі діагностування хвороби є обґрунтованим. Рекурентні нейронні мережі володіють унікальними перевагами, особливо для обробки послідовних даних, які є типовими у медичних даних.

Завдяки своїй здатності до роботи з послідовними даними та врахуванню контексту, рекурентні нейронні мережі можуть ефективно моделювати залежності між різними симптомами та їх динамікою в часі. Це дозволяє враховувати не лише окремі симптоми, а й їх взаємодію та вплив на діагноз хвороби.

Застосування рекурентних нейронних мереж в медичному діагностичному процесі може сприяти покращенню точності та ефективності діагностики, дозволяючи лікарям та медичним фахівцям швидше та точніше визначати хвороби та розробляти плани лікування.

Отже, обравши рекурентні нейронні мережі для вирішення задачі визначення хвороби, можна розраховувати на їх потужний потенціал у роботі з медичними даними та забезпеченням якісної та точної медичної діагностики.

1.3 Обгрунтування вибору аналогу до програмного модуля діагностування захворювань

Було знайдено програму аналогічного призначення [3], яка виконує діагностування захворювань за симптомами, але використовує для цього різні методи машинного навчання. Зокрема, використовуються такі методи:

1) SVM – метод опорних векторів,
2) Naive Bayes – наївний метод Байєса,
3) Random Forest – метод випадкового лісу,
4) Combined Model – комбінована модель.

Результати достовірності діагностування захворювань за цими методами представлено на рисунку 1.5.

SVM Accuracy: 60.53%
Naive Bayes Accuracy: 37.98%
Random Forest Accuracy: 68.98%
Combined Model Accuracy: 60.64%

Рисунок 1.5 — Результати достовірності діагностування захворювань аналога [3]

Як видно з рисунка 1.5, достовірність діагностування захворювань при використанні класичних методів машинного навчання не перевищує 70%. Це є недостатнім, так як не говорить про високу точність діагностичної роботи.

Також було знайдено другий аналог [4], який побудовано на основі нейронної мережі типу багатошаровий персептрон [5]. Ця програма має достовірність діагностування захворювань за симптомами майже 82%. Це вже краще, але все-одно недостатньо. Таким чином, можна зробити висновок, що головним недоліком існуючих програм є невелика достовірність прогнозування захворювань. Звідси випливає мета даної роботи – підвищення

достовірності діагностування захворювань на основі медичних даних та симптомів.

1.4 Висновок до розділу 1

У розділі описано детальну постановку задачі діагностування захворювань на основі медичних даних та симптомів. Було розглянуто як традиційні методи діагностування захворювань, так і методи, що ґрунтуються на нейронних мережах, у тому числі глибоких, згорткових та рекурентних. Як найбільш перспективний і застосовний до даної задачі було обрано метод на основі нейромереж. Було показано, що з різних типів нейронних мереж найбільше підходить для вирішення поставленої задачі рекурентна нейронна мережа LSTM. Крім цього, було обгрунтовано вибір аналогів розробленого модуля діагностування захворювань за симптомами.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ДІАГНОСТУВАННЯ ЗАХВОРЮВАНЬ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

2.1 Обґрунтування вибору нейронної мережі для діагностування захворювань

Розглянемо можливість використання різних типів нейронних мереж для визначення захворювання залежно від набору даних (таблиця 2.1).

Таблиця 2.1 – Взірець набору даних

itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	prognosis
1	1	1	0	0	Fungal infection
0	0	1	1	0	Allergy
0	0	0	0	1	GERD
1	1	0	1	1	Drug Reaction

Тепер розглянемо різні нейронні мережі для виконання завдання.

Рекурентні нейронні мережі (Recurrent Neural Networks, RNN): (рисунок 2.1) – ці мережі призначені для роботи з послідовними даними, де кожен вхід може бути пов'язаним з попередніми входами [6].

У випадку, коли ваш проект передбачає аналіз часових рядів, таких як медичні часові ряди симптомів, економічні дані або текстові дані, RNN може бути корисним для виявлення зв'язків між послідовними подіями.

Основна особливість RNN полягає в тому, що вона має здатність зберігати попередні стани або контекст і використовувати їх для обробки нових вхідних даних. Давайте розглянемо деякі ключові аспекти та застосування RNN:

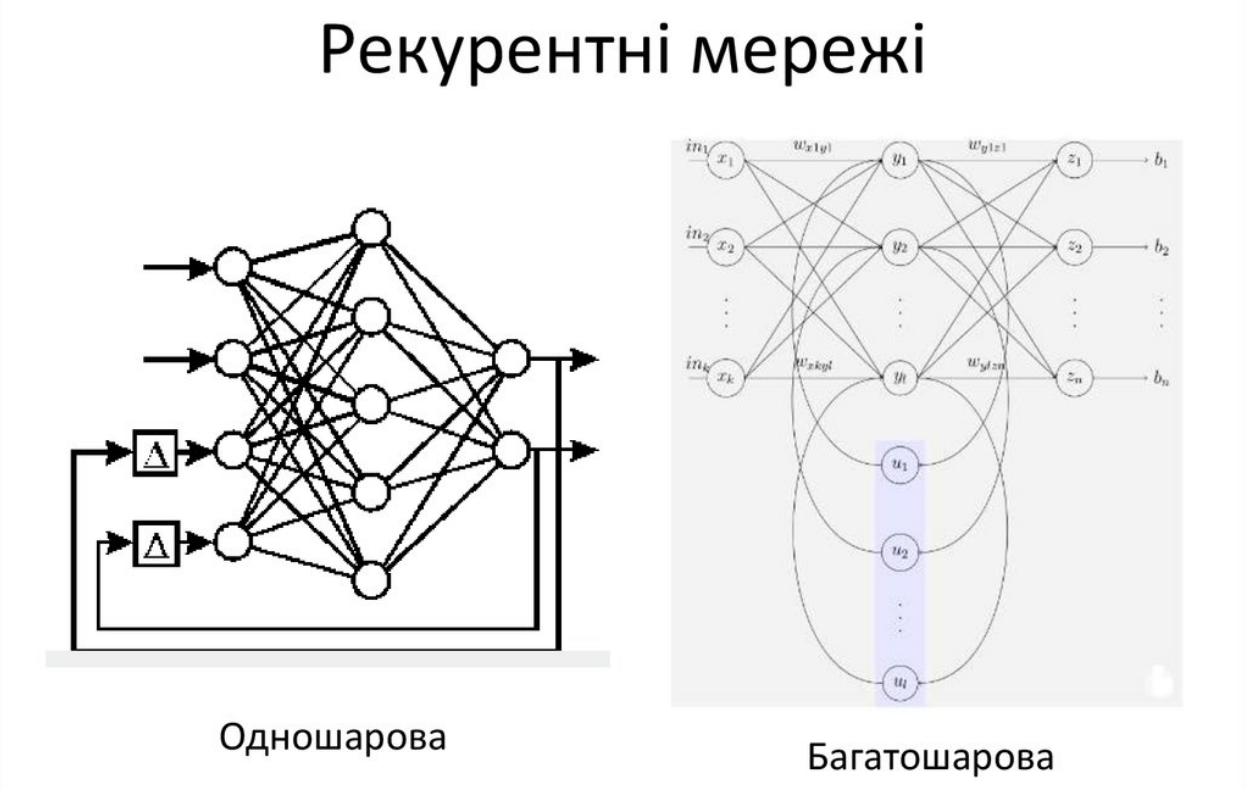

Рисунок 2.1 – Структура рекурентної нейронної мережі

Модель з пам'яттю:

Однією з основних переваг RNN є їх здатність зберігати і використовувати інформацію з попередніх кроків. Кожен вихід RNN залежить від вхідних даних і попередніх станів (пам'яті).

Застосування в аналізі послідовних даних:

RNN використовуються в багатьох областях, де даними є послідовність. Це може бути аналіз тексту (машинний переклад, генерація тексту), обробка аудіо- та відеоданих, прогнозування часових рядів, робота з медичними часовими рядами симптомів, аналіз фінансових даних та інше.

Моделювання довгострокових залежностей:

Однією з проблем стандартних нейронних мереж у роботі з послідовними даними є здатність моделювати довгострокові залежності. RNN дозволяють вирішувати цю проблему, зберігаючи контекст з попередніх кроків і використовуючи його для передбачення наступних.

Варіації RNN:

На базі основної архітектури RNN було розроблено кілька варіацій, таких як Long Short-Term Memory (LSTM) і Gated Recurrent Unit (GRU). Ці варіації мають покращену здатність моделювання довгострокових залежностей і уникнення проблеми зникаючого градієнту.

Застосування в медичному аналізі:

У медичній сфері RNN можуть бути корисними для аналізу медичних часових рядів симптомів, моніторингу пацієнтів у реальному часі, прогнозування медичних показників та роботи з медичними зображеннями.

Згорткові нейронні мережі (Convolutional Neural Networks, CNN): (рисунок 2.2) – Ці мережі призначені для роботи з послідовними даними, де кожен вхід може бути пов'язаним з попередніми входами [7].

У випадку, коли ваш проект передбачає аналіз часових рядів, таких як медичні часові ряди симптомів, економічні дані або текстові дані, RNN може бути корисним для виявлення зв'язків між послідовними подіями [6].

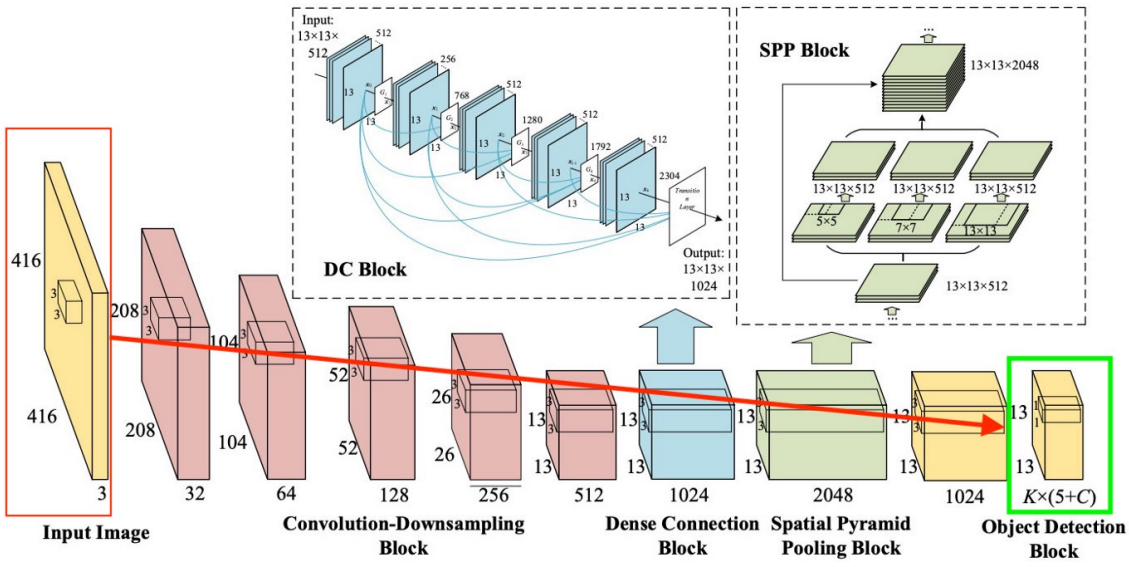

Рисунок 2.2 – Структура згорткової нейронної мережі

Переглянемо основні характеристики та застосування згорткових нейронних мереж (CNN):

Здатність аналізувати просторові залежності:

Основна особливість CNN полягає в їхній здатності аналізувати просторові залежності в даних. Зазвичай вони використовуються для обробки зображень, де просторові залежності можуть включати розподіл об'єктів, текстур, контурів та інших візуальних ознак.

Використання згорткових шарів і пулінгу:

CNN складаються зі згорткових шарів, які використовують фільтри для виявлення важливих ознак у вхідних даних, та пулінгових шарів, які зменшують розмірність даних. Це дозволяє зберігати важливу інформацію та зменшує обсяг даних для подальшого аналізу [8].

Застосування в обробці зображень:

Основне застосування CNN полягає у роботі з зображеннями. Вони використовуються для розпізнавання об'єктів на зображеннях, класифікації зображень, виявлення об'єктів, відстеження об'єктів на відео та багато іншого.

Використання в інших областях:

Крім обробки зображень, CNN також можуть бути застосовані в інших областях, де даними є послідовності. Наприклад, вони можуть бути корисними для аналізу тексту, обробки аудіо- та відеоданих, а також для аналізу геномних послідовностей та інших послідовних даних.

Висока точність та ефективність:

CNN часто демонструють вражаючі результати в багатьох завданнях, особливо в обробці зображень. Вони можуть автоматично виявляти та класифікувати об'єкти на зображеннях з високою точністю, що робить їх популярним вибором для багатьох додатків [9].

У медичному моделюванні згорткові нейронні мережі (CNN) можуть бути використані для різних завдань, які вимагають аналізу медичних даних та зображень. Ось деякі способи, які вони можуть бути застосовані в цій області:

Розпізнавання патологій на медичних зображеннях: Згорткові нейронні мережі можуть бути використані для автоматичного виявлення та класифікації

патологій на медичних зображеннях, таких як рентгенівські знімки, комп'ютерні томографії (КТ) та магнітно-резонансна томографія (МРТ). Наприклад, вони можуть допомагати у виявленні раку на рентгенівських знімках легень або мозку на МРТ-зображеннях [8].

Сегментація органів та тканин: CNN можуть використовуватися для автоматичної сегментації органів та тканин на медичних зображеннях. Це може бути корисним для планування лікування, наприклад, для планування хірургічних втручань.

Діагностування хвороб: Використання згорткових нейронних мереж для діагностування хвороб може включати аналіз симптомів, лабораторних даних та інших клінічних даних для визначення діагнозу або прогнозування хірургічного результату.

Аналіз часових рядів: У медичному моделюванні згорткові нейронні мережі можуть бути використані для аналізу часових рядів, наприклад, для виявлення аномалій у пульсі, електрокардіограмах або інших часових даних, що можуть вказувати на наявність хвороб або стан пацієнта.

Нейронна мережа прямого поширення (Feedforward Neural Network, FNN) (рисунок 2.3) – це тип штучної нейронної мережі, в якій сигнали рухаються в одному напрямку, від вхідних шарів до вихідних, без циклічних зв'язків. Це одна з найпоширеніших і простих архітектур нейронних мереж, яка зазвичай складається з трьох типових шарів: вхідного шару, прихованих шарів і вихідного шару [6].

Зазвичай FNN складається з трьох типових шарів:

Вхідний шар: Цей шар отримує вхідні дані і передає їх далі до прихованих шарів для обробки. Кількість нейронів у цьому шарі залежить від кількості вхідних ознак.

Приховані шари: Ці шари приймають сигнали від попереднього шару, обробляють їх за допомогою ваг та активаційних функцій і передають їх далі

до наступного шару. Кількість та розмір прихованих шарів може варіюватися залежно від складності завдання.

Вихідний шар: Цей шар отримує оброблені дані від останнього прихованого шару і генерує вихід, який може бути результатом прогнозування, класифікації або іншої операції відповідно до завдання.

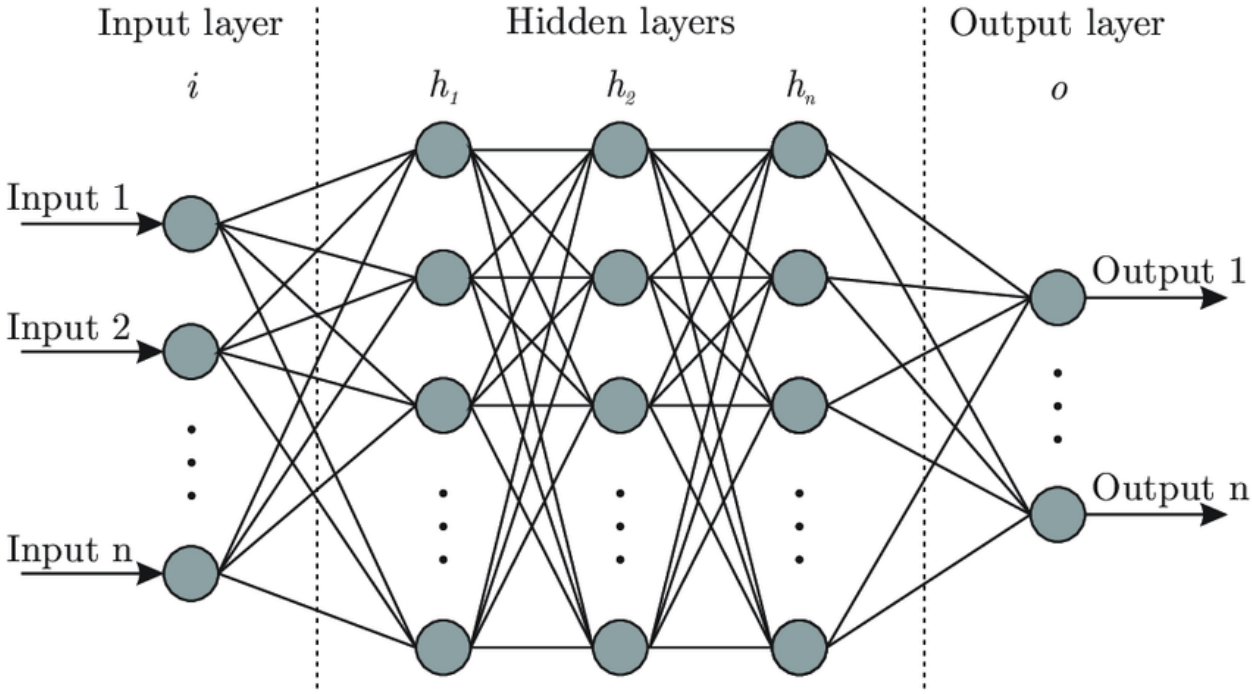

Рисунок 2.3 – Структура нейронної мережі прямого поширення

Нейронні мережі прямого поширення широко використовуються для різних завдань, таких як класифікація тексту, розпізнавання об'єктів на зображеннях, прогнозування часових рядів тощо. Вони можуть бути успішно застосовані і в медичному моделюванні для аналізу клінічних даних, діагностування хвороб або виявлення аномалій.

Обравши рекурентні нейронні мережі (RNN), ми врахували їхню здатність ефективно працювати з послідовними даними, такими як часові ряди симптомів чи медичні звіти. RNN можуть ураховувати залежності в часі і контекстуальні зв'язки між даними, що є важливим для медичного моделювання, де симптоми та хвороби можуть змінюватися з часом. Крім того, вони ефективно працюють з великими обсягами даних та можуть

враховувати складні взаємозв'язки, що допомагає побудувати точні моделі для прогнозування або класифікації у медичному контексті.

2.2 Аналіз структура рекурентної нейронної мережі LSTM

Структурно, RNN складається з одного або кількох рекурентних шарів, де кожен шар має певну кількість нейронів (рекурентних одиниць). Кожен нейрон у рекурентному шарі приймає вхідні дані, власний попередній вихід та попередній виходи інших нейронів у цьому ж шарі. Ця структура дозволяє мережі зберігати попередні стани і використовувати їх для прийняття рішень про поточний вихід (рисунок 2.4).

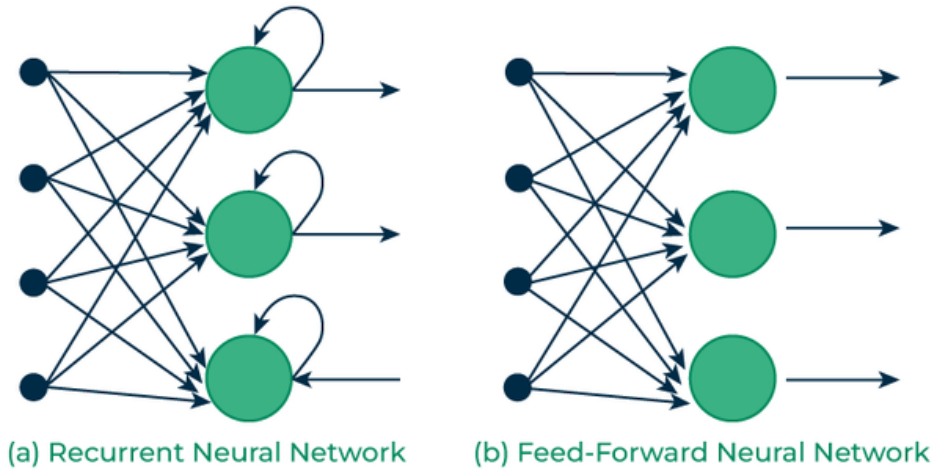

Рисунок 2.4 — RNN і FNN мережі

Математична модель RNN включає в себе набір рівнянь, що визначають співвідношення між входами, вагами, зсувами та активаційними функціями нейронів у кожному шарі. Зазвичай використовуються активаційні функції, такі як гіперболічний тангенс або релу, для нелінійної обробки вхідних даних та вагових коефіцієнтів.

Плюси рекурентних нейронних мереж (RNN) полягають у їхній здатності до моделювання послідовних залежностей у даних та роботі з даними різної довжини. Ось докладніші пояснення:

Моделювання послідовних залежностей: RNN добре пристосовані для роботи з послідовними даними, де кожен елемент вхідної послідовності може бути пов'язаний зі своїми попередниками та наступниками. Це робить їх ефективними для завдань, таких як аналіз часових рядів, обробка природної мови та машинний переклад.

Робота з даними різної довжини: Оскільки RNN можуть обробляти послідовності будь-якої довжини, вони підходять для завдань, де довжина вхідних даних може змінюватися. Це дозволяє їм працювати з даними різного формату та розміру, забезпечуючи гнучкість в застосуванні.

Незважаючи на переваги, у рекурентних нейронних мережах є деякі недоліки, які варто врахувати:

Проблема зникновення або вибуху градієнта: Під час тренування RNN градієнти можуть швидко зростати або спадати, що призводить до нестабільності навчання. Це може ускладнити оптимізацію параметрів мережі та призвести до погіршення її якості.

Обмежена здатність до врахування довготермінових залежностей: У RNN є обмежена пам'ять, тому вони можуть мати складності з врахуванням довготермінових залежностей у даних. Це може призвести до труднощів у моделюванні складних завдань, де важливо врахувати контекст на великих відстанях.

Навчання рекурентних нейронних мереж (RNN) вимагає особливих підходів та алгоритмів через їхню особливу структуру та здатність до роботи з послідовностями. Ось докладніші пояснення:

Алгоритм зворотнього поширення помилки через час (BPTT): Для навчання RNN використовується алгоритм зворотнього поширення помилки через час, який є модифікацією алгоритму зворотнього поширення помилки, адаптованого до послідовностей даних. BPTT враховує часову залежність між вхідними та вихідними даними, дозволяючи ефективно навчати мережу враховуючи історію вхідних даних.

Обсяг даних для навчання: RNN вимагають більш об'ємних даних для навчання, особливо при роботі зі складними завданнями, оскільки вони намагаються вивчити складні залежності між вхідними та вихідними даними. Додатковий обсяг даних допомагає запобігти перенавчанню та покращити узагальнюючу здатність моделі.

Основні аспекти використання RNN включають:

Підбір гіперпараметрів: Вибір правильних значень гіперпараметрів, таких як кількість шарів, розмір шарів, швидкість навчання тощо, є важливою частиною процесу розробки RNN. Недоліки у підборі гіперпараметрів можуть призвести до неправильної роботи мережі та погіршення її результатів.

Управління довжиною послідовності: Керування довжиною послідовності важливо для ефективності роботи RNN. Довгі послідовності можуть призвести до проблем зі зникненням градієнта або вибуху, тому важливо розглядати стратегії обрізання або підсилення даних.

Уникання перенавчання: Важливо використовувати методи регуляризації та перевірки для уникнення перенавчання RNN. Це включає в себе використання технік, таких як випадковий вимкнення (dropout), а також використання валідаційної вибірки для оцінки ефективності моделі на незалежних даних.

Вибір відповідних функцій активації та оптимізаційних алгоритмів: Вибір підходящих функцій активації та оптимізаційних алгоритмів впливає на швидкість та якість навчання RNN. Наприклад, використання функцій активації, які добре працюють з послідовностями, може покращити результати, а вибір оптимізаційного алгоритму може вплинути на швидкість збіжності навчання.

RNN не вивчає довгострокові залежності і страждає від проблем з короткостроковою пам'яттю. LSTM може ефективно подолати ці проблеми [10].

Довга короткочасна пам'ять, відома як LSTM, є різновидом рекурентної нейронної мережі, здатної вивчати довготривалі залежності. Сепп Хохрайтер і Юрген Шмідхубер [10] представили її для вирішення проблеми довгострокових залежностей. LSTM мережі мають такий самий потік керування як і рекурентна нейронна мережа. Вона обробляє дані послідовно, передаючи інформацію в міру її поширення вперед. На відміну від RNN, операції всередині комірки LSTM дозволяють мережі забувати або зберігати інформацію протягом тривалого періоду часу.

LSTM комірки. Основною концепцією LSTM є стан комірки та її різні вентилі. На рисунку 2.5 показана базова комірка LSTM.

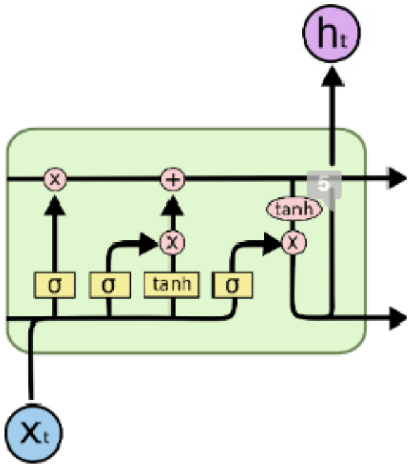

Рисунок 2.5 — Комірка LSTM

Стан комірки поводиться як довгострокова пам'ять, яка зберігає довгострокові залежності та патерни. Він передає відносну інформацію по всьому ланцюжку послідовності. Навіть інформація з попередніх часових кроків може враховуватися аж до останнього часового кроку, таким чином зменшуючи ефект короткочасної пам'яті. У стані клітини градієнт стабілізується завдяки механізму воріт. Тому менша ймовірність виникнення проблеми зникаючого градієнта, яка є основною проблемою при навчанні RNN. Стан комірки складається з трьох воріт: вхідних, забування та вихідних. Ворота – це механізми, які контролюють, як інформація надходить у стан

комірки та виходить з нього. Вони дозволяють або забороняють проходження даних під час процесу навчання в залежності від важливості даних для прогнозування. Ворота складаються з сигмоїдного шару нейронної мережі з операцією точкового множення.

Повторювана структура LSTM.

На рисунку 2.6 показано ілюстрацію повторюваного LSTM модуля.

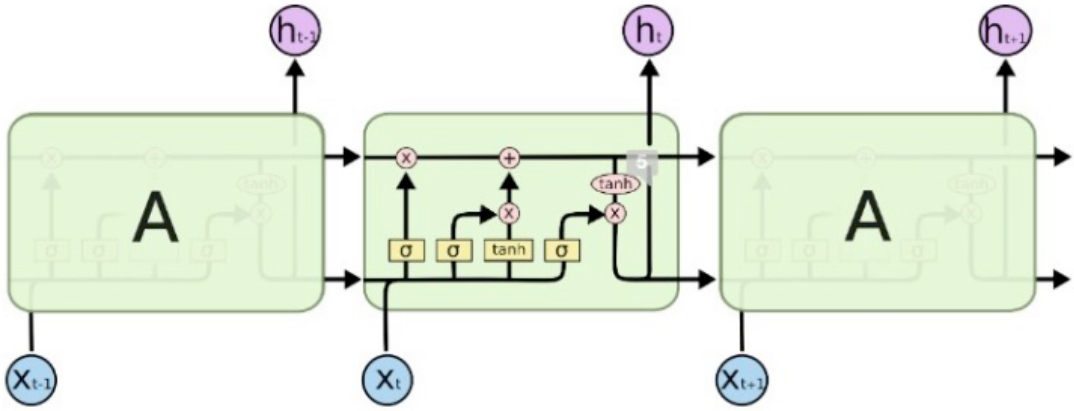

Рисунок 2.6 — Повторювана LSTM структура

Стекова LSTM.

Класична LSTM модель складається з одного прихованого LSTM шару, за яким слідує вихідний шар прямого поширення. LSTM модель з більш ніж одним LSTM шаром є стековою LSTM архітектурою. На рисунку 2.7 показано приклад стекової LSTM моделі. Додавання прихованих LSTM шарів робить модель глибшою. Збільшення кількості прихованих шарів нейронних мереж зазвичай сприяє збільшенню точності. Чим більше прихованих шарів, тим більше кожен шар може рекомбінувати вивчене представлення з попередніх шарів і створювати нові представлення з вищим ступенем абстракції [10].

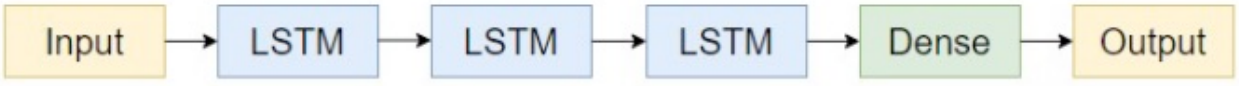

Рисунок 2.7 — Стекова LSTM

Двонаправлена LSTM.

Двонаправлена LSTM – це тип рекурентної нейронної мережі, яка навчається одночасно в двох напрямках (вперед і назад) і може використовувати інформацію з обох сторін. На рисунку. 2.8 показано двонаправлена LSTM. Вона дублює шар LSTM, так що тепер є два шари, які вирівняні поруч.

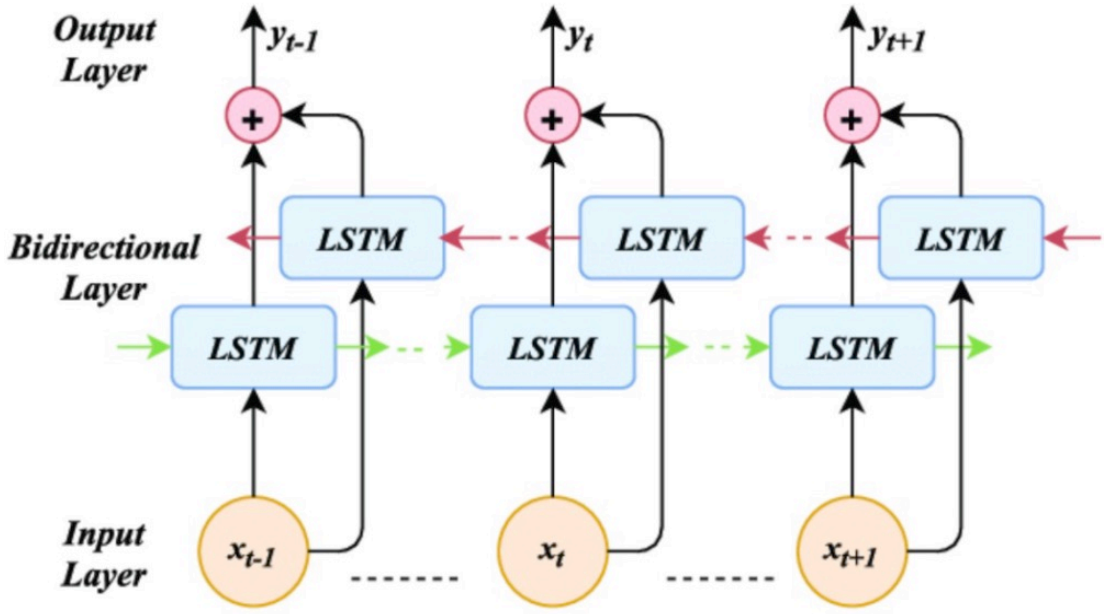

Рисунок 2.8 — Розгорнута двонаправлена LSTM

Двонаправлені LSTM структури використовують дві незалежні LSTM мережі. Одна мережа навчається на прямій послідовності даних. Інша навчається на зворотній послідовності даних. Окремі комірки LSTM можуть вивчати контекст з майбутньої інформації, надаючи обернену копію вхідних даних. Це дозволяє мережі розуміти взаємозв'язки між значеннями в обох напрямках. У будь-який момент часу вихід кожного прямого і зворотного виходів комірки об'єднуються за допомогою функції активації для отримання єдиного виходу. Таким чином, мережа в будь-який момент часу може обробляти як минулу, так і майбутню інформацію, на відміну від односпрямованої LSTM, яка може обробляти лише минулу інформацію.

2.3 Розробка алгоритму роботи програмного модуля діагностування захворювань

Основний компонент програми для визначення захворювання за симптомами базується на рекурентній нейронній мережі LSTM. Розглянемо ключові етапи розробки цього компоненту:

Завантаження даних: Перший етап включає можливість завантаження даних про симптоми з різних джерел або існуючих наборів даних.

Підготовка даних: На цьому етапі здійснюється підготовка вхідних даних. Вона може включати очищення, перетворення формату даних, кодування категоріальних змінних та розділення набору даних на тренувальний і тестовий.

Побудова моделі та навчання: Основний компонент програми містить код для побудови та навчання рекурентної нейронної мережі LSTM. Це включає визначення архітектури мережі, компіляцію з вибором оптимального оптимізатора та функції втрат, а також навчання моделі на тренувальних даних.

Тестування моделі: Після навчання модель тестується на тестовому наборі даних для оцінки її ефективності та точності в класифікації захворювань за симптомами.

Особливості цього компоненту включають:

Використання рекурентних нейронних мереж: Оскільки задача полягає у класифікації захворювань за симптомами, використання рекурентної мережі LSTM дозволяє ефективно моделювати залежності між послідовними симптомами та їхнім впливом на діагноз.

Вибір оптимального оптимізатора та функції втрат: Важливо підібрати оптимальний оптимізатор та функцію втрат для ефективного навчання моделі.

Візуалізація результатів: Для оцінки ефективності моделі можна використовувати графіки, такі як графіки втрат та точності, матриця

невідповідності та інші. Це допомагає зрозуміти, наскільки добре модель класифікує захворювання за симптомами. Узагальнену взаємодію компонентів системи можна побачити на рисунку 2.9.

Рисунок 2.9 – Взаємодія компонентів програмного засобу

Загальний алгоритм роботи основного компоненту програми для визначення захворювання за симптомами з використанням рекурентної нейронної мережі можна описати наступним чином:

— Збір та завантаження даних: Збираються дані про симптоми, які пов'язані з різними захворюваннями. Ці дані можуть бути отримані з медичних джерел, баз даних або інших джерел, які містять інформацію про клінічні симптоми.

— Підготовка даних: Очищення та передобробка даних, включаючи кодування категоріальних змінних, нормалізацію даних та розділення набору даних на тренувальний і тестовий.

— Побудова рекурентної нейронної мережі: Визначення архітектури мережі, включаючи кількість і розмір шарів LSTM, кількість вузлів та функції активації. Після цього компілюється модель з вибором оптимального оптимізатора та функції втрат.

— Навчання моделі: Модель навчається на тренувальних даних, використовуючи алгоритм зворотнього поширення помилки через час (BPTT). Під час навчання використовується пакетна обробка для покращення ефективності навчання [9].

— Відстеження навчання та відновлення: Використання методів відстеження навчання, таких як Early Stopping, для зупинки навчання моделі, якщо виявлено перенавчання або недооптимізацію.

— Тестування та оцінка моделі: Модель тестується на тестовому наборі даних для оцінки її точності та ефективності. Оцінюються метрики якості, такі як точність, чутливість, специфічність та F1-оцінка.

— Використання моделі в реальному часі: Оптимально навчена модель може використовуватися для класифікації захворювань за симптомами в реальному часі, шляхом введення нових даних та отримання відповідного діагнозу від моделі [11].

Поміж іншими аспектами, крім головного компонента, важливо забезпечити спілкування між цим компонентом та користувачем.

Деталізовану взаємодію користувача з навченою моделлю можна побачити на рисунку 2.10.

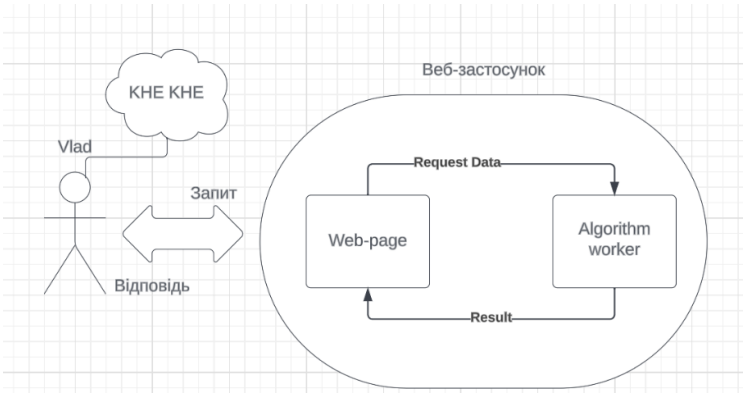

Рисунок 2.10 – Взаємодія користувача з нейромережею

Більш детальний алгоритм роботи основного модуля наведено на рисунку 2.11.

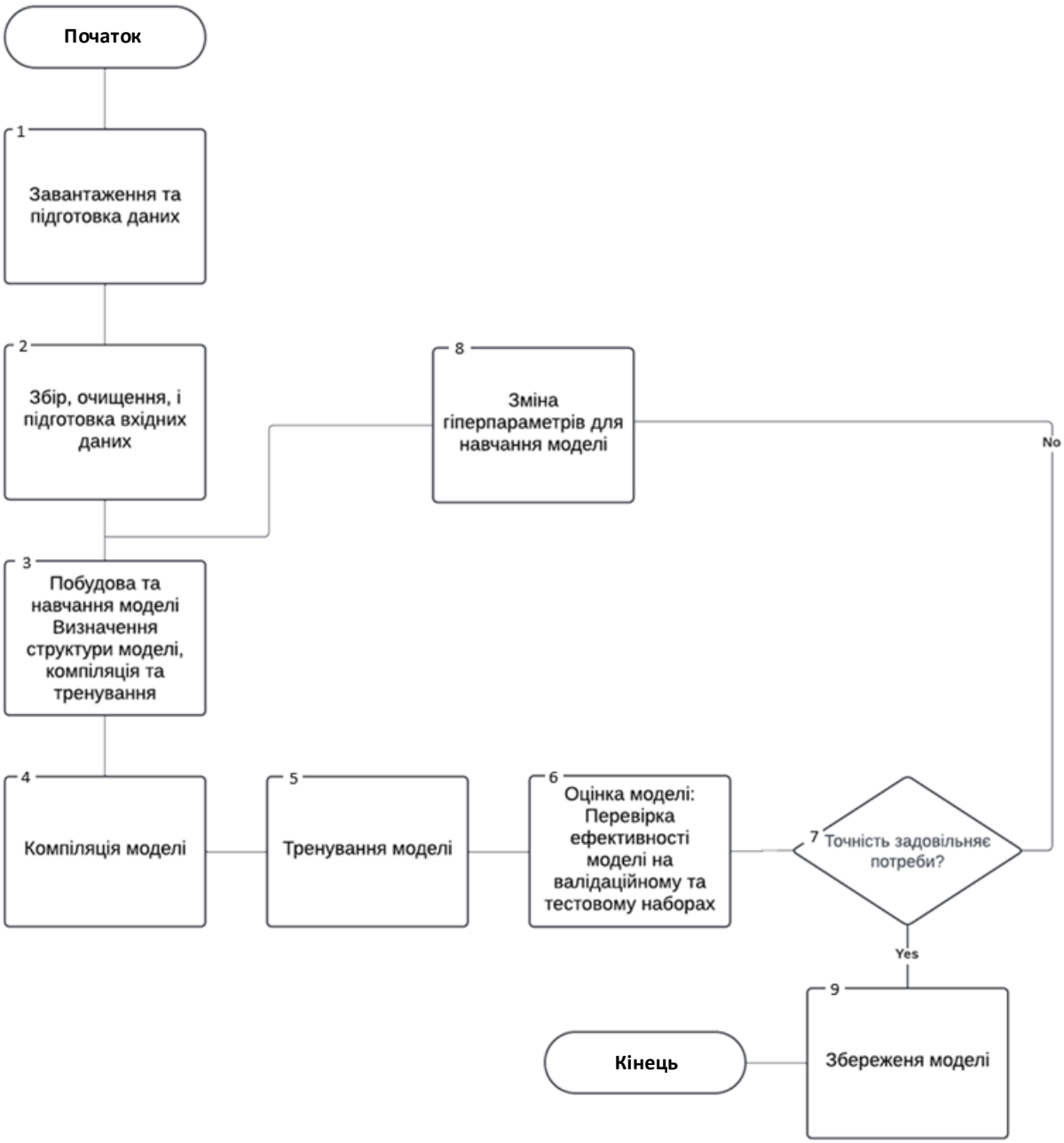

Рисунок 2.11 — Схема алгоритму роботи програмного модуля діагностування захворювань

Окрім базового моніторингу тренування нейронної мережі, буде розроблено додатковий модуль тестування. Цей модуль включатиме декілька тестів для остаточної оцінки готовності моделі до інтеграції в API. Користувачі ж зможуть взаємодіяти з API через веб-інтерфейс, що забезпечить швидкість та зручність.

2.5 Висновок до розділу 2

У розділі було обгрунтовано вибір архітектури нейронної мережі LSTM для діагностування захворювань за симптомами. Було проаналізовано структуру рекурентної нейронної мережі LSTM, принципи її функціонування та навчання. Також було розроблено алгоритм роботи програмного модуля діагностування захворювань по симптомах на основі нейронної мережі LSTM.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ ДІАГНОСТУВАННЯ ЗАХВОРЮВАНЬ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

3.1 Обґрунтування вибору мови та середовища програмування

Проведемо порівняльний аналіз мов програмування та інтегрованих середовищ програмування у формі таблиць 3.1 та 3.2.

Таблиця 3.1 Порівняльний аналіз мов програмування

Особливість	Python	R
Використання	Використовується для широкого спектру завдань програмування, включаючи наукові обчислення, веб-розробку, автоматизацію процесів та інші. Є дуже популярним в області машинного навчання та глибокого навчання.	Часто використовується для статистичного аналізу даних та наукових досліджень. Має розширені можливості в аналізі даних, включаючи візуалізацію та моделювання.
Простота вивчення	Вважається однією з найбільш доступних мов програмування для новачків. Синтаксис Python легкий для читання та розуміння.	Мова R спеціалізується на аналізі даних, тому вивчення може бути складнішим для тих, хто не має досвіду в статистиці або аналізі даних.
Бібліотеки та фреймворки	Має широкий вибір бібліотек та фреймворків для машинного навчання та глибокого навчання, таких як TensorFlow, PyTorch, Keras, scikit-learn.	R має потужні бібліотеки для статистичного аналізу та візуалізації даних, такі як ggplot2, dplyr і caret.
Платформи розповсюдження	Є популярним у широкому спектрі галузей та має значну кількість відкритих проектів та спільнот. Рахується мультиплатформеним	Часто використовується в академічних та дослідницьких галузях, а також серед професіоналів у сфері статистики та аналізу даних. Рахується мультиплатформеним

Таблиця 3.2 Порівняльний аналіз середовищ програмування

Особливість	PyCharm	Visual Studio Code
Доступність	Доступний у платних (Professional та Ultimate) та безкоштовній (Community) версіях.	Безкоштовний та відкритий код.локальних комп'ютерах.
Оптимізація для машинного навчання	Має плагін PyTorch та інші інструменти для розробки ML/DL.	Має розширення для TensorFlow, Jupyter Notebooks та інших ML/DL бібліотек.
Командний режим роботи	Підтримує спільну роботу над проектами через Git	Має вбудований Git та підтримку GitHub.
Інтеграція з іншими сервісами	Інтегрується з Jira, Docker, Kubernetes та іншими інструментами.	Має безліч розширень для інтеграції з різними сервісами.
Відладка та підтримка	Має потужний вбудований відладчик та доступ до підтримки JetBrains.	Має базовий відладчик та покладається на спільноту для підтримки.

Після ретельного аналізу мов програмування Python та R, а також середовищ розробки PyCharm та Visual Studio Code, виявлено, що Python та PyCharm є найкращим вибором для виконання поставленого завдання [12].

Python має ряд переваг, які роблять його ідеальним для наукових досліджень, аналізу даних, машинного навчання та розробки прототипів програмного забезпечення:

Широкий спектр інструментів: Python володіє великою кількістю бібліотек та фреймворків для розробки нейромереж, що робить його потужним інструментом для дослідників та розробників [12].

Простота використання: Синтаксис Python лаконічний та зрозумілий, що робить його доступним для людей з різним досвідом програмування.

Універсальність: Python використовується не лише для наукових досліджень, але й для веб-розробки, системного адміністрування та інших сфер.

PyCharm також має ряд переваг, які роблять його зручним середовищем для розробки на Python:

Просунута підтримка Python: PyCharm пропонує широкий спектр функцій для полегшення розробки на Python, таких як автодоповнення, перевірка коду та відладка.

Хмарні обчислення: PyCharm може використовувати хмарні потужності для виконання ресурсомістких завдань, таких як навчання нейромереж.

Інтеграція з іншими інструментами: PyCharm інтегрується з системами контролю версій, Git, та іншими інструментами, що робить процес розробки більш ефективним [13].

Виходячи з вищезазначеного, Python та PyCharm є кращим вибором для виконання поставленого завдання завдяки своїй універсальності, простоті використання та потужності.

3.2 Опис набору даних симптомів захворювань для навчання та тестування модуля

Було знайдено декілька датасетів з медичними даними симптомів і відповідних їм хвороб, але найбільш популярним є датасет [14]. Він налічує 132 симптоми, 41 хворобу, кількість навчальних прикладів – 4428, кількість тестових прикладів – 492 (рисунок 3.1)

```
Number of Features in a Input Vector:  132
Number of Classes(Disease):  41
Number of Vocabulary:  132
Number of Train Samples:  4428
Number of Test Samples:  492
```

Рисунок 3.1 — Основні параметри датасету захворювань та симптомів

Симптоми (symptoms) — це конкретні ознаки або прояви захворювань у пацієнтів.

Хвороба (prognosis) — в датасеті відображає діагноз або прогноз захворювання, який ставлять лікарі пацієнтам на підставі їх симптомів та інших характеристик.

Кожен рядок у датасеті містить інформацію про симптоми, які спостерігаються у пацієнта, а також прогнозовану хворобу, яка може бути результатом аналізу цих симптомів.

У датасеті ця характеристика представляє собою текстове значення, що вказує на конкретну хворобу або діагноз, наприклад "Грибкова інфекція", "Грип", "Ангіна" тощо (рисунок 3.2).

```
Disease Dictionary:
{0: '(vertigo) Paroymsal  Positional Vertigo', 1: 'AIDS', 2: 'Acne', 3: 'Alcoholic hepat
itis', 4: 'Allergy', 5: 'Arthritis', 6: 'Bronchial Asthma', 7: 'Cervical spondylosis', 8
: 'Chicken pox', 9: 'Chronic cholestasis', 10: 'Common Cold', 11: 'Dengue', 12: 'Diabete
s ', 13: 'Dimorphic hemmorhoids(piles)', 14: 'Drug Reaction', 15: 'Fungal infection', 16
: 'GERD', 17: 'Gastroenteritis', 18: 'Heart attack', 19: 'Hepatitis B', 20: 'Hepatitis C
', 21: 'Hepatitis D', 22: 'Hepatitis E', 23: 'Hypertension ', 24: 'Hyperthyroidism', 25:
'Hypoglycemia', 26: 'Hypothyroidism', 27: 'Impetigo', 28: 'Jaundice', 29: 'Malaria', 30:
'Migraine', 31: 'Osteoarthristis', 32: 'Paralysis (brain hemorrhage)', 33: 'Peptic ulcer
diseae', 34: 'Pneumonia', 35: 'Psoriasis', 36: 'Tuberculosis', 37: 'Typhoid', 38: 'Urina
ry tract infection', 39: 'Varicose veins', 40: 'hepatitis A'}
[ 4 11 36 ... 12 15 33]
```

Рисунок 3.2 — Перелік хвороб датасету

А перелік можливих симптомів подано на рисунку 3.3.

Структура даних. Дані структуровані та організовані у вигляді таблиці для зручності обробки.

```
Features Vocabulary list and their index:
 [(0, 'abdominal_pain'), (1, 'abnormal_menstruation'), (2, 'acidity'), (3, 'acute_liver_
failure'), (4, 'altered_sensorium'), (5, 'anxiety'), (6, 'back_pain'), (7, 'belly_pain')
, (8, 'blackheads'), (9, 'bladder_discomfort'), (10, 'blister'), (11, 'blood_in_sputum')
, (12, 'bloody_stool'), (13, 'blurred_and_distorted_vision'), (14, 'breathlessness'), (1
5, 'brittle_nails'), (16, 'bruising'), (17, 'burning_micturition'), (18, 'chest_pain'),
(19, 'chills'), (20, 'cold_hands_and_feets'), (21, 'coma'), (22, 'congestion'), (23, 'co
nstipation'), (24, 'continuous_feel_of_urine'), (25, 'continuous_sneezing'), (26, 'cough
'), (27, 'cramps'), (28, 'dark_urine'), (29, 'dehydration'), (30, 'depression'), (31, 'd
iarrhoea'), (32, 'dischromic_patches'), (33, 'distention_of_abdomen'), (34, 'dizziness')
, (35, 'drying_and_tingling_lips'), (36, 'enlarged_thyroid'), (37, 'excessive_hunger'),
(38, 'extra_marital_contacts'), (39, 'family_history'), (40, 'fast_heart_rate'), (41, 'f
atigue'), (42, 'fluid_overload'), (43, 'foul_smell_of_urine'), (44, 'headache'), (45, 'h
igh_fever'), (46, 'hip_joint_pain'), (47, 'history_of_alcohol_consumption'), (48, 'incre
ased_appetite'), (49, 'indigestion'), (50, 'inflammatory_nails'), (51, 'internal_itching
'), (52, 'irregular_sugar_level'), (53, 'irritability'), (54, 'irritation_in_anus'), (55
, 'itching'), (56, 'joint_pain'), (57, 'knee_pain'), (58, 'lack_of_concentration'), (59,
'lethargy'), (60, 'loss_of_appetite'), (61, 'loss_of_balance'), (62, 'loss_of_smell'), (
63, 'malaise'), (64, 'mild_fever'), (65, 'mood_swings'), (66, 'movement_stiffness'), (67
, 'mucoid_sputum'), (68, 'muscle_pain'), (69, 'muscle_wasting'), (70, 'muscle_weakness')
, (71, 'nausea'), (72, 'neck_pain'), (73, 'nodal_skin_eruptions'), (74, 'obesity'), (75,
'pain_behind_the_eyes'), (76, 'pain_during_bowel_movements'), (77, 'pain_in_anal_region'
), (78, 'painful_walking'), (79, 'palpitations'), (80, 'passage_of_gases'), (81, 'patche
s_in_throat'), (82, 'phlegm'), (83, 'polyuria'), (84, 'prominent_veins_on_calf'), (85, '
puffy_face_and_eyes'), (86, 'pus_filled_pimples'), (87, 'receiving_blood_transfusion'),
(88, 'receiving_unsterile_injections'), (89, 'red_sore_around_nose'), (90, 'red_spots_ov
er_body'), (91, 'redness_of_eyes'), (92, 'restlessness'), (93, 'runny_nose'), (94, 'rust
y_sputum'), (95, 'scurring'), (96, 'shivering'), (97, 'silver_like_dusting'), (98, 'sinu
s_pressure'), (99, 'skin_peeling'), (100, 'skin_rash'), (101, 'slurred_speech'), (102, '
small_dents_in_nails'), (103, 'spinning_movements'), (104, 'spotting_urination'), (105,
'stiff_neck'), (106, 'stomach_bleeding'), (107, 'stomach_pain'), (108, 'sunken_eyes'), (
109, 'sweating'), (110, 'swelled_lymph_nodes'), (111, 'swelling_joints'), (112, 'swellin
g_of_stomach'), (113, 'swollen_blood_vessels'), (114, 'swollen_extremeties'), (115, 'swo
llen_legs'), (116, 'throat_irritation'), (117, 'toxic_look_'), (118, 'typhos'), (119, 'u
lcers_on_tongue'), (120, 'unsteadiness'), (121, 'visual_disturbances'), (122, 'vomiting'
), (123, 'watering_from_eyes'), (124, 'weakness_in_limbs'), (125, 'weakness_of_one_body_
side'), (126, 'weight_gain'), (127, 'weight_loss'), (128, 'yellow_crust_ooze'), (129, 'y
ellow_urine'), (130, 'yellowing_of_eyes'), (131, 'yellowish_skin')]
```

Рисунок 3.3 — Перелік симптомів датасету

Ці таблиці датасету представлені у форматі файлів CSV або JSON для зручності обробки та аналізу.

Із рисунку 3.3 також видно, що датасет розбито на 2 частини: навчальні приклади (Train Samples: 4428) та тестові приклади (Test Samples: 492). Склад навчального датасету по хворобах представлено на рисунку 3.4. Там видно, що розподіл прикладів по хворобах є більш-менш рівномірним, тобто по кожній хворобі є від 99 до 114 прикладів (в середньому по 107 прикладів).

Each Class with their number of Train Samples taken for train this model:

	Classes	Number of Train Samples
0	(vertigo) Paroymsal Positional Vertigo	107
1	AIDS	109
2	Acne	108
3	Alcoholic hepatitis	107
4	Allergy	107
5	Arthritis	106
6	Bronchial Asthma	112
7	Cervical spondylosis	109
8	Chicken pox	102
9	Chronic cholestasis	107
10	Common Cold	106
11	Dengue	106
12	Diabetes	107
13	Dimorphic hemmorhoids(piles)	99
14	Drug Reaction	101
15	Fungal infection	110
16	GERD	107
17	Gastroenteritis	105
18	Heart attack	111
19	Hepatitis B	110
20	Hepatitis C	116
21	Hepatitis D	108
22	Hepatitis E	110
23	Hypertension	103
24	Hyperthyroidism	105
25	Hypoglycemia	111
26	Hypothyroidism	108
27	Impetigo	108
28	Jaundice	109
29	Malaria	106
30	Migraine	110
31	Osteoarthristis	106
32	Paralysis (brain hemorrhage)	112
33	Peptic ulcer diseae	112
34	Pneumonia	110
35	Psoriasis	104
36	Tuberculosis	111
37	Typhoid	106
38	Urinary tract infection	113
39	Varicose veins	110
40	hepatitis A	114

Рисунок 3.4 — Склад навчального датасету по хворобах

Рівномірність розподілу прикладів по хворобах є дуже корисною при оцінюванні точності мультикласової класифікації.

3.3 Програмна реалізація модуля діагностування захворювань

Розробка програмного модуля діагностування захворювання складалася з кількох послідовних етапів:

Завантаження та підготовка даних:

Дані завантажуються з CSV-файлу за допомогою пакету Pandas [15].

Розділення даних на ознаки (X) та цільову змінну (y) важливо для підготовки даних перед використанням їх у моделі машинного навчання. Ось деякі причини, чому це робиться:

Розділення функцій і міток: Ознаки (X) представляють собою незалежні змінні, за допомогою яких модель буде робити прогнози. Цільова змінна (y) - це змінна, яку ми намагаємося передбачити за допомогою моделі.

Розділення цих змінних допомагає визначити, які дані будуть використовуватися для навчання моделі, а які для перевірки її ефективності.

Стандартизація даних: Нормалізація або стандартизація ознак (X) до певного діапазону значень допомагає уникнути проблем, пов'язаних з великими різницями в масштабі між різними ознаками. Це полегшує процес оптимізації моделі, оскільки допомагає збільшити швидкість навчання та покращує загальну стабільність моделі.

Таким чином, розділення даних на ознаки та цільову змінну, а також їх нормалізація, є важливими етапами перед використанням цих даних для навчання моделі машинного навчання.

Побудова моделі:

Використання послідовної моделі Keras Sequential [16] дозволяє легко створювати моделі нейронних мереж за допомогою послідовного додавання шарів. Ось які переваги і навіщо в даному випадку використовується послідовна модель з додаванням трьох шарів LSTM та шарів викидання:

Легкість створення моделі: Використання Sequential дозволяє додавати шари один за одним в послідовному порядку без потреби явного визначення входів для кожного шару.

Шари LSTM для роботи з послідовностями: LSTM (Long Short-Term Memory) — це спеціальний тип рекурентного шару, призначений для роботи з послідовністю даних, таких як текст або часові ряди. Використання LSTM дозволяє моделі враховувати залежності в послідовних даних та здійснювати прогнози на їх основі.

Зменшення кількості нейронів у кожному наступному шарі: Зменшення кількості нейронів у кожному наступному шарі дозволяє зменшити складність

моделі та контролювати перенавчання. Це допомагає уникнути перевантаження моделі та покращити її загальну здатність до узагальнення.

Додавання шарів викидання для запобігання перенавчанню: Шари викидання (Dropout) випадковим чином вимикають нейрони під час навчання, тим самим ускладнюючи процес навчання моделі. Це допомагає уникнути перенавчання та покращити здатність моделі до узагальнення на нові дані.

Фрагмент коду побудови моделі:

```
model = Sequential()
model.add(LSTM(256, input_shape=(X_train.shape[1], 1), return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(128, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(64))
model.add(Dense(len(label_encoder.classes_), activation='softmax'))
```

Компіляція моделі:

Використання оптимізатора Adam зі зменшеною швидкістю навчання, а також використання функції втрати 'sparse_categorical_crossentropy' та метрики точності 'accuracy' має наступні переваги та призначення:

Оптимізатор Adam зі зменшеною швидкістю навчання: Оптимізатор Adam — це швидкий та ефективний алгоритм оптимізації, який адаптується до швидкості навчання для кожного параметра моделі. Зменшення швидкості навчання допомагає уникнути перехоплення глобального мінімуму та забезпечити стабільніше навчання моделі.

Фрагмент коду компіляції моделі:

```
# Компіляція моделі з оптимізатором Adam і зменшеною швидкістю навчання
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)
model.compile(optimizer=optimizer, loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

Функція втрати 'sparse_categorical_crossentropy': Ця функція втрати використовується для задач класифікації з багатьма класами (багатокласовою класифікацією). Вона обчислює різницю між фактичними та передбаченими ймовірностями для кожного класу та використовується для оцінки точності передбачень моделі.

Метрика точності 'accuracy' [17]: Ця метрика використовується для визначення відсотка правильно класифікованих випадків у загальній кількості випадків. Вона дозволяє оцінити ефективність моделі шляхом вимірювання того, наскільки точно модель передбачає класи.

Навчання моделі:

Перетворення даних для LSTM за допомогою np.expand_dims: LSTM шари очікують вхідні дані у вигляді тривимірного тензора, де розмірність останнього виміру відповідає часовим крокам, а інші розмірності відповідають розміру вхідних ознак. Функція np.expand_dims використовується для додавання нового виміру до вхідних даних, тим самим перетворюючи їх у формат, придатний для використання в LSTM шарах.

Метод навчання fit: Метод fit використовується для навчання моделі на тренувальних даних. Під час навчання модель адаптується до вхідних даних шляхом оновлення внутрішніх параметрів, що дозволяє їй робити більш точні прогнози на майбутніх даних.

Використання Early Stopping: Early Stopping — це стратегія, яка дозволяє автоматично зупинити процес навчання, якщо метрика, яка відстежується на валідаційному наборі, не покращується впродовж певної кількості епох (періодів навчання). Це допомагає уникнути перенавчання та забезпечує, що модель не продовжує навчання, коли вона перестає вдосконалюватися на валідаційних даних.

Фрагмент коду навчання моделі:

```
# Використання Early Stopping для зупинки навчання, якщо точність на валідаційному
наборі перестає покращуватися
early_stopping = EarlyStopping(monitor='val_accuracy', patience=10,
restore_best_weights=True)

# Навчання моделі та збереження історії
history = model.fit(X_train_lstm, y_train, epochs=150, batch_size=64,
validation_data=(X_test_lstm, y_test), callbacks=[early_stopping])
```

Оцінка моделі на тестовому наборі:

Після завершення навчання модель оцінюється на тестовому наборі даних, який не брав участі в навчанні. Це дозволяє оцінити загальну ефективність моделі на нових, раніше не бачених даних. Оцінка включає визначення точності моделі (як часто вона правильно класифікує дані) та втрат (якість передбачень моделі в порівнянні з реальними мітками).

Фрагмент коду оцінки точності моделі на тестовому наборі::

```
# Оцінка точності моделі на тестовому наборі
accuracy = model.evaluate(X_test_lstm, y_test)
print(f"Test Accuracy: {accuracy[1]*100:.2f}%")
print(f"Test Loss: {accuracy[0]:.4f}")
print(f"Model Summary: {model.summary()}")
```

Збереження структури моделі та ваг:

Після навчання моделі та оцінки її ефективності, важливо зберегти структуру моделі разом з її вагами. Це дозволить використовувати навчену модель у майбутньому без необхідності повторного навчання. Збереження моделі і ваг дозволяє легко використовувати їх для класифікації нових даних або інтеграції моделі в інші програмні продукти чи сервіси.

У розробленій моделі нейронної мережі використовується три шари LSTM та один Dense шар. Кожен з LSTM шарів має відповідно 256, 128 і 64 нейронів.

3.4 Висновок до розділу 3

У розділі було обґрунтовано вибір мови програмування Python, середовища розробки PyCharm та спеціалізованих бібліотек TensorFlow, Keras, Sklearn, Numpy, Pandas, Matplotlib для програмної реалізації модуля діагностування захворювань на основі нейронної мережі. Проведено аналіз набору даних симптомів захворювань для навчання та тестування модуля, який містить 4428 навчальних прикладів та 492 тестових приклади. Описано основні етапи програмної реалізація та функціонування модуля діагностування захворювань на основі нейронної мережі, що складається з імпорту бібліотек, завантаження набору даних, побудови моделі запропонованої нейромережі, навчання нейромережі, тестування результатів роботи розробленої нейромережі.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМНОГО МОДУЛЯ ДІАГНОСТУВАННЯ ЗАХВОРЮВАНЬ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

4.1 Тестування програмного модуля діагностування захворювань на основі нейронної мережі

Програмна реалізація модуля діагностування захворювань на основі нейронної мережі представляє собою веб-застосунок, побудований на основі фреймворку Django. Нижче описані ключові етапи та кроки взаємодії з цим застосунком:

1) запустити Python код в інтрпретаторі,
2) написати в консоль команду «python manage.py runserver»,
3) відкрити посилання, яке видасть консоль, воно відкриється в браузері,
4) на сторінці з’явиться форма (рисунок 4.1),
5) заповнити поля форми (рисунок 4.2), а саме – у верхнє поле ввести ім’я пацієнта або ID пацієнта та у п’яти наступних полях вибрати наявні симптоми з випадаючого списку,
6) після заповнення полів, натиснути кнопку «Прогнозувати хворобу» внизу вікна програми,
7) веб-додаток визначить можливий діагноз (рисунок 4.2) на основі введених симптомів і відобразить його в нижній частині сторінки.

Для параметрів в формі не потрібна валідація, тому що вони вибираються саме із випадаючого списку.

Якщо потрібно провести наступне діагностування, то після нового заповнення полів, потрібно знов натиснути кнопку «Прогнозувати хворобу» внизу вікна програмиі веб-додаток видасть нове повідомлення діагнозу.

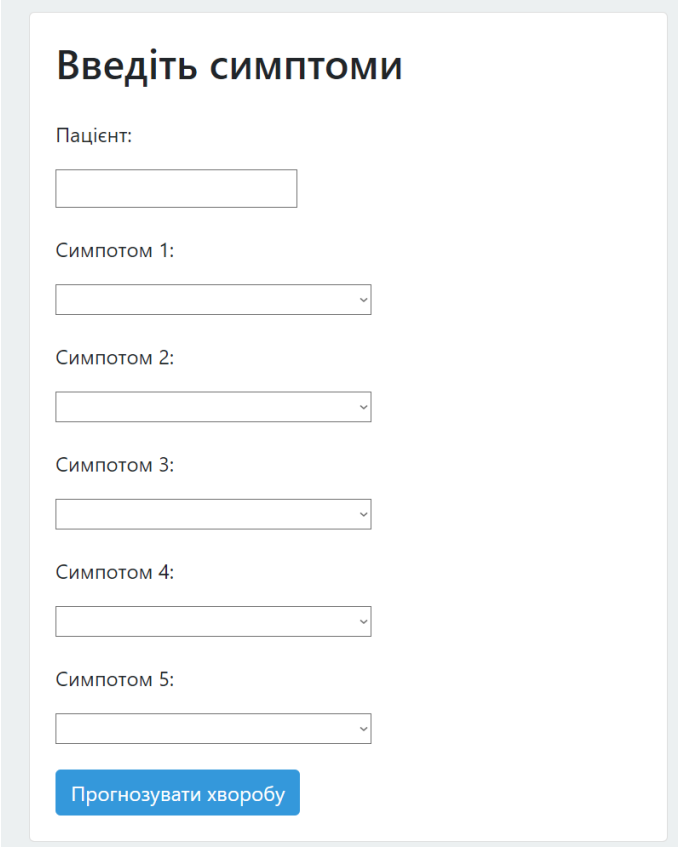

Рисунок 4.1 – Інтерфейс користувача.

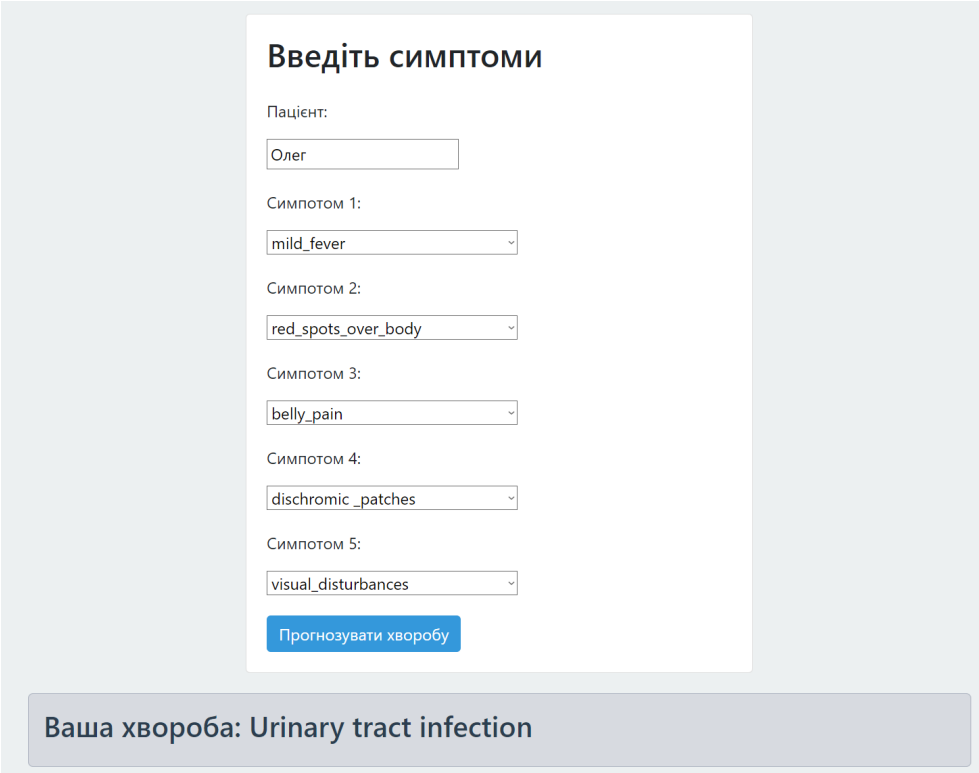

Рисунок 4.2 – Введення даних та отримання результату передбачення

4.2 Аналіз результатів роботи програмного модуля діагностування захворювань на основі нейронної мережі

Аналіз результатів роботи програмного модуля діагностування захворювань на основі нейронної мережі полягав у дослідженні її ефективності та точності в розпізнаванні виду захворювання за симптомами. Нижче наведено основні кроки та результати цього аналізу:

Підготовка тестового набору даних: Для тестування програми був підготовлений набір даних, що містив різноманітні симптоми та відповідні захворювання.

Виконання тестів з використанням тестового набору даних: Програма була запущена на тестовому наборі даних, і для кожного набору симптомів програма намагалася правильно визначити відповідне захворювання.

Порівняння результатів з очікуваними значеннями: Результати, отримані від програми, були порівняні з відомими діагнозами з тестового набору даних. Було оцінено, наскільки точно програма змогла розпізнати симптоми та призначити правильний діагноз.

Обчислення метрик продуктивності: Були обчислені різні метрики продуктивності, такі як точність (accuracy), чутливість (sensitivity), специфічність (specificity) та інші, щоб кількісно оцінити ефективність програми.

Аналіз помилок: Були проаналізовані випадки, коли програма неправильно призначала діагноз за симптомами. Це дозволило виявити можливі проблеми або обмеження програми та розробити стратегії для їх виправлення.

Статистична обробка результатів: Отримані результати були піддані статистичному аналізу для визначення середніх значень метрик продуктивності, їх дисперсії та інші статистичні характеристики.

Додатково до аналізу результатів тестування програми діагностування захворювань, були розроблені й виконані юніт-тести для тестування моделі.

Ось деякі основні аспекти цього тестування.

Розробка тестових сценаріїв: Були розроблені тестові сценарії, які охоплювали різні аспекти роботи програми, включаючи введення різних наборів симптомів, перевірку правильності визначення захворювань, обробку введення користувача та інші важливі функціональності.

Реалізація тестових випадків: Для кожного тестового сценарію були розроблені відповідні тестові випадки, які перевіряли правильність роботи окремих частин програми або програми в цілому.

Виконання тестів: Тестові випадки були виконані для перевірки роботи програми на різних етапах виконання, включаючи обробку введених даних, роботу з моделлю та правильність виводу результатів.

Автоматизація тестування: Для забезпечення ефективності та повторюваності тестування було використано автоматизовані тести, які дозволяли автоматично виконувати тестові випадки та перевіряти їх результати.

Аналіз результатів тестування: Після виконання тестових випадків було проведено аналіз результатів для виявлення будь-яких помилок або проблем у роботі програми. Цей аналіз допоміг виявити та виправити помилки та покращити функціональність програми.

Використовується навчена модель gnb, щоб передбачити клас (у нашому випадку, хворобу чи ймовірність виникнення хвороби) для вхідних даних inputtest, тестові дані зображені в таблиці 4.1. Функція predict приймає на вхід вектор з даними і повертає передбачений клас чи значення.

predicted=predict[0]

Таблиця 4.1 — Тестовий набір даних

№	Symptom_1	Symptom_2	Symptom_3	Symptom_4	Symptom_5	Діагноз
1	constipation	abdominal_pain	yellow_urine	acute_liver_failure	throat_irritation	Common Cold
2	back_pain	mild_fever	acute_liver_failure	malaise	redness_of_eyes	Fungal infection
3	toxic_look_(typhos)	foul_smell_of urine	increased_appetite	muscle_pain	belly_pain	Typhoid
4	Movement_stiffness	brittle_nails	small_dents_in_nails	pus_filled_pimples	irritability	Dengue
5	bruising	pain_during_bowel_movements	history_of_alcohol_consumption	receiving_unsterile_injections	knee_pain	Hepatitis B

З отриманого результату передбачення (predict), витягується перше значення, оскільки ми передали лише один вхідний вектор (список inputtest).. Це значення, що отримане після передбачення, зберігається у змінній predicted.

Також є готові юніт тести, для тестування цього кейсу, які представлено на рисунку 4.3

Після тестування було проведено порівняння результатів роботи розробленого модуля та програми-аналога, яке зведено у таблицю 4.2

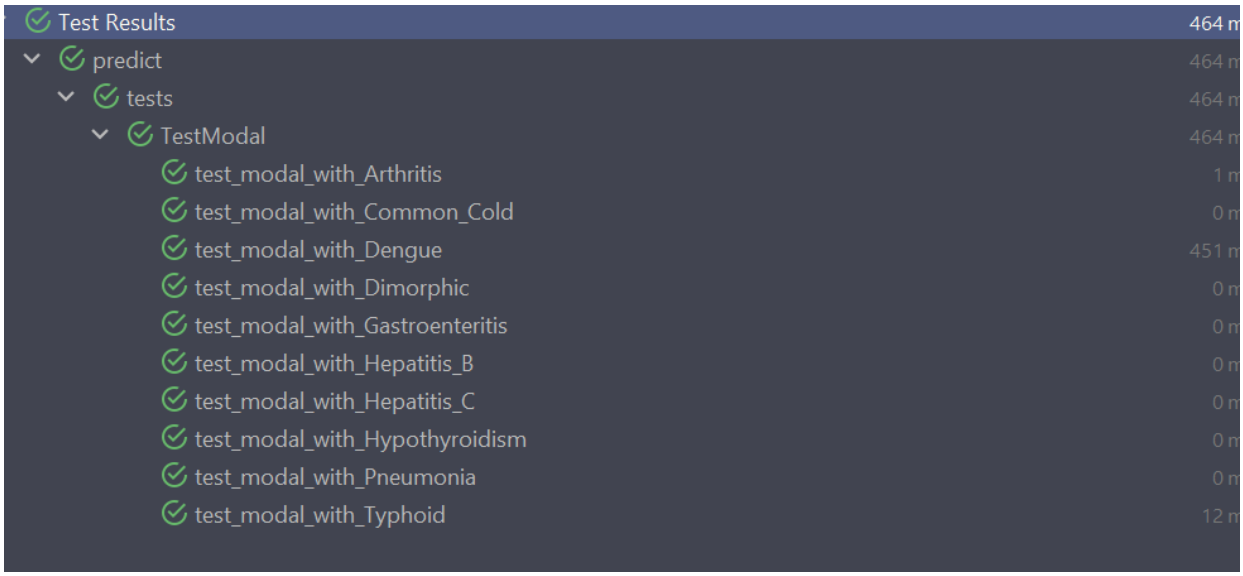

Рисунок 4.3 — Результат юніт тестів

Таблиця 4.2 – Порівняння результатів роботи розробленого програмного модуля та програми-аналога

Програмний засіб	К-ть записів про захворювання у тест. вибірці	Неправильно спрогнозовано	Достовірність прогнозування, %
Програма-аналог	492	89	81,91
Розроблений програмний модуль	492	54	89,02

Із таблиці 4.2 видно, що достовірність діагностування захворювання на тестовому наборі для запропонованого модуля має величину 89%, а для програми-аналога [3] – майже 82%. Тобто достовірність діагностування захворювання розробленого програмного модуля вища на 7% (89-82=7), ніж у програми аналога.

Таким чином, можна зробити висновок, що розроблений програмний модуль діагностування захворювань на основі нейронної мережі має порівняно з аналогом збільшену на 7% достовірність діагностування

захворювань на тестовому наборі. Тобто мета роботи досягнута – достовірність діагностування захворювань підвищена.

4.3 Висновок до розділу 4

У розділі в результаті тестування програмного модуля діагностування захворювань на основі нейронної мережі було доведено його працездатність та відповідність поставленому завданню. Було розглянуто екранну форму модуля та як у нього вводиться інформація про симптоми та як виводиться діагноз. Після виконання тестових випадків було проведено аналіз результатів для виявлення будь-яких помилок або проблем у роботі програми. Цей аналіз допоміг виявити та виправити помилки та покращити функціональність програми. Розроблений програмний модуль має достовірність діагностування захворювання 89%, а програма-аналог майже 82%. Тобто мета роботи досягнута – достовірність діагностування захворювань підвищена на 7%.

ВИСНОВКИ

У роботі було розглянуто задачу діагностування захворювань за симптомами. Описано детальну постановку задачі діагностування захворювань на основі медичних даних та симптомів. Було розглянуто як традиційні методи діагностування захворювань, так і методи, що ґрунтуються на нейронних мережах, у тому числі глибоких, згорткових та рекурентних. Як найбільш перспективний і застосовний до даної задачі було обрано метод на основі нейромереж. Було показано, що з різних типів нейронних мереж найбільше підходить для вирішення поставленої задачі рекурентна нейронна мережа LSTM. Крім цього, було обгрунтовано вибір аналогів розробленого модуля діагностування захворювань за симптомами.

Було обгрунтовано вибір архітектури нейронної мережі LSTM для діагностування захворювань за симптомами. Було проаналізовано структуру рекурентної нейронної мережі LSTM, принципи її функціонування та навчання. Також було розроблено алгоритм роботи програмного модуля діагностування захворювань по симптомах на основі нейронної мережі LSTM.

Було обґрунтовано вибір мови програмування Python, середовища розробки PyCharm та спеціалізованих бібліотек TensorFlow, Keras, Sklearn, Numpy, Pandas, Matplotlib для програмної реалізації модуля діагностування захворювань на основі нейронної мережі. Проведено аналіз набору даних симптомів захворювань для навчання та тестування модуля, який містить 4428 навчальних прикладів та 492 тестових приклади. Описано основні етапи програмної реалізація та функціонування модуля діагностування захворювань на основі нейронної мережі, що складається з імпорту бібліотек, завантаження набору даних, побудови моделі запропонованої нейромережі, навчання нейромережі, тестування результатів роботи розробленої нейромережі.

У результаті тестування програмного модуля діагностування захворювань на основі нейронної мережі було доведено його працездатність

та відповідність поставленому завданню. Було розглянуто екранну форму модуля та як у нього вводиться інформація про симптоми та як виводиться діагноз. Після виконання тестових випадків було проведено аналіз результатів для виявлення будь-яких помилок або проблем у роботі програми. Цей аналіз допоміг виявити та виправити помилки та покращити функціональність програми. Розроблений програмний модуль має достовірність діагностування захворювання 89%, а програма-аналог майже 82%. Тобто мета роботи досягнута – достовірність діагностування захворювань підвищена на 7%.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Machine Learning in Healthcare: Applications, Pros, and Cons. [Електронний ресурс]. — Режим доступу до ресурсу: https://www.healthcareweekly.com/machine-learning-in-healthcare/

2 Machine Learning and Healthcare: A Review. [Електронний ресурс]. — Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5877962/

3 Disease Prediction Using Machine Learning [Електронний ресурс] – Режим доступу: https://www.geeksforgeeks.org/machine-learning/disease-prediction-using-machine-learning/

4 Disease Predition based on Symptoms using Neural Network [Електронний ресурс] – Режим доступу: https://github.com/YeasirArafatZim/Disease-Prediction-Based-on-their-Symptoms-using-Neural-Network/blob/master/Disease_Prediction.ipynb

5 Руденко О.В. Штучні нейронні мережі: Навчальний посібник / О.В.Руденко, Є.В.Бодянський. - Харків : ТОВ «Компанія СМІТ», 2006. — 404 с. - ISBN 966-8630-73-X.

6 Introducing Recurrent Neural Networks. [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/introducing-recurrent-neural-networks-f359653d7020

7 Weather Forecasting using DT. [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/code/priteepriyadarshini/weather-forecasting-using-dt/input

8 Flask’s documentation. [Електронний ресурс]. – Режим доступу: https://flask.palletsprojects.com/en/3.0.x/

9 How to run Flask App on Google Colab? [Електронний ресурс]. – Режим доступу: https://github.com/alloc7260/Project-Snippets/blob/main/run

10 Гудфеллоу Я. Глубокое обучение / пер. с анг. А. А. Слинкина. 2-е изд., испр. М.: ДМК Пресс, 2018. 652 с.: цв. ил.

11 Machine Learning in Healthcare: Applications, Pros, and Cons. [Електронний ресурс]. — Режим доступу до ресурсу: https://www.healthcareweekly.com/machine-learning-in-healthcare/

12 Python’s standard documentation. [Електронний ресурс]. – Режим доступу: https://www.python.org/doc/

13 Pycharm [Електронний ресурс]. – Режим доступу:https://www.jetbrains.com/pycharm/#

14 Disease Symptoms and Patient Profile Dataset [Електронний ресурс] – Режим доступу: https://www.kaggle.com/datasets/uom190346a/disease-symptoms-and-patient-profile-dataset

15 Pandas. Python Data Analysis Library [Електронний ресурс] – Режим доступу:https://pandas.pydata.org/

16 Keras [Електронний ресурс] – Режим доступу: https://keras.io/

17 Confusion Matrix. [Електронний ресурс]. – Режим доступу: https://www.sciencedirect.com/topics/engineering/confusion-matrix

18 Django forum [Електронний ресурс]. — Режим доступу до ресурсу: https://readthedocs.org/projects/django/downloads/

19 Методичні вказівки до виконання бакалаврської кваліфікаційної роботи для студентів денної та заочної форм навчання спеціальності 122 - «Комп’ютерні науки» / Укладачі А.А.Яровий, О.В.Сілагін, І.В.Богач. – Вінниця: ВНТУ, 2022. – 47 с.

Додаток А (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Програмний модуль діагностування захворювань на основі нейронної мережі

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 5,46 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

- ☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Паночишин Ю.М.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Коновалова М.С.
(підпис) (прізвище, ініціали)

Додаток Б (обов’язковий)

Лістинг програми

```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
from keras.callbacks import EarlyStopping

# Завантаження та підготовка даних
data = pd.read_csv('../../diseaseprediction/predict/utils/training_data.csv')
X = data.drop('prognosis', axis=1)
y = data['prognosis']

# Нормалізація даних
X = (X - X.min()) / (X.max() - X.min())

# Кодування категоріального стовпця "prognosis"
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Розділення на тренувальний і тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.3, random_state=40)

# Побудова моделі
model = Sequential()
model.add(LSTM(256, input_shape=(X_train.shape[1], 1), return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(128, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(64))
model.add(Dense(len(label_encoder.classes_), activation='softmax'))

# Компіляція моделі з оптимізатором Adam і зменшеною швидкістю навчання
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)
model.compile(optimizer=optimizer, loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Перетворення даних для LSTM
X_train_lstm = np.expand_dims(X_train.values, axis=2)
X_test_lstm = np.expand_dims(X_test.values, axis=2)

# Використання Early Stopping для зупинки навчання, якщо точність на валідаційному наборі перестає покращуватися
early_stopping = EarlyStopping(monitor='val_accuracy', patience=10, restore_best_weights=True)

# Навчання моделі та збереження історії
history = model.fit(X_train_lstm, y_train, epochs=150, batch_size=64, validation_data=(X_test_lstm, y_test),
callbacks=[early_stopping])

# Оцінка точності моделі на тестовому наборі
accuracy = model.evaluate(X_test_lstm, y_test)
print(f"Test Accuracy: {accuracy[1]*100:.2f}%")
print(f"Test Loss: {accuracy[0]:.4f}")
print(f"Model Summary: {model.summary()}")


# Збереження моделі
model.save('model/disease_diagnosis_model_lstm_updated.keras')
np.save('classes.npy', label_encoder.classes_)
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Styled Form</title>
```

```
    <!-- Підключення Bootstrap і Font Awesome -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.6.0/css/bootstrap.min.css" />
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.2/css/all.min.css" />

    <style>
        .result-container {
        margin-top: 20px;
        padding: 15px;
        background-color: #f8d7da; /* Червоний фон для відображення помилок */
        border: 1px solid #f5c6cb;
        border-radius: 5px;
    }

    .result-container h3 {
        color: #721c24; /* Темно-червоний колір тексту */
    }
        body {
            background-color: #f8f9fa;
        }

        .main-form {
            max-width: 500px;
            margin: 0 auto;
        }

        .form-group {
            margin-bottom: 20px;
        }

        .login-button {
            margin-right: 10px;
        }
    </style>
</head>
<body>

    <div class="container mt-5">
        <div class="card main-form">
            <div class="card-body">

                <h2 class="mb-4">Введіть симптоми</h2>

                <form action="#" method="post">
                    {% csrf_token %}
                    {% for field in form %}
                        <div class="form-group">
                            <label for="{{ field.id_for_label }}">{{ field.label_tag }}</label>
                            <div class="input-group">
                                {{ field }}
                            </div>
                        </div>
                    {% endfor %}

                    <input type="hidden" name="method_predict" value="" id="method_predict_input">


        <button type="button" class="btn btn-success login-button" onclick="setMethodValue('NaiveBayes')">Прогнозувати
хворобу</button>

        <script>
            function setMethodValue(value) {
                document.getElementById('method_predict_input').value = value;
                document.querySelector('form').submit();
            }
        </script>

                </form>

            </div>
        </div>
    {% if disease %}
    <div class="result-container">
    <span>
```

```
        <h3>Ваша хвороба: {{ disease }}</h3>
    </span>
    </div>
    {% endif %}

    </div>

    <!-- Підключення скриптів Bootstrap і jQuery -->
    <script src="https://code.jquery.com/jquery-3.6.4.slim.min.js" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js" integrity="sha384-
c7Jq2qZB+3GB9Qk99Lqf8qzQj7qJSKbGIYdksRSeUzO3uTNi3UAMh1cFboCJlHD" crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.6.0/js/bootstrap.min.js"></script>

</body>
</html>
<style>
    .result-container {
        margin-top: 20px;
        padding: 15px;
        background-color: #d7dae0; /* Сірий фон для відображення помилок */
        border: 1px solid #abb2bf;
        border-radius: 5px;
    }

    .result-container h3 {
        color: #2c3e50; /* Темно-синій колір тексту */
    }
    body {
        background-color: #ecf0f1;
    }

    .main-form {
        max-width: 500px;
        margin: 0 auto;
    }

    .form-group {
        margin-bottom: 20px;
    }

    .login-button {
        margin-right: 10px;
        background-color: #3498db; /* Синій колір кнопки */
        border: none;
    }
</style>
from django.http import HttpResponse
from django.shortcuts import render
from django.template.loader import render_to_string
from django.views.generic import View
from .utils import predict
from .form import PredictForm  # Додавання імпорту форми


class PredictView(View):
    def post(self, request, *args, **kwargs):
        form = PredictForm(request.POST)
        if form.is_valid():
            Symptom_1 = form.cleaned_data['symptom_choice_1']
            Symptom_2 = form.cleaned_data['symptom_choice_2']
            Symptom_3 = form.cleaned_data['symptom_choice_3']
            Symptom_4 = form.cleaned_data['symptom_choice_4']
            Symptom_5 = form.cleaned_data['symptom_choice_5']
            disease = “”
            disease = predict (Symptom_1, Symptom_2, Symptom_3, Symptom_4, Symptom_5)
            return render(
                request,
                'predict/predict_disease.html',
                {'form': form, 'disease': disease}
            )

        else:
```

```
        return HttpResponse("Form is not valid")

    def get(self, request, *args, **kwargs):
        form = PredictForm()
        return render(
            request,
            'predict/predict_disease.html',
            {'form': form}
        )
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

«ПРОГРАМНИЙ МОДУЛЬ ДІАГНОСТУВАННЯ ЗАХВОРЮВАНЬ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ»

Виконала: студентка 4 курсу,групи 3КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Коновалова М. С.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф.КН
Паночишин Ю.М.
(прізвище та ініціали)
«03» червня 2025 р.

Вінниця ВНТУ - 2025 рік

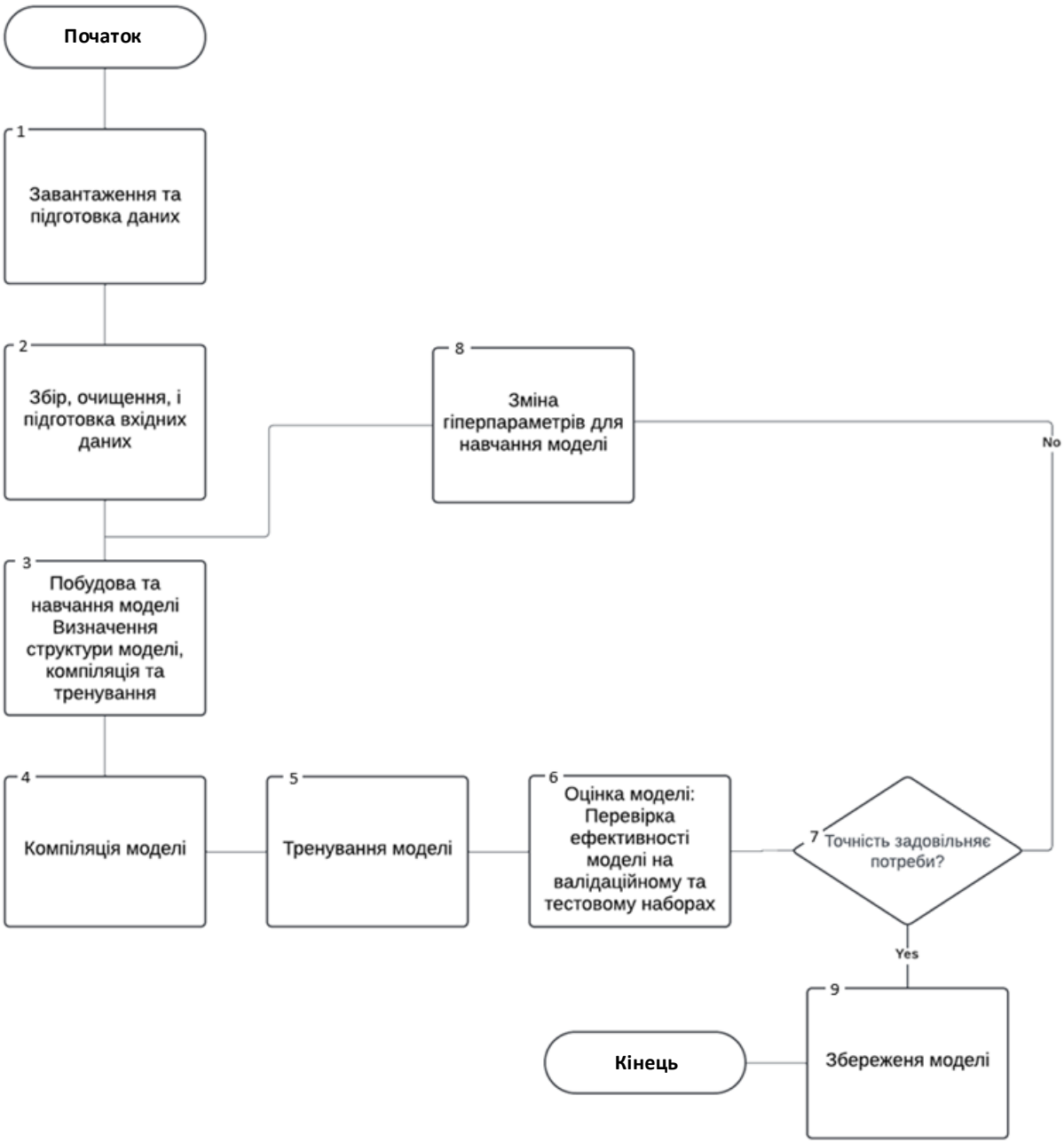

Рисунок В.1– Схема алгоритму роботи програмного модуля діагностування захворювань на основі нейронної мережі

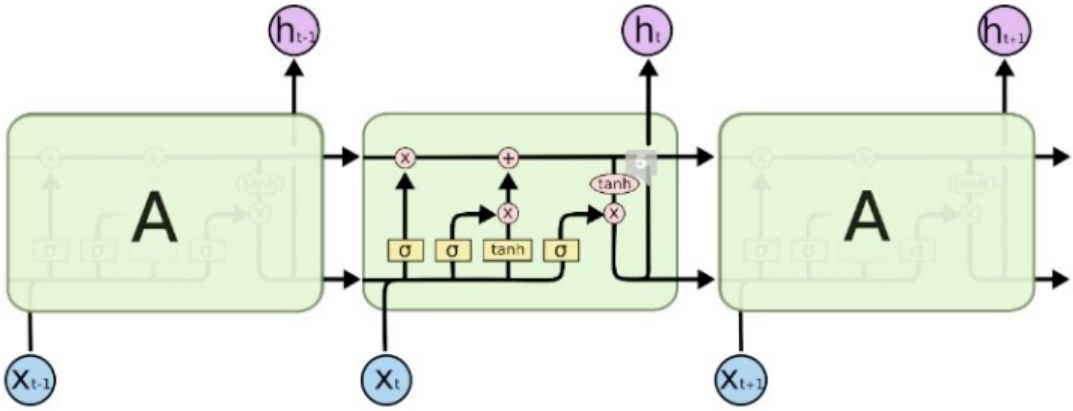

Рисунок В.2–Повторювана LSTM структура

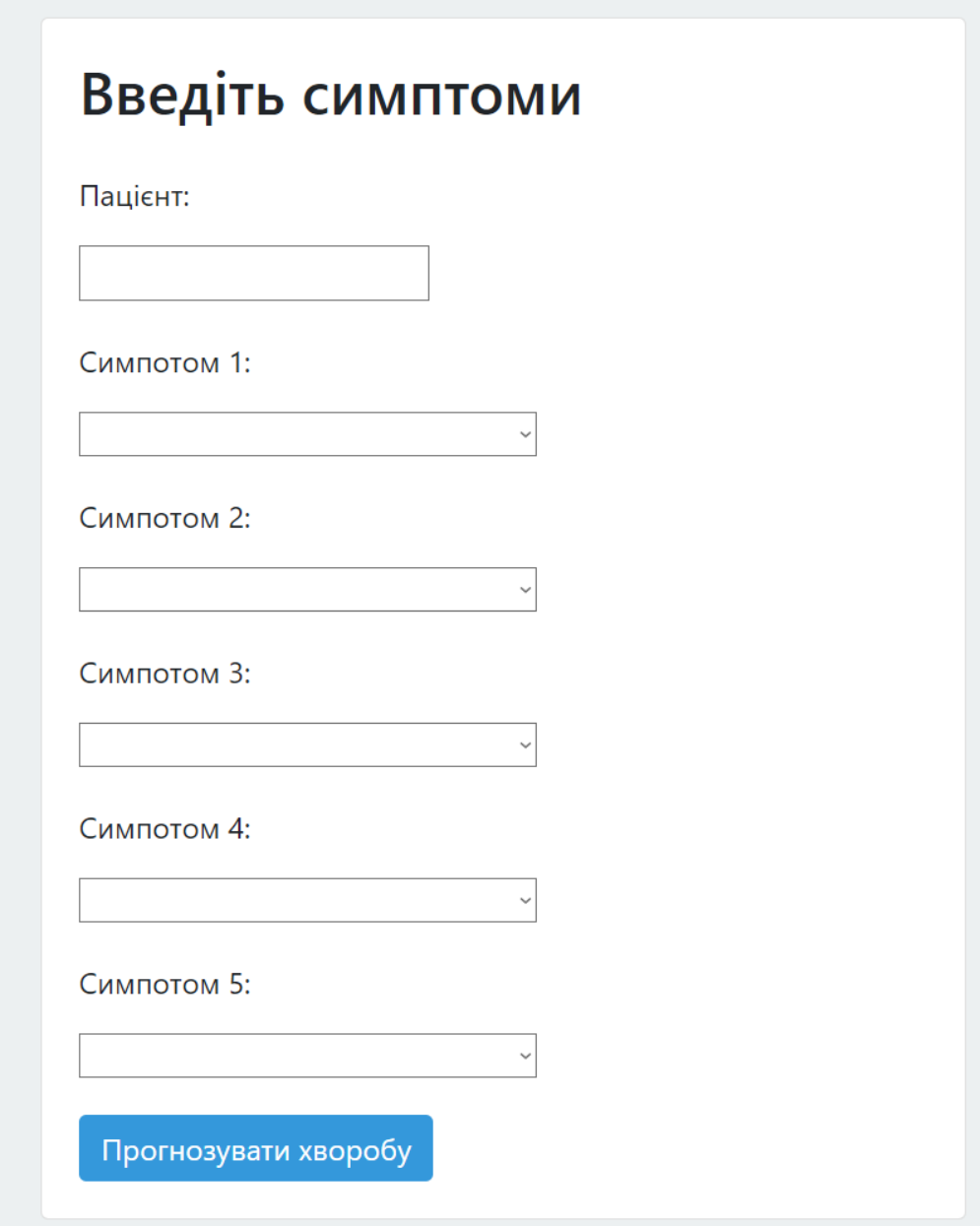

Рисунок В.3 - Інтерфейс користувача

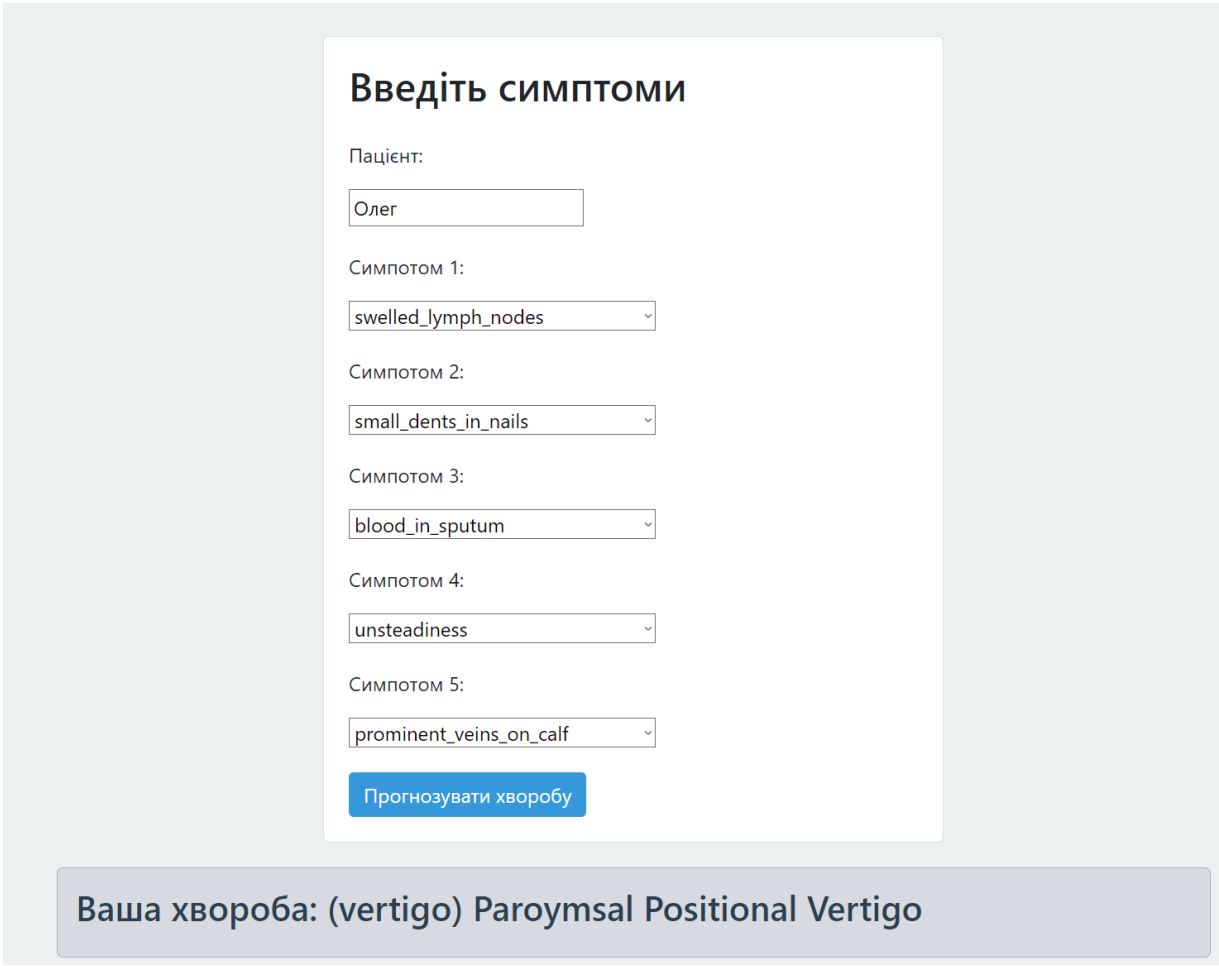

Рисунок В.4 – Введення даних та отримання результату передбачення

Таблиця В.1 – Порівняння результатів роботи розробленого програмного модуля та програми-аналога

Програмний засіб	К-ть записів про захворювання у тест. вибірці	Неправильно спрогнозовано	Достовірність прогнозування, %
Програма-аналог	492	89	81,91
Розроблений програмний модуль	492	54	89,02