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

Бакалаврська дипломнна робота на тему:

ІНТЕЛЕКТУАЛЬНИЙ МОДУЛЬ ПРОГНОЗУВАННЯ РИЗИКУ ІНСУЛЬТУ

Виконала: студентка 4 курсу, групи 2КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Скирська В. В.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф.КН
Паночишин Ю. М.
(прізвище та ініціали)
«07» 06 2024 р.

Рецензент: к.т.н., проф. каф.КСУ
Биков М. М.
(прізвище та ініціали)
«07» 06 2024 р.

Допущено до захисту
Зав. кафедри _____ Яровий А.А.
«07» 06 2024 р.

Вінниця ВНТУ - 2024 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.
«07» 03 2023 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ ДИПЛОМННУ РОБОТУ СТУДЕНТУ

Скирській Валентині Василівні

1. Тема роботи: Інтелектуальний модуль прогнозування ризику інсульту.
керівник роботи: Паночишин Юрій Миколайович, к.т.н., доц.
затверджені наказом вищого навчального закладу "11" 03 2024 року №10

2. Термін подання студентом роботи 27 05.2024р.

3. Вихідні дані до роботи :
кількість класів прогнозування – 2, вид класифікатора – нейронна мережа, кількість навчальних зразків – не менше 4000, кількість тестових зразків– не менше 1000, кількість параметрів пацієнта – не менше 8, використання об'єктно-орієнтованої мови програмування.

4. Зміст текстової частини (перелік питань, які потрібно розробити):
проаналізувати існуючі методи прогнозування ризику інсульту та вибрати найбільш ефективний;
вибрати нейронну мережу, спроектувати її структуру та метод навчання для модуля прогнозування ризику інсульту;
розробити алгоритм роботи модуля прогнозування ризику інсульту;
спроектувати та програмно реалізувати модуль прогнозування ризику інсульту за допомогою нейронної мережі;
протестувати роботу програми прогнозування ризику інсульту за допомогою нейронної мережі.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
Схема алгоритму роботи програми
Структура нейронної мережі
Фрагмент набору даних
Результати роботи програми

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 07.03.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	06.05.24	10.05.24	
2	Обгрунтування методу розв'язання задачі, проектування інтелектуального модуля прогнозування ризику інсульту	11.05.24	23.05.24	
3	Програмна реалізація інтелектуального модуля прогнозування ризику інсульту	24.05.24	01.06.24	
4	Аналіз результатів тестування інтелектуального модуля прогнозування ризику інсульту	02.06.24	03.06.24	
5	Оформлення матеріалів до захисту БДР	04.06.24	06.06.24	
6				

Студент _____ (підпис) **Скирська В. В.** (прізвище та ініціали)

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

АНОТАЦІЯ

Скирська В. В. Інтелектуальний модуль прогнозування ризику інсульту. Бакалаврська дипломна робота скаладається з 64 сторінок формату А4, на яких є 24 рисунки, 2 таблиці, список використаних джерел містить 16 найменувань.

Метою роботи є підвищення специфічності прогнозування ризику інсульту за рахунок використання нейронної мережі.

У роботі було обґрунтовано вибір типу нейронної мережі для модуля прогнозування ризику інсульту - глибока нейромережа, яка має 10 входів, 6 прихованих шарів відповідно по 600, 500, 400, 300, 200, 100 нейронів та вихідний шар із 2 нейронів. Було обрано функцію активації Leaky-ReLu у прихованих шарах та функцію активації Softmax у вихідному шарі. Для програмної реалізації інтелектуального модуля прогнозування ризику інсульту було обґрунтовано вибір мови програмування Python та спеціалізованих бібліотек Keras, NumPy та Pandas. Розроблений інтелектуальний модуль має специфічність прогнозування ризику інсульту на тестовій вибірці 78%, а програма-аналог на основі методу CatBoost має специфічність прогнозування ризику інсульту на тестовій вибірці 2%. Таким чином, розроблений інтелектуальний модуль має порівняно з аналогом збільшену на 76% специфічність прогнозування ризику інсульту.

Ключові слова: класифікація, машинне навчання, нейронна мережа, ризик інсульту.

ABSTRACT

Skyrska V. V. Intelligent stroke risk prediction module. The bachelor's qualification paper consists of 64 pages of A4 format, on which there are 24 figures, 2 tables, the list of used sources contains 16 names.

The purpose of the work is to increase the specificity of predicting the risk of stroke by using a neural network.

The work justified the choice of the type of neural network for the stroke risk prediction module - a deep neural network that has 10 inputs, 6 hidden layers of 600, 500, 400, 300, 200, 100 neurons, respectively, and an output layer of 2 neurons. The Leaky-ReLu activation function in the hidden layers and the Softmax activation function in the output layer were chosen. The choice of the Python programming language and specialized Keras, NumPy, and Pandas libraries was justified for the software implementation of the intelligent stroke risk prediction module. The developed intelligent module has the specificity of predicting the risk of stroke on the test sample of 78%, and the analogue program based on the CatBoost method has the specificity of predicting the risk of stroke on the test sample of 2%. Thus, the developed intelligent module has a 76% increased specificity of stroke risk prediction compared to its analogue.

Keywords: classification, machine learning, neural network, stroke risk.

ЗМІСТ

ВСТУП 3
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ КЛАСИФІКАЦІЇ ОБ`ЄКТІВ 5
1.1 Постановка задачі 5
1.2 Огляд відомих методів класифікації об`єктів 7
1.3 Обгрунтування вибору аналогу до інтелектуального модуля прогнозування ризику інсульту 14
1.4 Висновок до розділу 1 15
2 ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНОГО МОДУЛЯ ПРОГНОЗУВАННЯ РИЗИКУ ІНСУЛЬТУ НА ОСНОВІ НЕЙРОМЕРЕЖІ 16
2.1 Обґрунтування вибору типу нейронної мережі для модуля прогнозування ризику інсульту 16
2.2 Структура процесів обробки інформації інтелектуального модуля прогнозування ризику інсульту 23
2.3 Архітектура глибокої нейронної мережі прямого поширення 23
2.4 Розробка алгоритму роботи інтелектуального модуля прогнозування ризику інсульту 26
2.5 Висновок до розділу 2 27
3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОГО МОДУЛЯ ПРОГНОЗУВАННЯ РИЗИКУ ІНСУЛЬТУ НА ОСНОВІ НЕЙРОМЕРЕЖІ 28
3.1 Обґрунтування вибору мови та спеціалізованих бібліотек 28
3.2 Опис набору даних для навчання та тестування модуля 31
3.3 Попередня обробка інформації із набору даних 33
3.4 Програмна реалізація інтелектуального модуля прогнозування ризику інсульту 36
3.4 Висновок до розділу 3 41
4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ІНТЕЛЕКТУАЛЬНОГО МОДУЛЯ ПРОГНОЗУВАННЯ РИЗИКУ ІНСУЛЬТУ НА ОСНОВІ НЕЙРОМЕРЕЖІ 42
ВИСНОВКИ 50
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 52
ДОДАТОК А (ОБОВ’ЯЗКОВИЙ) ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ 54
ДОДАТОК Б (ОБОВ’ЯЗКОВИЙ) ЛІСТИНГ ПРОГРАМИ 55
ДОДАТОК В (ОБОВ’ЯЗКОВИЙ) ІЛЮСТРАТИВНА ЧАСТИНА 59

ВСТУП

Актуальність досліджень. Інсульт - це захворювання, яке вражає артерії, що ведуть до мозку та всередині нього. Інсульт виникає, коли кровоносна судина, яка переносить кисень і поживні речовини до мозку, блокується тромбом або розривається. За даними ВООЗ, інсульт займає друге місце серед причин смерті в усьому світі. У всьому світі 3% населення страждає від субарахноїдальних крововиливів, 10% - від внутрішньомозкових крововиливів і більшість 87% - від ішемічного інсульту. У 80% випадків ці інсульти можна запобігти.

Інсульт, який іноді називають крововиливом у мозок, виникає, коли щось блокує кровопостачання частини мозку або коли кровоносна судина в мозку лопається. У будь-якому випадку частини мозку пошкоджуються або гинуть. Інсульт може спричинити тривале пошкодження мозку, тривалу втрату працездатності або навіть смерть.

Ознаки інсульту у чоловіків і жінок:

– Раптове оніміння або слабкість в обличчі, руці або нозі, особливо на одній стороні тіла.

– Раптова плутанина у мові, проблеми з розмовою або труднощі з розумінням мови.

– Раптовий сильний головний біль без відомої причини.

Потрібно діяти швидко для визначення інсульту.

Раннє виявлення інсульту є вирішальним кроком для ефективного лікування. Найкраще ефективне лікування інсульту можливе лише якщо інсульт розпізнано та діагностовано протягом 3 годин після перших симптомів. Машинне навчання (ML) — це найкраща технологія, яка може допомогти медичним працівникам приймати клінічні рішення та прогнози.

Конкретна мета розроблюваної програми – передбачити, чи буде у людини мозковий інсульт на основі введених параметрів. Ці введені параметри

аналізуються моделлю машинного навчання для прогнозування цільового класу.

Метою дослідження є підвищення специфічності прогнозування ризику інсульту за рахунок використання штучної нейронної мережі. На відміну від аналогів, які використовують традиційні методи машинного навчання без використання нейромереж, в роботі пропонується застосувати штучну нейронну мережу для навчання прогнозуванню на основі прикладів.

Об’єктом дослідження є процес комп’ютеризованого прогнозування ризику інсульту на основі нейромережі.

Предметом дослідження є алгоритми та програмні засоби прогнозування ризику інсульту на основі нейронної мережі та достовірність їх роботи.

Задачі дослідження:

1. Проаналізувати відомі методи прогнозування ризику інсульту та обрати напрямок досліджень;
2. Розробити структуру інтелектуального модуля прогнозування ризику інсульту на основі нейронної мережі;
3. Обгрунтувати вибір архітектури нейромережі;
4. Розробити алгоритм роботи інтелектуального модуля прогнозування ризику інсульту;
5. Спроектувати модуль прогнозування ризику інсульту на основі нейронної мережі;
6. Здійснити програмну реалізацію інтелектуального модуля прогнозування ризику інсульту;
7. Провести тестування інтелектуального модуля прогнозування ризику інсульту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ КЛАСИФІКАЦІЇ ОБ`ЄКТІВ

1.1 Постановка задачі

Інсульт, який іноді називають крововиливом у мозок, виникає, коли щось блокує кровопостачання частини мозку або коли кровоносна судина в мозку лопається. У будь-якому випадку частини мозку пошкоджуються або гинуть. Інсульт може спричинити тривале пошкодження мозку, тривалу втрату працездатності або навіть смерть.

Ознаки інсульту у чоловіків і жінок:

– Раптове оніміння або слабкість в обличчі, руці або нозі, особливо на одній стороні тіла.

– Раптова плутанина у мові, проблеми з розмовою або труднощі з розумінням мови.

– Раптове порушення зору на одне або обидва ока.

– Раптові проблеми з ходьбою, запаморочення, втрата рівноваги або відсутність координації.

– Раптовий сильний головний біль без відомої причини.

Потрібно діяти швидко для визначення інсульту.

Найкраще ефективне лікування інсульту можливе лише якщо інсульт розпізнано та діагностовано протягом 3 годин після перших симптомів. Пацієнти з інсультом можуть не отримати ефективне лікування, якщо вони не прибудуть до лікарні вчасно.

Якщо ви думаєте, що у когось інсульт, дійте швидко і проведіть наступний тест:

1) Обличчя: попросіть людину посміхнутися. Чи опускається одна сторона обличчя?

2) Руки: попросіть людину підняти обидві руки. Одна рука дрейфує вниз?

3) Мовлення: попросіть людину повторити просту фразу. Мова невиразна чи дивна?

4) Час: якщо ви бачите будь-які з цих ознак, негайно зателефонуйте у швидку допомогу.

У роботі будемо використовувати набір даних [1] з Kaggle . Це параметри пацієнтів, які досліджувались на наявність інсульту. Характеристика набору даних для визначення ризику інсульту показана на рис. 1.1.

```
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5110 non-null   int64
 1   gender             5110 non-null   object
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64
 4   heart_disease      5110 non-null   int64
 5   ever_married       5110 non-null   object
 6   work_type          5110 non-null   object
 7   Residence_type     5110 non-null   object
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object
 11  stroke             5110 non-null   int64
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

Рисунок 1.1 – Характеристика набору даних для визначення ризику інсульту

Із рис.1.1 бачимо, що у наборі даних є 12 параметрів (від 0 до 11), причому параметр id є ідентифікатором рядка набору даних і тому не є значимим. Параметри 1-10 є вхідними параметрами для системи прогнозування ризику інсульту, а параметр 11 є вихідним (цільовим) параметром системи прогнозування ризику інсульту. Таким чином, нам треба побудувати систему класифікації (прогнозування), яка класифікує об`єкти за 10 параметрами на 2 класи (1-й клас – є ризик інсульту, 2-й клас – немає ризику інсульту),

В принципі, для цієї задачі можна застосувати будь-який із відомих класифікаторов, але в цій роботі ми застосуємо класифікатор на основі штучної нейронної мережі.

Мета нейронної мережі, яка створюється, - передбачити клас пацієнта на основі вхідних атрибутів. Це означає, що потрібно створити модель, яка буде описувати зв'язок між значеннями атрибутів і ризиком виникнення інсульта у пацієнта.

1.2 Огляд відомих методів класифікації об`єктів

Класифікація - це процес визначення класу, до якого відноситься конкретний об`єкт, причому число класів та їхні назви відомі наперед, а також відомі типові параметри об`єктів, які належать цим класам.

Існує велика кількість різних видів завдань класифікації, які потрібно виконувати. Для кожного завдання класифікації зазвичай потрібен свій алгоритм, так як кожне з них застосовується для розв`язання конкретної проблеми.

Для відповідної проблеми потрібно обрати вірний алгоритм. Питання полягає в тому, як це зробити найкраще. Якщо у вас є достатньо обчислювальних ресурсів, то можна протестувати декілька алгоритмів і налаштувань параметрів. У такому випадку головне питання полягає в тому, як адекватно оцінити і порівняти якість алгоритмів.

В машинному навчанні алгоритми класифікації використовують навчальні дані для прогнозування імовірності того, що нові дані потраплять в одну із завчасно визначених категорій. Одне з найпоширеніших застосувань класифікації – медичне діагностування пацієнтів на "хворий" і "здоровий".

Іншими словами, класифікація - це різновид "розпізнавання образів", при якому алгоритми класифікації використовуються до навчальних прикладів, щоб знайти схожий образ (подібні слова, параметри пацієнтів, послідовності чисел тощо) у нових наборах даних.

Дослідження класифікації в медицині дуже широке, і існує кілька видів алгоритмів класифікації, які можна застосовувати в залежності від набору

даних, з яким ви працюєте. Наведемо перелік найпоширеніших алгоритмів класифікації у машинному навчанні:

1. K-найближчих сусідів
2. Логістична регресія
3. Дерево рішень
4. Машини опорних векторів
5. Ансамблеві методи
6. Нейромережева класифікація

K-найближчих сусідів.

K-найближчих сусідів (k-NN) - це метод класифікації образів, який використовує навчальний набір даних для пошуку k «найближчих» прикладів до нового образу.

Коли k-NN використовується для класифікації, потрібно обчислитиі відстані між векторами ознак, щоб віднести нові дані до класу найближчого сусіда.

Логістична регресія.

Логістична регресія - це метод класифікації, який застосовується для передбачення бінарного результату: або щось відбувається, або ні. Це можна представити як "Так/Ні", "Здоровий/Хворий" тощо.

Незалежні параметри аналізуються для визначення бінарного результату, і як результат - потрапляння в одну з двох класів. Незалежні параметри можуть бути або числовими або категоріальними, але залежна змінна завжди є категоріальною. Записується це так:

$$P(Y=1|X) \text{ or } P(Y=0|X)$$

За цією формулою обчислюється імовірність залежної змінної Y, враховуючи незалежний вектор X. Її використовують для обчислення імовірності того, що якесь слово має або позитивну або негативну конотацію (0 або 1 чи за шкалою від 0 до 1). Або її використовують для визначення об'єкта,

що зображено на фотографії (квітка, дерево, трава тощо), і при цьому кожному об'єкту привласнюється імовірність між 0 та 1.

Дерево рішень

Дерево рішень - це метод навчання з учителем, який підходить ідеально для задач класифікації, так як він здатний впорядковувати категорії. Він працює як граф-схема, розділяючи групи даних на дві подібні категорії від "стовбура дерева" до "гілок" і "листя", де класи стають все більш схожими. Це створює класи всередині класів, що дозволяє натурально класифікувати дані під обмеженим наглядом людини.

Для прикладу зі спортом, дерево рішень має вид як показано на рис. 1.2.

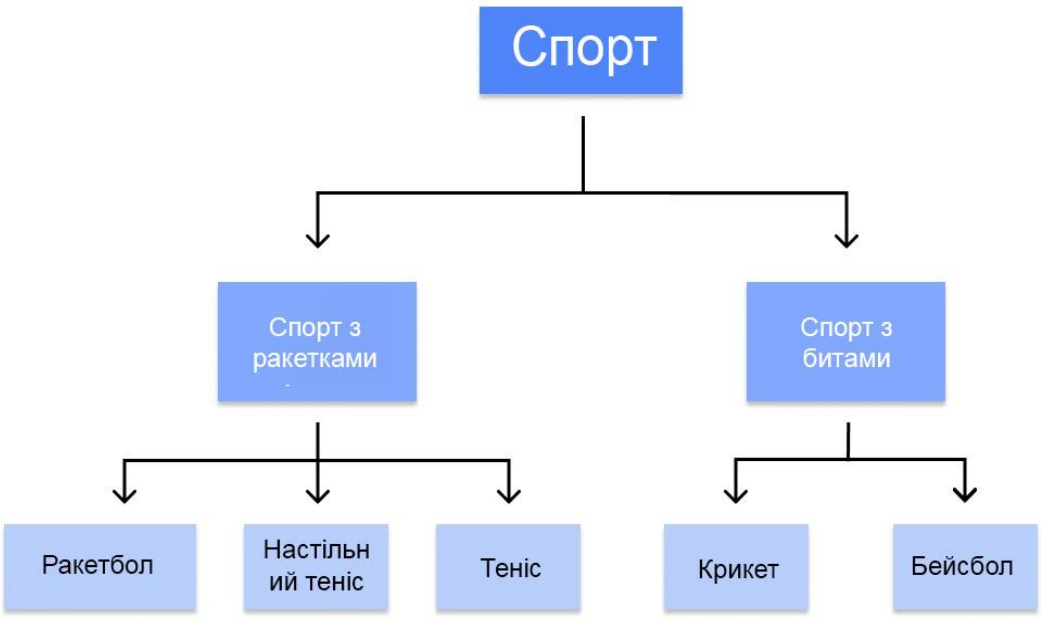

Рисунок 1.2 – Дерево рішень для прикладу зі спортом

Машини опорних векторів

Машини опорних векторів (SVM) використовують алгоритми навчання для класифікації даних у межах ступенів полярності, виводячи їх на рівень, який виходить за межі прогнозування X/Y (рис. 1.3).

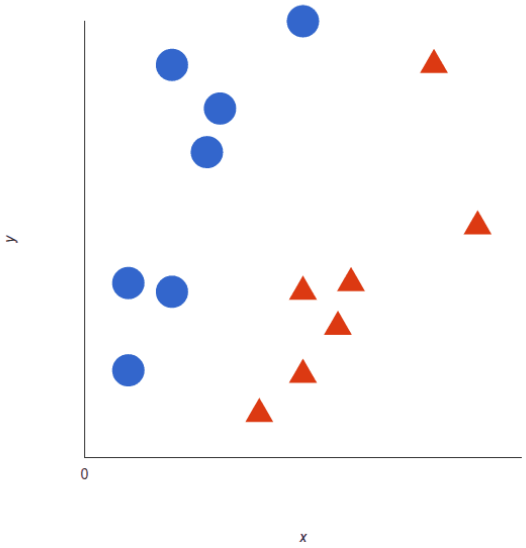

Рисунок 1.3 – Просте візуальне пояснення розподілу даних двох класів

Для нескладного візуального пояснення застосовано дві мітки: синю і червону, з двома характеристиками даних: X та Y, а потім навчено класифікатор виводити координату X/Y як синю або червону.

Потім SVM визначає гіперплощину, яка розділяє мітки найкраще (рис. 1.4). У двовимірному просторі гіперплощина - це пряма лінія. Все, що розташовується по один бік від лінії, є синім, а все, що по інший бік лінії - червоним.

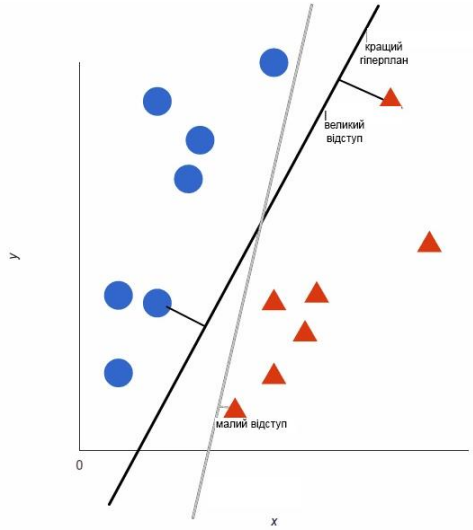

Рисунок 1.4 – SVM визначає гіперплощину (пряму лінію), яка розділяє класи найкраще

Для того, щоб оптимізувати машинне навчання, найкращою є та гіперплощина, яка має найбільшу відстань між кожною парою міток.

Однак, оскільки набори даних іноді є складними, то може виявитися, що неможливо провести єдину лінію для розділення даних на два класи. Приклад цього подано на рис. 1.5.

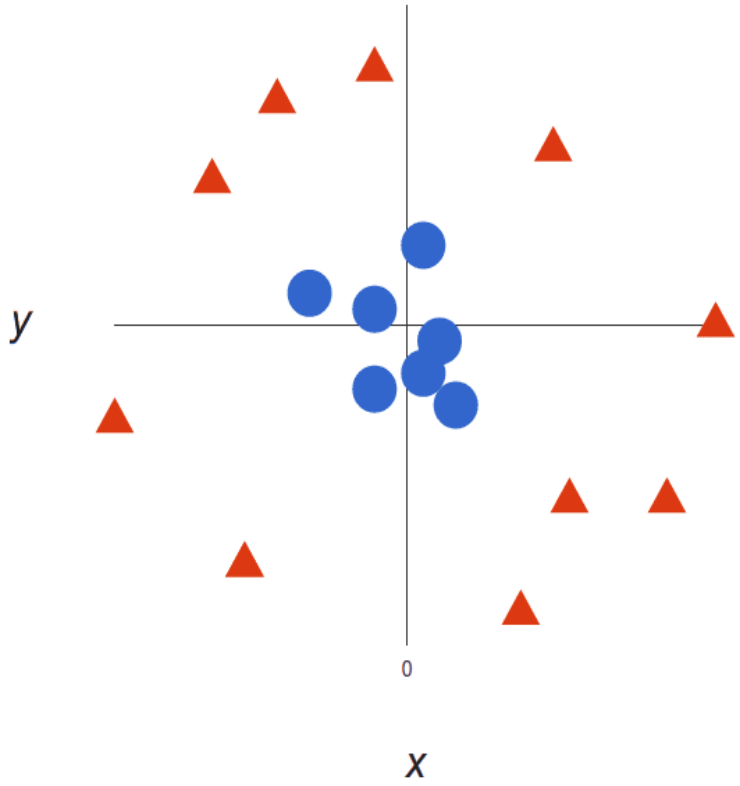

Рисунок 1.5 – Приклад неможливості проведення єдиної лінії для класифікації на дві категорії

При використанні SVM, чим складніші дані, тим точнішим буде класифікатор. Якщо уявити все вищесказане у тривимірному просторі, додавши вісь z, щоб вийшло коло, то з найкращою гіперплощиною це виглядає так, як вказано на рис. 1.6.

SVM має більш точні результати машинного навчання, так як є багатовимірним.

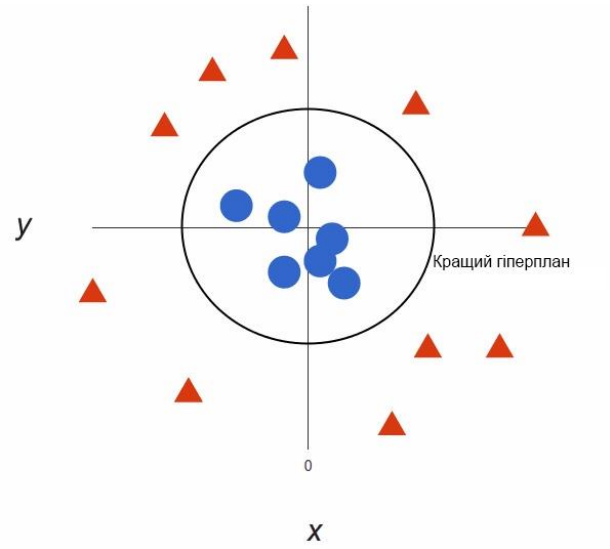

Рисунок 1.6 – Приклад найкращої лінії, що розділяє (коло)

Ансамблеві методи.

Ансамблеві методи – це потужний інструмент для побудови моделей машинного навчання. Команди, які використовують їх у змаганнях на Kaggle, займають переможні місця. Ансамблі дозволяють збільшити точність моделі до 90+, причому досить прості в розумінні.

Метод машинного навчання, де кілька моделей навчаються для вирішення однієї і тієї ж проблеми та об'єднуються для отримання кращих результатів називається ансамблевим методом. Основна передумова полягає в тому, що результат роботи кількох моделей буде точнішим, ніж результат тільки однієї моделі.

Коли йдеться про ансамблі, то вводиться поняття слабкого учня (звичайні моделі на кшталт лінійної регресії чи дерева рішень). Набір слабких учнів є будівельними блоками для складніших моделей. Об'єднання слабких учнів задля поліпшення якості моделі, зменшення зміщення чи розкиду, називається сильним учнем.

Види ансамблевих методів.

Найбільш популярними ансамблевими методами є: стекінг, бегінг, бустинг.

Стекінг. Використовується кілька слабких учнів. Їх навчають та поєднують для побудови прогнозу, заснованого на результатах різних слабких моделей.

Бегінг. І тут однорідні моделі навчають на різних наборах даних, і об'єднують. Отримують прогноз шляхом усереднення. Якщо використовувати як слабкого учня дерева рішень, то вийде випадковий ліс RandomForestClassifier/RandomForestRegressor.

Бустинг. З використанням цього методу кілька однорідних моделей послідовно навчаються, виправляючи помилки одне одного.

Нейронні мережі.

Нейронна мережа - математична модель мережі біологічних нейронів. Вона складається з великої кількості простих пристроїв – нейронів, які з'єднані один з одним. Кожен нейрон отримує певний вхідний сигнал, якось перетворює його, а потім надсилає цей сигнал до інших нейронів.

Особливість нейронних мереж – їх здатність до навчання на вирішення певних завдань. Під час навчання відбувається підбір коефіцієнтів зв'язків між нейронами. Нейронна мережа здатна у процесі навчання віднаходити складні взаємозв'язки між характеристиками вхідних та вихідних даних. Також, нейронні мережі мають здатність до узагальнення, тобто здатність видавати вірний вихід для вхідних даних, які не використовувалися при навчанні.

Порівнюючи ці методи, можна зазначити, що метод опорних векторів має велику обчислювальну складність, і для його застосування треба мати більший простір ознак, ніж є в даній задачі. Відсутня від початку статистична інформація про вхідні дані ускладнює застосування методу байєсівського класифікатора. Ансамблеві методи машинного навчання є більш точними, але також вимагають великих обчисювальних ресурсів і низьку швидкодію навчання.

Нейромережі спроможні розв`язувати такі задачі класифікації, у яких набори даних є лінійно-нероздільними. До того ж, нейронні мережі здатні до апроксимації довільної багатозначної функції кількох аргументів із заданою

точністю [2,3]. Наведені аргументи свідчать про те, що використання нейромереж для вирішення даної задачі прогнозування ризику інсульту є кращим варіантом.

1.3 Обгрунтування вибору аналогу до інтелектуального модуля прогнозування ризику інсульту

На теперішній час існує багато відомих програмних додатків, що здатні здійснювати діагностування різних хвороб. Діагностування хвороб є класичною задачею класифікації, тому різні дослідники застосовують для цієї задачі свої методи та алгоритми класифікації.

У роботі [4] було описано модель прогнозування ризику інсульту за допомогою методу ансамблевого навчання, поєднуючи результати кількох моделей у ансамблі. Там було порівняно кілька методів машинного навчання (табл.1.1) і показано, що найкращим є метод Categorical Boosting.

Таблиця 1.1 – Результати порівняння кількох методів машинного навчання при прогнозуванні ризику інсульту

	Метод	accuracy	precision	recall	f1_score
0	K-Nearest Neighbors	0.951076	0.000000	0.00	0.000000
1	Logistic Regression	0.949119	0.000000	0.00	0.000000
2	Linear Discriminant Analysis	0.925636	0.175000	0.14	0.155556
3	Quadratic Discriminant Analysis	0.914873	0.215385	**0.28**	**0.243478**
4	Stochastic Gradient Descent	0.921722	0.200000	0.20	0.200000
5	Bagging Classifer using Random Subspaces	0.951076	0.000000	0.00	0.000000
6	Bagging Classifer using Random Patches	0.951076	0.000000	0.00	0.000000
7	Bagging Classifer (Bootstrap)	0.951076	0.000000	0.00	0.000000
8	Bagging Classifer using (Pasting)	0.951076	0.000000	0.00	0.000000
9	Random Forest	0.951076	0.000000	0.00	0.000000
10	Extra Trees	0.951076	0.000000	0.00	0.000000
11	Histogram Gradient Boosting	0.938356	0.117647	0.04	0.059701
12	Categorical Boosting	**0.950098**	**0.333333**	0.02	0.037736

Із табл. 1.1 видно, що найкращим методом є Categorical Boosting, тому саме його оберемо як аналог для порівняння із розроблюваним інтелектуальним модулем.

До недоліків цієї програми можна віднести невисоке значення такої метрики як влучність (precision), а також не використовується така метрика як специфічність (specificity), яка показує точність визначення помилок 1-го роду, коли хвора людина ідентифікується як здорова. Звідси випливає завдання розробки нового програмного засобу для прогнозування ризику інсульту з підвищеними метриками якості роботи, а саме - підвищення специфічності та влучності.

1.4 Висновок до розділу 1

У розділі описано детальну постановку задачі прогнозування ризику інсульту. Також розглядаються різні методи розв'язання задачі класифікації і оцінюється їх застосовність до завдання прогнозування ризику інсульту. Серед таких методів машинного навчання як метод K-найближчих сусідів, логістичної регресії, дерев рішень, машини опорних векторів, ансамблевих методів та нейромережевої класифікації як найбільш перспективний і застосовний до даної задачі було обрано нейромережевий метод. Крім цього, було обгрунтовано вибір аналогу до розроблюваного модуля та на основі його недоліків сформульовано мету роботи – підвищення специфічності прогнозування ризику інсульту.

2 ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНОГО МОДУЛЯ ПРОГНОЗУВАННЯ РИЗИКУ ІНСУЛЬТУ НА ОСНОВІ НЕЙРОМЕРЕЖІ

2.1 Обґрунтування вибору типу нейронної мережі для модуля прогнозування ризику інсульту

Для задачі прогнозування ризику інсульту, в принципі, може бути застосовано дуже багато типів штучних нейромереж, які здатні працювати в режимі класифікації образів. З метою вибору найбільш ефективної для задачі прогнозування ризику інсульту, розглянемо такі штучні нейронні мережі [2]:

- багатошаровий персептрон,
- глибока нейронна мережа прямого поширення,
- нейро-нечітка мережа Такагі-Сугено-Канг.

В основі процесу навчання всіх цих нейромереж лежить набір навчальних пар ($\mathbf{X_i}$, $\mathbf{d_i}$), де $\mathbf{X_i}$ - це вектор параметрів, а $\mathbf{d_i}$ - це вектор міток класу. Розглянемо їх детальніше з метою вибору найоптимальнішої для нашої задачі.

Нейромережі – це моделі, що імітують діяльність мозку людини. Вони, як і мозок, складаються з великої кількості однотипних елементів – нейронів, зв'язаних між собою. На рис. 2.1. показано схему штучного нейрона [3].

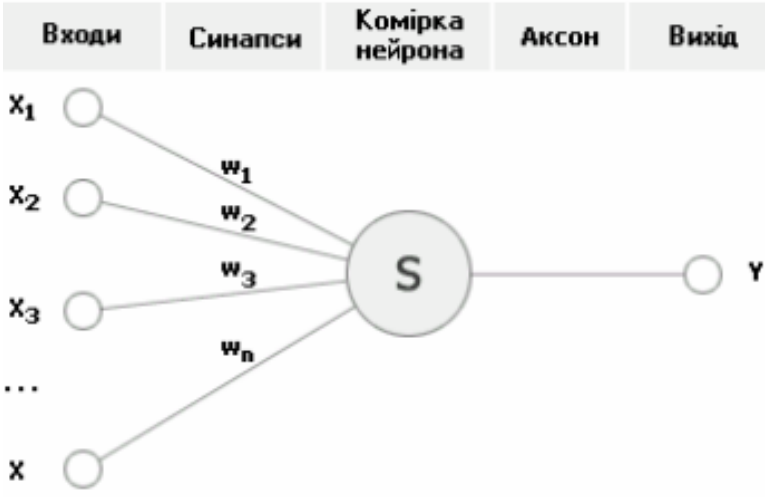

Рисунок 2.1 – Схема моделі нейрона

З рисунку 2.1 видно, що штучний нейрон, як і біологічний, складається з синапсів, які пов'язують входи нейрона із його ядром, самого ядра нейрона, яке виконує обробку вхідних сигналів та аксона, який пов'язує нейрон з нейронами наступного шару. Кожний синапс має вагу, яка визначає ступінь впливу відповідного входу нейрона на його стан. Стан нейрона обчислюється за формулою:

$$S = \sum_{i=1}^{n} x_i w_i \tag{2.1}$$

де n - кількість входів нейрону, x_i – величина значення і-го входу нейрона, w_i – вага і-го синапсу. Значення виходу обчислюється за формулою:

$$Y=f(S) \tag{2.2}$$

де f –функція, яка називається активаційною. Найчастіше як активаційну функцію, використовують сигмоїду, яка має наступний вид:

$$f(x) = \frac{1}{1+e^{-\alpha x}} \tag{2.3}$$

Видатною перевагою сигмоїди є те, що вона диференційована по всій осі абсцис і має просто обчислювану похідну:

$$f'(x) = \alpha f(x)(1-f(x)) \tag{2.4}$$

При зменшенні параметра α сигмоїда стає більш похилою та вироджується у горизонтальну лінію на рівні 0,5 при a=0. При збільшенні α сигмоїда стає ближче до функції Хевісайда.

2.1.1 Багатошаровий персептрон

Багатошаровий персептрон [3, 5] - це одна з найбільш відомих нейромереж, яка складається з множини простих нейроноподібних елементів обробки сигмоїдної функції активації, згрупованих у шари. Функції активації в багатошаровому персептроні можуть бути різні: логарифмічна f(x)=1/(1+exp(-x)), гіперболічний тангенс, сігнум (кусково-постійна) або лінійна. Класична нейромережа складається з одного прихованого шару, за яким йде вихідний шар нейронів.

Інформація обробляється локально у кожному елементі шляхом обчислення скалярного добутку вхідного вектору та вагового вектору відповідного нейрона. Перед тренуванням вагові вектори ініціюються випадковими числами. Навчання мережі для отримання бажаного вихідного вектора $\mathbf{d_i}$, якщо відомий вхідний вектор $\mathbf{x_i}$, включає зазвичай зміну вагових векторів усіх нейронів доти, поки мережа не виведе бажаний результат на виході, маючи при цьому допустиму помилку (похибку). Це повторюється під час усього циклу тренування. Тобто навчання зводить до мінімуму вимірювані помилки для всього навчального набору даних на протязі кінцевої кількості циклів навчання для запобігання перенавчанню [3].

Градієнтні моделі є основою найбільш ефективних методів навчання. Градієнтні вектори в багатошаровій нейромережі розраховуються із застосуванням алгоритму зворотного поширення помилки [3,5]. У градієнтному методі навчання вагові вектори w підбираються від циклу до циклу згідно з інформацією про градієнт функції помилки:

$$w(k+1) = w(k) + \eta p(k),$$

де η - швидкість навчання, що змінюється в кожному циклі, а p (k) - це вектор мінімізації в k-му циклі. Після закінчення етапу навчання знайдені вагові

вектори «застабілізовуються» і потім застостовуються в тестовому режимі, в якому вхідний вектор x, що оброблений нейромережею для утворення вихідного сигналу, відповідає за визначення мітки класу. Зазвичай нейрон з найбільшим вихідним сигналом відповідає визначеному класу.

Узагальнення - фундаментальна властивість, яка дозволяє визначити здатність нейромережі до розпізнавання нових образів, які не входили в навчальний набір. Якщо число ваг мережі занадто велике або число навчальних прикладів дуже мале, то буде велика кількість нейромереж, які відповідають навчальним даним, але лише деякі будуть точно описувати простір правильних рішень. Отже, більш переважним є слабке узагальнення. Щоб підвищити імовірність вірного узагальнення, потрібно мінімізувати число вільних параметрів (ваг). Однак, це потрібно зробити таким чином, щоб розмір нейромережі не зменшився до області, в якій уже не можливо досягти бажаного результату. Додатково, потрібно контролювати число циклів навчання, аби уникнути надмірної відповідності мережі тренувальним даним (перенавчання нейромережі).

Інший спосіб – це перехресна перевірка достовірності (перехресна валідація). У цьому методі дані розбиваються на навчальний, валідаційний і тестовий набори. Набір валідаційних даних використовується для перевірки здатності мережі, навченої на тренувальних даних, до узагальнення. Розмір мережі, що забезпечує мінімальну помилку валідації, приймається за оптимальний.

2.1.2 Глибокі нейронні мережі

Глибока нейронна мережа (ГНМ, англ. deep neural network, DNN) — це штучна нейронна мережа (ШНМ) із кількома шарами між шарами входу та виходу.[6,7] Існують різні типи нейронних мереж, але вони завжди складаються з тих же складових: нейронів, синапсів, ваг, зміщень та функцій[7]. Ці складові

в цілому функціонують у спосіб, що імітує функціювання людського мозку, і їх, як і будь-який інший алгоритм машинного навчання, можливо тренувати.

Наприклад, ГНМ, тренована розпізнавати породи собак, проходитиме заданим зображенням й обчислюватиме ймовірність того, що зображений собака належить до певної породи. Користувач може переглядати результати й обирати, які ймовірності мережа повинна відображувати (вище певного порогу тощо) й повертати запропоновану мітку. Кожну математичну маніпуляцію як таку вважають шаром, і складні ГНМ мають багато шарів, звідси й назва «глибокі» мережі.

ГНМ можуть моделювати складні нелінійні зв'язки. Архітектури ГНМ (рис. 2.2) породжують композиційні моделі, де об'єкт виражають багатошаровою композицією примітивів. Додаткові шари дозволяють комбінувати ознаки з нижчих шарів, потенційно моделюючи складні дані меншою кількістю вузлів, ніж неглибокі мережі з подібною продуктивністю. Наприклад, було доведено, що розріджені багатовимірні многочлени експоненційно легше наближувати за допомогою ГНМ, ніж за допомогою неглибоких мереж [2].

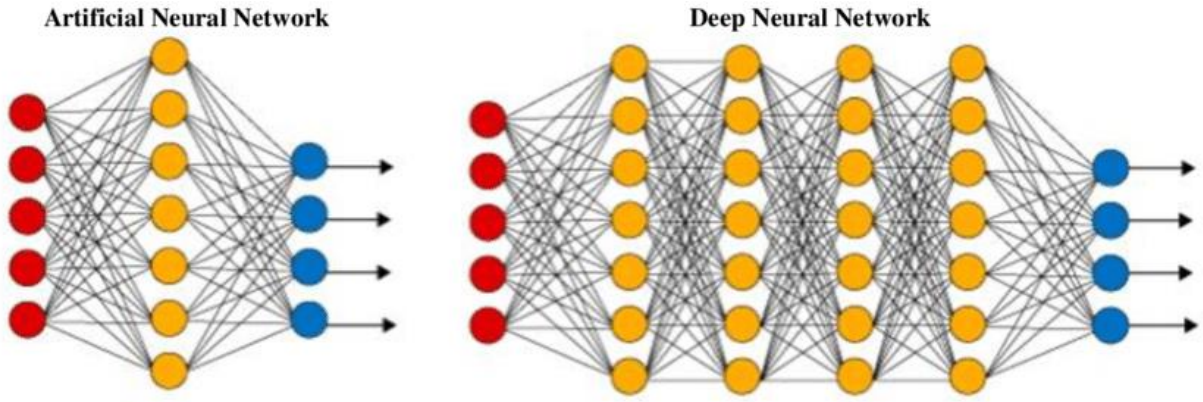

Рисунок 2.2 – Структура глибокої нейронної мережі у порівнянні зі звичайною багатошаровою нейромережею.

До глибоких архітектур належать багато варіантів кількох основних підходів. Кожна архітектура досягла успіху в певних областях. Не завжди

можливо порівняти продуктивність кількох архітектур, якщо їх оцінювали не на однакових наборах даних.

ГНМ, як правило, є мережами прямого поширення, в яких дані проходять з шару входу до шару виходу без повернення назад. Спочатку ГНМ створює карту віртуальних нейронів і призначає зв'язкам між ними випадкові числові значення, або «ваги». Ваги та входи перемножуються й повертають результат між 0 та 1. Якщо мережа не розпізнає певний образ точно, алгоритм підлаштовує ці ваги. Таким чином, алгоритм може робити певні параметри впливовішими, доки не визначить правильну математичну операцію для повної обробки даних.

2.1.3 Нейро-нечітка мережа Такагі-Сугено-Канг

Нечіткий підхід до класифікації з учителем заснований на застосуванні модифікованих моделей нейромережі Такагі-Сугено-Канг [7,8]. Було доведено, що система нейро-нечіткого логічного виведення Такагі-Сугено-Канг може бути застосована як універсальний апроксиматор даних з довільною точністю. Функція апроксимації Такагі-Сугено-Канг y(x) може бути спрощеною:

$$y(x) = \sum_{i=1}^{K} \mu_i(x) \left[p_{i0} + \sum_{j=1}^{N} p_{ij} x_j \right],$$

де μi (x) отримано раніше, а pij - коефіцієнти лінійних функцій Такагі-Сугено-Канг.

Структура нечіткої нейромережі відноситься до цієї модифікованої мережі Такагі-Сугено-Канг $f_i(x) = p_{i0} + \sum_{j=1}^{N} p_{ij} x_j$ як представлено на рис. 2.3, де fi(x) для i = 1, 2, ..., K, є лінійною функцією Такагі-Сугено-Канг, яка пов'язана з кожним довільним правилом.

Параметри частини передумов (значення елементів μ (xj) вибираються дуже суворо з використанням самоорганізованого алгоритму Густавсона-

Кесселя [7,8]. Далі вони заморожуються і не беруть участі у подальшій адаптації. Це означає, що коли вхідний вектор x подається в нейромережу, значення елементів μi (x) залишаються незмінними.

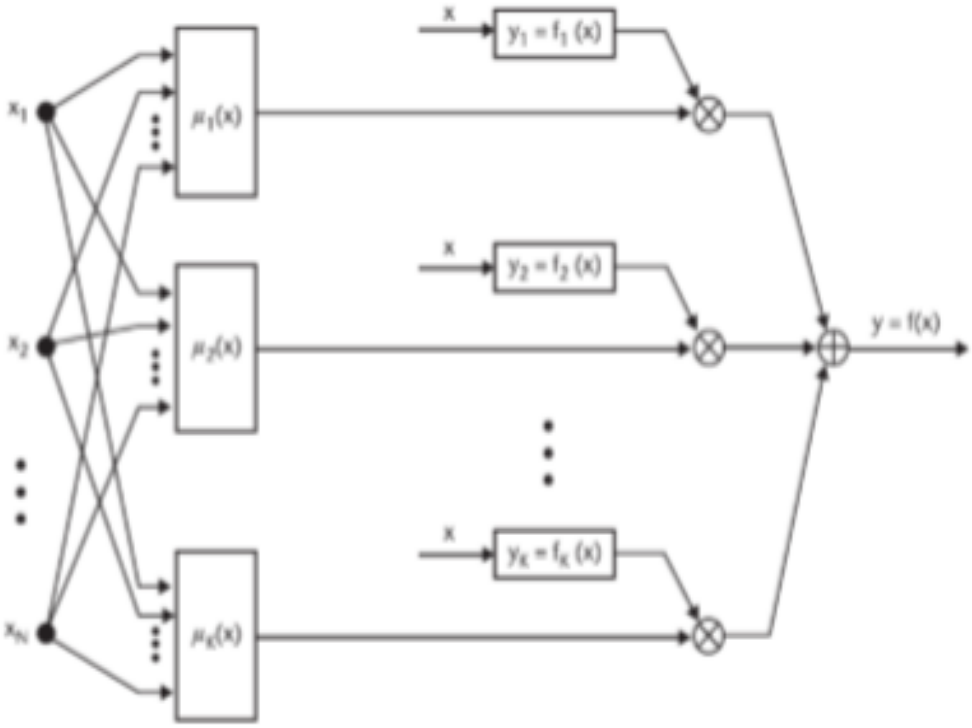

Рисунок 2.3 - Структура нейро-нечіткої мережі згідно модифікованої формули Такагі-Сугено-Канг

Решта параметрів лінійної функції Такагі-Сугено-Канг pij потім легко отримуються шляхом розв`язання допустимого набору лінійних рівнянь, які випливають з прирівнювання поточних значень y (xj) до значень dj для j = 1, 2,...,p. Визначення цих змінних виконується в одну дію при використанні методу сингулярного розкладання і методу псевдообернення [8].

У цій роботі пропонується обрати саме глибоку нейронну мережу, оскільки вона краще підійде для нашої задачі через здатність розділяти складні області об`єктів, а її узагальнююча спроможність буде достатньою для нашого набору даних.

2.2 Структура процесів обробки інформації інтелектуального модуля прогнозування ризику інсульту

Структура процесів обробки інформації інтелектуального модуля прогнозування ризику інсульту складається з таких етапів:

1. Завантаження набору даних із файла.
2. Попередня обробка даних (заповнення пропусків у даних, групування, визначення важливості параметрів).
3. Розбиття набору даних на навчальну вибірку (80%) та тестову вибірку (20%).
4. Створення моделі нейронної мережі.
5. Навчання створеної нейронної мережі.
6. Прогнозування ризику інсульту за допомогою моделі нейронної мережі.
7. Оцінювання точності моделі створеної нейронної мережі.

Процеси навчання та оцінювання точності мають вирішальне значення для будь-якої штучної нейронної мережі. Ці процеси, зазвичай, виконуються за допомогою двох наборів даних, одного набору для навчання, а іншого - тестового набору - для перевірки точності навченої нейромережі. У реальному житті ми зазвичай маємо лише один набір даних, а потім розбиваємо його на два окремі набори. Так і тут, для навчального набору ми використовуємо 80% прикладів і ще 20% прикладів використовуємо для оцінки точності моделі нейронної мережі.

2.3 Архітектура глибокої нейронної мережі прямого поширення

Розробимо структуру нейромережі для модуля прогнозування ризику інсульту. Оскільки в наборі даних пацієнти характеризуються 10-ма вхідними параметрами (gender, age, hypertension, heart_disease, ever_married, work_type, Residence_type, avg_glucose_level, bmi, smoking_status – див. рис. 1.1), то

вхідний шар нашої нейромережі повинен мати 10 нейронів (рис. 2.4). А кількість класів (станів пацієнтів), які потрібно прогнозувати, є 2 (немає ризику інсульта, є ризик інсульта), тому вихідний шар нейромережі повинен мати 2 нейрони, що будуть давати ймовірність того, що вхідні параметри пацієнта відповідають кожному із двох класів.

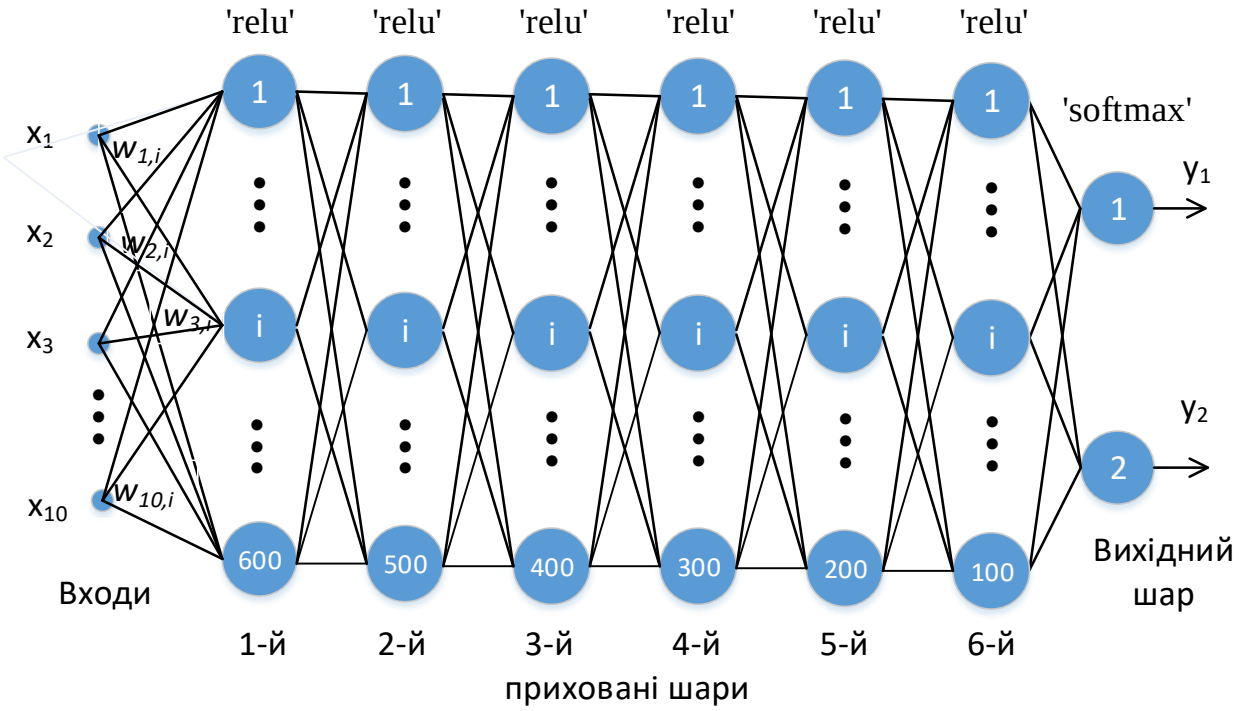

Рисунок 2.4 – Структура глибокої нейронної мережі прямого поширення для модуля прогнозування ризику інсульту

Остаточна класифікація може бути визначена вихідним нейроном з найвищим вихідним сигналом з плаваючою комою. Якби вихідні числа дорівнювали 0,24, 0,76, то такого пацієнта можна було б класифікувати як такого, що має імовірність 0,76 ризику інсульту (2-й клас).

У *i*-тий нейрон першого прихованого шару (рис. 2.4) надходять вхідні значення, наприклад, у нашому випадку 10 параметрів пацієнта: $x_1, x_2, \ldots x_{10}$ (елементи вхідного вектора) з відповідними вагами $w_{1,i}, w_{2,i}, \ldots w_{10,i}$. Далі, всередині нейрона відбувається обчислення двох операцій, а саме, композиції лінійної та нелінійної функції:

Було обрано 6 прихованих шарів,. Число нейронів у першому прихованому шарі - 600, у другому прихованому шарі - 500, у третьому прихованому шарі - 400, у четвертому прихованому шарі - 300, у п`ятому прихованому шарі - 200, у шостому прихованому шарі - 100. Функція активації була обрана Leaky-relu [3] для всіх прихованих шарів, а для вихідного шару – Softmax [3].

У 2-му, 3-му та 4-му прихованих шарах передбачено використання дропауту. Коефіцієнти дропауту для 2-го та 3-го шарів 0,2, а для 4-го шару – 0,3.

Дропаут або викидання (від англ. dropout) - метод регуляризації штучних нейронних мереж, призначений для зменшення перенавчання мережі за рахунок запобігання складним коадаптаціям окремих нейронів на тренувальних даних під час навчання [9]. Термін "dropout" (вибивання, викидання) характеризує виключення певного відсотка (наприклад 30%) випадкових нейронів (які знаходяться як у прихованих, так і видимих шарах) на різних ітераціях (епохах) під час навчання нейронної мережі. Це дуже ефективний спосіб усереднення моделей усередині нейронної мережі. В результаті найбільш навчені нейрони отримують у мережі більшу вагу [9]. Такий прийом значно збільшує швидкість навчання, якість навчання на тренувальних даних, а також підвищує якість прогнозування моделі на нових тестових данних [9].

Для завдання викидання у різних бібліотеках машинного навчання необхідно вибрати значення гіперпараметра, який вказує на ймовірність занулення елемента нейронної мережі. В результаті генерується маска, що складається з 0 і 1, яка вказує, які елементи будуть прибрані. Як правило, маска генерується незалежно на кожному кроці градієнтного спуску. Важливо, що на етапі прогнозування dropout нічого не змінює. На етапі передбачення dropout «вимикається»: внутрішні значення використовуються як є без множення на маску. А щоб шар знав, навчається він зараз чи прогнозує, у нейромережевих бібліотеках у класі шару зазвичай реалізовано перемикання між цими режимами (наприклад, булев прапор training у pytorch-модулях) [10].

2.4 Розробка алгоритму роботи інтелектуального модуля прогнозування ризику інсульту

Згідно з метою роботи та постановкою задачі було розроблено алгоритм інтелектуального модуля прогнозування ризику інсульту, представлений на рис. 2.5.

Рисунок 2.5 – Схема алгоритму роботи інтелектуального модуля прогнозування ризику інсульту

Першим кроком в інтелектуальному модулі прогнозування ризику інсульту буде завантаження бібліотек (блок 1). Потім йде завантаження набору даних (блок 2) та розділення набору даних на навчальну та тестову вибірки (блок 3).

Далі переходимо до попередньої обробки набору даних (блок 4). Потім проводиться створення нейронної мережі (блок 5). За тим починається процес навчання нейромережі (блок 6), який триває 200 епох (блок 7), після чого проводиться оцінка специфічності роботи нейромережі (блок 8) та виведення специфічності прогнозування ризику інсульту на тестовій вибірці (блок 9).

2.5 Висновок до розділу 2

У розділі було обґрунтовано вибір типу нейронної мережі - глибока нейромережа прямого поширення (в яких дані проходять з шару входу до шару виходу без повернення назад) для модуля прогнозування ризику інсульту. Обрано архітектуру глибокої нейромережі, яка має 10 входів, 6 прихованих шарів відповідно по 600, 500, 400, 300, 200, 100 нейронів та вихідний шар із 2 нейронів. Число нейронів у першому прихованому шарі - 600, у другому прихованому шарі - 500, у третьому прихованому шарі - 400, у четвертому прихованому шарі - 300, у п`ятому прихованому шарі - 200, у шостому прихованому шарі - 100.Було обрано функцію активації Leaky-ReLu у прихованих шарах та функцію активації Softmax у вихідному шарі. Розроблено структуру процесів обробки інформації інтелектуального модуля. Також розроблено алгоритм роботи програмного модуля прогнозування ризику інсульту.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОГО МОДУЛЯ ПРОГНОЗУВАННЯ РИЗИКУ ІНСУЛЬТУ НА ОСНОВІ НЕЙРОМЕРЕЖІ

3.1 Обґрунтування вибору мови та спеціалізованих бібліотек

Для реалізації інтелектуального модуля прогнозування ризику інсульту за допомогою глибокої нейронної мережі обрано мову програмування Python та програмне середовище Anaconda.

Python [11] (читається — «Пайтон») —об'єктно-орієнтована інтерпретована мова програмування високого рівня із суворою динамічною типізацією. Наявність структури даних високого рівня вкупі із динамічною семантикою та динамічним зв'язуванням роблять її зручною для швидкої розробки програм. Python підтримує пакети модулів та модулі і сприяє повторному використанню кодів та модульності. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на усіх платформах. У мові Python підтримується кілька парадигм програмування, зокрема: функціональна, об'єктно-орієнтована, аспектно-орієнтована та процедурна.

Переваги мови Python:

- зручний синтаксис (для виділення блоків застосовуються відступи);
- властивість перенесення програм (властиво більшості мов програмування);
- стандартний дистрибутив має велике число корисних модулів (включно з модулем розробки графічного інтерфейсу);
- можливість використовувати Python у діалоговому режимі (корисне для розв'язання простих завдань та експериментування);
- стандартний дистрибутив має просте, але потужне середовище розробки IDLE, яке написане на Python;

- зручний для вирішення математичних задач (має засоби роботи з цілими числами довільної величини, комплексними числами, може використовуватися як потужний калькулятор);
- відкритий код (можливість редагування коду іншими користувачами).

Недоліки:

Невисока швидкодія. Python, як і більшість інших інтерпретованих мов, що не застосовують JIT-компілятори, мають цей загальний недолік — відносно невелику швидкість виконання програм. Однак, у випадку з Python цей недолік компенсується зменшенням тривалості часу розробки програми. У середньому, програма на Python, в 2-4 рази займає менше місця, ніж її аналог на C++ або Java. Збереження байт-коду (файли типу .pyc і .pyo) дозволяє інтерпретатору, на відміну, від мови Perl, не витрачати час на перекомпіляцію коду модулів при кожному запуску. Крім того, існує спеціальна JIT-бібліотека psyco (на жаль призводить до збільшення споживання оперативної пам'яті). Ефективність psyco суттєво залежить від архітектури програми.

Оскільки ми використовуємо нейромережі, то для цієї роботи знадобляться бібліотеки Keras [12], NumPy [13] та Pandas [14].

Keras — це відкрита нейромережева бібліотека, написана на мові Python. Вона може працювати над TensorFlow, R, Microsoft Cognitive Toolkit, Theano та PlaidML. ЇЇ було спроектовану для уможливлення швидких експериментів з мережами глибокого навчання, та зосереджено на тому, аби вона була модульною, зручною у використанні та розширюваною. Keras було створено як частину дослідницьких дій проєкту ONEIROS (англ. Open-ended Neuro-Electronic Intelligent Robot Operating System).

У 2017 році команда TensorFlow Google вирішила підтримувати Keras в основній бібліотеці TensorFlow. Було роз`яснено, що Keras замислювався більше як інтерфейс, аніж як самостійна система машинного навчання.

NumPy (скорочено від Numerical Python) — це фреймворк з відкритим кодом для мови Python. NumPy має наступні можливості:

- підтримка багатомірних масивів (матриць);

- підтримка високорівневих матфункцій для роботи з багатомірними масивами.

Pandas — програмна бібліотека [14], написана для мови Python для маніпулювання даними та їх аналізу. Вона, пропонує структури даних та операції над часовими рядами та чисельними таблицями. Pandas є вільним програмним забезпеченням, що випускається за ліцензією BSD. Назва Pandas походить від терміну «панельні дані» (англ. panel data), який в економетрії означає багатомірні структуровані набори даних.

Можливості бібліотеки:

- Інструменти для записування та зчитування даних між структурами даних у пам'яті та із різними форматами файлів.
- Об'єкт DataFrame для маніпулювання даними із вбудованим індексуванням.
- Вирівнювання даних та підтримка пропущених даних.
- Отримання зрізів за мітками, отримання піднаборів з великих наборів даних та індексування з розширеними можливостями.
- Переформатовування з метою отримання зведених наборів даних.
- Вставлення та вилучення стовпчиків у структурах даних.
- Злиття та з'єднання у наборах даних.
- Рушій групування, який дозволяє робити над наборами даних операції розділення-об'єднання-зміни (англ. split-apply-combine).
- Функціональність для роботи із часовии рядами: породження діапазонів дат та перетворення частоти, статистики рухливого вікна, зсування дат та запізнювання.
- Ієрархічне індексування осей для роботи з даними високої розмірності в структурі даних нижчої розмірності.

Також у роботі використовуються бібліотеки для графічного представлення: Matplotlib, Seaborn.

Для масштабування та передискретизації використовуються бібліотеки Sklearn (попередня обробка) та Imblearn.

3.2 Опис набору даних для навчання та тестування модуля

У роботі використовується набір даних [1] з Kaggle . Це параметри пацієнтів, які досліджувались на наявність інсульту. Фрагмент набору даних для визначення інсульту показано на рис. 3.1.

1	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
2	9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
3	51676	Female	61	0	0	Yes	Self-employed	Rural	202.21	N/A	never smoked	1
4	31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
5	60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
6	1665	Female	79	1	0	Yes	Self-employed	Rural	174.12	24	never smoked	1
7	56669	Male	81	0	0	Yes	Private	Urban	186.21	29	formerly smoked	1
8	53882	Male	74	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
9	10434	Female	69	0	0	No	Private	Urban	94.39	22.8	never smoked	1
10	27419	Female	59	0	0	Yes	Private	Rural	76.15	N/A	Unknown	1

Рисунок 3.1 – Фрагмент набору даних для визначення інсульту

Розглянемо цей набір даних детально.

У наборі даних є 5110 записів по 11 параметрів для кожного пацієнта (за винятком id), включаючи цільову змінну:

1) Стать (gender): визначає стать пацієнта. Чоловік або жінка;

2) Вік (age): коливається від 0,08 років до 82,0 років;

3) Гіпертонія (hypertension): 0 - немає гіпертонії, 1 - є гіпертонія;

4) Хвороби серця (heart_disease): 0 - немає хвороби серця,1 - є хвороба серця;

5) Сімейний стан (ever_married): Ні - ніколи не був одружений, Так - у шлюбі або хоча б один раз був;

6) Тип роботи (work_type): Загальна сфера роботи пацієнта: діти (Children), державна робота (Govt_job), ніколи не працював (Never_worked), приватний сектор (Private), самозайнятість (Self-employed);

7) Тип місця проживання (Residence_type): Тип місцевості, в якій вони проживають: сільська (Rural), міська (Urban);

8) Рівень глюкози (avg_glucose_level): Середній рівень глюкози в крові: діапазон від 55,12 до 271,14;

9) Індекс маси тіла (bmi): Цей атрибут означає індекс маси тіла людини. Це числові дані: коливається від 10,3 до 97,6;

10) Чи палить пацієнт (smoking_status): раніше курив (formerly smoked), ніколи не курив (never smoked), курить (smokes), невідомо (unknown) - означає, що інформація про паління для цього пацієнта недоступна;

11) Інсульт (stroke): 0 - не було інсульту, 1 - був інсульт;

Набір даних містить параметри 5110 пацієнтів, причому 4861 (клас 0) з них не мали інсульту, а 249 мали інсульт (клас 1) – див. рис. 3.2.

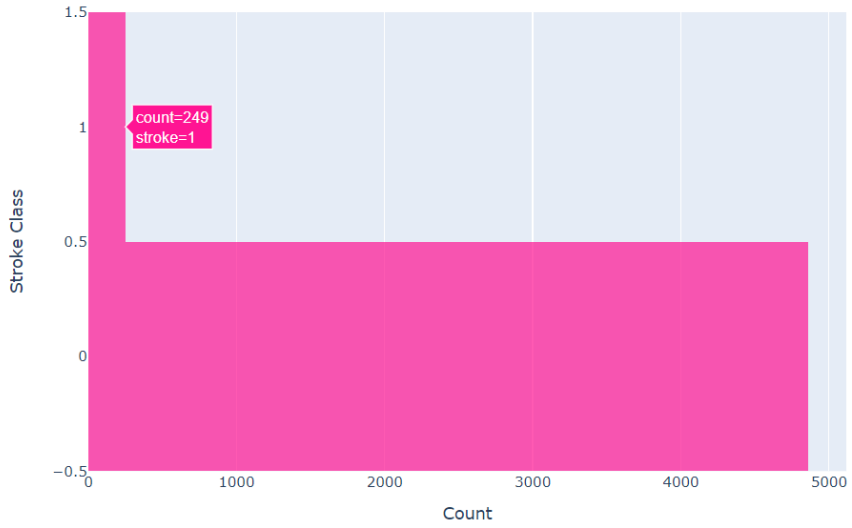

Рисунок 3.2 – Кількість пацієнтів з інсультом (strok=1) та без інсульту (strok=0) у наборі даних.

Тобто у цьому наборі даних наявний класовий дисбаланс – це коли розподіл прикладів по класах не є рівномірним (не наближається до подібних розмірів) і натомість сильно спотворений на користь деяких класів. Ці проблеми зазвичай називають незбалансованою класифікацією. Випадки, коли є невеликий дисбаланс, розглядаються як звичайна проблема класифікації, тоді

як випадки, коли є серйозний дисбаланс, вимагають спеціальних методів для побудови належної моделі.

Клас або класи з більшою кількістю прикладів у незбалансованому наборі даних називають мажоритарним класом, тоді як класи, які мають мало екземплярів, називають міноритарним класом. Ця проблема класу створює проблему побудови належної моделі, оскільки клас більшості може створювати хибне відчуття точності під час побудови моделі. Модель може правильно передбачати у 95% випадків, але це навряд чи багато значить, коли 95% вибірок зміщено до одного класу. Крім того, більшість моделей машинного навчання припускають рівномірний розподіл класів, тому, якщо використовувати незбалансований набір даних, він не дасть справді точних прогнозів.

Маємо приблизно 5:95 відношення класового дисбалансу. Це вважається серйозно незбалансованим співвідношенням класів, де більшість – це пацієнти, у яких не було інсульту, а меншість – пацієнти, які мали інсульт.

3.3 Попередня обробка інформації із набору даних

Етап попередньої обробки даних має справу з відсутніми значеннями в наборі даних, перетворює числові значення в тип рядка, щоб ми могли виконати однократне кодування, а також обробляємо недостатню вибірку цільового атрибута.

• Аналіз даних показує, що існує дуже низька кореляція між атрибутами, найвища кореляція спостерігалася між віком та ІМТ зі значенням 0,32, усі інші значення кореляції були меншими за 0,3.

• Значення NAN були замінені на середнє значення ІМТ, яке становило 36,6.

• було додано фіктивні функції за допомогою методу get dummies(), наявного в бібліотеці Pandas. Він перетворює категоріальні дані у фіктивні змінні Dummies або індикатор. Фіктивна змінна — це числова змінна, яка кодує категоріальну інформацію, подібну до однократного кодування.

У фіктивній змінній:

«1» кодує наявність категорії, а «0» кодує відсутність категорії. Якщо категорій більше однієї, створюється окремий стовпець.

• Під час EDA (Exploratory Data Analysis - дослідницький аналіз даних) було виявлено, що набір даних, використаний у цьому документі, містить лише 249 записів про людей, які перенесли інсульт, і 4861 записів про людей без інсульту, що робить набір даних дуже незбалансованим: лише близько 4,8% від загальної кількості записів належать до меншини «інсульт». Якби алгоритми машинного навчання застосували до такого набору даних, це призвело б до низької продуктивності на меншинному класі, продуктивність якого є найважливішою.

• Щоб подолати цю проблему незбалансованих класів, Використовується випадкове розширення прикладів класу «інсульт».

• Випадкове розширення прикладів класу «інсульт» включає вибір випадкових прикладів із класу меншості із заміною та доповненням навчальних даних кількома копіями цього екземпляра, отже, можливо, що один екземпляр може бути обраний кілька разів.

Групування по віку:

- Дитина 0-11,9 років (Child) ,
- Підліток 12-17,9 років (Teenager),
- Молодь 18-29,9 років,
- Тридцятирічні 30-39,9 років (Thirties),
- Сорокарічні 40-49,9 років (Forties),
- П'ятидесятирічні 50-59,9 років (Fifties),
- Шестидесятирічні 60-69,9 років (Sixties),
- Сімдесятирічні 70-79,9 років (Seventies),
- Вісімдесятирічні+ 80- … років (Eighties).

```
df_stroke_categorical["Age Bracket"] = pd.cut(x=df_stroke_numeric['age'], bins=[0.0,11.9,17.9,29.9,39.9,49.9,
59.9,69.9,79.9,np.inf],
```

```
                                        labels=["Child","Teenager","18-29.9","Thirties","Forties","Fifties"
,"Sixties","Seventies","Eighties +"])
        =["Child","Teenager","18-29.9","Thirties","Forties","Fifties","Sixties","Seventies","Eighties +"])
```

Групування по середньому рівню глюкози.

Визначено такі діапазони eAG (estimate Average Glucose - оцінка середнього рівня глюкози):

- Нормальний діапазон: менше 114 mg/dL
- Діапазон переддіабету: більше 114 mg/dL і менше 140 mg/dL
- Діапазон діабету: більше 140 mg/dL

```
df_stroke_categorical["Glucose Category"] = pd.cut(x=df_stroke_numeric['avg_glucose_level'], b
ins=[-np.inf,114,140,np.inf],
                                          labels=["Normal","Prediabetes","Diabetes"])
```

Групування по індексу маси тіла (IMT). Діапазони IMT:

- Нижче 18,5 – недостатня вага (Underweight),
- 18,5 – 24,9 – здорова вага (Healthy Weight),
- 25,0 – 29,9 – надмірна вага (Overweight),
- 30,0 і вище – ожиріння (Obese),

```
df_stroke_categorical["BMI Category"] = pd.cut(x=df_stroke_numeric['bmi'], bins=[-np.inf,18.4,
24.9,29.9,np.inf],
                                           labels=["Underweight","Healthy Weight","Overweigh
t","Obese"])
```

Об`єднання колонок «гіпертонія» та «хвороби серця».

Поєднання гіпертонії та серцевих захворювань може бути сильнішим показником інсульту, ніж будь-яке з них окремо, тому людина, яка має і гіпертонію, і серцеву хворобу, буде піддаватися більшому ризику інсульту, ніж той, хто має лише одне з двох.

```
df_stroke_numeric["hyp_hd"] = df_stroke_numeric["heart_disease"] + df_stroke_numeri
c["hypertension"]
```

Дослідницький аналіз даних. Важливість параметрів. Тут ми використовуємо CatBoostClassifier, щоб визначити найважливіші параметри.

Було визначено, що ознаки hyp_hd (яка вимірює, чи є у пацієнта гіпертонія та хвороба серця, чи є одна, чи інша, чи жодної немає) і вік мають однакові рівні важливості. На третьому місці у нас avg_glucose_level. Це має сенс, оскільки вищий рівень глюкози підвищує ризик інсульту. Як не дивно, ever_married має трохи більше значення, ніж heart_disease. Повний рейтинг важливості параметрів показано на рис.3.3.

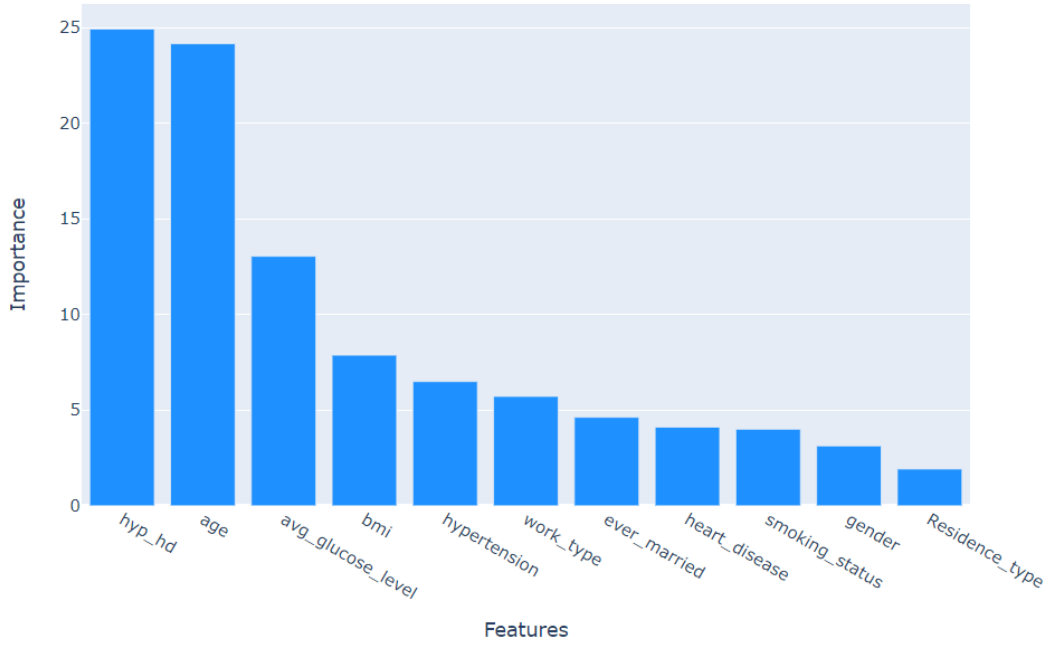

Рисунок 3.3 – Повний рейтинг важливості параметрів

3.4 Програмна реалізація інтелектуального модуля прогнозування ризику інсульту

Програмна реалізація здійснювалась на мові програмування Python у середовищі розробки Anaconda. Програма є консольною і не має користувацького графічного інтерфейсу.

Першим кроком в модулі є завантаження бібліотек (рис. 3.4):

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')


import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go
from plotly.offline import plot, iplot, init_notebook_mode
init_notebook_mode(connected=True)

sns.set(rc={'figure.figsize':(16,8)})

RANDOM_STATE = 115
```

Рисунок 3.4 – Завантаження бібліотек

Далі треба завантажити набір даних (рис. 3.5).

```
df_stroke_original = pd.read_csv('../input/stroke-prediction-dataset/healthcare-dataset-stroke-data.csv')
# df_stroke_original = df_stroke_original.drop(["id"],axis = 1)
df_stroke_original.shape
```

Рисунок 3.5 – Завантаження набору даних

Далі ми розділяємо отримані дані на два набори – навчальний та тестовий (рис. 3.6).

```
split = StratifiedShuffleSplit(n_splits = 1, test_size = 0.2, random_state=RANDOM_STATE)
for train_index,test_index in split.split(df_stroke_gender,df_stroke_gender[stratify_on]):
    train_set = df_stroke_gender.loc[train_index]
    test_set = df_stroke_gender.loc[test_index]

train_set = train_set.reset_index(drop=True)

X_train = train_set.drop(["stroke"],axis=1)
y_train = train_set["stroke"].copy(deep = True)

X_test = test_set.drop(["stroke"],axis=1)
y_test = test_set["stroke"].copy(deep = True)

X_train_prepped = full_pipeline.fit_transform(X_train)
X_test_prepped = full_pipeline.transform(X_test)

n_cols = X_train_prepped.shape[1]
target = to_categorical(y_train)
target_test = to_categorical(y_test)
```

Рисунок 3.6 – Розділення набору даних на навчальний та тестовий

Оскільки наш набір даних дуже незбалансований, ми надаємо перевагу класу меншості. Це робиться шляхом встановлення ваги класу меншості:

$$\frac{1}{samples_{minority} * weight}$$

Скалярне значення ваги збільшує або зменшує знаменник, фактично збільшуючи/зменшуючи вагу класу меншості в моделі.

Для основного класу ми встановили вагу:

$$\frac{1}{samples_{majority}}$$

Код для встановлення ваг класів подано на рис. 3.7.

```
weight_multiplier = 0.7
counts = np.bincount(df_stroke_original["stroke"])
weight_for_0 = 1.0/counts[0]
weight_for_1 = 1.0/(counts[1]*weight_multiplier)
class_weight = {0: weight_for_0, 1: weight_for_1}
```

Рисунок 3.7 – Фрагмент коду для встановлення ваг класів

Тепер нам потрібно вибрати модель, яку ми будемо використовувати. У нашій задачі ми намагаємося передбачити клас ризику інсульту на основі даних атрибутів. Тому ми обрали глибоку нейронну мережу (мережа з послідовним слідуванням шарів – **Sequential**) – рис. 3.8.

```
model = Sequential()
model.add(Dense(600,activation="leaky_relu",input_shape=(n_cols,),
          kernel_initializer = tf.keras.initializers.HeNormal(seed=RANDOM_STATE),
          bias_initializer = tf.keras.initializers.Zeros(),
          kernel_regularizer=tf.keras.regularizers.L2(1e-7)))
model.add(Dense(500,activation="leaky_relu",
          kernel_initializer = tf.keras.initializers.HeNormal(seed=RANDOM_STATE),
          bias_initializer = tf.keras.initializers.Zeros(),
          kernel_regularizer=tf.keras.regularizers.L2(1e-3)))
model.add(Dropout(0.2))
model.add(Dense(400,activation="leaky_relu",
          kernel_initializer = tf.keras.initializers.HeNormal(seed=RANDOM_STATE),
          bias_initializer = tf.keras.initializers.Zeros(),
          kernel_regularizer=tf.keras.regularizers.L2(1e-3)))
model.add(Dropout(0.2))
model.add(Dense(300,activation="leaky_relu",
          kernel_initializer = tf.keras.initializers.HeNormal(seed=RANDOM_STATE),
          bias_initializer = tf.keras.initializers.Zeros()))
model.add(Dropout(0.3))
model.add(Dense(200,activation="leaky_relu",
          kernel_initializer = tf.keras.initializers.HeNormal(seed=RANDOM_STATE),
          bias_initializer = tf.keras.initializers.Zeros()))
model.add(Dense(100,activation="leaky_relu",
          kernel_initializer = tf.keras.initializers.HeNormal(seed=RANDOM_STATE),
          bias_initializer = tf.keras.initializers.Zeros()))
model.add(Dense(2,activation="softmax"))
model.compile(optimizer="adam",
              loss="categorical_crossentropy",
              metrics=["accuracy",
                       tf.keras.metrics.AUC(name="auc",from_logits=True),
                       tf.keras.metrics.Precision(name="precision"),
                       tf.keras.metrics.Recall(name="recall")])
```

Рисунок 3.8 – Фрагмент коду створення моделі нейромережі

Із рис. 3.8 видно, що нейромережа містить шість прихованих повнозв`язних (**Dense**) шарів з відповідно 600, 500, 400, 300, 200 та 100

нейронами в кожному, причому входів у першого шару нашої нейромережі відповідає числу колонок набору даних (`input_shape=(n_cols)`. Кількість вихідних класів нейромережі, які потрібно прогнозувати, - 2 («інсульт не можливий» і «інсульт можливий»), тому вихідний шар нейромережі має 2 нейрони (**Dense(2)**), що будуть давати ймовірність того, що вхідні параметри відповідають кожному із двох класів. А імовірність виходить тому, що для вихідного шару була обрана функція активації **softmax**. Для всіх прихованих шарів була обрана функція активації **leaky-relu**.

Для навчання нейромережі використовуємо алгоритм оптимізації "Adam" `(optimizer="adam")`, який є модифікацією методу зворотного розповсюдження помилки ('backprop').

Для оцінки втрат при навчанні використовуєо параметр `categorical_crossentropy`. Для оцінки продуктивності моделі ШНМ використовуємо такі параметри достовірності класифікації як `"auc"`, `"precision"` та `"recall"`.

Потім, ми навчаємо нашу мережу – рис. 3.9.

```
history = model.fit(X_train_prepped,
                    target,
                    epochs=200,
                    validation_split=0.3,
                    verbose=False,
                   class_weight = class_weight)
y_pred_neural_net_proba = model.predict(X_test_prepped)
y_pred = y_pred_neural_net_proba.argmax(axis=-1)
```

Рисунок 3.9 – Фрагмент коду навчання моделі нейромережі

Бачимо, що обрано тривалість навчання в 200 епох та частка валідаційних прикладів – 0,3.

Нарешті, проводимо оцінку роботи розробленої нейронної мережі – рис. 3.10.

```
s = list(np.array(score_summary(y_test,y_pred)))
s.append("Neural Network (with Weighted Classes)")
k = pd.DataFrame(s).T
k.columns = ensb.columns
ensb = pd.concat([ensb,k],axis=0)
```

Рисунок 3.10 – Фрагмент коду оцінювання точності роботи моделі нейромережі

Під точністю мережі тут розуміється достовірність розпізнавання на тестовій вибірці. Результати оцінювання точності роботи моделі нейромережі детально будут проаналізовані у наступному розділі.

3.4 Висновок до розділу 3

У розділі було обґрунтовано вибір мови програмування Python та спеціалізованих бібліотек Keras (відкрита нейромережева бібліотека, написана на мові Python) NumPy (фреймворк з відкритим кодом для мови Python) та Pandas (програмна бібліотека для маніпулювання даними та їх аналізу) для програмної реалізації інтелектуального модуля прогнозування ризику інсульту. Проведено аналіз набору даних прогнозування ризику інсульту для навчання та тестування модуля. Розроблено процедури попередньої обробки інформації із набору даних. Обрано алгоритм оптимізації "Adam", який є модифікацією методу зворотного розповсюдження помилки ('backprop').

Описано основні етапи програмної реалізації та функціонування інтелектуального модуля прогнозування ризику інсульту, що складається з завантаження бібліотек, завантаження набору даних із параметрами пацієнтів та їх ідентифікатором класу, аналізу цих параметрів, створення, тренування, оцінки точності моделі нейромережі та виведення результатів.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ІНТЕЛЕКТУАЛЬНОГО МОДУЛЯ ПРОГНОЗУВАННЯ РИЗИКУ ІНСУЛЬТУ НА ОСНОВІ НЕЙРОМЕРЕЖІ

Розроблена програма є консольною, вона не має графічного інтерфейсу користувача, а тому запускається із середовища розробки програм, виконує всі закладені в неї функції обробки набору даних та виводить отриманий результат. Коли ми запускаємо код програми, то крім інших, отримуємо такий результат:

```
Epoch 1/200
341/341 [==============================] - 1s 1ms/step - loss: 0.3076 - get_f1: 0.0
111
Epoch 2/200
341/341 [==============================] - 0s 1ms/step - loss: 0.1653 - get_f1: 0.0
000e+00
Epoch 3/200
341/341 [==============================] - 0s 1ms/step - loss: 0.1476 - get_f1: 0.0
000e+00
…

Epoch 198/200
341/341 [==============================] - 0s 1ms/step - loss: 0.1250 - get_f1: 0.0
792
Epoch 199/200
341/341 [==============================] - 0s 1ms/step - loss: 0.1259 - get_f1: 0.0
567
Epoch 200/200
341/341 [==============================] - 0s 1ms/step - loss: 0.1320 - get_f1: 0.1
169

Accuracy: 68.6%, Precision: 98.4%, Sensitivity: 68.1 %, Specificity: 78 %
```

Тобто виводиться результат кожної із 200 епох навчання у вигляді: номер епохи/всього епох/тривалість епохи/похибка навчання на цій епосі (loss).

Взагалі, на вхід нейромережі надходять параметри пацієнтів двох класів: «не мали інсульту» і «мали інсульт». І можливі наступні варіанти вихідного сигналу нейромережі:

1) пацієнта, який не мав інсульту, ідентифіковано як такого, що не мав інсульту (вірнопозитивний результат – True positive – TP);

2) пацієнта, який не мав інсульту, ідентифіковано як такого, що мав інсульт (хибнонегативний результат – False Negative – FN, помилка 2-го роду);

3) пацієнта, який мав інсульт, ідентифіковано як такого, що мав інсульт (вірнонегативний результат – True Negative – TN);

4) пацієнта, який мав інсульт, ідентифіковано як такого, що не мав інсульту (хибнопозитивний результат – False positive – FP, помилка 1-го роду).

Детально ці 4 випадки відображено у матриці невідповідностей [15] (confusion matrix) на рис. 4.1

	Результат ідентифікації	
	пацієнт не мав інсульту	пацієнт мав інсульт
На вході пацієнт, який не мав інсульту (загальна кількість пацієнтів без інсульту P)	Вірнопозитивний True positive – **TP**	Хибнонегативний False Negative – **FN** помилка 2-го роду
На вході пацієнт, який мав інсульт (загальна кількість пацієнтів з інсультом N)	Хибнопозитивний False positive – **FP** помилка 1-го роду	Вірнонегативний True Negative – **TN**

Рисунок 4.1 – Матриця невідповідностей (можливі варіанти вхідного сигналу та результату ідентифікації)

Вірнопозитивні та вірнонегативні – це результати, які правильно ідентифіковані і, отже, показані зеленим кольором. А хибнопозитивні та хибнонегативні – це результати, які неправильно ідентифіковані (помилкові) і, отже, показані червоним кольором, їх потрібно мінімізовувати.

На основі цих чотирьох параметрів розраховуються такі показники якості класифікатора : достовірність (accuracy), влучність (precision), чутливість (sensitivity) та специфічність (specificity) [15].

Достовірність. Достовірність є найбільш інтуїтивно зрозумілим показником якості класифікаторів, і це просто відношення кількості вірних

результатів ідентифікації прикладів (із тестової вибірки) до загальної кількості прикладів, поданих на ідентифікацію.

$$Accuracy = (TP + TN) / (TP + FP + FN + TN)$$

Але достовірність не є найкращим показником якості класифікаторів. Так, достовірність є гарним показником, але лише тоді, коли у нас є симетричні набори даних, тобто коли кількість хибнопозитивних та хибнонегативних результатів майже однакові. Тому часто для характеристики якості класифікаторів застосовують інші параметри, щоб всебічно оцінити якість класифікаторів.

Влучність. Влучність – це відношення вірнопозитивних (True positive – TP) результатів ідентифікації до загальної кількості позитивних результатів ідентифікації.

$$Precision = TP / (TP + FP)$$

Ця метрика відповідає на питання про те, яка частка тих пацієнтів, що ідентифіковані як без інсульту, дійсно не перенесли інсульт, Цей параметр показує частку вірно ідентифікованих пацієнтів без інсульту із усіх пацієнтів, які ідентифіковані як «без інсульту». Висока влучність пов'язана з низькою кількістю хибнопозитивних.

Чутливість (Sensitivity) (або інша назва - повнота (recall)) – це відношення вірнопозитивних (True positive – TP) результатів ідентифікації до загальної кількості пацієнтів, які насправді не перенесли інсульту. Чутливість (рівень істинно позитивних результатів) — це ймовірність позитивного результату тесту за умови, що індивід справді позитивний.

$$Recall = TP / (TP + FN)$$

Специфічність (specificity) – це відношення вірнонегативних (True negative – TN) результатів ідентифікації до загальної кількості пацієнтів, які насправді перенесли інсульт. Специфічність (істинно негативний рівень) — це ймовірність негативного результату тесту за умови, що індивід справді негативний.

Specificity = TN / (TN + FP)

У медицині чутливість і специфічність математично описують точність тесту, який повідомляє про наявність або відсутність захворювання. Якщо особи, які не мають захворювання, вважаються «позитивними», а ті, у кого є захворювання, вважаються «негативними», тоді чутливість є мірою того, наскільки добре тест може ідентифікувати справді позитивні результати, а специфічність — це міра того, наскільки добре тест може ідентифікувати справді негативні результати.

Достовірність працює найкраще, якщо хибнопозитивні та хибнонегативні результати мають однакову вартість. Якщо вартість хибнопозитивних та хибнонегативних результатів сильно різниться, краще поглянути і на чутливість і на специфічність.

Що стосується оцінки ефективності розробленого модуля, достовірність (Accuracy) не є найкращим показником через нерівномірність розподілу даних по класам. Слід переглянути інші показники: чутливість і специфічність. Ці показники дають більш чітке уявлення про те, які конкретні риси ми хочемо отримати від нашої моделі, а потім можемо використовувати бажану для конкретних випадків використання (наприклад, якщо ми хочемо мінімізувати кількість хибнопозитивних результатів, ми віддамо перевагу моделі з вищою оцінкою специфічності).

Чутливість (Sensitivity) вимірює фактичні позитивні результати із загальної кількості позитивних результатів, поданих на прогнозування. Ця метрика намагається визначити частку вірних позитивних ідентифікацій.

Простіше кажучи, вищий показник чутливості зменшує кількість хибнонегативних (FN). Тож у контексті нашої проблеми хибнонегативним результатом буде діагностувати ризик інсульту у пацієнта, який насправді немає ризику інсульту (або, у більш конкретному медичному контексті, здоровій людині сказати, що вона хвора). Влучність прагне зменшити кількість таких помилок. Але влучність не такий важливий показник, як, скажімо, чутливість (Sensitivity), оскільки здорову людину, яку неправильно діагностовано як хвору, можна додатково обстежити, щоб з'ясувати справжність діагностованої їй хвороби, якщо вона взагалі є. А от сказати хворій людині, що вона здорова – це більш фатальна помилка, бо людина заспокоїться, не буде лікуватись і хвороба буде швидко прогресувати.

Специфічність (specificity) вимірює вірнонегативні прогнозовані результати із загальної кількості негативних результатів, поданих на ідентифікацію. Специфічність прагне відповісти на питання, які пропорції фактичних негативів були визначені правильно. Простіше кажучи, вищий показник специфічності зменшує кількість хибнопозитивних результатів, тобто негативних результатів, визначених як позитивні. Тож у контексті нашої проблеми хибнопозитивним результатом буде пацієнт, у якого є ризик інсульту, і якому неправильно поставлено діагноз, що він не має ризику інсульту (або, у більш конкретному медичному контексті, хворій людині сказали, що вона здорова). Зрозуміло, що це очевидна проблема, оскільки інсульт є серйозним наслідком для здоров'я, і його ризик слід негайно виявити. Що стосується оцінки розробленого модуля, то специфічність (specificity) буде основним показником, на який потрібно звертати увагу, оскільки це зменшить кількість хворих пацієнтів, які можуть бути помилково визначені здоровими.

Матриця невідповідностей, отримана при тестуванні розробленого модуля на основі нейронної мережі показана на рис. 4.2. А в табл. 4.1 представлено результати розрахованих метрик.

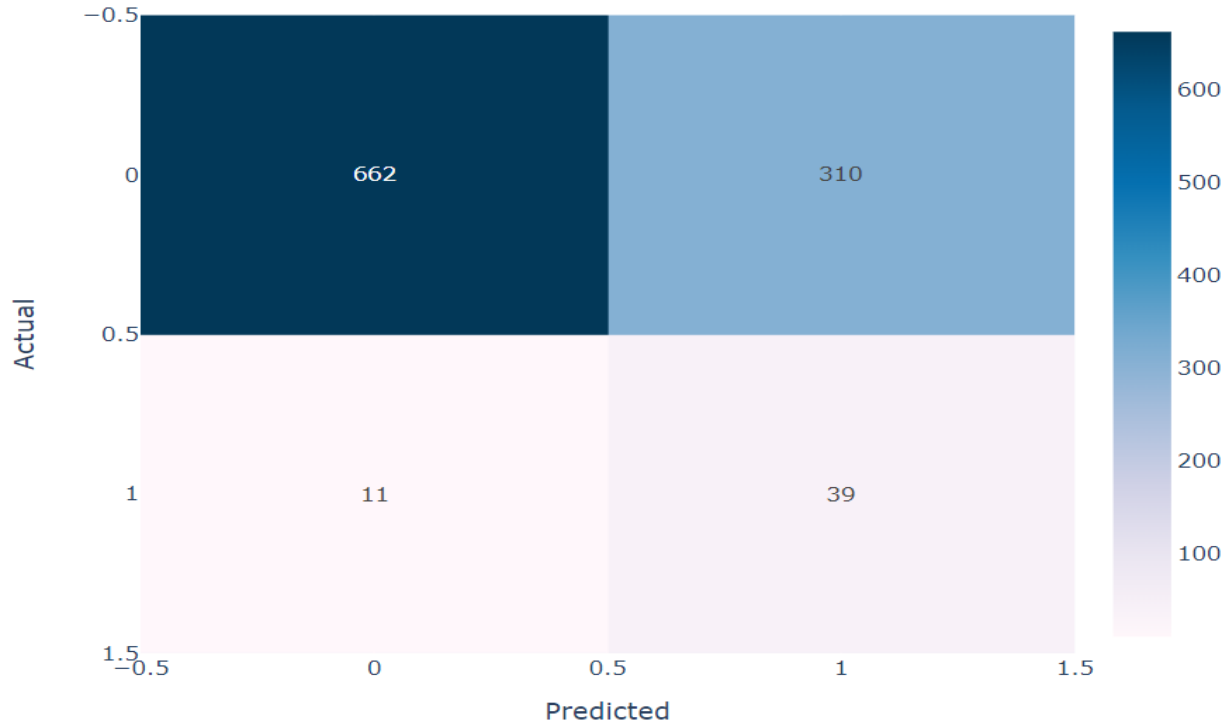

Рисунок 4.2 –Матриця невідповідностей, отримана при тестуванні розробленого модуля

Таблиця 4.1 - Результати розрахованих метрик якості роботи для розробленого модуля і аналога

№ п/п	Метрика	Розроблений модуль (NN)	Аналог (CatBoost)
1	Достовірність (Accuracy)	(662+39)/ (662+39+310+11)= = **68.6** %	(968+1)/ (968+1+4+49)= = **94.8** %
2	Влучність (Precision)	662/(662+11)= = **98.4** %	968/(968+49)= = **95.2** %
3	Чутливість (Sensitivity)	662/(662+310)= = **68.1** %	968/(968+4)= = **99.6** %
4	Специфічність (Specificity)	39/(39+11)=**78** %	1/(49+1)=**2** %

Матриця невідповідностей, отримана при тестуванні аналога на основі методу машинного навчання Cat Boost (Categorical Boosting) [4] показана на рис. 4.3. А в табл. 4.1 представлено також результати розрахованих метрик для аналога (у порівнянні з розробленим модулем).

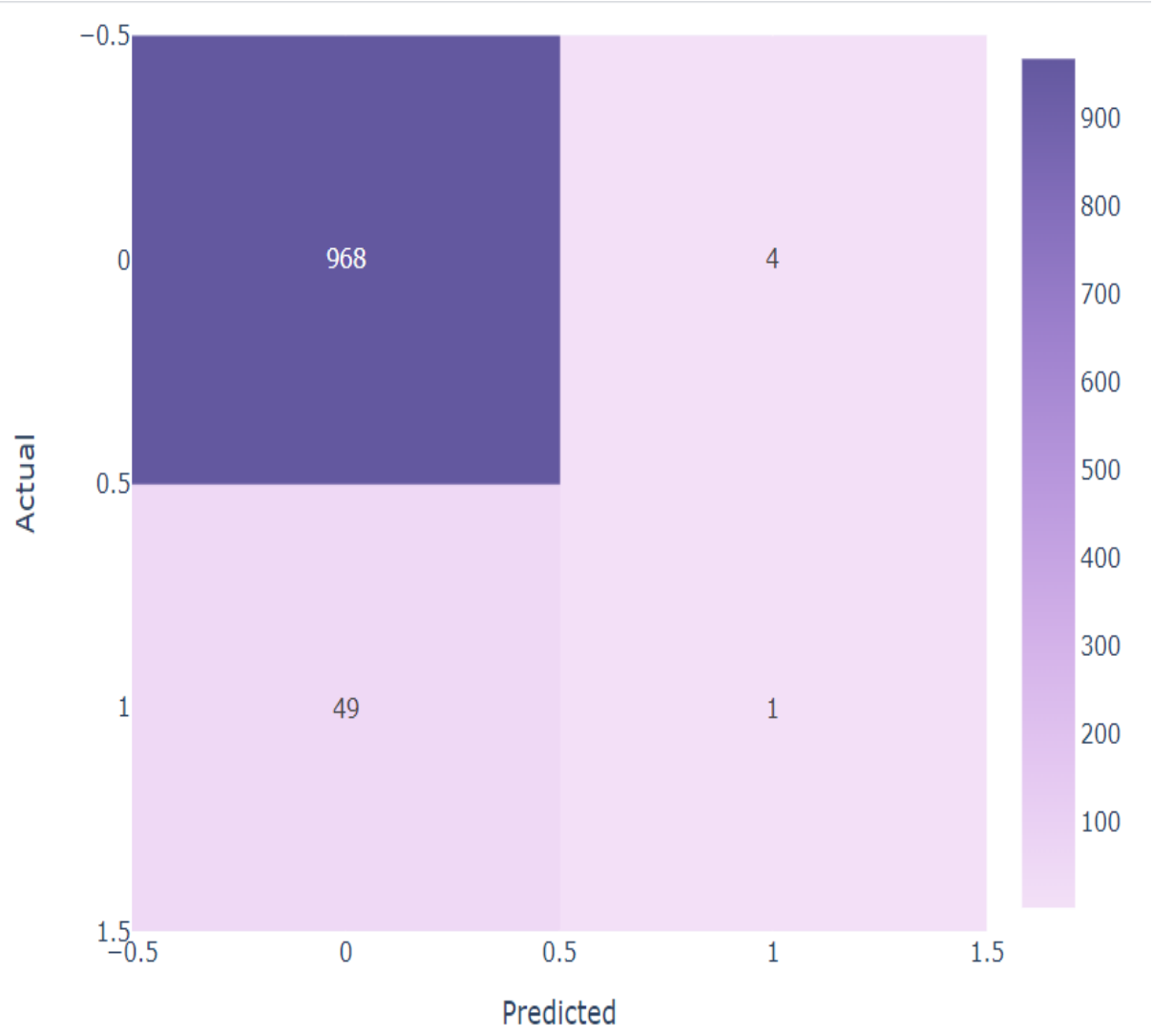

Рисунок 4.3 –Матриця невідповідностей, отримана при тестуванні аналога

Із табл. 4.1 видно, що розроблений модуль на основі нейромережі переважає аналог за такими метриками як влучність (98.4 % проти 95,2%) та специфічність (78 % проти 2%), але поступається аналогу за такими метриками як достовірність (68.6 % проти 94,8%) та чутливість (68.1 % проти 99,6%). Але

оскільки ми визначили, що найважливішою метрикою у нашому випадку є специфічність, то по цьому параметру розроблений модуль значно переважає аналог - 78 % проти 2%.

Отже, ми отримали специфічність 78% прогнозування ризику інсульту для свого модуля, що є досить непоганим показником. Таким чином, можна зробити висновок, що розроблений інтелектуальний модуль прогнозування ризику інсульту на основі нейромережі має порівняно з аналогом збільшену на 76% специфічність прогнозування ризику інсульту. Тобто мета роботи досягнута – специфічність прогнозування ризику інсульту підвищена.

Висновок до розділу 4

У розділі в результаті тестування програми було доведено її повну працездатність, наведено відповідні скріншоти її роботи, описано процес її роботи (що подається на вхід, вихід тощо) та відповідність поставленому завданню. Було проаналізовано матриці невідповідностей отримані при тестуванні аналога на основі методу машинного навчання Cat Boost та розробленого модуля на основі нейронної мережі. Розроблений інтелектуальний модуль має специфічність прогнозування ризику інсульту на тестовій вибірці 78%, а програма-аналог на основі методу CatBoost має специфічність прогнозування ризику інсульту на тестовій вибірці 2%. Було обгрунтовано та проаналізовано метрики якості прогнозування інстульту такі як: достовірність (accuracy), влучність (precision), чутливість (sensitivity) та специфічність (specificity). Таким чином, розроблений інтелектуальний модуль має порівняно з аналогом збільшену на 76% специфічність прогнозування ризику інсульту. Тобто мета роботи досягнута – специфічність прогнозування ризику інсульту підвищена.

ВИСНОВКИ

У роботі було розглянуто задачу прогнозування ризику інсульту на основі глибокої нейронної мережі. В ході аналізу предметної області було описано детальну постановку задачі прогнозування ризику інсульту. Також розглянуто різні методи розв'язання задачі класифікації і оцінено їх застосовність до завдання прогнозування ризику інсульту. Серед таких методів машинного навчання як метод K-найближчих сусідів, логістичної регресії, дерев рішень, машини опорних векторів, ансамблевих методів та нейромережевої класифікації як найбільш перспективний і застосовний до даної задачі було обрано нейромережевий метод. Крім цього, було обгрунтовано вибір аналогу до розроблюваного модуля та на основі його недоліків сформульовано мету роботи – підвищення специфічності прогнозування ризику інсульту.

У другому розділі було обґрунтовано вибір типу нейронної мережі - глибока нейромережа прямого поширення (в яких дані проходять з шару входу до шару виходу без повернення назад) для модуля прогнозування ризику інсульту. Обрано архітектуру глибокої нейромережі, яка має 10 входів, 6 прихованих шарів відповідно по 600, 500, 400, 300, 200, 100 нейронів та вихідний шар із 2 нейронів. Число нейронів у першому прихованому шарі - 600, у другому прихованому шарі - 500, у третьому прихованому шарі - 400, у четвертому прихованому шарі - 300, у п`ятому прихованому шарі - 200, у шостому прихованому шарі - 100.Було обрано функцію активації Leaky-ReLu у прихованих шарах та функцію активації Softmax у вихідному шарі. Розроблено структуру процесів обробки інформації інтелектуального модуля. Також розроблено алгоритм роботи програмного модуля прогнозування ризику інсульту.

У третьому розділі було обґрунтовано вибір мови програмування Python та спеціалізованих бібліотек Keras (відкрита нейромережева бібліотека, написана на мові Python) NumPy (фреймворк з відкритим кодом для мови Python) та Pandas (програмна бібліотека для маніпулювання даними та їх

аналізу) для програмної реалізації інтелектуального модуля прогнозування ризику інсульту. Проведено аналіз набору даних прогнозування ризику інсульту для навчання та тестування модуля. Розроблено процедури попередньої обробки інформації із набору даних. Обрано алгоритм оптимізації "Adam", який є модифікацією методу зворотного розповсюдження помилки ('backprop').

Описано основні етапи програмної реалізації та функціонування інтелектуального модуля прогнозування ризику інсульту, що складається з завантаження бібліотек, завантаження набору даних із параметрами пацієнтів та їх ідентифікатором класу, аналізу цих параметрів, створення, тренування, оцінки точності моделі нейромережі та виведення результатів.

У четвертому розділі в результаті тестування програми було доведено її повну працездатність, наведено відповідні скріншоти її роботи, описано процес її роботи (що подається на вхід, вихід тощо) та відповідність поставленому завданню. Було проаналізовано матриці невідповідностей отримані при тестуванні аналога на основі методу машинного навчання Cat Boost та розробленого модуля на основі нейронної мережі. Розроблений інтелектуальний модуль має специфічність прогнозування ризику інсульту на тестовій вибірці 78%, а програма-аналог на основі методу CatBoost має специфічність прогнозування ризику інсульту на тестовій вибірці 2%. Було обгрунтовано та проаналізовано метрики якості прогнозування інстульту такі як: достовірність (accuracy), влучність (precision), чутливість (sensitivity) та специфічність (specificity). Таким чином, розроблений інтелектуальний модуль має порівняно з аналогом збільшену на 76% специфічність прогнозування ризику інсульту. Тобто мета роботи досягнута – специфічність прогнозування ризику інсульту підвищена.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Stroke Prediction Dataset. 11 clinical features for predicting stroke events [Електронний ресурс] – Режим доступу: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

2 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.

3 Любунь З. М. Основи теорії нейромереж: текст лекцій / З. М. Любунь. – Львів : Видавничий центр ЛНУ ім. Івана Франка, 2006.

4 Stroke Analysis|Tensorflow, CatBoost, Ensembling [Електронний ресурс] – Режим доступу: https://www.kaggle.com/code/renadope/stroke-analysis-tensorflow-catboost-ensembling

5 Багаторівнева архітектура [Електронний ресурс] – режим доступу: https://metanit.com/sharp/mvc5/23.5.php .

6 ГНМ

7 Abdellah Benzaouia, Ahmed El Hajjaji Advanced Takagi–Sugeno Fuzzy Systems: Delay and Saturation (Studies in Systems, Decision and Control, 8) Springer; 2014th edition (July 1, 2014)

8 Tahmasebi, P. A hybrid neural networks-fuzzy logic-genetic algorithm for grade estimation (англ.) // Computers & Geosciences : journal. — 2012. — Vol. 42. — P. 18—27.

9 Warde-Farley, David; Goodfellow, Ian J.; Courville, Aaron; Bengio, Yoshua (2013-12-20). "An empirical analysis of dropout in piecewise linear networks". arXiv:1312.6197

10 PyTorch GET STARTED [Електронний ресурс] – Режим доступу: https://pytorch.org/

11 Welcome to Python [Електронний ресурс] – Режим доступу: https://www.python.org

12 Keras: Simple. Flexible. Powerful. [Електронний ресурс] – Режим доступу: https://keras.io/

13 Numpy [Електронний ресурс] – Режим доступу: https://numpy.org/

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

15 Confusion matrix [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Confusion_matrix

16 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 НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інтелектуальний модуль прогнозування ризику інсульту

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 90,67% Схожість 9,33%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Озеранський В.С.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Скирська В.В.

Керівник роботи _____ Паночишин Ю.М.

Додаток Б (обов'язковий)

Лістинг програми

Фрагменти лістінгу коду програми

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')


import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go
from plotly.offline import plot, iplot, init_notebook_mode
init_notebook_mode(connected=True)

sns.set(rc={'figure.figsize':(16,8)})

RANDOM_STATE = 115


df_stroke_original = pd.read_csv('../input/stroke-prediction-dataset/healthcare-dat
aset-stroke-data.csv')
# df_stroke_original = df_stroke_original.drop(["id"],axis = 1)
df_stroke_original.shape


split = StratifiedShuffleSplit(n_splits = 1, test_size = 0.2, random_state=RANDOM_S
TATE)
for train_index,test_index in split.split(df_stroke_gender,df_stroke_gender[stratif
y_on]):
    train_set = df_stroke_gender.loc[train_index]
    test_set = df_stroke_gender.loc[test_index]

train_set = train_set.reset_index(drop=True)

X_train = train_set.drop(["stroke"],axis=1)
y_train = train_set["stroke"].copy(deep = True)

X_test = test_set.drop(["stroke"],axis=1)
y_test = test_set["stroke"].copy(deep = True)

X_train_prepped = full_pipeline.fit_transform(X_train)
X_test_prepped = full_pipeline.transform(X_test)

n_cols = X_train_prepped.shape[1]
target = to_categorical(y_train)
target_test = to_categorical(y_test)


df_stroke_categorical["Age Bracket"] = pd.cut(x=df_stroke_numeric['age'], bins=[0.0
,11.9,17.9,29.9,39.9,49.9,59.9,69.9,79.9,np.inf],
```

```
                                          labels=["Child","Teenager","18-29.9","Thi
rties","Forties","Fifties","Sixties","Seventies","Eighties +"])


df_stroke_categorical["Glucose Category"] = pd.cut(x=df_stroke_numeric['avg_glucose
_level'], bins=[-np.inf,114,140,np.inf],
                                          labels=["Normal","Prediabetes","Diabetes"
])


df_stroke_categorical["BMI Category"] = pd.cut(x=df_stroke_numeric['bmi'], bins=[-n
p.inf,18.4,24.9,29.9,np.inf],
                                          labels=["Underweight","Healthy Weight","O
verweight","Obese"])


df_stroke_categorical = df_stroke_categorical[list(df_stroke_categorical)].astype("
category")


df_stroke_numeric["hyp_hd"] = df_stroke_numeric["heart_disease"] + df_stroke_numeri
c["hypertension"]


df_stroke_new = pd.concat([df_stroke_numeric,df_stroke_categorical],axis = 1)


class_dist = px.histogram(df_stroke_numeric,y="stroke",color_discrete_sequence=["de
eppink"],opacity = .7)
class_dist.update_layout(title="Target (Stroke) Class Count")
class_dist.update_xaxes(title="Count")
class_dist.update_yaxes(title="Stroke Class")
class_dist.show()


n = 6
target = "stroke"
corr = df_stroke_numeric.corr()

x = corr.nlargest(n,target).index
corr_df =  df_stroke_numeric[list(x)]
corr = corr_df.corr()
corr = np.around(corr,decimals=3)
fig = px.imshow(corr,color_continuous_scale = "pubu",text_auto=True)
fig.update_layout(title="Top "+str(n)+" Features Correlated With "+str(target).capi
talize())
fig.show()


import numpy
from catboost import CatBoostClassifier
model = CatBoostClassifier(cat_features=cat_indicies,verbose=0)
fit_model = model.fit(X_SMOTE_resampled, y_SMOTE_resampled)


def feature_importance_visual(cols,
                              fit_model,
                              x_lab:str,
```

```
                                y_lab:str,
                                model_name:str,
                                title:str,
                                sort_as:bool):
    re:bool = False
    if sort_as == True:
        rev = True

    features = sorted(list(zip((cols),fit_model.get_feature_importance())),key=lamb
da x:x[1],reverse=rev)
#     print(features)
    x = [features[i][0] for i in range(0,len(features))]
    y = [features[i][1] for i in range(0,len(features))]

    fig= px.bar(x=x,y=y,color_discrete_sequence=["dodgerblue"])
    fig.update_xaxes(title=x_lab)
    fig.update_yaxes(title=y_lab)
    fig.update_layout(title = title+" using "+model_name)
    fig.show()


feature_importance_visual(X_SMOTE_resampled,
                                fit_model,
                                x_lab="Features",
                                y_lab="Importance",
                                model_name="CatBoostClassifier",
                                title="Feature Importance",
                                sort_as=True)


democracy = VotingClassifier(best_unfitted_estimators.copy(), voting="soft")
pringles = StackingClassifier(best_unfitted_estimators.copy(),final_estimator = His
tGradientBoostingClassifier())
democracy.fit(X_train_prepped,y_train)
pringles.fit(X_train_prepped,y_train)


weight_multiplier = 0.7
counts = np.bincount(df_stroke_original["stroke"])
weight_for_0 = 1.0/counts[0]
weight_for_1 = 1.0/(counts[1]*weight_multiplier)
class_weight = {0: weight_for_0, 1: weight_for_1}


model = Sequential()
model.add(Dense(600,activation="leaky_relu",input_shape=(n_cols,),
          kernel_initializer = tf.keras.initializers.HeNormal(seed=RANDOM_STATE),
          bias_initializer = tf.keras.initializers.Zeros(),
          kernel_regularizer=tf.keras.regularizers.L2(1e-7)))
model.add(Dense(500,activation="leaky_relu",
          kernel_initializer = tf.keras.initializers.HeNormal(seed=RANDOM_STATE),
          bias_initializer = tf.keras.initializers.Zeros(),
          kernel_regularizer=tf.keras.regularizers.L2(1e-3)))
model.add(Dropout(0.2))
model.add(Dense(400,activation="leaky_relu",
          kernel_initializer = tf.keras.initializers.HeNormal(seed=RANDOM_STATE),
          bias_initializer = tf.keras.initializers.Zeros(),
          kernel_regularizer=tf.keras.regularizers.L2(1e-3)))
```

```
model.add(Dropout(0.2))
model.add(Dense(300,activation="leaky_relu",
          kernel_initializer = tf.keras.initializers.HeNormal(seed=RANDOM_STATE),
          bias_initializer = tf.keras.initializers.Zeros()))
model.add(Dropout(0.3))
model.add(Dense(200,activation="leaky_relu",
          kernel_initializer = tf.keras.initializers.HeNormal(seed=RANDOM_STATE),
          bias_initializer = tf.keras.initializers.Zeros()))
model.add(Dense(100,activation="leaky_relu",
          kernel_initializer = tf.keras.initializers.HeNormal(seed=RANDOM_STATE),
          bias_initializer = tf.keras.initializers.Zeros()))
model.add(Dense(2,activation="softmax"))
model.compile(optimizer="adam",
              loss="categorical_crossentropy",
              metrics=["accuracy",
                       tf.keras.metrics.AUC(name="auc",from_logits=True),
                       tf.keras.metrics.Precision(name="precision"),
                       tf.keras.metrics.Recall(name="recall")])


history = model.fit(X_train_prepped,
                    target,
                    epochs=200,
                    validation_split=0.3,
                    verbose=False,
                   class_weight = class_weight)
y_pred_neural_net_proba = model.predict(X_test_prepped)
y_pred = y_pred_neural_net_proba.argmax(axis=-1)
```

```
s = list(np.array(score_summary(y_test,y_pred)))
s.append("Neural Network (with Weighted Classes)")
k = pd.DataFrame(s).T
k.columns = ensb.columns
ensb = pd.concat([ensb,k],axis=0)
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

«ІНТЕЛЕКТУАЛЬНИЙ МОДУЛЬ ПРОГНОЗУВАННЯ РИЗИКУ ІНСУЛЬТУ»

Виконала: студентка 4 курсу,групи 2КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Скирська В. В.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф.КН
Паночишин Ю.М.
(прізвище та ініціали)

«07» 06 2024 р.

Вінниця ВНТУ - 2024 рік

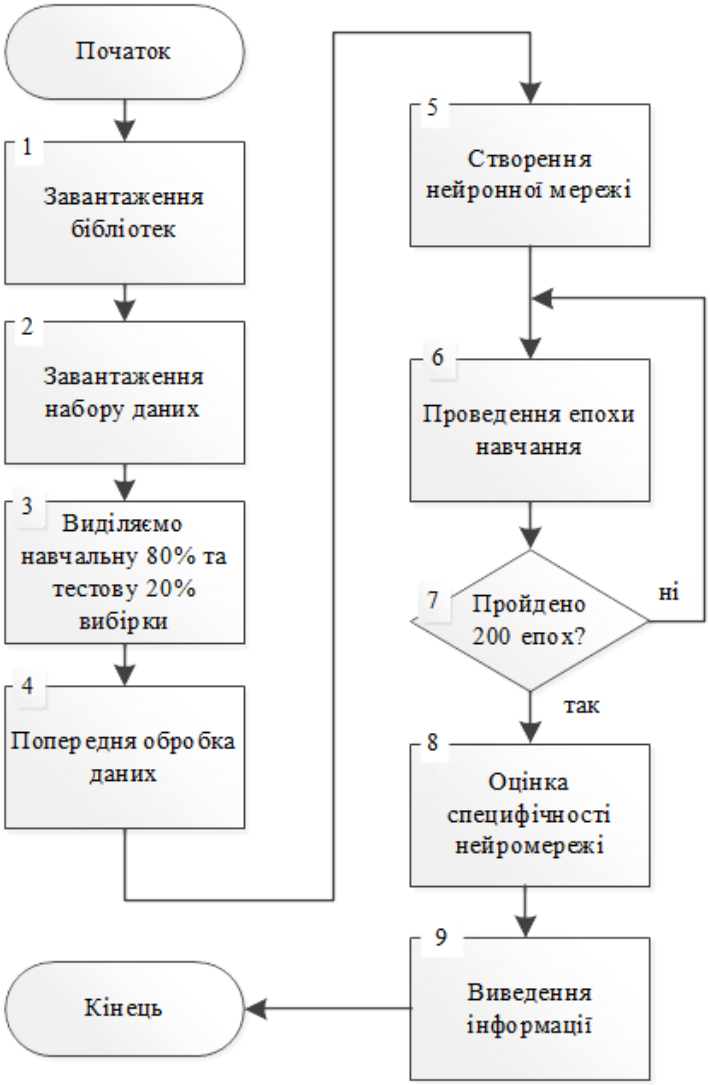

Рисунок В.1 – Схема алгоритму роботи інтелектуального модуля прогнозування ризику інсульту

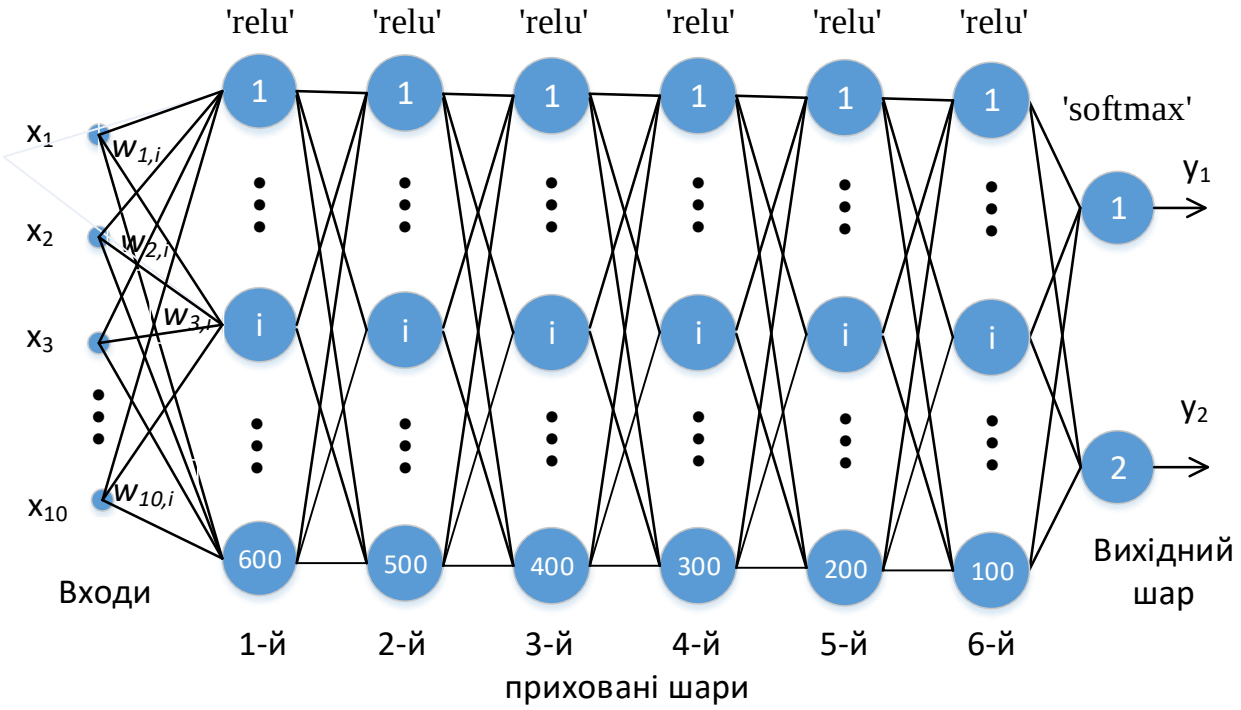

Рисунок В.2 – Структура глибокої нейронної мережі для модуля прогнозування ризику інсульту

1	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
2	9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
3	51676	Female	61	0	0	Yes	Self-employed	Rural	202.21	N/A	never smoked	1
4	31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
5	60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
6	1665	Female	79	1	0	Yes	Self-employed	Rural	174.12	24	never smoked	1
7	56669	Male	81	0	0	Yes	Private	Urban	186.21	29	formerly smoked	1
8	53882	Male	74	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
9	10434	Female	69	0	0	No	Private	Urban	94.39	22.8	never smoked	1
10	27419	Female	59	0	0	Yes	Private	Rural	76.15	N/A	Unknown	1

Рисунок В.3 – Фрагмент набору даних для визначення інсульту

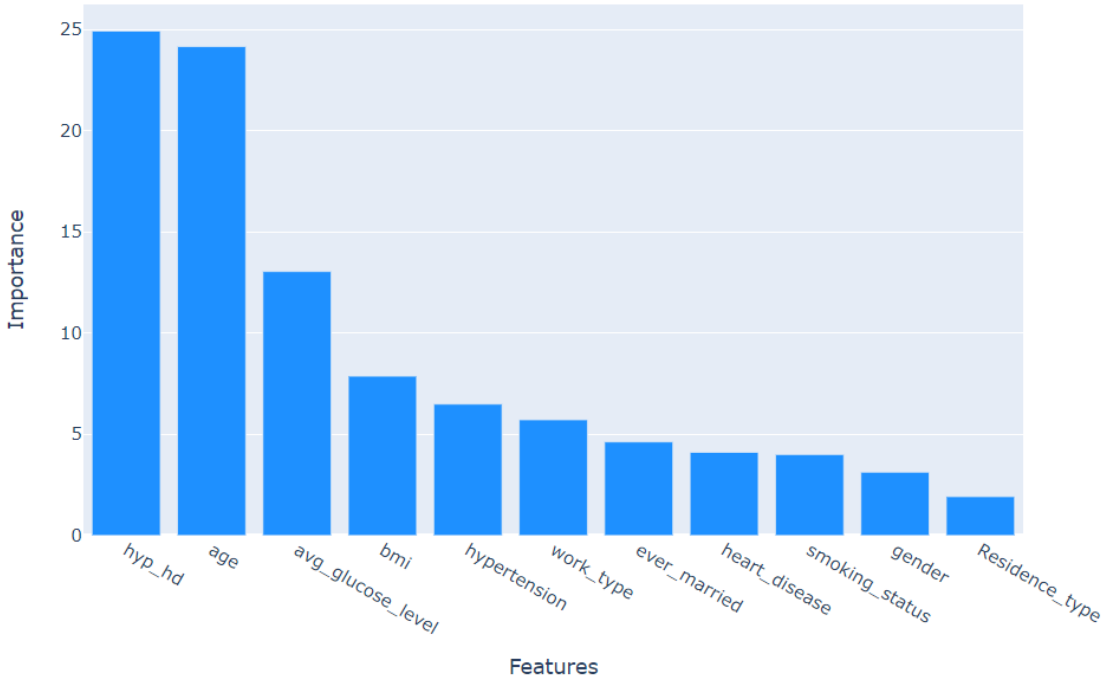

Рисунок В.4 – Повний рейтинг важливості параметрів

```
Epoch 1/200
341/341 [==============================] - 1s 1ms/step - loss: 0.3076 - get_f1: 0.0
111
Epoch 2/200
341/341 [==============================] - 0s 1ms/step - loss: 0.1653 - get_f1: 0.0
000e+00
Epoch 3/200
341/341 [==============================] - 0s 1ms/step - loss: 0.1476 - get_f1: 0.0
000e+00
...

Epoch 198/200
341/341 [==============================] - 0s 1ms/step - loss: 0.1250 - get_f1: 0.0
792
Epoch 199/200
341/341 [==============================] - 0s 1ms/step - loss: 0.1259 - get_f1: 0.0
567
Epoch 200/200
341/341 [==============================] - 0s 1ms/step - loss: 0.1320 - get_f1: 0.1
169

Accuracy: 68.6%, Precision: 98.4%, Sensitivity: 68.1 %, Specificity: 78 %
```

Рисунок В.5 – Результат роботи програми

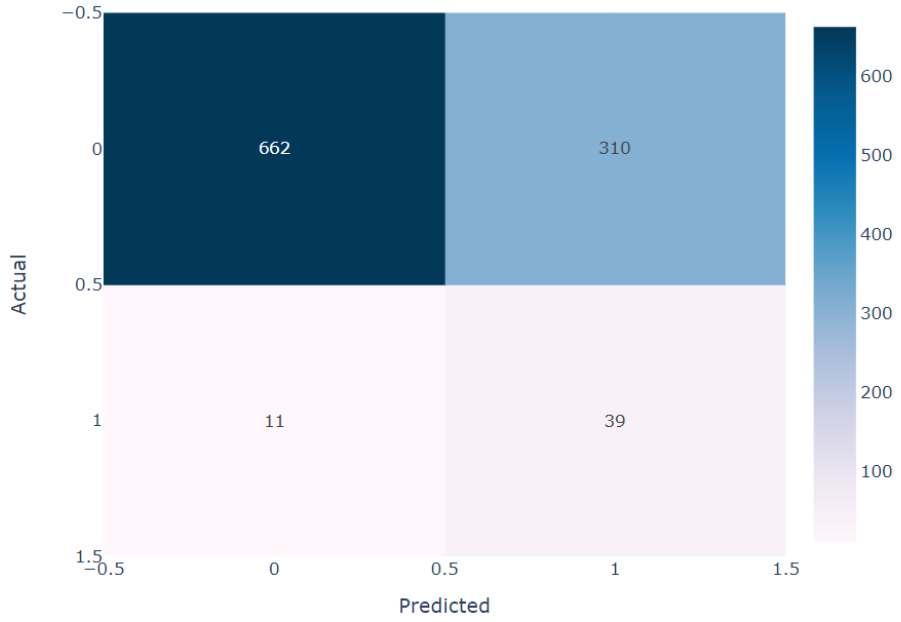

Рисунок В.6 –Матриця невідповідностей, отримана при тестуванні розробленого модуля

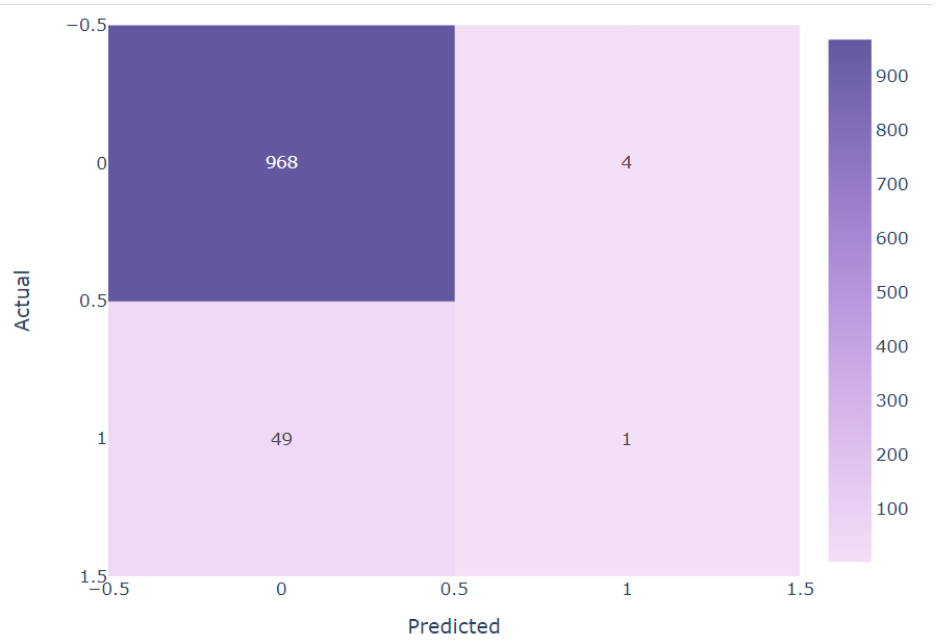

Рисунок В.7 –Матриця невідповідностей, отримана при тестуванні аналога

Таблиця В.1 - Результати розрахованих метрик якості роботи для розробленого модуля і аналога

№ п/п	Метрика	Розроблений модуль (NN)	Аналог (CatBoost)
1	Достовірність (Accuracy)	(662+39)/ (662+39+310+11)= = **68.6** %	(968+1)/ (968+1+4+49)= = **94.8** %
2	Влучність (Precision)	662/(662+11)= = **98.4** %	968/(968+49)= = **95.2** %
3	Чутливість (Sensitivity)	662/(662+310)= = **68.1** %	968/(968+4)= = **99.6** %
4	Специфічність (Specificity)	39/(39+11)=**78** %	1/(49+1)=**2** %