

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
ІНТЕЛЕКТУАЛЬНИХ МЕТОДІВ ДЛЯ МЕДИЧНОГО ДІАГНОСТУВАННЯ
RESEARCH AND SOFTWARE IMPLEMENTATION OF INTELLIGENT
METHODS FOR MEDICAL DIAGNOSIS

Виконав: студент(ка) 2 курсу, групи КНТ-119м

Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Саман М. В

(прізвище та ініціали)

Керівник Колпакова Т. О.

(прізвище та ініціали)

Рецензент Казурова А. Є.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут , факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С. О. Субботін
 “ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Саман Маргарита Володимирівни
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація інтелектуальних методів для медичного діагностування

Research and software implementation of intelligent methods for medical diagnosis

керівник проєкту (роботи) Колпакова Тетяна Олексіївна к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “03” листопада 2020 року
 № 301

2. Строк подання студентом проєкту (роботи) 11 грудня 2020 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити 1. Аналіз предметної області 2. Вибір інструментів розробки програмного забезпечення 3. Основні рішення щодо реалізації компонентів системи 4. Керівництво програміста 5 Тестування та експериментальне дослідження програми 6 Економічне обґрунтування роботи 7. Охорона праці та безпека у надзвичайних ситуаціях

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділи	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Колпакова Т. О., доцент		
6 Економіко-організаційна частина	Пожуєва Т. О., професор		
7 Охорона праці та безпека в надзвичайних ситуаціях	Коробко О.В., ст.викладач		
Нормоконтроль	Дейнега Л. Ю., ст. викладач		

7. Дата видачі завдання “02” вересня 2020 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-4 тижні	Розділ 1
3	Вибір засобів розробки програмного продукту	5 тиждень	Розділ 2
	Розробка архітектури програми	6 тиждень	Розділ 3
4	Розробка програмного продукту	7-9 тижні	Розділ 4-5
5	Економічне обґрунтування роботи	10 тиждень	Розділ 6
6	Написання розділу охорона праці та безпека у надзвичайних ситуаціях	11 тиждень	Розділ 7
7	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	12 тиждень	Додатки
8	Захист роботи	13 тиждень	

Студент(ка)

Керівник проєкту (роботи)

(підпис)

(підпис)

Саман М. В.

(прізвище та ініціали)

Колпакова Т. О.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
161 с., 33 таблиці, 63 рис., 3 додатки, 60 джерел.

ГЕНЕТИЧНИЙ АЛГОРИТМ, КЛАСИФІКАЦІЯ, НЕЙРОННА
МЕРЕЖА, ПРОГНОЗУВАННЯ, ДІАБЕТ, KERAS, PYTHON, PSO

Об'єкт дослідження – процес прогнозування та виявлення цукрового діабету за медичними показниками стану здоров'я людини.

Предмет дослідження – методи для прогнозування та класифікації захворюваності.

Мета роботи – дослідження, модифікація та програмна реалізація методів прогнозування захворюваності на цукровий діабет.

Метод дослідження – традиційні моделі, нейронні мережі, комбіновані методи (нейронні мережі і генетичні алгоритми, нейронні мережі і мультиагентні системи).

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, мова програмування Python, середовище програмування Sublime Text 3, персональний комп'ютер з процесором Intel Core 3 під управлінням операційної системи Microsoft Windows 10.

Результати. Створено програмний продукт для прогнозування ризику виникнення цукрового діабету у пацієнта, що визначає фактор ризику виникнення захворювання.

Висновки. Розроблено програмний продукт, який дозволяє визначати фактор ризику виникнення цукрового діабету на основі клінічних даних про здоров'я людини.

Галузь використання – визначення фактору ризику виникнення цукрового діабету у медичних закладах лікарями-ендокринологами.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 161 pages, 63 figures, 33 tables, 3 appendixes, 60 sources.

CLASSIFICATION, DIABETES, FORECASTING, GENETIC ALGORITHM, KERAS, NEURAL NETWORK, PSO, PYTHON.

The object of the research is the process of predicting and detecting diabetes by medical indicators of human health.

The subject of the research are methods of predicting classifying morbidity.

The purpose of the research are research, modification and software implementation of methods for predicting the incidence of diabetes.

The method of research are traditional models, neural networks, combined methods (neural networks and genetic algorithms, neural networks and multi-agents systems).

Materials, methods and technical means – structural and object-oriented programming, Python programming language, Sublime Text 3 programming environment, personal computer with Intel 3 Core processor running Microsoft Windows 10 operating system.

Results. A software product has been created to predict the risk of developing diabetes in a patient, which determines the risk factor for the disease.

Conclusions. A software product has been developed to determine the risk factor for diabetes based on clinical data on human health.

The field of use is determining the risk factors for diabetes in medical institution by endocrinologist.

ЗМІСТ

Перелік скорочень та умовних позначень	10
Вступ	11
1 Аналіз предметної області.....	12
1.1 Аналіз понять медичного діагностування та цукрового діабету	
12	
1.1.1 Медичне діагностування.....	12
1.1.2 Медичне діагностування на основі даних.....	12
1.1.3 Поняття діабету.....	13
1.2 Аналіз галузі штучного інтелекту для прогнозування	
захворюваності	14
1.2.1 Байєсова мережа	15
1.2.2 Алгоритм NNge (Non-nested generalized exemplars).....	16
1.2.3 Random Forest Classifier	17
1.2.4 HMM (Прихована марковська модель)	19
1.2.5 RBFNetwork (radial basis function networks).....	21
1.2.6 AdaBoost	22
1.2.7 ENORA.....	23
1.2.8 Порівняння класичних алгоритмів, що використовуються	
для задачі прогнозування	24
1.2.9 MultilayerPecptron.....	26
1.2.10 Згорткова нейронна мережа (CNN)	27
1.2.11 LSTM.....	28
1.2.12 Dense neural network	30

1.2.13	Порівняння розглянутих нейронних мереж.....	31
1.3	Аналіз методів для медичного діагностування	32
1.4	Постановка завдання	41
1.5	Висновки за розділом 1	42
2	Вибір інструментів розробки програмного забезпечення	44
2.1	Вибір мови програмування	44
2.1.1	Мова програмування C++	44
2.1.2	Мова програмування R.....	45
2.1.3	Мова програмування Python.....	45
2.1.4	Мова програмування Matlab.....	46
2.2	Порівняння мов програмування	47
2.3	Огляд бібліотек машинного навчання.....	48
2.3.1	Бібліотека для створення нейронних мереж Keras	49
2.3.2	Бібліотека Scikit-learn.....	50
2.3.3	Бібліотека Matplotlib.....	51
2.3.4	Бібліотека для роботи з масивами NumPy	51
2.4	Вибір середовища програмування	52
2.4.1	Середовище розробки Visual Studio	52
2.4.2	Середовище розробки IntelliJ IDEA.....	53
2.4.3	Середовище розробки Sublime Text.....	53
2.4.4	Порівняння можливостей середовищ розробки	54
2.5	Висновки за розділом	54
3	Основні рішення щодо реалізації компонентів системи.....	56
3.1	Оптимізація нейронних мереж за допомогою генетичного алгоритму	56

3.2	Основні розроблені алгоритми.....	59
3.3	Використовувані моделі даних.....	63
3.4	Опис концепції програмної моделі в проєктованій системі	64
3.5	Висновки за розділом 3	70
4	Керівництво програміста.....	71
4.1	Призначення та умови застосування програми.....	71
4.1.1	Призначення програми.....	71
4.1.2	Умови застосування програми	71
4.2	Звернення до програми	72
4.3	Вхідні та вихідні дані	73
4.4	Повідомлення програмісту	75
4.5	Висновки за розділом	76
5	Керівництво користувача	77
5.1	Призначення та умови застосування програми.....	77
5.1.1	Призначення програми.....	77
5.1.2	Умови застосування програми	77
5.2	Виконання програми	78
5.2.1	Екран входу	78
5.2.2	Екран роботи алгоритму	78
5.3	Повідомлення оператору.....	81
5.4	Висновки за розділом	81
6	Експерименти та результати	82
6.1	Проведення експериментів	82
6.2	Висновки за розділом	101
7	Економічне обґрунтування роботи	102

7.1	Опис ідеї.....	102
7.2	Попередня характеристика потенційного ринку.....	103
7.3	Розроблення ринкової стратегії проєкту	106
7.4	Калькуляція витрат на реалізацію проєкту	107
7.4.1	Визначення складу, чисельності та фонду заробітної плати працівників	108
7.4.2	Визначення витрат на матеріали та вартості спожитих послуг	112
7.4.3	Розрахунок суми амортизації	116
7.4.4	Калькуляція кошторису витрат	117
7.5	Висновки за розділом	118
8	Охорона праці та безпека у надзвичайних ситуаціях.....	119
8.1	Аналіз потенційних небезпек	119
8.2	Заходи із забезпечення безпеки.....	121
8.3	Заходи із забезпечення виробничої санітарії та гігієни праці	123
8.4	Заходи безпеки у надзвичайних ситуаціях.....	128
8.4.1	Заходи з пожежної безпеки.....	128
8.4.2	Заходи з цивільного захисту. Єдина державна система цивільного захисту, її складові та режими функціонування	130
	Висновки	134
	Перелік джерел посилання.....	136
	Додаток А Технічне завдання.....	143
	Додаток Б Код програми	147
	Додаток В Слайди презентації.....	154

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

БШП – багатошаровий персептрон

ЗНМ – згорткова нейронна мережа

ПЗ – програмне забезпечення

РНМ – рекурентна нейронна мережа

ЦД – цукровий діабет

ШІ – штучний інтелект

ШНМ – штучна нейронна мережа

AROC – the area under the receiver operating characteristic curve

ATAR – abnormalities and trend analysis rules

CART – classification and regression tree

CNN – convolutional neural networks

CTA – correlation-based trend analysis

DCNN – deep convolutional neural network

DMIS – diabetes mellitus information system

DMPR – diabetes mellitus prediction rules

DMTL – deep multi-task learning

GBM – gradient boosting machine

HRBR – hybrid rule-based reasoning

IDPA – improved diabetes prediction algorithm

LSTM – long short-term memory

MLP – multilayer perceptron

NNge – non-nested generalized exemplars

RF – random forest

RBM – restricted Boltzmann machine

RSR – rough set reasoning

SKCNN – sigmoid kernel convolutional neural network

STL – single task learning

SVM – support vector machine

ВСТУП

Цукровий діабет – це глобальна медико-соціальна проблема сучасності. Рівень розповсюдження цукрового діабету зростає у всьому світі. Відповідно даних Всесвітньої організації охорони здоров'я захворювання її розповсюдженість стрімко зростає, наближаючись до глобальної епідемії [1].

Цукровий діабет має такі наслідки: рання інвалідність, високий рівень летальності, заходи, що направлені на реабілітацію хворих, призводять до великих затрат служб охорони здоров'я. Це спричинено тим, що хвороба і її лікування визначаються не з моменту порушення вуглеводного обміну, що визначається лише за допомогою проведення різних тестів навантаження (глюкозо толерантний тест і інші), а при прояві клінічних ознак захворювання, що в свою чергу, є причиною різних судинних та інших змін в тканинах.

Тож існує актуальність прогнозування захворюваності на цукровий діабет з використанням даних про стан здоров'я людини для планування профілактичних заходів та відповідних фінансових витрат. Таким чином, для проведення прогнозування необхідно розробити належне програмне забезпечення, яке могло б приймати файл з даними про стан здоров'я та видавати певне значення прогнозування хвороби.

Метою роботи є аналіз методів та алгоритмів класифікації даних, дослідження сучасної літератури щодо покращення існуючих засобів класифікації та розробка власного програмного засобу, яке буде придатним для виконання поставленої задачі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз понять медичного діагностування та цукрового діабету

1.1.1 Медичне діагностування

Медичне діагностування – це розділ клінічної медицини, що вивчає склад та методи процесу розпізнавання захворювань з метою встановлення правильного діагнозу [1].

Медичне діагностування включає в себе [1]:

- семіотику – вчення про симптоми захворювань та їх діагностичне значення;
- методи обстеження пацієнта;
- методологічні основи встановлення діагнозу.

1.1.2 Медичне діагностування на основі даних

Медичне діагностування на основі даних про стан здоров'я пацієнта виконується з використанням відповідного програмного забезпечення.

Сьогодні лікарі користуються спеціальними системами, що допомагають не лише автоматизувати їх роботу, а й такими, що здатні встановлювати попередній діагноз. До таких систем відносять і медичні інформаційні системи.

Загальними завданнями медичних інформаційних систем в клінічних установах є [2]:

- об'єктивізація трактування результатів досліджень (за деякими даними, помилково позитивні або помилково негативні, клінічно недостатні або неправильно витлумачені результати рентгенологічного, електрокардіологічного та лабораторних досліджень призводять в 31% випадків до помилкового діагнозу);

- автоматизація обробки інформації на етапі попередньої роботи медичного персоналу з визначенням діагнозу і вироблення тактики лікування (лікар приймає остаточне рішення з питань діагностики та лікування хворого);
- автоматизація лабораторних досліджень;
- створення баз (банків) даних: накопичення відомостей про кожного хворого для подальшого аналізу матеріалу, організація обробки цієї інформації відповідним математичним забезпеченням;
- створення баз знань: накопичення знань експертів в галузі медицини та охорони здоров'я, необхідних для розробки експертних систем діагностики, лікування та реабілітації, профоглядів, експертизи планування та управління;
- впорядкування потоку інформації всередині установи.

Практично всі перераховані задачі можуть бути ефективно вирішені із застосування експертних систем – одного прогресивних напрямків сучасних комп'ютерних технологій, що суттєво базується на ідеях створення штучного інтелекту.

1.1.3 Поняття діабету

Діабет – це хронічна хвороба, яка розвивається в тих випадках, коли підшлункова залоза не виробляє достатню кількість інсуліну або коли організм не може ефективно використовувати інсулін, що виробляється. Інсулін – це гормон, що регулює рівень вмісту цукру в крові. Загальним результатом неконтрольованого діабету є гіперглікемія, або завищений рівень вмісту цукру в крові, що з часом призводить до серйозних ушкоджень багатьох систем організму, особливо нервової системи та кровоносних судин [3].

Цукровий діабет (ЦД) входить в число найпоширеніших у світі хронічних захворювань. В останній час ця хвороба почала вивчатися як

соціальна проблема, що стає все більш актуальною. Це пов'язано зі збільшенням кількості людей, що страждають на ЦД, хронічним характером протікання хвороби, розвитком різного роду ускладнень, які призводять до зниження якості життя і скороченню її тривалості [4].

Згідно з даними Міжнародної федерації діабету (IDF), на даний час у світі зареєстровано 415 млн людей, які хворіють на ЦД. До 2040 року прогнозується зростання числа хворих на діабет до 642 млн [4].

У зв'язку з тим, що дана проблема тягне за собою великі економічні затрати та соціальні збитки, існує необхідність прогнозування захворюваності населення на діабет.

1.2 Аналіз галузі штучного інтелекту для прогнозування захворюваності

Розглянута проблема має під собою не тільки медичний зміст, а й економічний й соціальний. Нині для прогнозування різних захворювань використовують різні методи та моделі, серед яких методи аналізу часові ряди, регресійні та авторегресійні моделі, методи, що базуються на машинному навчанні – байєсові мережі та штучні нейронні мережі, техніки, що базуються на вирішенні задач фільтрації.

Для порівняння точності прогнозування захворюваності на ЦД було проаналізовано декілька різних методів, опис та порівняння яких наведено далі.

В якості даних використано інформацію щодо захворювання на діабет представників корінних народів Північної Америки. Джерелом інформації про захворюваність є Національний інститут діабету, захворювань органів травлення та нирок (США) [5]. Набір даних формувався від жінок 21 року і більше народності Піма. Набір складається з 9 атрибутів та 768 екземплярів:

- а) кількість вагітностей;

- б) концентрація глюкози в плазмі через 2 години після введення в пероральному глюкозотолерантному тесті;
- в) діастолічний артеріальний тиск (мм рт.с.);
- г) товщина шкірної складки в районі трицепса (мм);
- г) концентрація інсуліну в сироватці крові (мкЕд/мл);
- д) індекс маси тіла (вага в кг/(зріст в м)²);
- е) функція, що описує генетичну схильність до діабету (diabetes pedigree);
- є) вік (роки);
- ж) клас, що показує чи був у пацієнта ЦД.

1.2.1 Байєсова мережа

Байєсова мережа – це ймовірнісна графова модель (різновид статистичної моделі), яка представляє набір випадкових змінних та їхніх умовних залежностей за допомогою орієнтованого ациклічного графу. Наприклад, байєсова модель може представляти ймовірнісні зв'язки між захворюваннями та симптомами.

Існують ефективні алгоритми, що виконують висновки та навчання в байєсових мережах. Байєсові мережі, що моделюють послідовності змінних називають динамічними байєсовими мережами. Узагальнення байєсових мереж, що можуть представляти та розв'язувати задачі ухвалення рішень за умов невизначеності, називають діаграмами впливу.

Дану мережу можливо навчати за допомогою алгоритмів машинного навчання для отримання більш точних значень прогнозованого параметра.

Перевагою такої Байєсової мережі є те, що її навчання є набагато простішим, ніж метод зворотного поширення помилки. Проте істотним недоліком мережі є її розмір, оскільки вона фактично вміщує в собі всі навчальні дані, потребує багато пам'яті і може повільно працювати. Архітектура мережі наведена на рис. 1.1 [6].

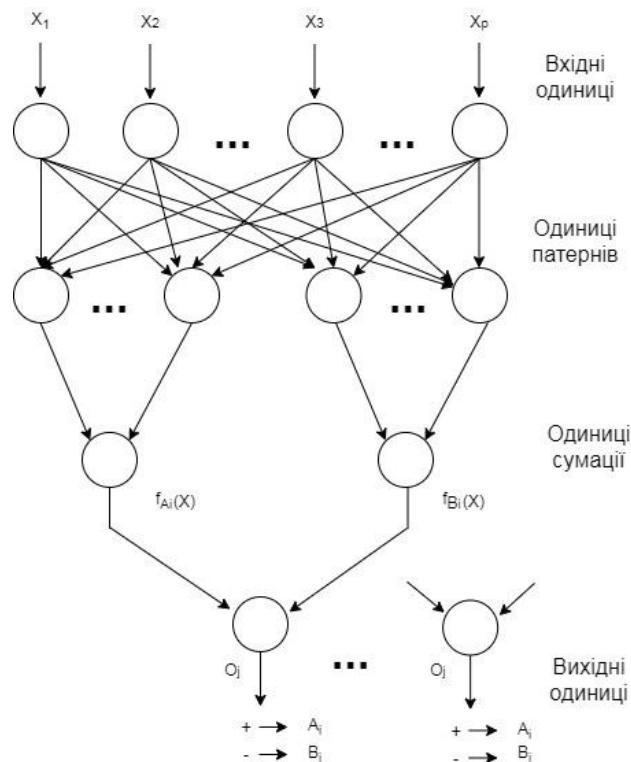


Рисунок 1.1 – Архітектура Байєсової нейронної мережі [6]

1.2.2 Алгоритм NNge (Non-nested generalized exemplars)

NNge – метод найближчого сусіда для генерації правил при використанні неукладених узагальнених зразків. Він виконує узагальнення шляхом злиття зразків, утворюючи гіперпрямокутники в просторі атрибутів, що представляють зв'язні правила з внутрішньою диз'юнкцією.

Алгоритм формує узагальнення кожен раз, коли новий екземпляр додається в базу даних, приєднуючи його до найближчого сусіда того ж класу [7].

NNge навчається поступово спочатку класифікуючи, а потім узагальнюючи кожен новий екземпляр. При класифікації, наприклад, може бути виявлений один або кілька екземплярів, що належать до неправильного класу. В такому випадку алгоритм відкидає цей екземпляр, щоб він перестав бути його членом. Після класифікації новий екземпляр екстраполюється

шляхом злиття його з найближчим зразком подібного класу. На рис. 1.2 приведено блок-схему алгоритму NNge [5].

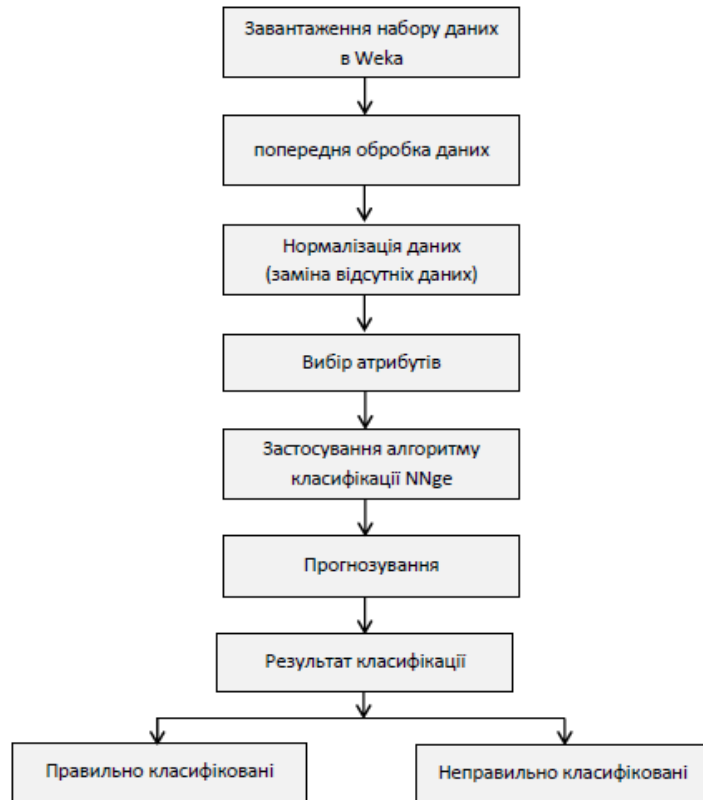


Рисунок 1.2 – Блок-схема роботи алгоритму NNge [5]

1.2.3 Random Forest Classifier

Random Forest – це комбінація дерев передбачень таких, що кожне дерево залежить від значень випадкового вектора незалежно відібраних значень та з однаковим розподілом всіх дерев в лісі.

Похибка узагальнення для лісів прямує до мінімуму, коли кількість дерев стає великою.

Random Forest – це особливий вид методів навчання ансамблю для класифікації, який будує безліч дерев рішень під час навчання та виводить класи за окремими деревами. RF створює ансамбль дерев CART за допомогою механізму пакетування. За допомогою пакетування кожен вузол

дерева вибирає невеликий набір ознак для поділу, що дозволяє алгоритму створювати класифікатори для великих наборів даних дуже швидко [8]. Коли ознаки вибрані, дерева будуються на основі кращого поділу. Кожне дерево дає висновок, який розглядається як «голос» від цього дерева. Вихід, який отримує найбільшу кількість «голосів» обирається лісом в якості кінцевого результату. Приклад роботи алгоритму зображено на рис. 1.3 [9].

До переваг даного методу відносять:

- здатність ефективно обробляти дані з великою кількістю ознак і класів;
- нечутливі до масштабування значень ознак;
- однаково добре оброблюються як неперервні, так і дискретні ознаки;
- існують методи оцінки значимості окремих ознак в моделі;
- здатність працювати паралельно в багатьох потоках.

Проте алгоритм схильний до перенавчання та має великий розмір отримуваних моделей, що потребує великої кількості пам'яті для зберігання моделей.

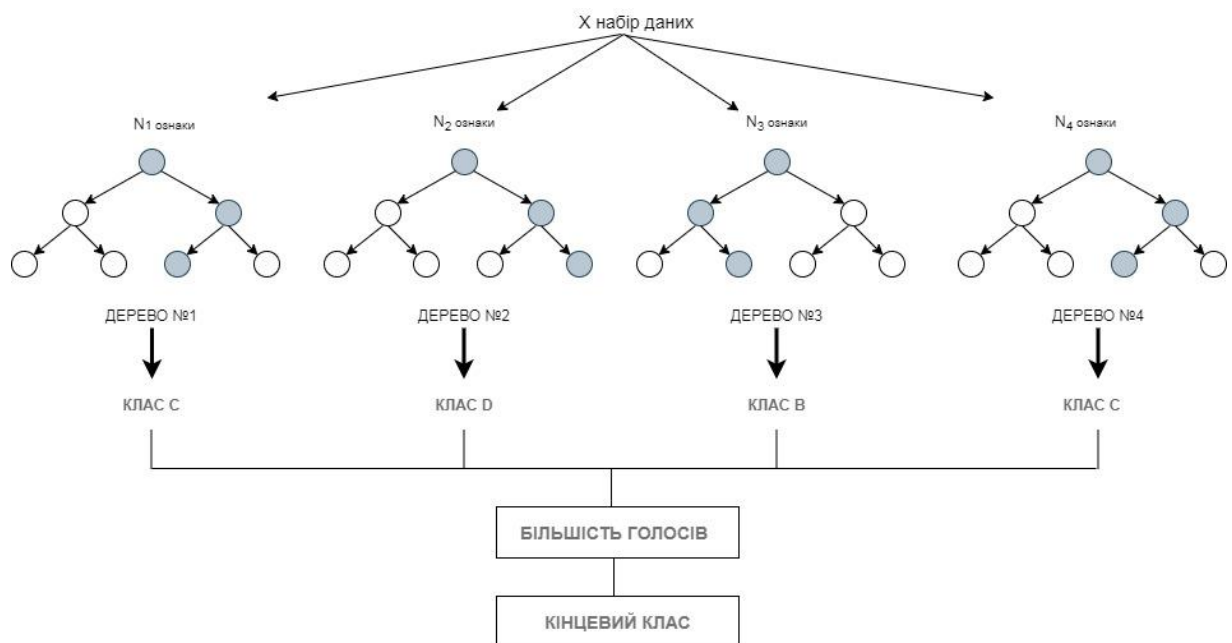


Рисунок 1.3 – Схема роботи алгоритму Random Forest [9]

1.2.4 НММ (Прихована марковська модель)

Прихована марковська модель – це статистична модель, у якій система, що моделюється, розглядається як марковський процес із неспостережуваними (прихованими) станами [10].

У простіших марковських моделях (таких як ланцюги Маркова) стан є безпосередньо видимим спостерігачеві, і тому ймовірності переходів станів є єдиними параметрами. У НММ стан не є видимим безпосередньо, але вихід, залежний від стану є видимим. Кожен стан має ймовірнісний розподіл усіх можливих вихідних значень. Отже, послідовність символів, згенерована НММ, дає якусь інформацію про послідовність станів [10].

Для визначення НММ необхідно визначити наступні елементи.

1. N – число станів у моделі. Незважаючи на те, що стани приховані від спостерігача, часто їм приписують деякий фізичний зміст. Для позначення множини станів моделі використовують запис $S = \{S_1, S_2, \dots, S_N\}$, а стани моделі в момент t позначають q_t .

2. M – число різноманітних символів спостережень, які можуть породжуватися моделлю. Символи спостережень відповідають виходу системи, що моделюється. Множина таких символів позначається як $V = \{v_1, v_2, \dots, v_M\}$.

3. $A = \{a_{ij}\}$ – розподіл імовірностей переходів між станами (або матриця переводних імовірностей), де

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i], 1 \leq i, j \leq N. \quad (1.1)$$

В окремому випадку, коли кожний стан може бути досягнутий з будь-якого іншого за один крок, маємо $a_{ij} > 0$ при всіх значеннях i та j . Для інших типів НММ будемо мати $a_{ij} = 0$ для однієї чи більше пар (i, j) .

4. Розподіл імовірностей виникнення символів спостережень в стані j , $B=\{b_j(k)\}$, де

$$b_j(k) = P[v_k \cdot at \cdot t | q_t = S_j], 1 \leq i \leq N, 1 \leq k \leq M. \quad (1.2)$$

5. Початковий розподіл імовірностей станів $\pi = \{\pi_i\}$

$$\pi_i = P[q_1 = S_i], 1 \leq i \leq N. \quad (1.3)$$

Використовуючи відповідні значення для N , M , A , B та π , НММ зможе генерувати деякі послідовності спостережень $O = O_1 O_2 \dots O_T$, де O_T – є символом алфавіту V , а T – це кількість символів в даній послідовності [11].

Звідси випливає, що НММ припускає завдання двох параметрів моделі (N та M), множини символів спостережень, а також трьох ймовірнісних мір A , B та π . Наочне представлення НММ зображено на рис. 1.4.

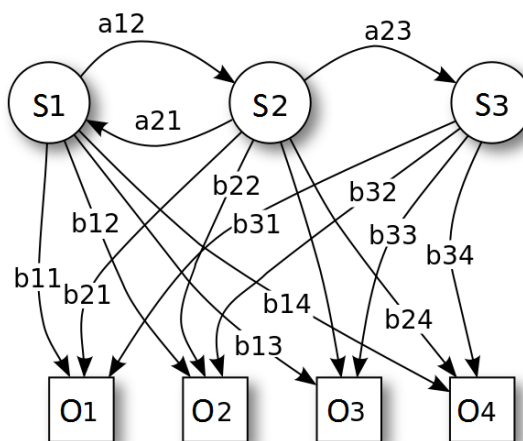


Рисунок 1.4 – Наочне відображення НММ [10]

Серед переваг таких моделей виділяють простоту математичної структури. Структура дозволяє моделювати складний ланцюжок

спостережень. Проте мітки розглядаються як одне ціле, тож неможливо виділити окремі ознаки.

1.2.5 RBFNetwork (radial basis function networks)

Радіальні функції – це просто клас функцій. Взагалі її можна використовувати в будь-яких моделях (лінійних чи нелінійних) і будь-яких мережах (одно- чи багатошарових) [12]. Мережі радіально базисних функцій зазвичай мають три шари: вхідний шар, прихований шар з нелінійною функцією активації та лінійний рівень. Вхід можна моделювати як вектор дійсних чисел $x \in R^n$. Вихід мережі, тоді є скалярною функцією вхідного вектора, $\varphi: R^n \rightarrow R$ і має вигляд

$$\varphi(x) = \sum_{i=1}^N a_i p(\|x - c_i\|), \quad (1.4)$$

де N – кількість нейронів в прихованому шарі;

c_i – центральний вектор для нейрона i ;

a_i – це вага нейрона i в лінійному виході нейронів.

Функції, які залежать лише від відстані від центру вектора, є радіально симетричними щодо цього вектора, і мають назву радіально базисних функцій.

Тим не менш, RBF networks традиційно асоціюють з радіальними функціями в одношарових мережах такими як показано на рисунку 1.5 [13].

RBF network є нелінійною, якщо базисні функції можуть переміщуватися або змінюватися, або, якщо існує більш, ніж один прихований шар [12].

Прихована частина перед вихідним шаром функціонує як стандартна MPL мережа, з сумою зважених прихованих функцій активації, що дає вихідні активації. Прихована функція активації дає базисні функції

$\phi_j(x, \mu_j, \sigma_j)$, що залежить від «ваг» $\{\mu_{ij}, \sigma_j\}$ і вхідних активацій $\{x_i\}$ не стандартним чином [13].

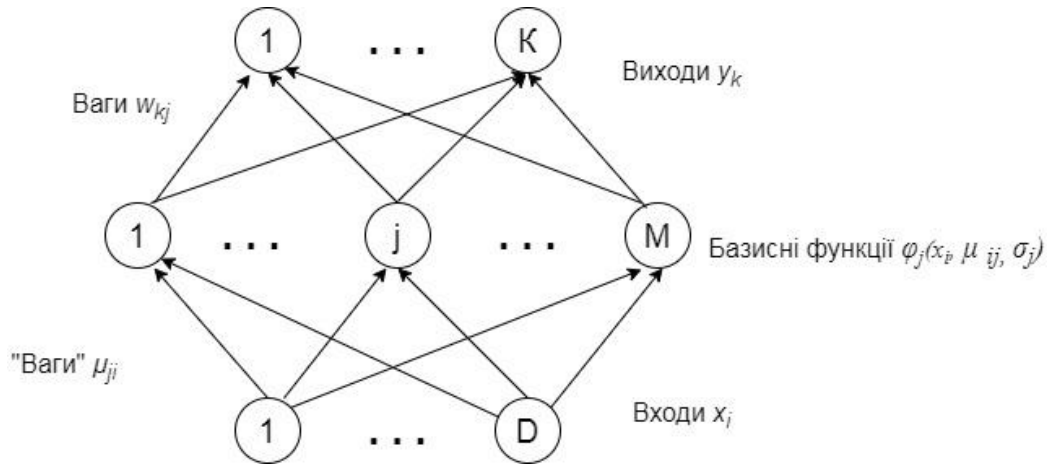


Рисунок 1.5 – Архітектура RBF network [13]

Однією з головних переваг RBF network є можливість визначення придатного прихованого параметру функції без необхідності проводити повну нелінійну оптимізацію всієї мережі [13].

1.2.6 AdaBoost

Бустинг – це підхід до машинного навчання, що базується на ідеї створення високої кваліфікації точного правила прогнозування, поєднуючи багато відносно слабких і неточних правил. Алгоритм AdaBoost був першим практичним алгоритмом бустинга і залишається одним з найбільш використовуваних і вивчених, що використовується в різних за стосунках багатьох галузей.

На початку роботи алгоритму задається навчальна вибірка $(x_1, y_1) \dots (x_m, y_m)$, де x_i -ті – це деякий параметр X , а клас $y_i \in \{-1; +1\}$. Для всіх $t=1, \dots, T$, розподіл D_t обчислюється для m навчальних екземплярів і даний слабкий алгоритм навчання для пошуку слабкої гіпотези $h_t: X \rightarrow \{-1; +1\}$, де мета слабого алгоритму знайти слабку гіпотезу з помилкою з низькою

вагою ε_t відносна до D_t . Кінцева або комбінована гіпотеза H обчислює ознаку зваженого поєднання слабких гіпотез

$$F(x) = \sum_{t=1}^T a_t h_t(x). \quad (1.5)$$

Це рівнозначно тому, що H обчислюється як зважена більшістю голосів слабка гіпотеза h_t , де кожному голосу присвоюється вага a_t [14].

До переваг алгоритму AdaBoost відносять:

- гарна узагальнююча властивість. В деяких задачах вдається будувати композиції, що мають перевагу в якості порівняно з базовими алгоритмами;
- простота реалізації;
- час побудови композиції практично повністю визначається часом навчання базових алгоритмів;
- можливість ідентифікувати об'єкти, що є шумовими викидами.

Проте даний алгоритм має значні недоліки пов'язані з перенавчанням, значними помилками в роботі базових алгоритмів, можливістю побудови неоптимального набору базових алгоритмів, бустинг може приводити до побудови громіздких композицій, що складаються з сотень алгоритмів [15].

1.2.7 ENORA

ENORA – це елітарний багатоцільовий еволюційний алгоритм на основі Паретто, який використовує $(\mu + \lambda)$ виживання (де μ – це розмір популяції, λ – це кількість народжених дітей), з рівномірною випадковою ініціалізацією та процесом відбору бінарних турнірів для вивчення простору атрибутів. Відстань переповнення використовується як міра визначення рейтингу ранжування на основі локального рівня не домінування разом із

самоадаптивним рівномірним кроссовером та самоадаптивною мутацією (рис.1.6) [16].

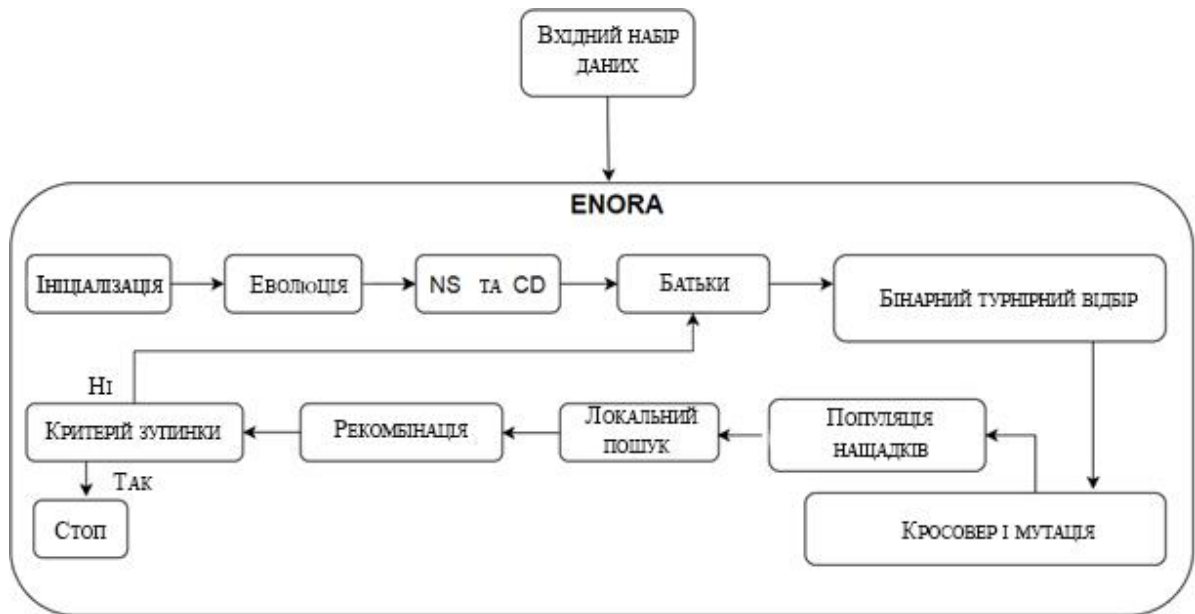


Рисунок 1.6 – Схема роботи алгоритму ENORA [16]

1.2.8 Порівняння класичних алгоритмів, що використовуються для задачі прогнозування

Розглянуті методи та алгоритми було протестовано з використанням набору даних про захворювання на діабет PIMA за допомогою ПЗ для аналізу даних та машинного навчання weka. Дані про точність прогнозування використаних алгоритмів наведено в таблиці 1.1 та на рис 1.7. При тестуванні було використано два способи поділу вибірки даних: поділ на 10 частин та поділ на навчальну та тестову вибірки у відсотковому відношенні 80/20%.

Таблиця 1.1 – Порівняльна таблиця точності прогнозування методів

№ п/п	Назва методу/алгоритму	Параметри			
		Cross-validation (10 folds)		Training/Test percentage split (80%)	
		Значення, %	Час роботи алгоритму, хв	Значення, %	Час роботи алгоритму, хв
1	Байєсова мережа	75.25	0,02	76.30	0,01
2	NNge	72.84	0,06	74.67	0,01
3	Random Forest	76.10	0,3	76.69	0,03
4	HMM	65.11	0,02	65.10	0.002
5	RBFNetwork	75.94	0,56	75.59	0,05
6	AdaBoost	73.38	0,03	73.77	0,01
7	ENORA	74.57	19,7	75.20	1,46

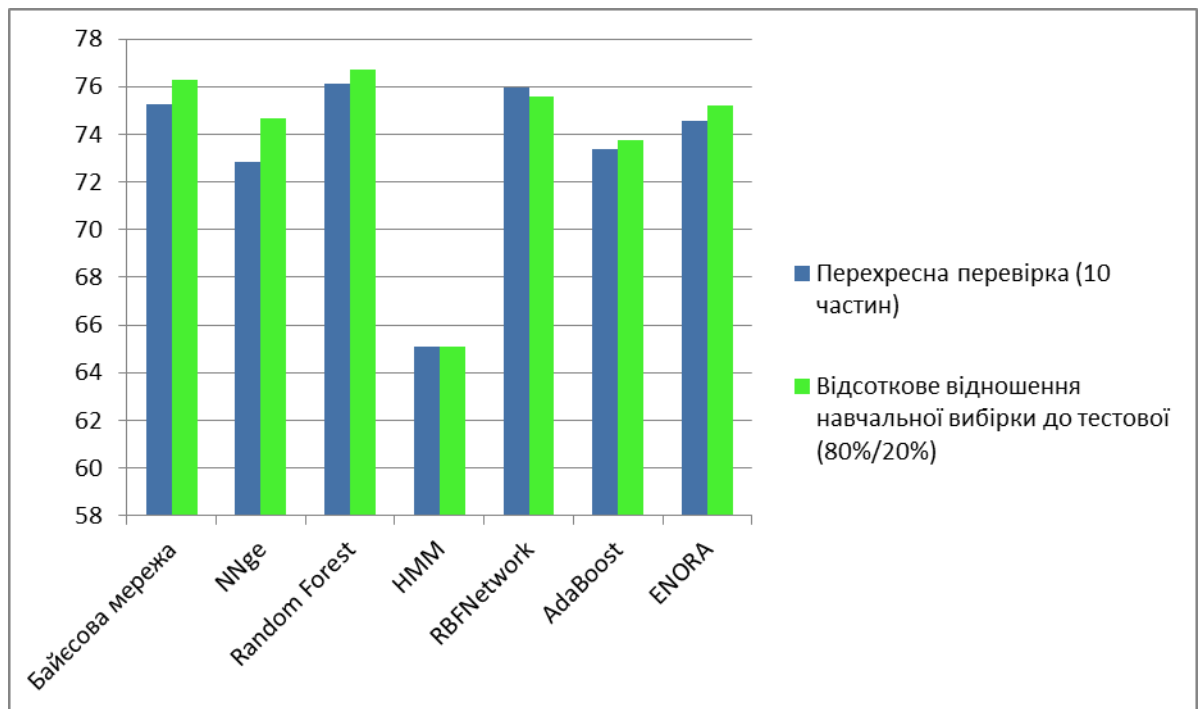


Рисунок 1.7 – Гістограма порівняння класичних алгоритмів за точністю прогнозування

Виходячи з таблиці порівняння можна зробити висновок про те, що кращим чином покази себе Байєсова мережа, точність класифікації якої при поділі вибірки на 10 частин та відсотковому поділі 20%/80% складає відповідно 75.25% та 76.30% та алгоритм Random Forest, який показав точність класифікації 76.10% та 76.69% при обраних параметрах поділу вибірки.

1.2.9 MultilayerPecptron

Штучна нейронна мережа (ШНМ) – це математична модель, а також її програмна або апаратна реалізація, побудована за принципом організації і функціонування біологічних нейронних мереж.

ШНМ являють собою систему з'єднаних і взаємодіючих між собою простих процесів (штучних нейронів) [17].

Multilayer персерптон (багатошаровий персерптон) – це багатошарова нейронна мережа прямого поширення, що складається з формальних нейронів і характеризується наступними параметрами і властивостями: M – кількість шарів мережі, N_m – кількість нейронів у m -му шарі, зв'язки між нейронами відсутні. Виходи нейронів m -го шару, $m=1,2,...,M-1$, надходять на входи нейронів тільки наступного $m+1$ -го шару. Зовнішній векторний сигнал x надходить на входи нейронів тільки першого шару, виходи нейронів останнього M -го шару утворюють вектор виходів мережі $y^{(M)}$ [18].

Кожен i -й нейрон m -го шару ((m, i) -й нейрон) перетворює вхідний вектор $x^{(m, i)}$ у вихідну скалярну величину $y^{(m, i)}$. Це перетворення складається з двох етапів: спочатку обчислюється дискримінантна функція,

$$\varphi^{(m,i)}(w^{(m,i)}, x^{(m,i)}), \quad (1.6)$$

яка далі перетворюється на вихідну величину

$$y^{(m,i)} = \psi^{(m,i)}(\varphi^{(m,i)}(w^{(m,i)}, x^{(m,i)})), \quad (1.7)$$

де $w^{(m,i)} = (w_0^{(m,i)}, w_1^{(m,i)}, \dots, w_{N_m}^{(m,i)})^T$ - вектор вагових коефіцієнтів нейрона;

$x_j^{(m,i)}$ - j -та компонента N_m – вимірного вхідного вектора $x^{(m,i)}$.

Якість рішення, одержуваного багат шаровим персептроном, буде істотно залежати від кількості шарів, кількості нейронів і кількості зв'язків між шарами.

Загальна кількість ваг і порогів в багат шаровому персептроні визначається за допомогою формули [18]:

$$N_w = \sum_{m=1}^M N_m (N_{m-1} + 1), N_0 = N. \quad (1.8)$$

До переваг багат шарового персептрона відносять можливість класифікувати будь-які об'єкти, що розділяються в просторі ознак.

До недоліків відносять складність і повільність, а також погану здібність до узагальнення [19].

1.2.10 Згорткова нейронна мережа (CNN)

Рекурентні нейронні мережі або багат шаровий персептрон з декількома прихованими шарами в ШНМ називають глибокими нейронними мережами (DNN). Згорткова нейронна мережа (CNN) – це один з видів рекурентних нейронних мереж. CNN має багато особливостей такі як проста структура, менші параметри тренувань та адаптивність [20].

CNN складається з пар шарів – шарів підвибірки і шарів згортки, кожен з яких в свою чергу складається з мап ознак. В таких мережах на перший шар нейронів подаються вхідні дані, останній шар є вихідним, а всі проміжні є повнозв'язними – кожен нейрон внутрішнього шару з'єднаний

зваженим ребром із кожним наступним нейроном попереднього шару. Взагалі, на внутрішньому шарі вектор виходів попереднього шару перемножується на деяку матрицю ваг повнозв'язного шару, після чого до отриманих значень застосовується деяка функція активації, що дозволяє ввести не лінійність операцій.

CNN відрізняються від класичних тим, що в них можуть бути присутніми не лише повнозв'язні шари. В основні види шарів додаються згорткові і pooling-шари. Загальна схема CNN наведена на рис.1.8 [21]-[22].

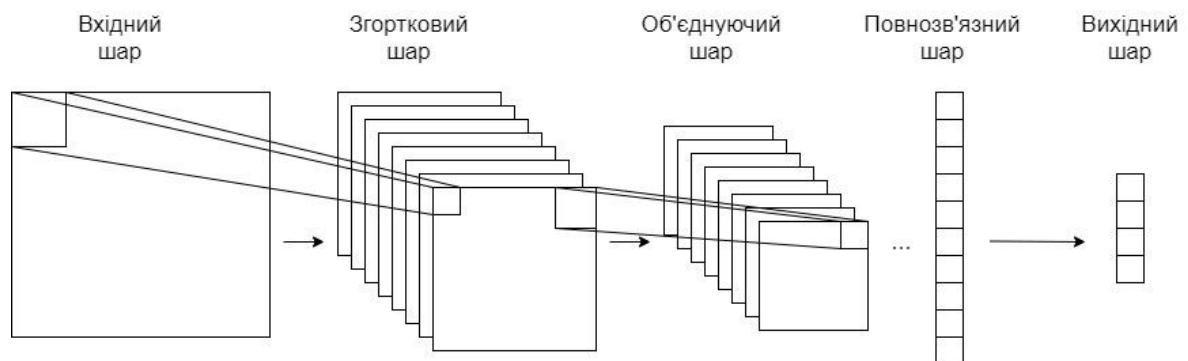


Рисунок 1.8 – Загальна схема CNN [18]

CNN зручні для обробки візуальних і інших двовірних даних. У порівнянні з іншими глибокими архітектурами, CNN показують чудові результати в обробці зображень і мовних застосувань. Вони також можуть бути навчені за допомогою стандартного зворотного поширення. CNN легше навчаються, ніж інші регулярні, глибокі нейронні мережі і мають менше параметрів для оцінки, що робить їх дуже привабливими [19].

1.2.11 LSTM

LSTM – це варіант рекурентний нейронних мереж (PHM), що здатна навчатися довготривалим залежностям. LSTM-мережі вперше були запропоновані Хохрайтером і Шмидхубером, а потім вдосконалені багатьма

іншими дослідниками. Вони добре підходять для широкого кола задач і є найбільш популярним типом РНМ.

В простій РНМ для реалізації рекурентності використовується комбінація прихованого стану на попередньому кроці і поточних вхідних даних в шарі з функцією активації. В LSTM-мережі рекурентність реалізується аналогічно, але $\tanh$ -шарів не один, а чотири, і взаємодіють вони досить специфічним чином [23].

LSTM-мережа – це штучна нейронна мережа, що складається з LSTM-модулів. LSTM-модуль – це рекурентний модуль мережі, що здатний запам'ятовувати значення як на короткі, та і на довгі проміжки часу. LSTM-модулі часто групуються в LSTM-блоки [24]. Схема роботи мережі LSTM зображена на рис. 1.9.

LSTM-блоки містять три або чотири «вентилі», які використовуються для контролю потоків інформації на входах і на виходах пам'яті даних блоків. Ці вентилі реалізуються в виді логічних функцій для обчислення значень в діапазоні $[0;1]$. Множення на це значення використовується для часткового допуску або заборони потоку інформації всередину і назовні пам'яті. Наприклад, «вхідний вентиль» контролює міру входження нового значення в пам'ять, а «вентиль забуття» контролює міру збереження значення в пам'яті. «Вихідний вентиль» контролює міру того, в якій мірі значення, що знаходиться в пам'яті, використовується при розрахунку вихідної функції активації для блоку [24].

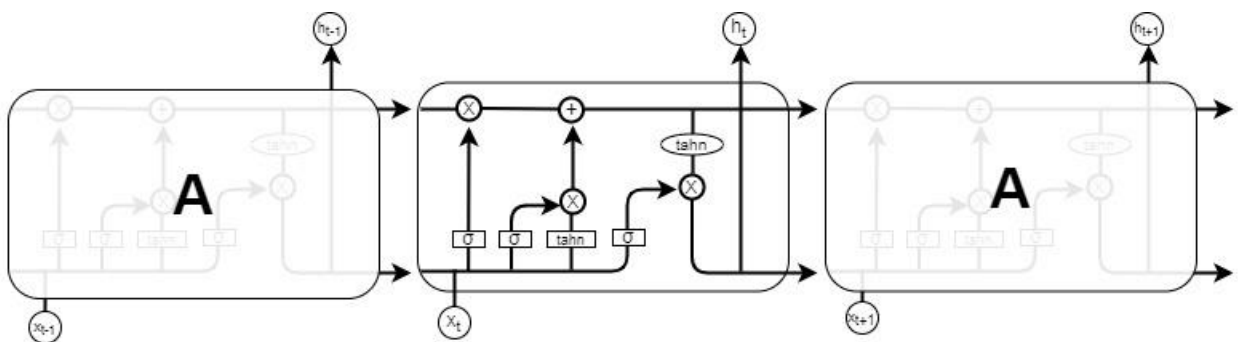


Рисунок 1.9 – Схема роботи LSTM [23]

Ваги в LSTM-блоці використовуються для завдання напряму оперування вентилів. Ці ваги визначені для значень, які подаються на блок (включаючи x_t і вихід з попереднього часового кроку h_{t-1}) для кожного з вентилів. Таким чином, LSTM-блок визначає, як розпоряджатися своєю пам'яттю як функцією з цих значень, і тренування ваг дозволяє LSTM-блоку вивчити функцію, що мінімізує втрати. LSTM-блоки зазвичай навчають за допомогою методу зворотного поширення помилки [24].

1.2.12 Dense neural network

Багат шарові нейронні мережі складаються з серії блоків, що називаються нейронами, що приймають вхідний сигнал і навчають своїм зваженим з'єднанням знижувати певні ваги вихідної функції. Більшість мере, що використовуються сьогодні мають структуру прямого зв'язку, що складається з ієрархії шарів. Це щільні нейронні мережі (DNN), коли вони мають велику кількість нейронів на кожному шарі, і вважаються «глибокими» (рис. 1.10), коли мають велику кількість шарів [25].

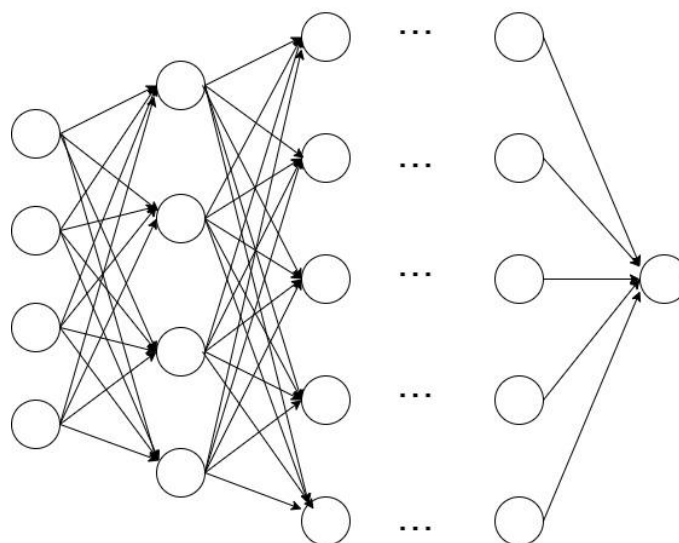


Рисунок 1.10 – Схема моделі DNN [25]

1.2.13 Порівняння розглянутих нейронних мереж

Розглянуті нейронні мережі було протестовано з використанням набору даних про захворювання на діабет PIMA. Дані про точність прогнозування використаних алгоритмів наведено в таблиці 1.2 та на рис 1.11. При тестуванні було використано спосіб поділу вибірки даних на навчальну та тестову вибірки у відсотковому відношенні 80/20%.

Таблиця 1.2 – Порівняння нейронних мереж

№ п/п	Назва нейронної мережі	Параметри	
		Значення точності, %	Час роботи алгоритму, хв
1	MultiLayerPerceptron (2 прихованих шари)	78.00	0,04
2	Згорткова мережа(CNN)	77.92	1,9
3	LSTM	89.6	~5
4	Dense neural network	88.67	~3

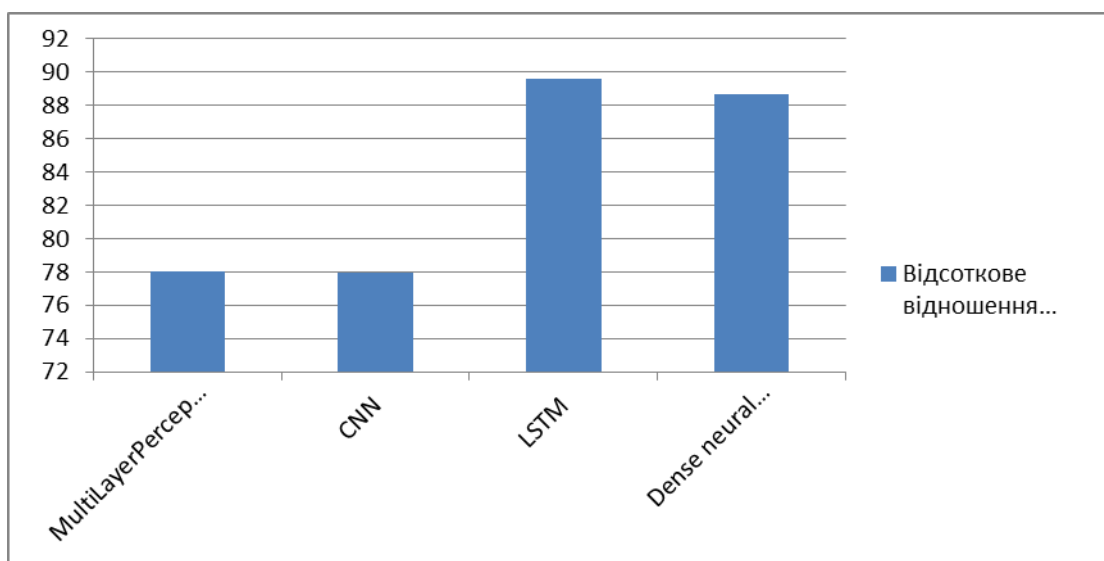


Рисунок 1.11 – Гістограма порівняння точності прогнозування нейронних мереж

Після порівняння розглянутих ШНМ очевидно, що кращу ступінь прогнозування має мережа LSTM – 89.6%, далі Dense neural network має точність 88.67%, MultiLayerPerceptron показав точність 78.00; і найнижчу точність показала згортова нейронна мережа 77.92%. Найшвидше відпрацювала мережа MultiLayerPerceptron час роботи склав 0,04 хв. Час роботи загорткової нейронної мережі склав 1,9 хв. На роботу Dense neural network знадобилося близько 3 хвилин, а час роботи мережі LSTM склав близько 5 хвилин.

1.3 Аналіз методів для медичного діагностування

Оскільки ЦД є хронічним захворюванням, що важко піддається лікуванню, то його діагностика є актуальним питанням в медичному діагностуванні. Дослідженням даної задачі займається багато науковців по всьому світу, і вже на даний час створено досить багато прогнозних моделей для діагностики та прогнозування ЦД, які є як специфічними (придатні лише для прогнозування/діагностування ЦД), так і загальними (придатні для використання для багатьох захворювань).

Для постановки завдання розробки, було проведено аналіз літератури, щодо існуючих прогнозних моделей опис яких наведено далі.

В роботі [26], представлена модель, що розроблена на основі нейронних мереж і меметичного алгоритму для прогнозування діабету. Дана модель розроблена на основі демографічних показників способу життя і клінічних характеристик. Для покращення точності прогнозування автори використали меметичний алгоритм для оновлення вагів нейронної мережі. Покращивши ваги нейромережевих моделей, автори отримали модель меметичного алгоритму, яка досягла точності моделі в 93%. Ця модель здатна забезпечити більшу точність прогнозування, ніж інші моделі, розглянуті в статті.

Дане дослідження має деякі обмеження. Меметичні алгоритми вимагають, порівняно більшого часу на навчання і більш потужний процесор, ніж моделі логістичної регресії. Також кількість даних дещо обмежувала якість дослідження. Модель, що представлена в роботі, можна покращити. Точність меметичного алгоритму може бути покращена новими даними і навчанням. Використання цих моделей іншими дослідниками може покращити надійність моделей і, таким чином, зекономити ресурси. В цьому дослідженні меметичні алгоритми були використані для оновлення ваг і покращення точності прогнозування нейронної мережі цукрового діабету.

Даною роботою [27] пропонується система повнозв'язного класифікатора багат шарового персепторона (БШП), який мінімізує емпіричний ризик і максимізує помилково класифіковані точки в навчальних даних (рис. 1.12).



Рисунок 1.12 – Схема моделі [27]

В дійсності згорткові нейронні мережі розроблюють множину детекторів ознак і використовують їх для розробки декількох карт, які називаються згортковими шарами. Протягом навчання, мережа визначає які ознаки є важливими для можливості сканування вхідних даних і для більш точної їх класифікації. Виходячи з цього, класифікатор сам розроблює власні особливості детекторів. В багатьох випадках функції, що розглядаються нейронною мережею, будуть непомітні для людського ока, а саме тому згорткові нейронні мережі є такими корисними.

Після реалізації алгоритма CNN-SVM для класифікації можна зробити висновок, що запропонована система може бути використана для навчання і розробки моделі. Цей аналіз буде включати розробку модифікованої нейронної мережі з глибоким навчання шляхом інтеграції методу опорних векторів. Для того, щоб перевірити систему, показники продуктивності будуть порівнюватися з уже існуючим методом. Запропонована система з модифікованим алгоритмом суттєво оптимізує загальну продуктивність системи визначення і ефективно знижує обчислювальну складність класифікатора.

Автори [28] пропонують модифікацію глибокої навчаючої загорткової нейронної мережі за рахунок інтеграції машини опорних векторів. Традиційна CNN використовує MLP алгоритми для класифікації похідних ознак від необроблених вхідних даних. Хоча ці алгоритми показали багатообіцяючі результати, MLP алгоритми мають різноманітні недоліки при класифікації нелінійних наборів даних. Для подолання цих недоліків, запропонована архітектура поєднує концепцію CNN методики виділення ознак з класифікатором SVM. Набори ознак від необроблених даних обчислюються шляхом математичної згортки цих даних з вагами Гаусівського ядра. Розмір цих мап ознак з використанням операції максимізації за виділеними наборами ознак. Ці оптимізовані мапи ознак далі класифікуються за допомогою SVM з функцією ядра для нелінійної класифікації даних.

Модифікований алгоритм суттєво оптимізує загальну продуктивність системи визначення і ефективно знижує обчислювальну складність класифікатора.

В роботі [29] описаний неінвазивний метод виявлення діабету. Метод включає в себе реалізацію одновимірної загорткової нейронної мережі для класифікації необроблених даних. Алгоритм CNN заплутує необроблені дані з фільтром на основі ядра для створення набору мап ознак. Розмірність цих мап зменшена шляхом використання операції мінімізації за зразками даних і тим самим знижуючи вибірку ознак. Далі ці мапи застосовуються до повнозв'язного класифікатора, який слідує за механізмом, що подібний до багатошарової нейронної мережі на основі персептрона. Даний алгоритм знижує витрати на обчислення і необхідність в оптимальних ознаках.

За результатами алгоритм виконав успішну класифікацію. Встановлено, що алгоритм значно зменшує середньоквадратичну помилку і оптимізує загальну продуктивність класифікатора.

В роботі [30] запропоновано інтелектуальну прогностичну модель, що використовує глибоке навчання для прогнозування фактору ризику виникнення хвороби у пацієнта і ступені тяжкості діабетиків з використанням набору умовних даних. Модель пропонує глибоке навчання у формі глибокої нейронної мережі, яка допомагає застосовувати прогнозну аналітику до набору даних про діабет для отримання оптимальних результатів. Існуючі прогностні моделі використовуються для прогнозування ступені тяжкості і факторів ризику діабетиків на основі даних. В запропонованій моделі алгоритм вибору запускається для процесу вибору, далі модуль глибокого навчання має глибоку нейронну мережу, яка використовує обмежену машину Больцмана (RBM) в якості базової одиниці для аналізу даних, присвоюючи ваги кожній гілці нейронної мережі. Результати тестування показали, що кількість досліджень проведених по діабету з використанням моделі глибокого навчання є невеликою. Це допоможе з більшою точністю прогнозувати діабет.

Модель може бути покращена шляхом збільшення розміру даних. Окрім цього, прогнозування може залежати від інших факторів, які відсутні у вказаному наборі даних. Прийняття до уваги цих факторів буде сприяти подальшому підвищенню точності прогнозування.

Точність прогнозування за допомогою даної моделі може досягати 86% в залежності від набору даних.

Автори роботи [31] побудували прогностичні моделі, використовуючи методи логічної регресії та градієнтного підсилювання (GBM). Площина під кривою робочої характеристики приймача (AROC) була використана для оцінки дискримінаційної здатності цих моделей. Автори використали метод скоригованого порогу і метод вагової категорії для покращення чутливості – кількості правильно класифікованих пацієнтів з цукровим діабетом розробленою моделлю. Дані моделі були порівнянні з іншими методами машинного навчання, такими як дерево рішень та випадковий ліс.

Результати моделей показали себе наступним чином: для градієнтного підсилювання чутливість склала 71.6%, для моделі логічної регресії – 73.4%.

В роботі [32] автори пропонують прогнозну модель, що має назву покращеного алгоритму прогнозування діабету (IDPA), який поєднує алгоритм ієрархічної кластеризації і алгоритм наївної байєсовської класифікації, що розроблений для виявлення та прогнозування цукрового діабету 2-го типу. В даному випадку модуль IDPA виконує групування даних на дві групи, тобто діабет та недіабет, шляхом застосування алгоритму ієрархічної кластеризації. Далі виконується фільтрації шляхом порівняння значення групи зі значенням класу з наступним застосуванням наївного байєсовського класифікатора для прогнозування діабету. Результати показуються, що запропонований метод може прогнозувати діабет з більш високим рівнем точності (96%), ніж традиційні та існуючі методи. Дана модель може бути використана для раннього прогнозування діабету, тим самим зменшуючи серйозні ускладнення діабету.

В роботі [33] пропонується гібридна модель для прогнозування у керування діабету 1-го та 2-го типу. Робота моделі складається з двох етапів: автономного та оперативного етапів. Оффлайн етап орієнтований на підготовку даних (структурування) із зовнішніх джерел знань, які представлені у вигляді неструктурованих клінічних діаграм і онлайн-керівництва по діабету і отримання знань з цих джерел. Отримання знань складається з ручних і автоматичних процедур. В керівництві процес, по-перше, клінічні діаграми пацієнтів перетворюються в структуровану форму, що називається інформаційна система цукрового діабету (DMIS) і керівництва за відхиленнями та правилами аналізу тенденцій (ATAR). На оффлайн-етапі отримання, набір грубих методів отримання знань, що базуються на наборі, використовуються для розробки правил прогнозування ЦД (DMPR) з DMIS. Правила зберігаються в базах знань на оффлайн-процесі. Онлайн-етап – це фаза реального часу або виконання моделі, яка виконує прогнозування і керування лікарів для підтримки їх в прийнятті рішення. Ця фаза активується, коли пацієнт прибуває на діагностику або коли пацієнт реєструється для подальшого спостереження. У випадку, якщо пацієнт реєструється вперше, запускається методологія HRBR. В HRBR діагнози з грубим набором міркувань (RSR) і прогнозує тип діабету. У випадку, коли пацієнт уже реєстрований, активується лише частина СТА. СТА надає можливість лікарям бачити всі попередні записи пацієнта у зведених формі і виявляти аномальні залежності. СТА також надає аналіз майбутніх тенденцій для всіх спостережень і підтримує їх для визначення потенційних ризиків в майбутньому. Відповідно, лікар може прийняти профілактичні заходи. Точність прогнозування такої моделі складає 95.91%.

Метод, запропонований в роботі [34], пропонує об'єднання KNN і генетичного алгоритму для підвищення точності класифікації даних діабету (рис. 1.13).

Автори [34] використали варіації генетичного алгоритму для відбору ознак і ранжування атрибутів, які більше сприяють класифікації. Кращі

ранжовані атрибути, що використовуються для прогнозування захворювання і алгоритм класифікації побудований на основі оцінених атрибутів. Класифікатор навчений класифікувати дані про діабет, що є як діабетичними так і недіабетичними. Запропонована система відповідальна за:

- вибір найкращої підмножини ознак з використанням генетичного алгоритму.
- побудову класифікатора і точності вимірювання класифікатора
- виявлення ускладнень, пов'язаних з діабетом, такі як проблеми з серцем та нирками, можливий інсульт та інші.

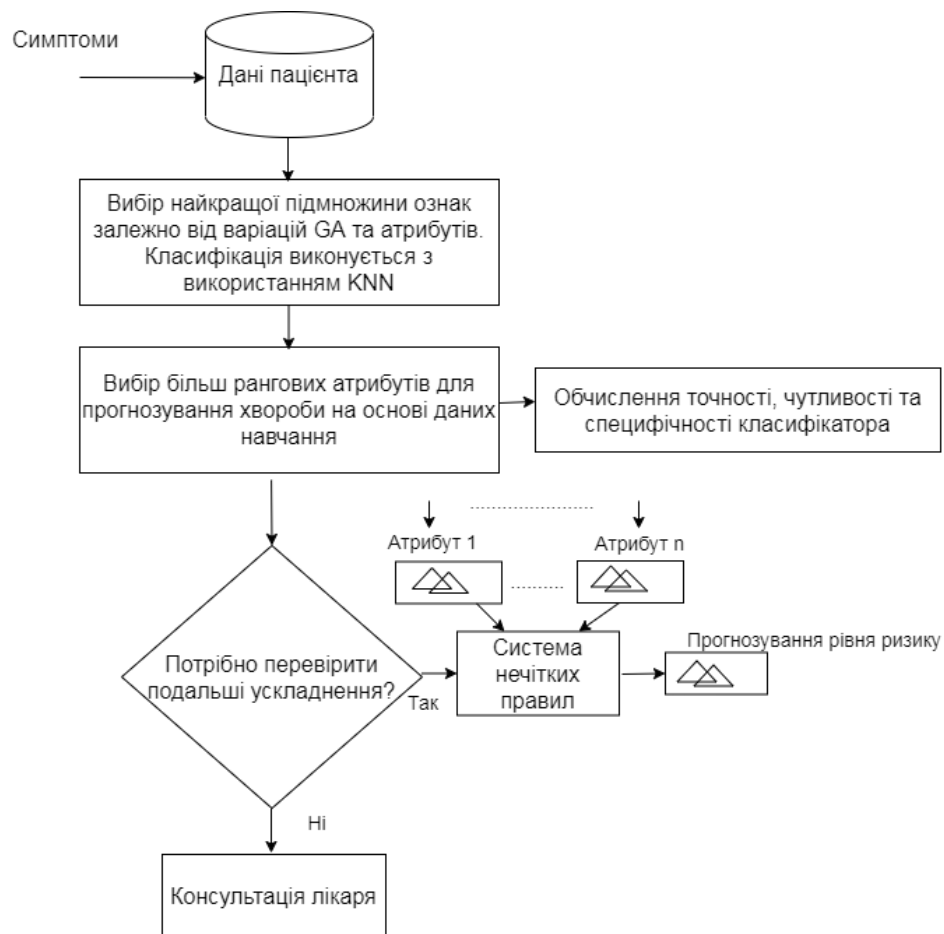


Рисунок 1.13 – Схема методу [34]

В роботі [35] запропонована оптимізована модель CNN (10-шарова) на основі DCNN та LeNet-5. Дана модель забезпечує точність прогнозування 89-95% (залежно від набору даних). Було внесено зміни між третім шаром max-

pooling та повнозв'язним шаром. Окрім цих шарів CNN також включає три шари згортки, три шари max-pooling та два повнозв'язних шари. Недостатня кількість шарів призводить до неадекватної здатності до вилучення CNN, в той час як, надмірна їх кількість збільшують витрати часу на розрахунок. В даній моделі шари згортки використовуються для вилучення складних параметрів вхідних мап ознак. Шари max-pooling зберігають основні особливості вхідних сигналів при зменшенні параметрів та обчислень, що допомогло уникнути виникнення надмірних розмірів та покращити здатність до узагальнення моделі CNN. Останні два повнозв'язні шари об'єднуються всі верхні мапи ознак в одновимірний масив, який використовується для класифікації виходу мережі. Загальна схема запропонованої моделі зображена на рис.1.14.

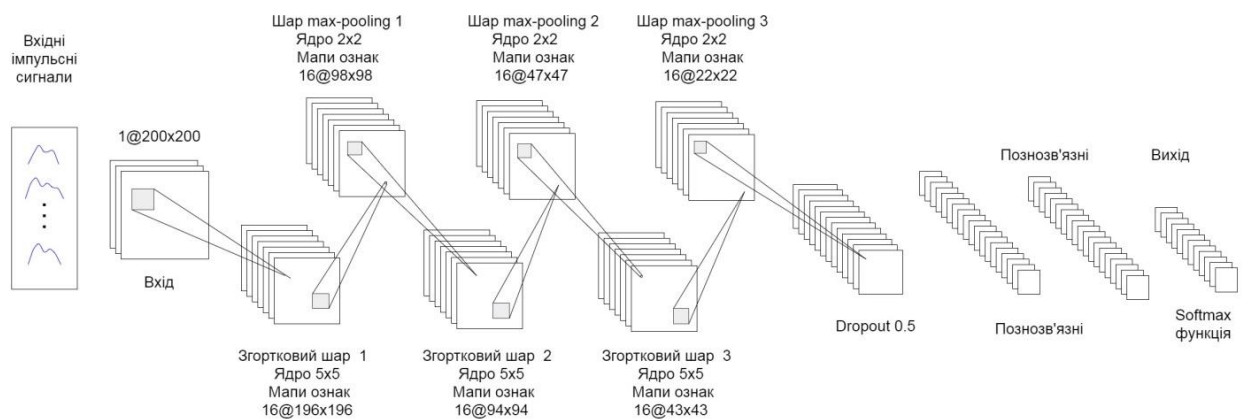


Рисунок 1.14 – Загальна схема моделі, що запропонована в [35]

Автори роботи [36] пропонують використовувати сигмоїдне ядро та CNN. З блок-схеми моделі SKCNN (рис.1.15) видно, що спочатку SKCNN розраховує сигмоїдне ядро для прогнозування захворювання та препаратів, структурну схожість препаратів та семантичну подібність хвороб. Сигмоїдне ядро препарату поєднується зі структурною схожістю препаратів, а сигмоїдне ядро хвороби поєднується з семантичною схожістю хвороб. Далі CNN для отримання функцій використовує комбіновану схожість препаратів та захворювань. На останньому кроці класифікатор Random Forest вводиться

для висновку, чи є препарати та хвороби взаємопов'язаними. Точність, що забезпечує дана модель приблизно становить 92,69%.

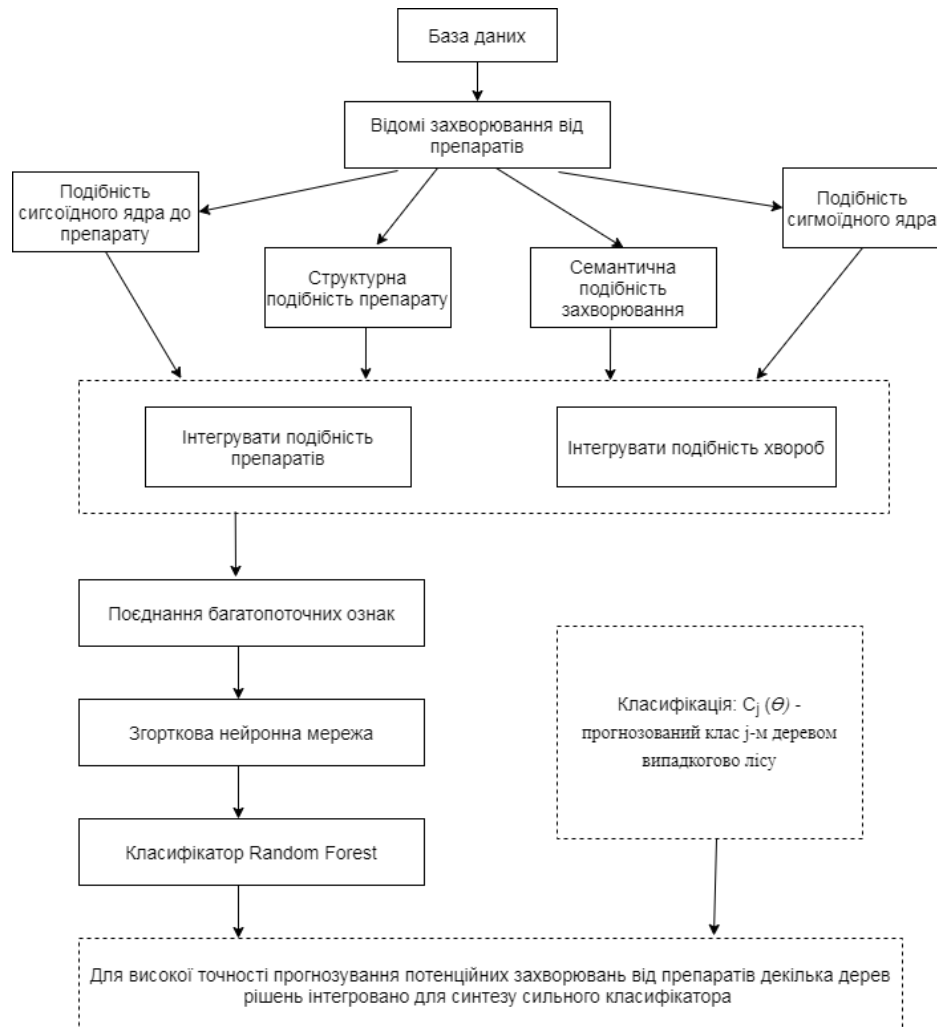


Рисунок 1.15 – Блок-схема роботи моделі SKCNN [36]

В роботі [37] представлена нова структура CNN, яка полягає у поєднанні ідей двох структур Койчі Іто та Гіла Леві, де Койчі використовував глибоке багатозадачне навчання, а Леві використовував поділ зображення на п'ять областей. Модифікація CNN, що описана в роботі, складається з двох частин: перша – це чотири загорткових шари з шарами об'єднання; друга частина формується з трьох повнозв'язних шарів. Далі STL та DMTL використовуються в процесі навчання, цей спосіб було визначено Койчі Іто.

Точність класифікації даної модифікації становить 80,11% при використанні CNN та методу навчання STL та 91,34% для CNN та методу навчання DMTL.

1.4 Постановка завдання

Метою роботи є дослідження та моделювання залежності розвитку цукрового діабету від медичних показників стану людини.

Математична залежність виникнення цукрового діабету від показників стану та анамнезу пацієнта можна визначити як функцію, в якій можливість розвитку захворювання буде залежати від показників пацієнта. Узагальнена модель залежності має вигляд (1.9):

$$K_{morb} = f(x_n), \quad (1.9)$$

де K_{morb} – показник, що характеризує наявність ризику виникнення цукрового діабету;

x – сукупність показників, що характеризують стан здоров'я пацієнта;

n – кількість показників.

Виходячи з наведених даних і аналізу статистичних даних, можна дійти висновку, що шукана математична модель буде не детермінованою, а скоріш стохастичною.

Відповідно до Уніфікованого клінічного протоколу первинної та вторинної (спеціалізованої) медичної допомоги [38] оцінка ризику виникнення у пацієнта цукрового діабету проводиться на основі діагностичних заходів та анамнезу людини. До основних факторів розвитку ЦД відносять наступні показники:

- вік пацієнта;
- концентрація глюкози в плазмі через 2 години після її введення в пероральному глюкозотолерантному тесті;

- концентрація інсульну в сироватці крові;
- значення діастолічного артеріального тесту;
- індекс маси тіла пацієнта;
- генетика.

В якості додаткових показників враховують також вагітності та товщину шкірної складки.

Окрім вищевказаних факторів впливу на розвиток ЦД є й інші, однак вони не несуть значного впливу на факт виникнення захворювання, тож враховувати їх не має сенсу.

Відповідно до вище вказаних показників здоров'я людини математичну залежність виникнення цукрового діабету можна визначити наступним чином (1.10) :

$$K_{morb} = f(x_{age}, x_{GC}, x_{IC}, x_{DBP}, x_{BMI}, x_{gen}, x_{NP}, x_{ST}), \quad (1.10)$$

де x_{age} – вік пацієнта;

x_{GC} – концентрація глюкози після її перорального глюкозотолерантного тесту;

x_{IC} – концентрація інсульну;

x_{DBP} – значення діастолічного артеріального тесту;

x_{BMI} – значення індексу маси тіла;

x_{gen} – генетична схильність;

x_{NP} – кількість вагітностей;

x_{ST} – товщину шкірної складки в районі трицепса.

1.5 Висновки за розділом 1

Відповідно до проведеного аналізу класичних та еволюційних методів, що здатні вирішувати завдання класифікацій та прогнозування

даних найкращим чином себе показали Байєсова мережа точність класифікації якої складає 76.30%, алгоритм Random Forest – точність класифікації складає 76.69% та мережа LSTM – точність класифікації складає 89.6% (рис. 1.16). Часові витрати на роботу даних алгоритмів є порівняно найбільш прийнятними.

Сформульовано та поставлено завдання до роботи.

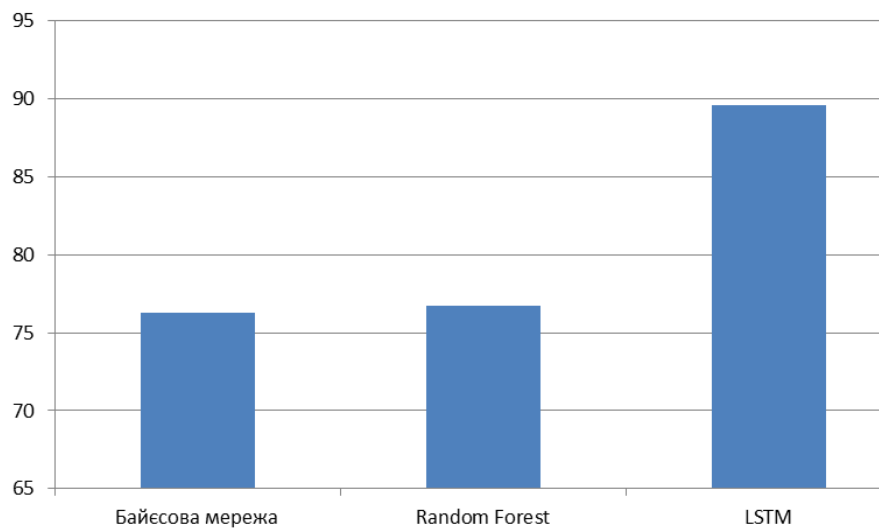


Рисунок 1.16 – Гістограма порівняння точності прогнозування методів, що показали себе найкращим чином

2 ВИБІР ІНСТРУМЕНТІВ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір мови програмування

Оскільки нейронні мережі можна реалізувати практично будь-якою мовою програмування, то питання її вибору є досить важливим. Від обраної мови буде залежати й правильність та легкість опрацювання даних, швидкість роботи моделі, підтримка алгоритмів тощо. Найчастіше для розробки нейронних мереж використовують мови C++, C#, Python та Matlab.

2.1.1 Мова програмування C++

C++ був розроблений з урахуванням продуктивності, ефективності і гнучкості, що робить його ідеальним вибором для багатьох проєктів програмування штучного інтелекту, які вимагають швидкості. У порівнянні з іншими мовами програмування, C++ має більш швидке виконання і більш низьку затримку, що робить його корисним для пошуку рішень складних проблем штучного інтелекту. Він також дозволяє швидко використовувати алгоритми і є ефективним засобом написання статичних методів ШІ, таких як нейронні мережі [38].

Переваг цієї мови модні віднести можливість роботи з пам'яттю та ядрами комп'ютера, а також можливість програм, які написані даною мовою запускатися на всіх операційних системах. Однак, нажаль, мова має один значний недолік – її складність роботи та важкість навчання. Для розробки будь-якого застосування необхідно буде витратити багато часу, для розуміння роботи з пам'яттю комп'ютера та підводних каменів, що можуть зустрітися під час роботи.

2.1.2 Мова програмування R

Мова R добре відома завдяки аналізу і обробці даних. В основному використовується в області статистики. Програміст може використовувати R для створення математичних символів, графіків і формул, коли це необхідно. Як мова програмування загального призначення, R має декілька мов, які можна використовувати для розробки ШІ. Вони включають в себе моделі G, RODBC, Tm і Class, що спрощує процес реалізації алгоритмів машинного навчання [39].

До переваг даної мови можна віднести її легкість у вивченні, велике співтовариство, широкі можливості у інтеграції з іншими мовами програмування, володіє великою кількістю безкоштовних пакетів з відкритим вихідним кодом тощо [39].

Однак незважаючи на велику кількість переваг R має обмеження, які є вагомими у галузі штучного інтелекту (ШІ). Мова має проблеми із захищеністю та керуванню пам'яттю, повільна, на ній не просто працювати з ООП.

2.1.3 Мова програмування Python

Python – це об'єктно-орієнтована високорівнева мова програмування з динамічною семантикою, який для виводу результатів використовує інтерпретатор. Вбудовані високорівневі структури даних в поєднанні з динамічною типізацією і зв'язуванням роблять мову привабливою для швидкої розробки (RAD). Окрім цього, її можна використовувати в якості сценарної мови для поєднання програмних компонентів [40].

Python містить велику стандартну бібліотеку модулів протестованого коду, які легко можуть бути імпортовані в вашу власні програми.

Перерахуємо деякі найважливіші особливості мови, які роблять її привабливою для програмістів:

- Python безкоштовний – це вільне програмне забезпечення з відкритим вихідним кодом;
- Python легкий у вивченні – має простий синтаксис;
- Python дозволяє створювати код, що легко читається;
- Python має модульну структуру;
- Python можна запускати на багатьох різних платформах;
- Python володіє багатим «арсеналом» – він має велику стандартну бібліотеку, яка легко інтегрується в ваші програми;
- Python розширюваний – дозволяє додавати низькорівневі модулі;
- Python універсальний – підтримує як процедурні, так і об'єктно-орієнтовані методи програмування [41].

2.1.4 Мова програмування Matlab

Matlab – це не тільки інтерпретована мова програмування, це повноцінний математичний пакет, призначений для вирішення задач обчислювальної математики та побудови чисельних моделей складних об'єктів і процесів [18].

MatLab є патентованою мовою програмування, що розроблена MathWorks. MatLab – одна з найстаріших, ретельно опрацьованих і перевірених часом систем автоматизації математичних розрахунків, побудована на розширеному поданні та застосуванні матричних операцій. Незважаючи на те, що спочатку MatLab призначався виключно для обчислень, в процесі еволюції в системі з'явилися широкі можливості для програмування. Серед недоліків системи MatLab можна відзначити невисоку інтегрованість середовища, не дуже виразну довідкову систему і специфічний редактор коду MatLab-програм. Сьогодні система широко використовується в техніці, науці та освіті для аналізу даних і організації обчислень [18], [42].

Варто відзначити, що пакет MatLab містить великий набір бібліотек, які реалізують сучасні чисельні методи моделювання, створені за останні декілька десятиріч. MatLab може бути застосований для розрахунків практично в будь-якій області науки та техніки. Також система містить потужні засоби графічної візуалізації. Серед недоліків варто відзначити низьку швидкість роботи [18], [42].

2.2 Порівняння мов програмування

Відповідно до проведеного аналізу мов програмування, що можуть використовуватися для реалізації ШІ, для більшої наочності, їх порівняння за основними критеріями зведено до таблиці 2.1.

Таблиця 2.1 – Порівняння мов програмування

Критерії порівняння	Мова програмування			
	C++	R	Python	Matlab
1	2	3	4	5
Легкість вивчення	-	+	+	-
Складність	+	-	-	+
Безкоштовність	+	+	+	-
Платформонезалежність	+	+	+	+
Наявність бібліотек машинного навчання	+	+	+	+
Висока продуктивність	+	-	+	-
Швидкість роботи	+	-	+	-
Швидкість розробки	-	+	+	-
Велике співтовариство	+	-	+	-
Робота з пам'яттю комп'ютера	+	-	+	+

Продовження таблиці 2.1

1	2	3	4	5
Підтримка ООП	+	-	+	+
Можливість інтеграції з іншими мовами	+	+	+	+

За результатами аналізу мов програмування, обрано мову Python, яка є високорівневою мовою загального призначення, що орієнтована на підвищення продуктивності розробника и читабельності коду. Стандартна бібліотека мови містить великий набір корисних функцій. Основними ознаками мови є динамічна типізація, автоматичне керування пам'яттю, механізм обробки виключень, підтримка багато потокових обчислень, високо рівневі структури даних [40]. Підтримується розбиття програми на модулі, які в свою чергу можуть поєднуватися в пакети. Python ідеально підходить для вирішення задач машинного навчання за рахунок наявності відповідних бібліотек. Для написання програм обраною мовою необхідно порівняно менший обсяг трудовитрат. Окрім того, програми, реалізовані на Python, запускаються без компіляції, за рахунок чого збільшується продуктивність [41].

Python поєднує в собі потужність мова програмування з простотою використання предметно-орієнтованих скрипкових мов таких як Matlab або R.

2.3 Огляд бібліотек машинного навчання

Python став загальноприйнятою мовою для багатьох галузей застосування науки про дані (data science). Однією з причин цього, є те що мова містить великий набір інструментів, які спрощують процес написання коду і, тим самим, скорочують час розробки. Python має бібліотеки для

завантаження даних, обробки природної мови, обробки зображень і багато інших [43].

Сьогодні для реалізації методів машинного навчання важливу роль відіграють спеціалізовані бібліотека та фреймворки, що надають повністю реалізовані базові структури та алгоритми для зручної розробки. До них належать Keras, Tensorflow, Theano, NumPy, Scikit-learn, Matplotlib та інші. Опис даних обраних для роботи інструментів наведено далі.

2.3.1 Бібліотека для створення нейронних мереж Keras

Keras – це API нейронних мереж високого рівня, написаний на Python і здатний працювати над Tensorflow або Theano. Бібліотека створена з метою швидкого експериментування та базується на вмінні переходити від ідеї до результату з найменшою можливою затримкою, що є ключем до хороших досліджень [32].

Оскільки Keras розроблявся з метою проведення швидких досліджень, то його ключовими характеристиками є [44]:

- можливість виконувати один і той же код на CPU або GPU;
- спрощення розробки прототипів моделей машинного навчання;
- включає в себе вбудовану підтримку загорткових мереж (для розпізнавання образів), рекурентних мереж (для обробки послідовностей) і будь-яких їх комбінацій;
- містить у собі підтримку довільних мережевих архітектур: моделей з множинними входами або виходами, спільне використання шарів, спільне використання моделей тощо.

З наведених характеристик можна зробити висновок, що Keras підходить для створення практично будь-яких моделей машинного навчання.

Бібліотека Keras поширюється на умовах вільної ліцензії і безкоштовно використовуватися для комерційних проєктів. Він сумісний з версіями Python від 2.7 і вище [43]. Keras викрішує велике коло задач та

використовується як академічними дослідниками та інженерами великих компаній, так і студентами та любителями.

Keras використовує модульний підхід, тобто до нього можна підключити декілька різних низькорівневих бібліотек. На сьогодні підтримуються три такі бібліотеки: Tensorflow, Theano та CNTK. Будь-який код, який використовує Keras, можна запускати з будь-якої з цих бібліотек без необхідності змінювати щось в коді: можна швидко перемикатися між ними під час розробки, що часом є уже корисним, наприклад, коли одна з бібліотек має більшу продуктивність при вирішенні конкретної задачі [43], [44].

2.3.2 Бібліотека Scikit-learn

Scikit-learn містить у собі реалізації алгоритмів, заснованих на навчанні з учителем і без. Він використовується при роботі на мові Python.

Scikit-learn – це бібліотека, що базується на бібліотеці SciPy (Scientific Python), яка призначена для виконання наукових та інженерних розрахунків. Тому SciPy необхідно встановити перед використанням Scikit-learn [45].

В Scikit-learn особлива увага приділяється питанням зручності користування, документації і оптимізації швидкості роботи. До переваг даної бібліотеки відносять простоту та ефективність інструментів аналізу даних, а також можливість використання відкритого вихідного коду.

Бібліотека орієнтована не на маніпуляції з даними, а на моделювання. Scikit-learn реалізує наступні основні методи [45]:

- кластеризація;
- класифікація;
- регресія;
- скорочення розмірності;

- оптимізації параметрів алгоритму (отримання максимально точного результату);
- алгоритмічні композиції (комбінації декількох моделей);
- алгоритми навчання з учителем та без нього.

2.3.3 Бібліотека Matplotlib

Matplotlib – це бібліотека на мові програмування Python призначена для побудови наукових графіків та візуалізації даних. Вихідні зображення можуть бути використані в якості ілюстрацій. Matplotlib є гнучким, а також легко конфігурованим пакетом, що разом з іншими бібліотеками, такими як NumPy, Scipy та IPython надає можливості, подібні Matlab [45].

Пакет може підтримувати наступні види графіків та діаграм: діаграми розкиду (scatter plot), графіки (line plot), гістограми (histogram), стовпчасті діаграми (bar chart), кругові діаграми (pie chart), контурні графіки (contour plot), стовбур-лист діаграми (stem plot), поля градієнтів (quiver) та спектральні діаграми (spectrogram) [43].

2.3.4 Бібліотека для роботи з масивами NumPy

NumPy – це один з основних пакетів для наукових обчислень мови програмування Python, що призначений для підтримки великих, багатовимірних масивів та матриць з великою кількістю математичних функцій високого рівня для роботи з ними [43].

Сьогодні майже всі нейронні мережі на Python використовують бібліотеку NumPy, яка надає можливість швидко і ефективно працювати з матрицями, таблицями в різних форматах, числовими даними, а також проводити велику кількість типових операцій, що потрібні в процесі роботи нейронної мережі.

2.4 Вибір середовища програмування

Інструменти розробки програмних засобів створені для спрощення, збільшення продуктивності та зручності процесу реалізації того чи іншого застосування, в яких міститься готова функціональність. Вибір такого середовища в цілому залежить від обраної мови програмування, цілей, набору готового функціоналу, можливості налаштування середовища під себе. Серед найпопулярніших серед інструментів для розробки методів штучного інтелекту виділяють Visual Studio, IntelliJ IDEA, Sublime Text.

2.4.1 Середовище розробки Visual Studio

Visual Studio – це професійний інструмент розробки програмного забезпечення для Windows, Mac, iOS, Android, а також вебзастосунків і хмарних сервісів.

Microsoft Visual Studio – повнофункціональне інтегроване середовище розробки (IDE) з підтримкою популярних мов програмування, серед яких C, C++, VB.NET, C#, JavaScript, Python.

Функціональність IDE охоплює всі етапи розробки ПЗ, надаючи сучасні інструменти для написання коду, проєктування графічних інтерфейсів, збірки, від лагодження і тестування за стосунків. Можливості Visual Studio можуть бути доповнені шляхом підключення необхідних розширень [46]. Дане середовище може похизуватися своїм зручним для використання дизайном, графічним виглядом. Також до переваг можна віднести інтегрування з багатьма програмами, які допомагають в написанні програмного коду. Серед недоліків варто виділити важкість освоєння з даними середовищем і загальну важкість роботи програми через її великий функціонал.

2.4.2 Середовище розробки IntelliJ IDEA

IntelliJ IDEA – це IDE, що в першу чергу орієнтована на розробку мовою Java, однак її універсальність надає можливість працювати на різних мовах. За допомогою даного середовища можна працювати на таких мовах програмування як Java, JavaScript та Python.

Сьогодні IntelliJ IDEA виділяється своїм гарним та інтуїтивним інтерфейсом, який дозволяє розібратися навіть початківцю. Однак, дане середовище вимагає багато оперативної пам'яті і зазвичай загальмовує роботу, особливо, якщо проєкт розробки досить великий, отже для роботи з IntelliJ IDEA потрібно використовувати потужний комп'ютер. Другий недолік – це складність розробки через її великий функціонал та орієнтованість на універсальність.

2.4.3 Середовище розробки Sublime Text

Sublime Text – це дуже зручний текстовий редактор коду. На перший погляд нагадує звичайний Notepad++, однак володіє великими можливостями. Особливістю Sublime є легкість середовища і можливість налаштувати його під свої потреби без значних витрат часу, зусиль та ресурсів комп'ютера. Варто відзначити, що незважаючи на всю легкість та простоту, Sublime Text містить в собі консоль та надає можливість працювати з дуже великою кількістю мов програмування.

Середовище доступне на всіх платформах. Sublime Text має вбудовану підтримку редагування Python-коду, а також багатий набір розширень, що розширюють можливості синтаксису і редагування.

Однак серед усіх переваг, варто відзначити, що середовище не є безкоштовним, хоча пробний період можна використовувати безкінечно. Встановлення розширень може перетворитися в доволі важку задачу.

2.4.4 Порівняння можливостей середовищ розробки

Відповідно до проведеного аналізу популярних середовищ розробки, що можуть бути використані для реалізації поставлених завдань, для більшої наочності, їх порівняння за основними критеріями зведено до таблиці 2.2.

В якості середовища розробки мовою Python обрано середовище Sublime Text 3 через її легкість роботи і можливість налаштувати середовище «під себе».

Таблиця 2.2 – Порівняння середовищ розробки

Критерії порівняння	Середовище розробки		
	Visual studio	IntelliJ IDEA	Sublime Text
Легкість та зрозумілість	-	-	+
Інтуїтивно зрозумілий інтерфейс	-	+	+
Можливість встановлення розширень	+	+	+
Безкоштовність	+	+	-
Гнучкість	±	±	+
Використання значних ресурсів комп'ютера	+	+	-
Кросплатформність	+	+	+
Можливість інтегрування з іншим ПЗ	+	+	-

2.5 Висновки за розділом

В розділі розглянуто інструменти розробки програмного забезпечення з використанням інтелектуальних методів та нейронних мереж. В якості мови

програмування обрано Python. Для реалізації нейронної мережі обрано бібліотеку Keras. Бібліотека Scikit-learn обрана для реалізації алгоритмів, заснованих на навчанні з учителем і без. Matplotlib – це бібліотека на мові програмування Python призначена для побудови наукових графіків та візуалізації даних. Для побудови графіків вирішено використовувати бібліотеку Matplotlib. Для роботи з масивами та математичними функціями високого рівня обрано пакет NumPy. В якості середовища розробки мовою Python обрано середовище Sublime Text 3, через його легкість та зрозумілість.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Оптимізація нейронних мереж за допомогою генетичного алгоритму

В теорії нейронних мереж існує дві актуальних проблеми, однією з яких є вибір оптимальної структури нейронної мережі (НМ), а іншою – побудова ефективного алгоритму навчання НМ [19], тобто розробка такого алгоритму, що дозволяє за мінімальний час налаштувати мережу для вирішення поставленого завдання.

Процес навчання нейронної мережі полягає в необхідності налаштування мережі таким чином, щоб для деякої множини входів видавати бажану множину виходів [19].

Розрізняють дві концепції навчання нейромереж: навчання з учителем та навчання без учителя [48].

Сьогодні існує багато різних методів навчання НМ [48], серед них:

- методи навчання, що засновані на правилі Хебба;
- дельта-правило;
- градієнтні методи;
- метод зворотного поширення помилки.

Основними недоліками вищевказаних методів є можливість передчасної зупинки через попадання до локального мінімуму, необхідність багаторазової всієї початкової вибірки, незадовільний час навчання НМ.

Для навчання нейронних мереж також можливо використовувати стохастичні методи. При використанні таких методів на початкових ітераціях виконують корекції значень вагових коефіцієнтів на великі величини, поступово зменшуючи їх у процесі пошуку [48].

До стохастичних методів відносять: методи випадкового пошуку, метод імітації відпалу, а також методи еволюційного пошуку. Значною

перевагою стохастичних методів є відсутність необхідності обчислення значень похідних цільової функції.

Еволюційний (генетичний) пошук включає групу багатовимірних, стохастичних, евристичних оптимізаційних методів, заснованих на ідеї еволюції за допомогою природного відбору. Методи генетичного пошуку отримані в процесі узагальнення та імітації в штучних системах таких властивостей живої природи, як природний відбір, пристосованість до умов середовища, спадкоємність життєво важливих властивостей від батьків та ін. [48].

На кожній ітерації генетичного пошуку метод працює не з єдиним рішенням, а з деякою множиною рішень, таким чином забезпечується паралельність обробки множини альтернативних рішень. При цьому кожна нова множина рішень залежить лише від попередньої та, в загальному випадку, є кращою.

Найбільш відомим представником еволюційних методів є генетичний алгоритм (ГА). Він полягає в паралельній обробці множини рішень, при цьому він фокусується на найбільш перспективних з них, що говорить про можливість використання ГА для вирішення широкого кола задач.

Для налаштування ваг нейронної мережі може бути використаний генетичний алгоритм. Для цього необхідно визначити цільову функцію (зазвичай це функція помилки) та спосіб подання ваг у хромосомі. Кожна хромосома $K = (k_1, k_2, \dots, k_i)$ – це вектор генів або, в даному випадку, вагових коефіцієнтів (ваги зчитуються з мережі за встановленим порядком – зліва направо та зверху вниз). Схематично хромосома виглядає як зображено на рисунку 3.1.

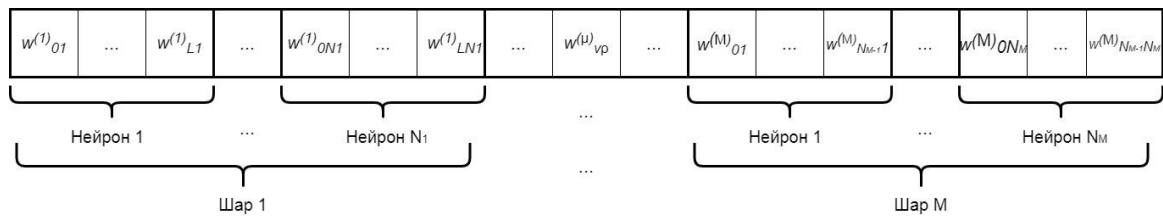


Рисунок 3.1 – Схематичне відображення хромосоми при налаштуванні ваг нейронної мережі [48]

Покоління – це набір хромосом (рішень).

Еволюція популяцій – це чергування поколінь, в яких хромосоми змінюють свої ознаки, таким чином, щоб кожна нова популяція найкращим чином пристосовувалася до зовнішнього середовища за допомогою обчислення фітнес-функції.

Для навчання мережі до початкової популяції застосовуються оператори селекції, схрещування та мутації, в результаті чого генеруються нові популяції. Правила селекції прагнуть залишити лише ті рішення, де досягається оптимуму цільової функції. Найбільш пристосовані особини мають можливість відтворювати нащадків за допомогою схрещування, що призводить до появи нових осіб. Після схрещування може відбуватися мутації – випадкові зміни у генах. Далі робота генетичного алгоритму – це ітераційний процес, який зупиняється коли досягається оптимальне значення цільової функції або закінчується кількість епох.

В загальному випадку процес оптимізації нейронної мережі із застосуванням генетичного алгоритму зображено на рисунку 3.2.

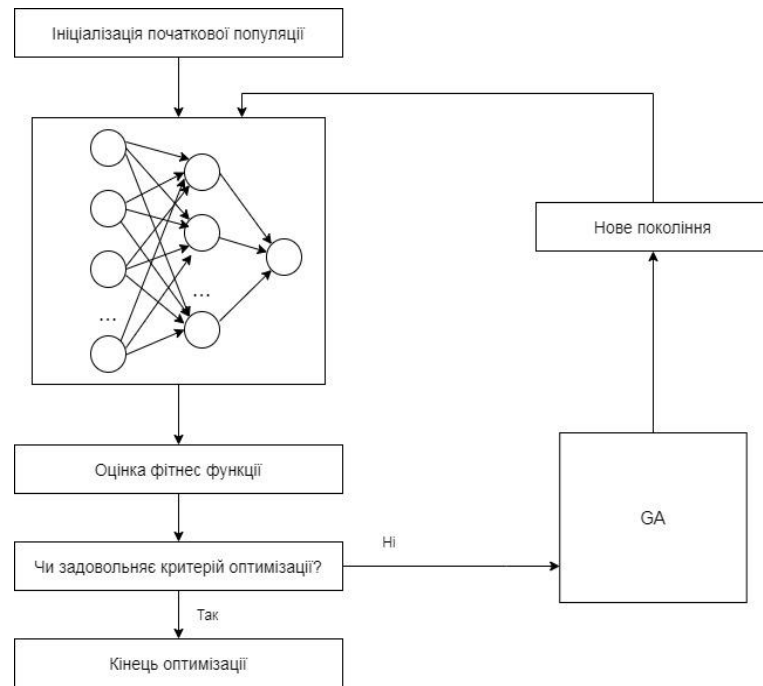


Рисунок 3.2 – Оптимізація нейронної мережі за допомогою генетичного алгоритму

3.2 Основні розроблені алгоритми

Генетичний алгоритм (ГА) відноситься до алгоритмів еволюційних обчислень [48]. ГА є ітераційним, і дає наближене рішення, що компенсується областю його застосування і швидкістю обчислень. Він може бути застосований для налаштування ваг нейронної мережі. Стандартний генетичний алгоритм складається з таких кроків:

- ініціалізація початкового покоління;
- оцінка пристосованості покоління;
- закінчення виконання алгоритму у випадку задоволення критерію зупинки;
- генерація нового покоління за допомогою виконання генетичних операторів.

В загальному випадку ГА зводиться до циклічної заміни одного покоління на більш пристосоване. Генерація нового покоління відбувається за допомогою виконання генетичних операторів, тож при правильному їх

застосуванні можна розраховувати на гарні рішення за менший час роботи алгоритму. Схема роботи генетичного алгоритму зображена на рисунку 3.3.

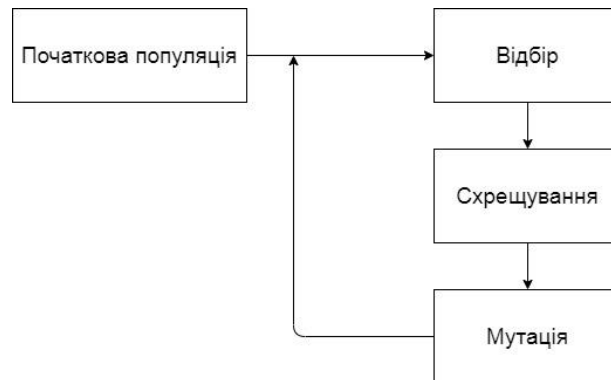


Рисунок 3.3 – Схема роботи генетичного алгоритму

В генетичному алгоритмі за формування батьківських пар відповідає оператор схрещування. Часто для цього обираються найпристосованіші особи після чого до пари осіб застосовують кросовер, який в заданій точці здійснює обмін генетичним матеріалом обраних осіб.

Однією з проблем, які виникають під час використання генетичних методів, є визначення не глобального, а локального екстремуму функції, що оптимізується. Для рішення цієї проблеми існує багато модифікацій генетичного методу. В даному проєкті запропоновано обирати батьківські пари на основі їх віддаленої спорідненості.

В природі існує два типи схрещування: споріднений та неспоріднений. Споріднене схрещування – це вид схрещування при, якому особи мають близьку ступінь спорідненості. Неспоріднене схрещування відбувається між особами, що не мають спорідненості або їх спорідненість низька.

Аналіз численних матеріалів показує, що ступінь спорідненості має великий вплив на нащадків. Якщо брати приклад з природи, то близька спорідненість супроводжується низькою життєздатністю потомства, має вплив на його пристосованість. При схрещуванні неспоріднених осіб таких проблем не виникає оскільки шкідливі рецесивні мутації, що знаходяться в

гомозиготному стані, перейдуть в гетерозиготний стан і не будуть впливати на життєздатність гібридного організму [49].

Таким чином, на етапі вибору осіб для формування батьківських пар, є сенс враховувати не лише показник пристосованості осіб, а й їх спорідненість між собою так, щоб пари формувалися з максимально віддалених осіб. Внаслідок цього алгоритм буде переглядати недосліджені області і попереджати збіжність до вже знайденого рішення.

Ступінь спорідненості можна виразити за допомогою формули (3.1):

$$d_{ij} = \sum_{k=1}^p |x_{ik} - x_{jk}|, \quad (3.1)$$

де x_{ik} – k -ий ген в хромосомі i -го батька;

x_{jk} – k -й ген в хромосомі j -го батька;

p – довжина хромосоми.

Довжина хромосоми відповідно до поставленої задачі оптимізації нейронної мережі вираховується за формулою (3.2):

$$p = N_1(L+1) + \sum_{\mu=1}^M N_{\mu}(N_{\mu-1}+1), \quad (3.2)$$

де N_{μ} – кількість нейронів на μ -ому шарі;

L – кількість ознак у навчаючій вибірці;

M – кількість шарів нейромережі.

Також в роботі пропонується дві модифікації оператора мутації. Наразі існують різні схеми виконання оператора мутації, що полягають імовірнісній мутації всього покоління шляхом перестановки генів місцями або зміні їх значень. В першому запропонованому методі мутації підлягають 20% популяції. Далі випадковим чином з хромосоми обирається два гени та вираховується середнє арифметичне цих значень (3.3).

$$H_A = \frac{h_{ij} + h_{ik}}{2}, \quad (3.3)$$

де h_{ij} – j -й ген i -ої хромосоми;

h_{ik} – k -й ген i -ої хромосоми.

Щоб у хромосому не поверталось два гени з однаковими значеннями, то пропонується змінювати їх значення шляхом додавання та віднімання певного числа із заданого діапазону значень (3.4, 3.5).

$$h_{ij}^* = H_A + rand(0,1), \quad (3.4)$$

$$h_{ik}^* = H_A - rand(0,1), \quad (3.5)$$

де h_{ij}^* , h_{ik}^* - значення j -го та k -го гена i -ї хромосоми в результаті мутації;

$rand(0,1)$ – випадкове значення від нуля до одиниці.

В якості другої модифікації оператора мутації пропонується обчислення середнього арифметичного значень двох випадково обраних генів (3.3), але на відміну від першої модифікації, в цій як значення одного гена йтиме значення середнього арифметичного, а як значення другого гена ставимо нуль.

$$h_{ij}^* = H_A + rand(0,1), \quad (3.6)$$

$$h_{ik}^* = 0, \quad (3.7)$$

де h_{ij}^* , h_{ik}^* - значення j -го та k -го гена i -ї хромосоми в результаті мутації;

$rand(0,1)$ – випадкове значення від нуля до одиниці.

Таким чином, з використанням запропонованих операторів, сенс оператора мутації зберігається, тобто виконується підвищення

різноманітності пошуку і введення нових хромосом в популяцію для більш повного дослідження простору пошуку.

3.3 Використовувані моделі даних

Дані про стан здоров'я пацієнта зберігаються та завантажуються з файлу `diabetes.csv`. У цьому файлі кожен рядок – це дані одного пацієнта, значення параметрів розділені між собою комами. В наборі даних є дев'ять параметрів, останній з яких – цільовий. Дані зберігаються у вигляді наведеному в таблиці 3.2, де:

- `preg` – кількість вагітностей;
- `plas` – концентрація глюкози в плазмі через 2 години після введення в пероральному глюкозотолерантному тесті;
- `pres` – діастолічний артеріальний тиск (мм рт.с.);
- `skin` – товщина шкірної складки в районі трицепса (мм);
- `insu` – концентрація інсуліну в сироватці крові (мкЕд/мл);
- `mass` – індекс маси тіла;
- `pedi` – генетична схильність до діабету;
- `age` – вік пацієнта;
- `class` – цільовий параметр, що характеризує наявність чи відсутність діабету.

Таблиця 3.2 – Спосіб збереження даних про стан здоров'я пацієнта

№ п/п	<code>preg</code>	<code>plas</code>	<code>pres</code>	<code>skin</code>	<code>Insu</code>	<code>mass</code>	<code>pedi</code>	<code>age</code>	<code>class</code>
1	6	148	72	35	0	33.6	0.627	50	1
2	1	85	66	29	0	26.6	0.351	31	0
3	8	183	64	0	0	23.3	0.672	32	1
4	1	89	66	23	94	28.1	0.167	21	0
5	0	137	40	35	168	43.1	2.288	33	1

3.4 Опис концепції програмної моделі в проєктованій системі

Для побудови моделі прогнозування ризику виникнення цукрового діабету в якості типу штучної нейронної мережі обрано багат шаровий персептрон. Вибір даної нейронної мережі виконано на основі вирішуваної задачі, а саме класифікації даних для визначення наявності ризику виникнення цукрового діабету.

В якості вхідних даних використано датасет, що складається показників здоров'я людини PIMA [5].

Розроблена нейронна модель має щільну структуру, в якій кожен нейрон зв'язаний з усіма нейронами наступного шару. Нейронна мережа складається зі вхідного шару, двох прихованих шарів та одного вихідного нейрона, який визначає кінцевий клас.

Зовнішній шар має 8 нейронів, відповідно до кількості вхідних параметрів, другий шар мережі має 14 нейронів, третій – 7 і вихідний шар має один нейрон.

В якості функції активації використано сигмоїдальну функцію (3.8), яка фактично видає значення ймовірності ризику захворювання:

$$f(x) = \frac{1}{1 + e^x}. \quad (3.8)$$

Для первинної ініціалізації синаптичних ваг використано нормальний розподіл (середнє значення дорівнює 0, а стандартне відхилення дорівнює 0,05).

У якості алгоритму оптимізації при навчанні мережі використано генетичний алгоритм з модифікованими операторами відбору та мутації.

Рішення задачі оцінюють за допомогою обчислення значень цільової функції. Вигляд цільової функції визначається характером розв'язуваної задачі, для якої використовується генетичний алгоритм. В даному випадку в

якості цільової функції взята функція (3.9), яка вираховує кількість правильних відповідей нейронної мережі:

$$f = \sum_{i=1}^N \sum_{j=1}^m D(y_{ij} - y_{ij}^t), \quad (3.9)$$

де m – кількість виходів нейронної мережі;

N – кількість зразків в навчальній вибірці;

y_{ij} – дійсне значення на j -му виході мережі для образу i ;

y_{ij}^t – значення, яке повинно бути на j -му виході мережі для образу i ;

D – значення, яке залежить від $|y - y^t| < d$, якщо нерівність вірна, то $D=1$, якщо ні, то $D=0$;

d – величина, яка визначає максимально допустиме відхилення реального значення від цільового.

В результаті обчислення фітнес-функції визначається чи буде продовжено виконання ГА. У випадку продовження виконання ГА виконуються послідовно операції відбору, схрещування та мутації.

В якості оператора відбору особин приймається модифікований спосіб вибору особин до батьківських пар, що враховує неспорідненість їх між собою таким чином, щоб пари формувалися з максимально віддалених осіб. Ступінь спорідненості можна виразити за допомогою формули (3.10):

$$d_{ij} = \sum_{k=1}^p |x_{ik} - x_{jk}|, \quad (3.10)$$

де x_{ik} – k -й ген в хромосомі i -го батька;

x_{jk} – k -й ген в хромосомі j -го батька;

p – довжина хромосоми.

Кросовер – це генетичний оператор схрещування, який здійснює обмін генами або частинами хромосом між обраними батьками з метою отримання нових хромосом. Даний оператор є настільки важливим оператором, що якщо його виключити із структури алгоритму, то генетичний пошук втрачає властивості, притаманні репродуктивно-популяційним алгоритмам.

В якості оператора схрещування приймається одноточковий кросовер, суть якого полягає в наступному.

1. Відповідно до обраної схеми схрещування формується «батьківська пара» із допустимих батьківських кодувань $x' = (x'_1, \dots, x'_L)$ та $x'' = (x''_1, \dots, x''_L)$.

2. Випадковим чином з рівною імовірністю $\frac{1}{L-1}$ обирається точка розриву $r \in \{1, 2, \dots, L-1\}$.

3. Кодування x' та x'' розриваються в одній і тій же точці r .

4. Із отриманих чотирьох частин батьківських кодувань

$$(x'_1, \dots, x'_r), (x'_{r+1}, \dots, x'_L), (x''_1, \dots, x''_r), (x''_{r+1}, \dots, x''_L)$$

формуються кодування двох нащадків

$$x^1 = (x'_1, \dots, x'_r), (x''_{r+1}, \dots, x''_L),$$

$$x^2 = (x''_1, \dots, x''_r), (x'_{r+1}, \dots, x'_L).$$

Схематично описані дії зображено на рисунку 3.4.

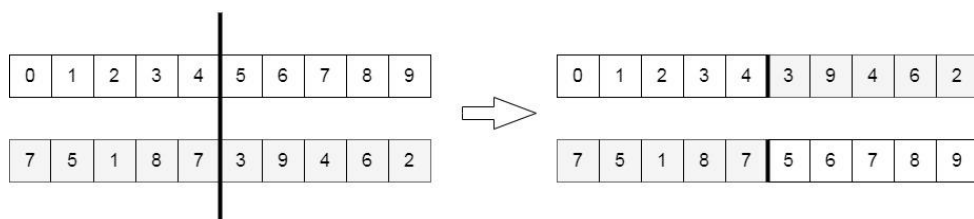


Рисунок 3.4 – Схема роботи одно точкового кросовера

Оператор мутації виконує важливу роль у виконанні алгоритму, він урізноманітнює пошук та вводить нові хромосоми у популяцію для більш повного дослідження простору пошуку. В даній роботі пропонується два оператори мутації, що відрізняються від класичного тим, що мутація здійснюється не лише для певного відсотка особин, а й певним чином змінюються значення генів у хромосомах.

Перша модифікація оператора мутації полягає у тому, що мутації підлягають 20% особин популяції. Далі, у обраних особин, відбувається зміна значень генів, що визначаються випадковим чином за формулою (3.11). Тобто, обчислюється середня арифметичне значень обраних генів та, щоб хромосома не містила двох генів з однаковими значеннями, їх значення змінюються шляхом додавання та віднімання певного числа із заданого діапазону.

$$H_A = \frac{h_{ij} + h_{ik}}{2}, \quad (3.11)$$

де h_{ij} – j -й ген i -ої хромосоми;

h_{ik} – k -й ген i -ої хромосоми.

В якості другої модифікації оператора мутації пропонується обчислення середнього арифметичного значень двох випадково обраних генів (3.11), але на відміну від першої модифікації, в цій як значення одного гена йтиме значення середнього арифметичного, а як значення другого гена виконуємо так звану white-мутацію, тобто обнуляємо значення.

Після виконання еволюційних операторів оцінюється розмір популяції, після чого її розмір повертається до вихідного. В нову популяцію потрапляють найбільш пристосовані особи.

Окрім цього, також реалізована оптимізація нейронної мережі методом рою часток.

Метод рою часток [50] (Particle Swarm Optimization, PSO) являє собою методику стохастичної оптимізації, яка бере ідею у соціальній поведінці зграї птахів або риб.

PSO має багато спільного з генетичними алгоритмами [50]. Система ініціалізується за допомогою популяції випадкових рішень та виконує пошук оптимумів шляхом оновлення поколінь цих рішень. Проте, на відміну від генетичних алгоритмів, PSO не має таких операторів еволюції, як кросовер та мутація. У PSO потенційні рішення, що називаються частками, пролітають у просторі рішень, орієнтуючись на частинки, що є оптимальними в кожний момент часу.

У кожній точці, де побувала частка, розраховується значення цільової функції. При цьому кожна частка запам'ятовує, яке (і де) краще значення цільової функції вона особисто знайшла, а також кожна частка знає де розташована точка, яка є кращою серед усіх точок, які розвідали частки. На кожній ітерації частки коректують свою швидкість (модуль і напрямок), щоб з одного боку бути ближче до кращої точки, яку частка знайшла сама (автори алгоритму назвали цей аспект поведінки "ностальгією"), і, в той же час, наблизитися до точки, яка в даний момент є глобально кращою. Через деяку кількість ітерацій частки повинні зібратися поблизу найбільш вдалої точки, хоча можливо, що частина частинок залишиться десь у відносно непоганому локальному екстремуму, але головне, щоб хоча б одна частка виявилася поблизу глобального екстремуму.

У початковому вигляді алгоритму корекція швидкості виглядала наступним чином:

$$v_{i,t+1} = v_{i,t} + \varphi_p r_p (p_i - x_{i,t}) + \varphi_g r_g (g_i - x_{i,t}), \quad (3.12)$$

де $v_{i,t}$ – i -та компонента швидкості при t -ій ітерації алгоритму;

$x_{i,t}$ – i -та координата частки при t -ій ітерації алгоритму;

p_i – i -та координата кращого рішення, знайденого часткою;

g_i – i -та координата найкращого рішення, знайденого всіма частками;

r_p і r_g – випадкові числа в інтервалі $(0, 1)$;

φ_p і φ_g – вагові коефіцієнти, що необхідно підібрати під конкретну задачу.

Після цього коректуємо координату кожної частки:

$$x_{i,t+1} = x_{i,t} + v_{i,t+1}. \quad (3.13)$$

Після цього обчислюється значення цільової функції у кожній новій точці, кожна частка перевіряє чи не стала нова координата найкращою серед всіх точок, які вона відвідала. Потім серед усіх нових точок перевіряється чи не знайдено глобально найкращу точку і, якщо знайдено, запам'ятовуються її координати і значення цільової функції в цій точці.

Узагальнена схема алгоритму наведено на рисунку 3.5:

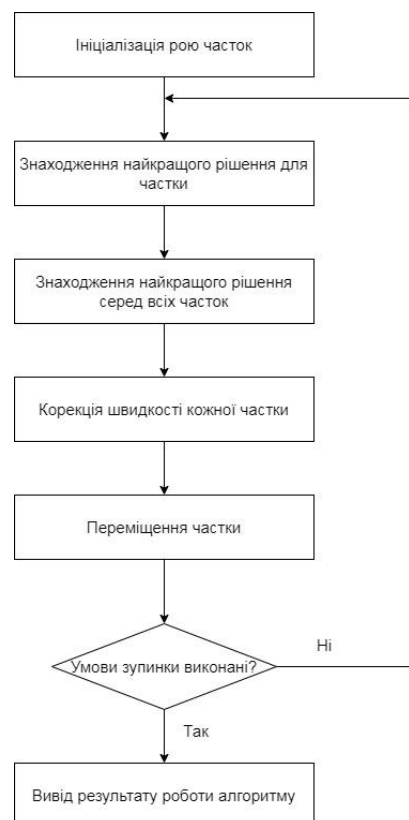


Рисунок 3.5 – Схема алгоритму рою часток [50]

У порівнянні з генетичними алгоритмами, перевагами PSO є легкість реалізації і невелика кількість параметрів, які треба встановлювати. PSO вдало застосовуються в багатьох областях таких, як оптимізація функцій, тренування штучних нейронних мереж, управління нечіткою системою та інші, де можна застосувати генетичні алгоритми [48].

В загальному вигляді оптимізації нейронної мережі виглядає як зображено на рисунку 3.6.

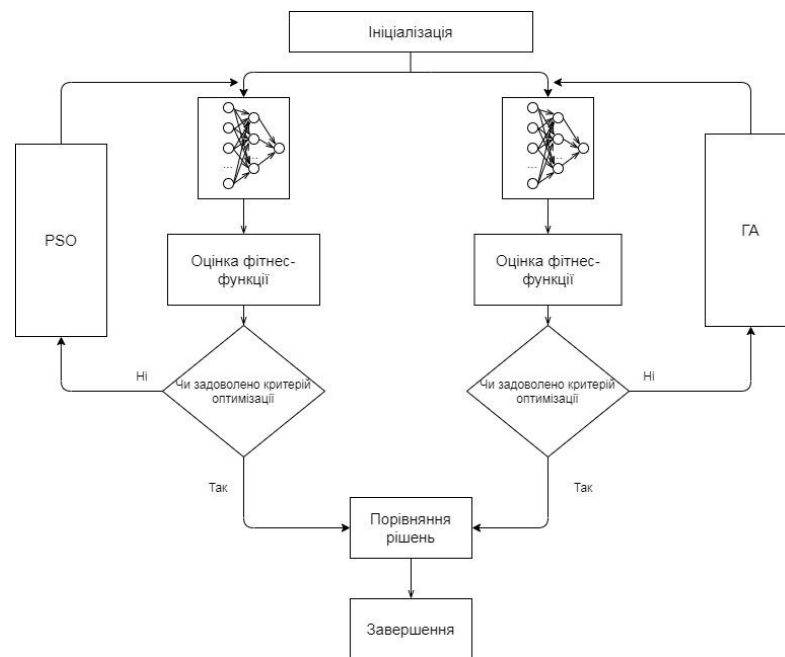


Рисунок 3.6 – Оптимізація нейронної мережі за допомогою ГА і PSO

3.5 Висновки за розділом 3

В результаті виконання розділу розглянуто спосіб оптимізації нейронних мереж за допомогою використання генетичного алгоритму та наведено схему функціонування даної оптимізації. Розроблено та описано модифікації генетичного алгоритму, а саме спосіб вибору осіб до батьківських пар за рахунок обчислення їх неспорідненості та двох операторів мутації. Визначено та описано модель даних, на яких тренується модель та описано схему її функціонування.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення та умови застосування програми

4.1.1 Призначення програми

Розроблений програмний продукт відповідає реалізовано відповідно до поставленого завдання та задовольняє всім необхідним вимогам.

Програмний продукт призначений для визначення ступеню ризику виникнення цукрового діабету у людини відповідно до факторів, що впливають на розвиток захворювання. Може бути використаний лікарями-ендокринологами для діагностування пацієнта та постановки діагнозу цукровий діабет.

4.1.2 Умови застосування програми

Зазначене програмне забезпечення написане мовою Python 3.7. Розробка і тестування проводилися на обладнанні з такими конфігураціями:

- процесор Intel Core i3 – 4005U, 1.7 GHz;
- 4 GB оперативної пам'яті;
- Операційна система Windows 10 (64);

Розроблене програмне забезпечення може бути використано на операційній системі сімейств Windows і Unix. Враховуючи використання в розробленому програмному пакеті компонентів, які розповсюджуються під ліцензією GPL та існуючі обмеження, що виникають за таких обставин, розроблене програмне забезпечення, також може використовуватися (цілком чи частково), змінюватися та розповсюджуватися ("як є") за умов ліцензії GPL 3. Компоненти програмного забезпечення мають низку залежностей, у разі недотримання яких робота пакету може бути некоректною та нестабільною. До таких основних залежностей належать:

- мова програмування Python 3.7;

- нейромережева бібліотека Keras;
- бібліотека машинного навчання Scikit-learn;
- бібліотека для побудови графіків та візуалізації даних Matplotlib;
- бібліотека, що дозволяє ефективно працювати з матрицями, таблицями та числовими даними NumPy.

4.2 Звернення до програми

Для запуску програми необхідно упевнитися у відповідності характеристик ЕОМ, на яку встановлюється програма, до мінімальних системних вимог, упевнитися у справності всіх компонентів ЕОМ та електронного носія, з якого здійснюватиметься запуск програми. Впевнитися, що всі необхідні залежності попередньо встановлені на ЕОМ.

Програмний продукт має наступну файлову структуру (рис. 4.1).

Папка `dataset` містить набір тренувального набору даних у двох форматах `.arff` та `.csv`, що використовувались під час тренування моделей.

Папка `model` призначена для зберігання файлів моделі та складається папок `main` та `optimizer`.

Папка `results` призначена для зберігання результатів роботи програми, а саме графіків.

`Main` – це папка, що містить файл `ANN.py`, який містить скрипт реалізації штучної нейронної мережі.

`Optimizer` – це папка, що містить у собі скрипти, що призначені для параметричної оптимізації нейронної мережі, а саме:

- `GA_firts_modified_mutation.py` – скрипт з реалізацією генетичного алгоритму з запропонованим першим оператором мутації та селекції;
- `GA_second_modified_mutation.py` – скрипт з реалізацією генетичного алгоритму з запропонованим другим оператором мутації та селекції;

- GA_standart.py – скрипт з реалізацією стандартного генетичного алгоритму;
- PSO.py – скрипт з реалізацією методу рою часток.

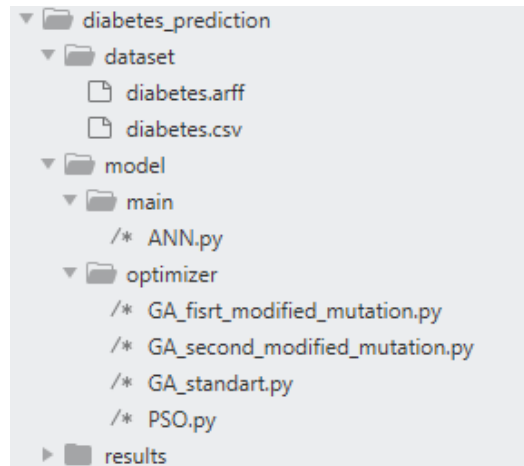


Рисунок 4.1 – Файлова структура

4.3 Вхідні та вихідні дані

В якості вхідних даних використано датасет, який містить дані про стан здоров'я пацієнтів і, які зберігаються та завантажуються з файлу `diabetes.csv`. У цьому файлі кожен рядок – це дані одного пацієнта, значення параметрів розділені між собою комами.

В наборі даних є дев'ять параметрів, останній з яких – цільовий:

- `preg` – кількість вагітностей;
- `plas` – концентрація глюкози в плазмі через 2 години після введення в пероральному глюкозотолерантному тесті;
- `pres` – діастолічний артеріальний тиск (мм рт.с.);
- `skin` – товщина шкірної складки в районі трицепса (мм);
- `insu` – концентрація інсуліну в сироватці крові (мкЕд/мл);
- `mass` – індекс маси тіла;
- `pedi` – генетична схильність до діабету;
- `age` – вік пацієнта;

– class – цільовий параметр, що характеризує наявність чи відсутність діабету.

Результат роботи моделі виводиться у консоль та має значення у відсотках. Приклад виводу результату зображено на рисунку 4.2.

```

10/768 [.....] - ETA: 0s - loss: 0.2661 - accuracy: 0.9000
260/768 [=====>.....] - ETA: 0s - loss: 0.3150 - accuracy: 0.8577
510/768 [=====>.....] - ETA: 0s - loss: 0.3057 - accuracy: 0.8725
760/768 [=====>.....] - ETA: 0s - loss: 0.2746 - accuracy: 0.8855
768/768 [=====] - 0s 201us/step - loss: 0.2779 - accuracy: 0.8828

32/768 [>.....] - ETA: 2s
768/768 [=====] - 0s 190us/step

accuracy: 88.67%

```

Рисунок 4.2 – Приклад виводу результату роботи моделі у консолі

Також результат виводиться у графічному вигляді (рис. 4.3). на рисунку 4.3 зображено графічне представлення результату тренування повнозв'язного перцептрона, а саме графік функції втрат.

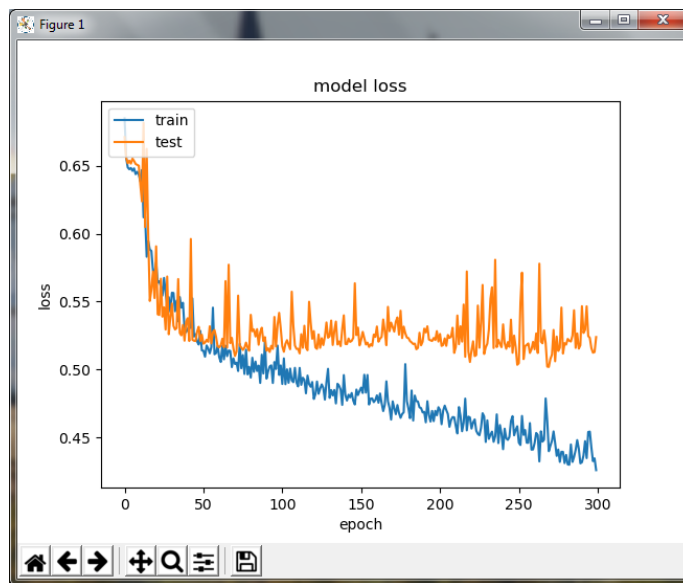


Рисунок 4.3 – Графічне представлення результату тренування повнозв'язного перцептрона

4.4 Повідомлення програмісту

Можливі повідомлення можуть виникати, коли встановлені залежності є некоректними (невірні версії бібліотек або мови програмування Python) або, коли залежності встановлені не в повній мірі (рис. 4.4-4.6).

На рисунку 4.4 наведено можливе повідомлення програмісту, відповідно до якого бібліотека tensorflow динамічно шукає правильну версію CUDA, і якщо не знаходить, то видає попередження і повертається в режим лише ЦП. Фактично, це також записується в журнал як інформаційне повідомлення відразу після попередження. Якщо на вашому комп'ютері немає графічного процесора з підтримкою CUDA або вас не хвилює відсутність прискорення графічного процесора, то можна нічого не робити. Однак, якщо прискорення графічного процесора є необхідним, то потрібно перевірити наявність встановленого CUDA: tensorflow 2.2 вимагає CUDA 10.1.

```
2020-11-18 13:33:48.326131: W tensorflow/stream_executor/platform/
default/dso_loader.cc:55] Could not load dynamic library '
cudart64_101.dll'; dLError: cudart64_101.dll not found
2020-11-18 13:33:48.327131: I tensorflow/stream_executor/cuda/
cudart_stub.cc:29] Ignore above cudart dlerror if you do not have
a GPU set up on your machine.
```

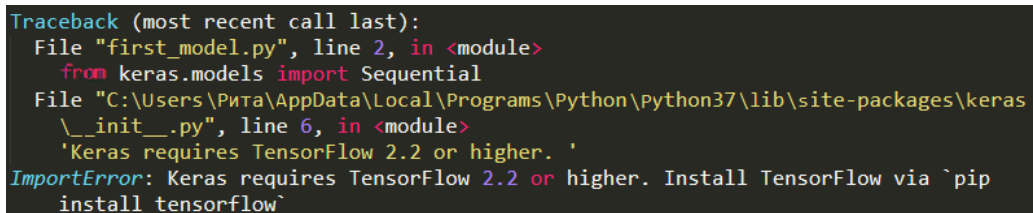
Рисунок 4.4 – Приклад повідомлення, що може отримати програміст

На рисунку 4.5 відображено повідомлення програмісту, яке повідомляє про відсутність бібліотеки Keras, якого вимагає програма. Вирішенням цієї проблеми є встановлення бібліотеки Keras. Встановити бібліотеку можна шляхом введення в терміналі команди `pip3 install keras`.

```
Traceback (most recent call last):
  File "first_model.py", line 2, in <module>
    from keras.models import Sequential
ModuleNotFoundError: No module named 'keras'
```

Рисунок 4.5 – Приклад повідомлення, що може отримати програміст

На рисунку 4.6 зображено повідомлення програмісту про невідповідність версій бібліотек Keras та tensorflow, а саме про те, що бібліотека Keras вимагає, щоб бібліотека tensorflow мала версію 2.2 або вище. Вирішити дану проблему можна встановленням бібліотеки tensorflow 2.2 встановивши за допомогою команди `pip3 install tensorflow==2.2.0` або оновити версію бібліотеки за допомогою команди `pip3 install --upgrade tensorflow`.



```
Traceback (most recent call last):
  File "first_model.py", line 2, in <module>
    from keras.models import Sequential
  File "C:\Users\Рита\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\__init__.py", line 6, in <module>
    'Keras requires TensorFlow 2.2 or higher. '
ImportError: Keras requires TensorFlow 2.2 or higher. Install TensorFlow via `pip install tensorflow`
```

Рисунок 4.6 – Приклад повідомлення, що може отримати програміст

Під час роботи додатку усі результати виконання окремих операцій будуть відображені у консолі. Для більш детальної інформації, а також під час налаштування додатку рекомендовано отримувати інформацію про роботу додатку через консоль Python. Отримана таким чином інформація є більш повною і може виявитись корисною для спеціаліста.

4.5 Висновки за розділом

У розділі наданий детальний опис для роботи з розробленим програмним продуктом. Надані параметри програмних та апаратних складових. Описані необхідні програмні залежності, попереднього встановлення яких потребує програмний продукт.

У розділі надано детальний опис методів розгортання програмного продукту, опис структурної схеми, вхідних та вихідних даних, а також можливих повідомлень під час роботи з ним.

5 КЕРІВНИЦТВО КОРИСТУВАЧА

5.1 Призначення та умови застосування програми

5.1.1 Призначення програми

Розроблений програмний продукт відповідає реалізовано відповідно до поставленого завдання та задовольняє всім необхідним вимогам.

Програмний продукт призначений для визначення ступеню ризику виникнення цукрового діабету у людини відповідно до факторів, що впливають на розвиток захворювання. Може бути використаний лікарями-ендокринологами для діагностування пацієнта та постановки діагнозу цукровий діабет.

5.1.2 Умови застосування програми

Зазначене програмне забезпечення написане мовою Python 3.7. Розробка і тестування проводилися на обладнанні з такими конфігураціями:

- процесор Intel Core i3 – 4005U, 1.7 GHz;
- 4 GB оперативної пам'яті;
- Операційна система Windows 10 (64);

Розроблене програмне забезпечення може бути використано на операційній системі сімейств Windows і Unix. Враховуючи використання в розробленому програмному пакеті компонентів, які розповсюджуються під ліцензією GPL та існуючі обмеження, що виникають за таких обставин, розроблене програмне забезпечення, також може використовуватися (цілком чи частково), змінюватися та розповсюджуватися ("як є") за умов ліцензії GPL 3. Компоненти програмного забезпечення мають низку залежностей, у разі недотримання яких робота пакету може бути некоректною та нестабільною. До таких основних залежностей належать:

- мова програмування Python 3.7;

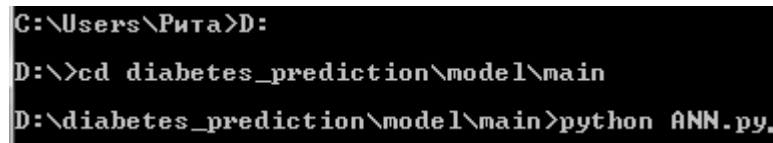
- нейромережева бібліотека Keras;
- бібліотека машинного навчання Scikit-learn;
- бібліотека для побудови графіків та візуалізації даних Matplotlib;
- бібліотека, що дозволяє ефективно працювати з матрицями, таблицями та числовими даними NumPy.

5.2 Виконання програми

5.2.1 Екран входу

Для запуску програми необхідно упевнитися у відповідності характеристик ЕОМ, на яку встановлюється програма, до мінімальних системних вимог, упевнитися у справності всіх компонентів ЕОМ та електронного носія, з якого здійснюватиметься запуск програми. Впевнитися, що всі необхідні залежності попередньо встановлені на ЕОМ.

Для того, щоб запустити програму та виконати діагностування необхідно відкрити інтерпретатор командного рядка, у випадку операційної системи Windows – це cmd.exe та прописати шлях до необхідного файлу, а саме ANN.py, як зображено на рисунку 5.1.



```
C:\Users\Pyta>D:
D:\>cd diabetes_prediction\model\main
D:\diabetes_prediction\model\main>python ANN.py
```

Рисунок 5.1 – Запуск програми через командний рядок

5.2.2 Екран роботи алгоритму

Якщо виконані всі вимоги до програмного продукту, йде виконання програми, яке супроводжується виводом певної інформації (рис. 5.2-5.3). В командному рядку виводиться інформація про кількість епох, значення функції втрат, точність, значення ROC-кривої.

```

Epoch 192/200
1/73 [.....] - ETA: 0s - loss: 0.2362 - accuracy: 1.00
46/73 [=====>.....] - ETA: 0s - loss: 0.4224 - accuracy: 0.80
73/73 [=====] - 0s 1ms/step - loss: 0.4428 - accuracy:
0.7846 - val_loss: 0.6742 - val_accuracy: 0.6154
Epoch 193/200
1/73 [.....] - ETA: 0s - loss: 0.2690 - accuracy: 0.90
45/73 [=====>.....] - ETA: 0s - loss: 0.4457 - accuracy: 0.79
73/73 [=====] - 0s 1ms/step - loss: 0.4435 - accuracy:
0.7915 - val_loss: 0.7742 - val_accuracy: 0.6154

```

Рисунок 5.2 – Виконання програми

```

AUC: 0.843
1/24 [>.....] - ETA: 0s - loss: 0.4314 - accuracy: 0.78
24/24 [=====] - 0s 1ms/step - loss: 0.3758 - accuracy:
0.8830
accuracy: 88.30%

```

Рисунок 5.3 – Дані про результат виконання програми

Також виконання програми супроводжується виводом графічної інформації, яка може виглядати наступним чином (рис. 5.4-5.6).

Графік на рисунку 5.4 відображає точність, яку має повнозв'язний перспетрон відповідно до кількості епох навчання. Відповідно до графіка кількість епохо навчання становить 200, а точність яку має повнозв'язний перспетрон на 200 епосі складає 90.82%.

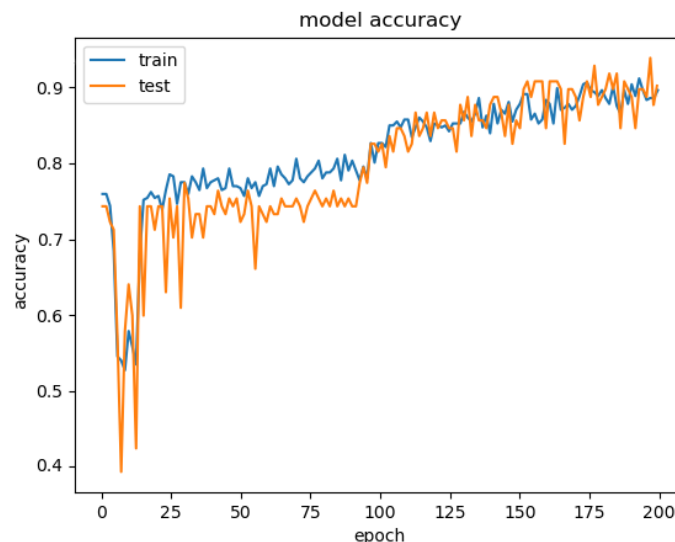


Рисунок 5.4 – Графік точності повнозв'язного перспетрона

На рисунку 5.5 відображено графік функції втрат протягом навчання повнозв'язного перспетрона. З графіка видно, що значення функції помилки

поступово знижується та на 200 епосі значення функції втрат складає 0,489 на тестових даних.

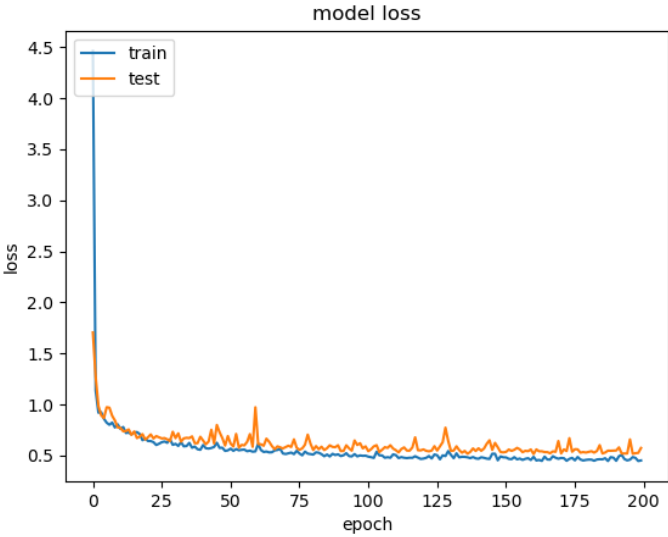


Рисунок 5.5 – Графік функції втрат повнозв'язного персептрона

Графік ROC-кривої зображено на рис. 5.6 та ілюструє чутливість класифікатору, показуючи скільки правильно класифікованих об'єктів можна отримати, дозволяючи більше випадків FP. Відповідне значення ROC-кривої склало 0,893.

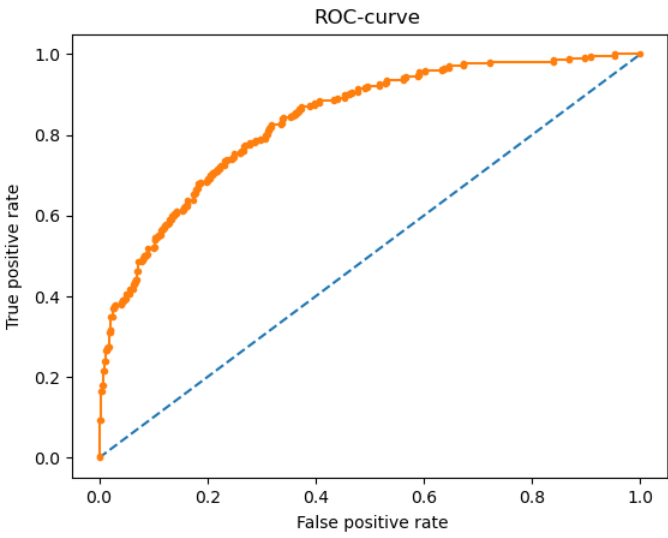


Рисунок 5.6 – Графік ROC-кривої

5.3 Повідомлення оператору

Повідомлення, що може видавати програма користувачеві, можуть полягати у невірних встановлених залежностях або не у повному їх обсязі. Також це інформація та графічне відображення виконання програми.

5.4 Висновки за розділом

Розділ містить опис роботи оператора з розробленим програмним продуктом. Надані параметри програмних та апаратних складових. Описані необхідні програмні залежності, попереднього встановлення яких потребує програмний продукт.

У розділі надано опис способу роботи з програмним продуктом, як виконується робота з ним, які дані та у якому вигляді отримує користувач під час роботи. Наведено приклади виводу графічної інформації.

6 ЕКСПЕРИМЕНТИ ТА РЕЗУЛЬТАТИ

6.1 Проведення експериментів

Після етапу проєктування та побудови моделі, наступним кроком є оцінювання ефективності моделі за допомогою спеціалізованих метрик та тестового набору даних. Метрики якості обираються залежно від типу задачі і мають безпосередній вплив на оцінювання алгоритмів та порівняння їх між собою.

Оцінка якості роботи класифікаторів базується на понятті матриці помилок, що для задачі бінарної класифікації має вигляд наведений в таблиці 6.1.

Таблиця 6.1 – Матриця помилок

	y=1	y=0
x=1	True Positive (TP)	False Positive (FP)
x=0	False Negative (FN)	True Negative (TN)

В даному випадку, оскільки розглядається задача медичної діагностики, $y=1$ означає наявність хвороби, а $y=0$ – її відсутність.

True Positive – випадок, коли реальне значення дорівнювало 1 ($x=1$) і відповідь алгоритму також 1 ($y=1$).

True Negative – у здорового пацієнта алгоритм діагностував відсутність хвороби.

False Positive – випадок, коли реальне значення дорівнювало нулю ($x=0$), а прогнозоване – одиниці ($y=1$).

False Negative – випадок, коли алгоритм визнав хвору людину здоровою.

Звідси виходить, що в ідеальному випадки модель повинна видавати нуль помилок класів False Positive та False Negative, але насправді така ситуація практично нереальна, тому варто мінімізувати кількість можливих помилок.

Тестування, як і навчання мережі проводилося на наборі даних, що містять інформацію щодо захворювання на діабет. Джерелом інформації про захворюваність є Національний інститут діабету, захворювань органів травлення та нирок (США) [5]. Набір даних формувався від жінок 21 року і більше народності Піма. Набір складається з 9 атрибутів та 768 екземплярів:

- а) кількість вагітностей;
- б) концентрація глюкози в плазмі через 2 години після введення в пероральному глюкозотолерантному тесті;
- в) діастолічний артеріальний тиск (мм рт.с.);
- г) товщина шкірної складки в районі трицепса (мм);
- д) концентрація інсуліну в сироватці крові (мкЕд/мл);
- е) індекс маси тіла (вага в кг/(зріст в м)²);
- ж) функція, що описує генетичну схильність до діабету (diabetes pedigree);
- з) вік (роки);
- и) клас, що показує чи був у пацієнта ЦД.

Створена мережа проходила навчання в продовж 200 епох. При тестуванні було використано спосіб поділу вибірки даних на навчальну та тестову вибірки в відсотковому відношенні 80/20%.

Правильність класифікації визначається як співвідношення між вірними відповідями та загальною кількістю відповідей (6.1):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \cdot \quad (6.1)$$

Як функція втрат використовується бінарна кросентропія (6.2). Бінарна кросентропія обчислює втрати шляхом обчислення наступного середнього:

$$loss = -\frac{1}{output\ size} \sum_{i=1}^{outputsize} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log(1 - \hat{y}_i), \quad (6.2)$$

де, $\hat{y}_i$ – і-та скалярна величина на вході моделі;

y_i – відповідне цільове значення;

output size – кількість скалярних значень даних на виході моделі.

ROC-крива – це графік, що ілюструє ефективність бінарної системи класифікації в залежності від заданого класу. ROC-крива надає детальну інформацію про поведінку класифікатора. Крива є результатом зображення на графіку взаємозв'язку True Positive rate (TPR) False Positive rate (FPR).

Отже, ROC-крива ілюструє чутливість класифікатора, показуючи скільки правильно класифікованих об'єктів можна отримати, дозволяючи більше випадків FP.

ROC-крива для якісної моделі має велику площу під собою. Відповідно, графік кривої для неефективного класифікатора лишатиме під собою дуже маленьку площу.

Ще одним показником, який був використаний для оцінки та порівняння моделей це час роботи моделі. Час, за який проводиться класифікація може бути свідком того, наскільки швидко можна обробити дані.

Таким чином, тестування розробленого програмного забезпечення має наступні результати.

Звичайна повнозв'язний персептрон показав такі результати (рис.6.1-6.3). Точність діагностування склала 78.78%. графічне відображення процесу навчання моделі та її точність відображена на рисунку 6.1 та в таблиці 6.2.

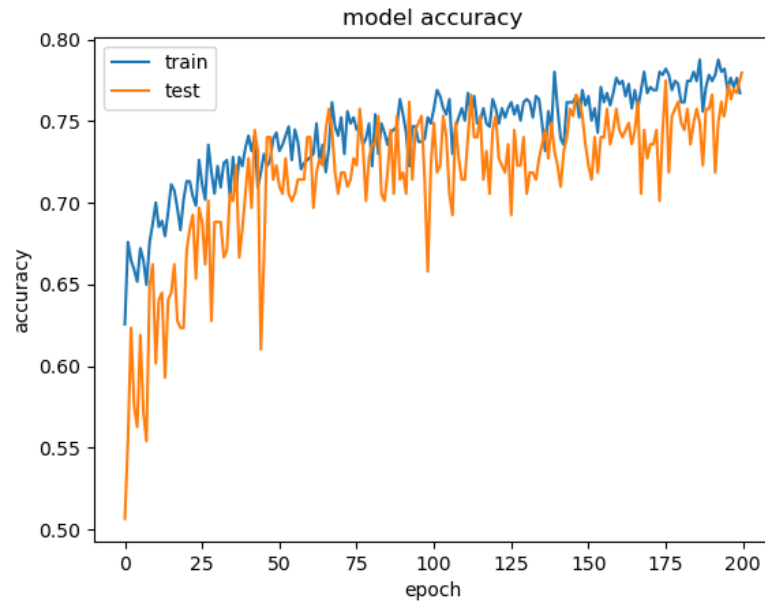


Рисунок 6.1 – Точність повнозв'язного персептрона

Процес навчання повнозв'язного персептрона відповідно до кількості епох відображено в таблиці 6.2, де наведено точність повнозв'язного персептрона через кожні 25 епох навчання на навчальній та тестовій вибірках.

Таблиця 6.2 – Точність повнозв'язного персептрона відповідно до епох навчання

Епоха	0	25	50	75	100	125	150	175	200
Точність на навчальній вибірці, %	62,71	70,77	73,28	74,22	76,15	76,13	76,24	77,76	76,73
Точність на тестовій вибірці, %	51,02	68,85	71,03	72,37	74,94	79,31	72,34	77,29	78,78

Зміну значення помилки відображено на графіку (рис. 6.2) та в таблиці 6.3, відповідно до якого спостерігається різке зниження значення

помилки, яке проявляє практично стале значення протягом навчання повнозв’язного персептрона.

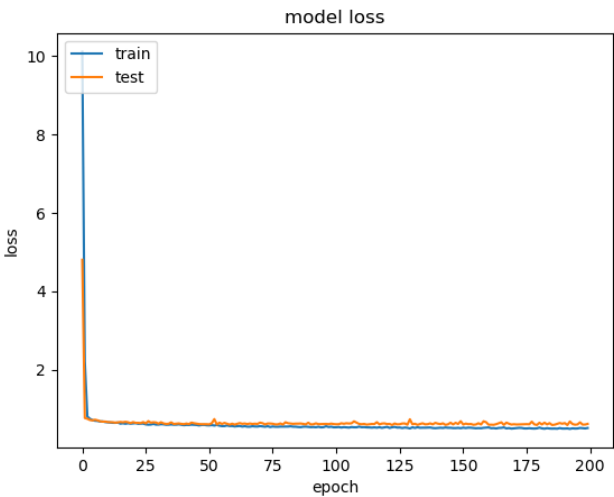


Рисунок 6.2 – Оцінка помилки повнозв’язного персептрона

Процес зниження значення функції втрат повнозв’язного персептрона у процесі навчання відповідно до кількості епох відображено в таблиці 6.3, де наведено значення функції втрат повнозв’язного персептрона через кожні 25 епох навчання на навчальній та тестовій вибірках.

Таблиця 6.3 – Значення функції помилки повнозв’язного персептрона відповідно до епох навчання

Епоха	0	25	50	75	100	125	150	175	200
Значення функції помилки на навчальній вибірці	10,23	0,641	0,639	0,652	0,649	0,647	0,647	0,647	0,647
Значення функції помилки на тестовій вибірці	10,07	0,61	0,62	0,57	0,59	0,58	0,60	0,59	0,58

Значення ROC-кривої – 0,811, графік якої зображено на рисунку 6.3 та в таблиці 6.4.

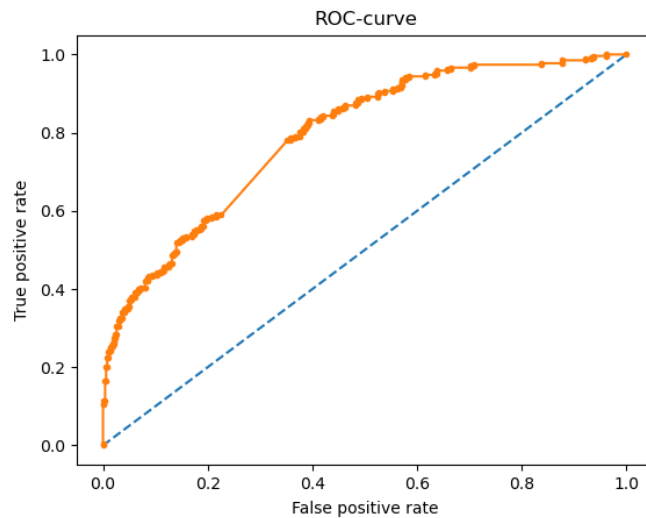


Рисунок 6.3 – ROC-крива повнозв'язного персептрона

Відповідно до збільшення значення False Positive rate відображено зміну значення True Positive rate у таблиці 6.4.

Таблиця 6.4 – Значення ROC-кривої

False rate	Positive	0.0	0.2	0.4	0.6	0.8	1.0
True rate	Positive	0,0	0,577	0,982	0,944	0,976	1,0

Результати повнозв'язного персептрона, навчання якого здійснювалося за допомогою ГА зображені на рисунках 6.4-6.6. Точність діагностування склала 89.3% та відображена на графіку (рис. 6.4) та в таблиці 6.5.

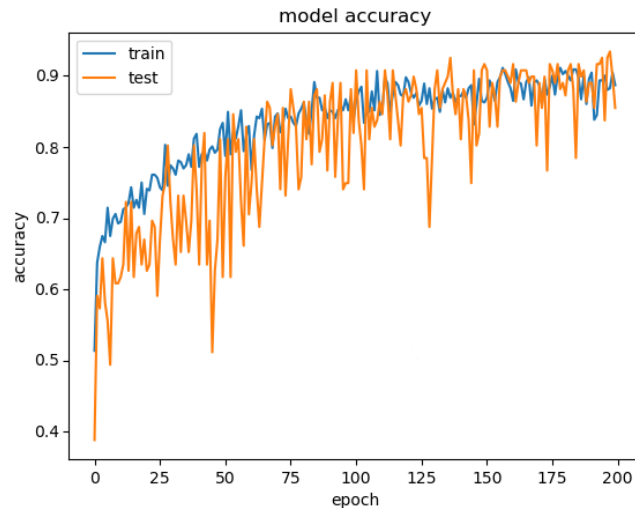


Рисунок 6.4 – Точність повнозв’язного персептрона, що навчався за допомогою ГА

Процес навчання повнозв’язного персептрона за допомогою генетичного алгоритму відповідно до кількості епох відображено в таблиці 6.5, де наведено точність повнозв’язного персептрона через кожні 25 епох навчання на навчальній та тестовій вибірках.

Таблиця 6.5 – Точність повнозв’язного персептрона відповідно до епох навчання

Епоха	0	25	50	75	100	125	150	175	200
Точність на навчальній вибірці, %	51,93	74,24	80,47	84,17	87,21	87,34	87,56	89,06	88,87
Точність на тестовій вибірці, %	38,71	67,86	77,12	8,78	90,74	84,34	91,02	89,98	89,3

Зміну значення помилки відображено на графіку (рис. 6.5) та в таблиці 6.6, відповідно до якого спостерігається зниження, проте її значення коливається протягом навчання повнозв’язного персептрона в межах від 0,6 до 1,0.

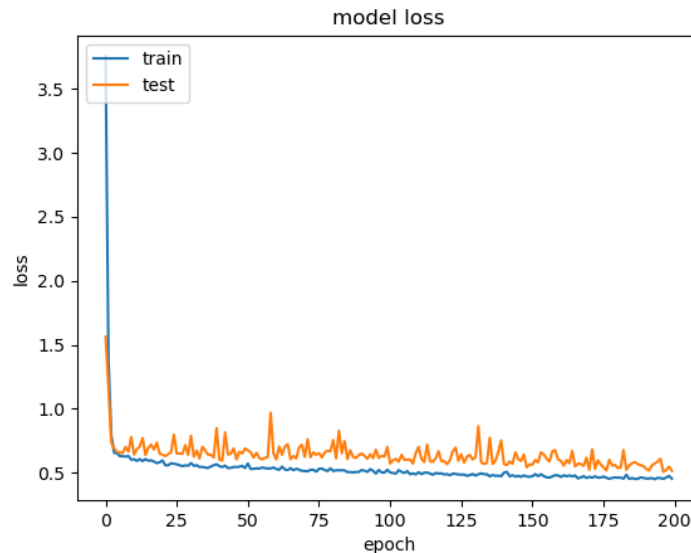


Рисунок 6.5 – Оцінка помилки повнозв’язного персептрона, що навчалася за допомогою ГА

Процес зниження значення функції втрат повнозв’язного персептрона у процесі навчання генетичним алгоритмом відповідно до кількості епох відображено в таблиці 6.6, де наведено значення функції втрат повнозв’язного персептрона через кожні 25 епох навчання на навчальній та тестовій вибірках.

Таблиця 6.6 – Значення функції помилки повнозв’язного персептрона відповідно до епох навчання

Епоха	0	25	50	75	100	125	150	175	200
Значення функції помилки на навчальній вибірці	3,582	0,567	0,551	0,527	0,503	0,497	0,496	0,491	0,487
Значення функції помилки на тестовій вибірці	1,575	0,637	0,671	0,646	0,528	0,530	0,531	0,513	0,511

Графік ROC-кривої зображено на рис. 6.6 та в таблиці 6.7, значення якої дорівнює 0,884.

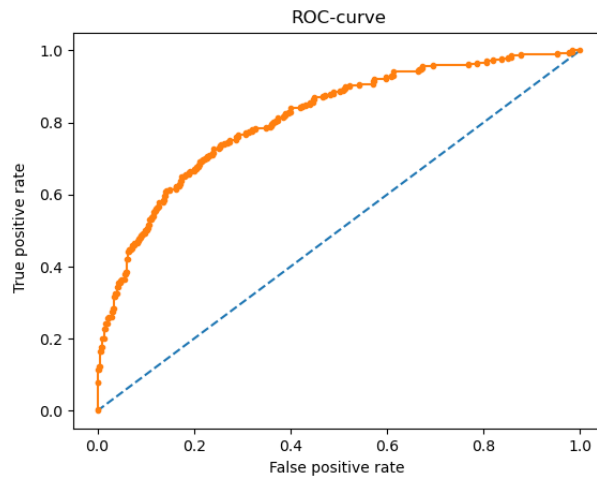


Рисунок 6.6 – ROC-крива повнозв’язного персептрона, що навчалася за допомогою ГА

Відповідно до збільшення значення False Positive rate відображено зміну значення True Positive rate у таблиці 6.7.

Таблиця 6.7 – Значення ROC-кривої

False rate	Positive	0.0	0.2	0.4	0.6	0.8	1.0
True rate	Positive	0,0	0,681	0,843	0,928	0,973	1,0

Результати повнозв’язного персептрона, навчання якого здійснювалося за допомогою PSO зображені на рисунках 6.7-6.9. Точність діагностування склала 90.82%, що видно з графіку точності моделі (рис. 6.7) та в таблиці 6.8.

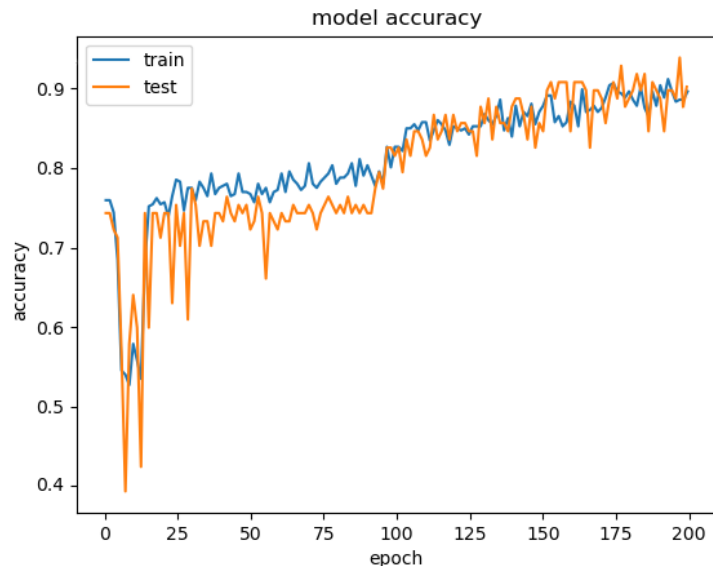


Рисунок 6.7 – Точність повнозв’язного персептрона, що навчалася за допомогою PSO

Процес навчання повнозв’язного персептрона за допомогою методу рою часток відповідно до кількості епох відображено в таблиці 6.8, де наведено точність повнозв’язного персептрона через кожні 25 епох навчання на навчальній та тестовій вибірках.

Таблиця 6.8 – Точність повнозв’язного персептрона відповідно до епох навчання

Епоха	0	25	50	75	100	125	150	175	200
Точність на навчальній вибірці, %	76,23	78,35	76,57	78,59	82,78	84,76	87,88	90,43	89,94
Точність на тестовій вибірці, %	74,49	73,05	73,31	75,47	82,38	84,53	89,55	89,27	90,82

Зміну значення помилки відображено на графіку (рис. 6.8) та в таблиці 6.9, відповідно до якого спостерігається зниження, проте її значення коливається протягом навчання мережі в межах від 0,5 до 1,0.

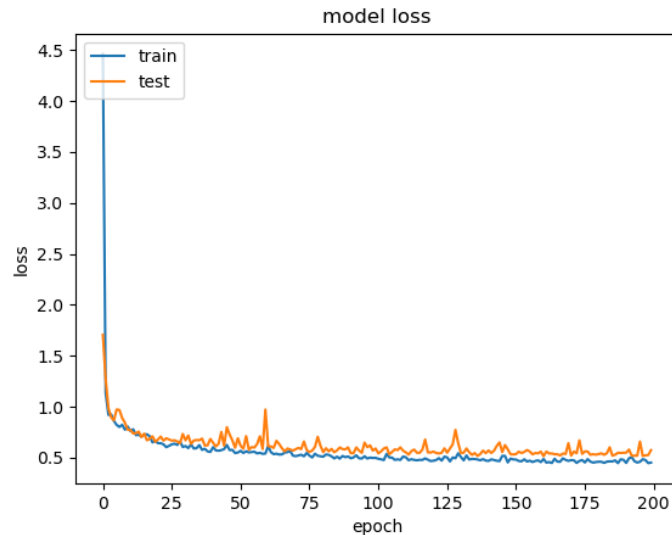


Рисунок 6.8 – Оцінка помилки мережі, що навчалася за допомогою PSO

Процес зниження значення функції втрат повнозв'язного персептрона, що навчався за допомогою методу рою часток у процесі навчання відповідно до кількості епох відображено в таблиці 6.9, де наведено значення функції втрат повнозв'язного персептрона через кожні 25 епох навчання на навчальній та тестовій вибірках.

Таблиця 6.9 – Значення функції помилки повнозв'язного персептрона відповідно до епох навчання

Епоха	0	25	50	75	100	125	150	175	200
Значення функції помилки на навчальній вибірці	4,497	0,637	0,552	0,537	0,498	0,501	0,509	0,510	0,489
Значення функції помилки на тестовій вибірці	1,72	0,69	0,575	0,568	0,597	0,566	0,567	0,566	0,564

Графік ROC-кривої наведено на рисунку 6.9 та в таблиці 6.10. Її значення склало 0,893.

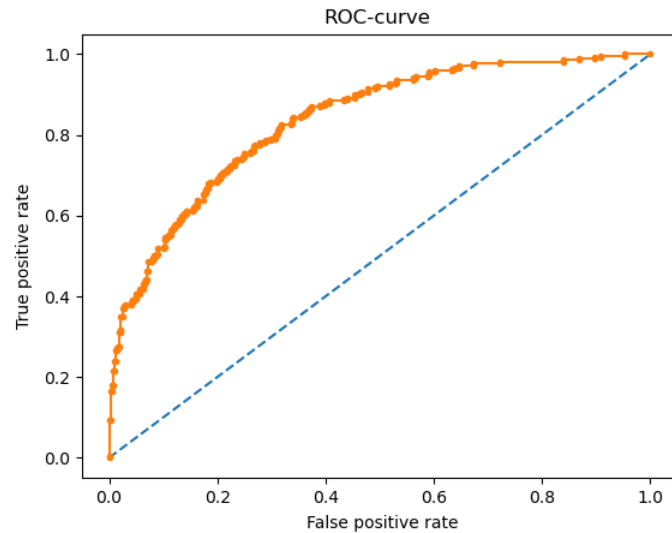


Рисунок 6.9 – ROC-крива повнозв'язного персептрона, що навчалася за допомогою PSO

Відповідно до збільшення значення False Positive rate відображено зміну значення True Positive rate у таблиці 6.4.

Таблиця 6.10 – Значення ROC-кривої

False rate	Positive	0.0	0.2	0.4	0.6	0.8	1.0
True rate	Positive	0,0	0,67	0,891	0,965	0,983	1,0

Результати повнозв'язного персептрона, навчання якої здійснювалося за допомогою ГА з модифікованими операторами селекції та з першим із запропонованих операторів мутації зображені на рисунках 6.10-6.12. графік точності моделі наведено на рис. 6.10 та в таблиці 6.11. Точність діагностування склала 91,3%.

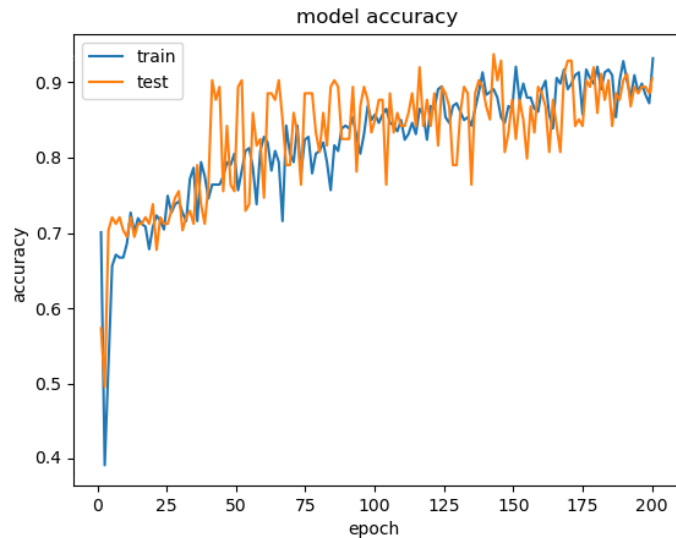


Рисунок 6.10 – Точність нейронної мережі, що навчалася за допомогою першої модифікації ГА

Процес навчання повнозв'язного персептрона за допомогою генетичного алгоритму із запропонованим оператором відбору та першого підходу до оператору мутації відповідно до кількості епох відображено в таблиці 6.11, де наведено точність повнозв'язного персептрона через кожні 25 епох навчання на навчальній та тестовій виборках.

Таблиця 6.11 – Точність повнозв'язного персептрона відповідно до епох навчання

Епоха	0	25	50	75	100	125	150	175	200
Точність на навчальній вибірці, %	70,58	73,85	77,56	82,53	85,37	85,41	88,90	89,12	83,26
Точність на тестовій вибірці, %	57,78	71,23	84,79	88,62	85,10	88,74	85,98	85,37	91,3

Зміну значення помилки відображено на графіку (рис. 6.11) та в таблиці 6.12, відповідно до якого спостерігається зниження значення з незначними коливаннями в межах значень 0,5-0,75.

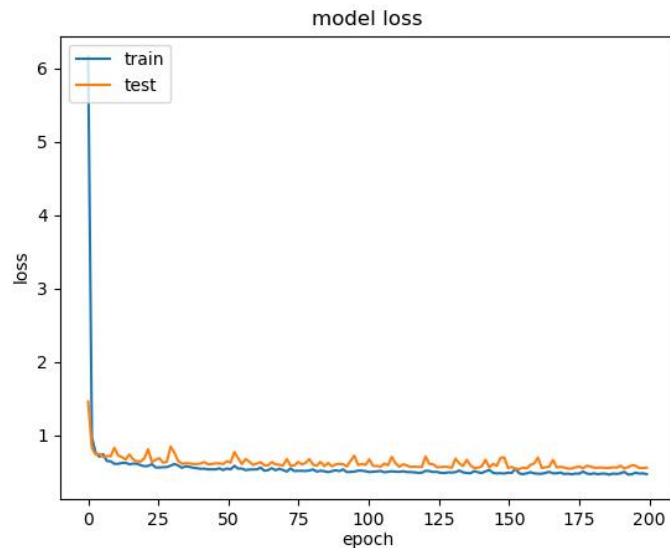


Рисунок 6.11 – Оцінка помилки повнозв'язного персептрона, що навчалася за допомогою першої модифікації ГА

Таблиця 6.12 – Значення функції помилки повнозв'язного персептрона відповідно до епох навчання

Епоха	0	25	50	75	100	125	150	175	200
Значення функції помилки на навчальній вибірці	6,975	0,563	0,471	0,468	0,403	0,407	0,410	0,406	0,404
Значення функції помилки на тестовій вибірці	1,50	0,679	0,619	0,617	0,615	0,613	0,612	0,611	0,610

Процес зниження значення функції втрат повнозв'язного персептрона, що навчався за допомогою генетичного алгоритму із запропонованим

оператором відбору та першого підходу до оператору мутації у процесі навчання відповідно до кількості епох відображено в таблиці 6.12, де наведено значення функції втрат повнозв'язного персептрона через кожні 25 епох навчання на навчальній та тестовій виборках.

Значення ROC-кривої складає 0,9, графік якої зображено на рис.6.12 та в таблиці 6.13.

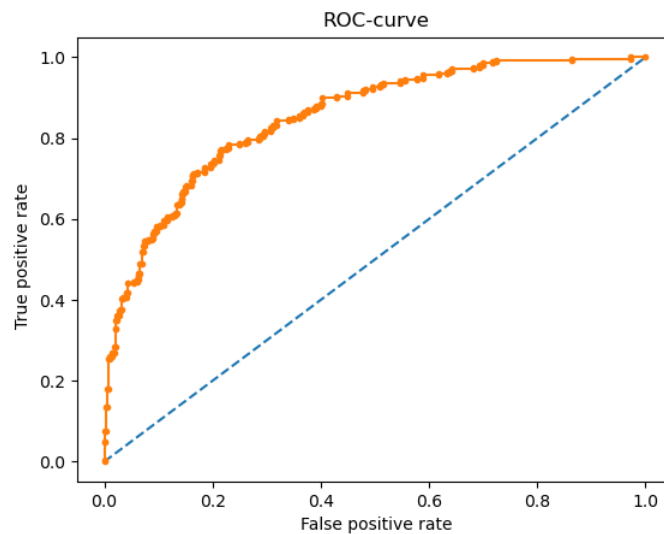


Рисунок 6.12 – ROC-крива повнозв'язного персептрона, що навчалася за допомогою першої модифікації ГА

Відповідно до збільшення значення False Positive rate відображено зміну значення True Positive rate у таблиці 6.4.

Таблиця 6.13 – Значення ROC-кривої

False rate	Positive	0.0	0.2	0.4	0.6	0.8	1.0
True rate	Positive	0,0	0,689	0,896	0,961	0,985	1,0

Результати повнозв'язного персептрона, навчання якого здійснювалося за допомогою ГА з модифікованими операторами селекції та з

другим із запропонованих операторів мутації зображені на рисунках 6.13-6.15. Точність діагностування склала 89,97% (рис. 6.13) та в таблиці 6.14.

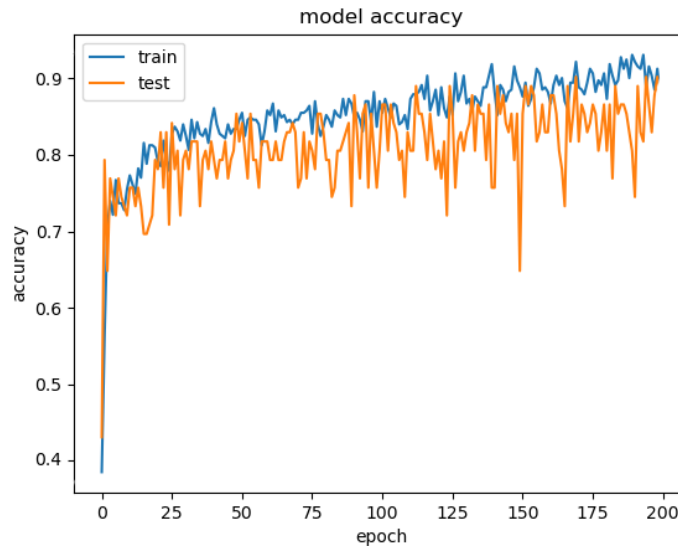


Рисунок 6.13 – Точність повнозв’язного персептрона, що навчалася за допомогою другої модифікації ГА

Процес навчання повнозв’язного персептрона за допомогою генетичного алгоритму із запропонованим оператором відбору та другого підходу до оператору мутації відповідно до кількості епох відображено в таблиці 6.14, де наведено точність повнозв’язного персептрона через кожні 25 епох навчання на навчальній та тестовій виборках.

Таблиця 6.14 – Точність повнозв’язного персептрона відповідно до епох навчання

Епоха	0	25	50	75	100	125	150	175	200
Точність на навчальній вибірці, %	36,74	82,94	84,52	82,98	86,69	87,43	87,85	90,38	89,14
Точність на тестовій вибірці, %	43,31	83,74	82,76	80,73	86,65	82,83	85,56	86,67	89,97

Зміну значення помилки відображено на графіку (рис. 6.14) та в таблиці 6.15, відповідно до якого спостерігається поступове зниження значення помилки, що зберігає стабільне значення починаючи з 75 епохи.

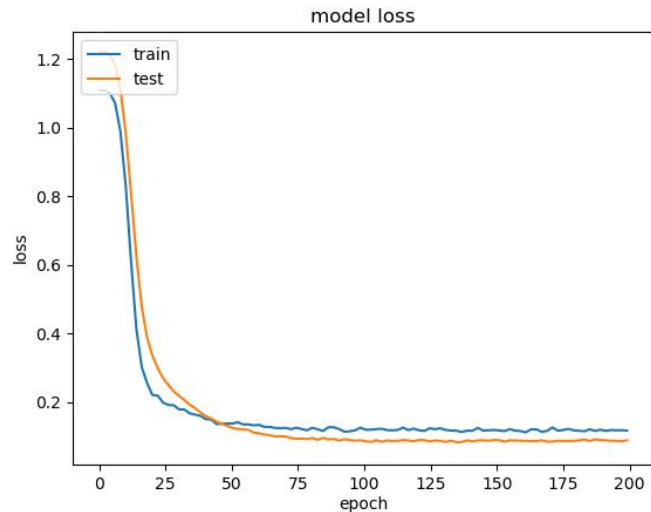


Рисунок 6.14 – Оцінка помилки повнозв’язного персептрона, що навчалася за допомогою другої модифікації ГА

Таблиця 6.15 – Значення функції помилки повнозв’язного персептрона відповідно до епох навчання

Епоха	0	25	50	75	100	125	150	175	200
Значення функції помилки на навчальній вибірці	1,157	0,198	0,146	0,125	0,123	0,124	0,123	0,124	0,123
Значення функції помилки на тестовій вибірці	1,33	0,264	0,132	0,088	0,081	0,080	0,081	0,080	0,080

Процес зниження значення функції втрат повнозв’язного персептрона, що навчання за допомогою генетичного алгоритму із запропонованим

оператором відбору та першого підходу до оператору мутації у процесі навчання відповідно до кількості епох відображено в таблиці 6.15, де наведено значення функції втрат повнозв'язного персептрона через кожні 25 епох навчання на навчальній та тестовій виборках.

Значення ROC-кривої дорівнює 0,899. Графік даної кривої зображено на рисунку 6.15 та в таблиці 6.16.

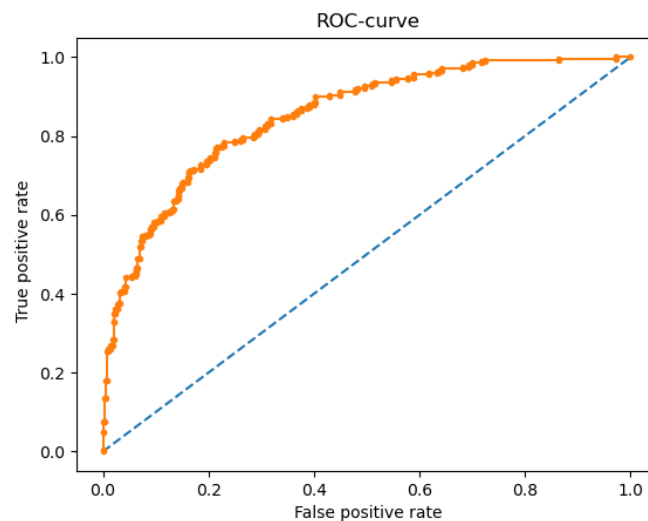


Рисунок 6.15 – ROC-крива повнозв'язного персептрона, що навчалася за допомогою другої модифікації ГА

Відповідно до збільшення значення False Positive rate відображено зміну значення True Positive rate у таблиці 6.4.

Таблиця 6.16 – Значення ROC-кривої

False rate	Positive	0.0	0.2	0.4	0.6	0.8	1.0
True rate	Positive	0,0	0,743	0,872	0,957	0,983	1,0

Для наочного відображення результатів отриманих в процесі дослідження моделей, дані занесені до таблиці 6.17, а точність класифікації відображена на гістограмі (рис. 6.16).

Таблиця 6.17 – Порівняльна таблиця результатів оцінювання моделей

№ п/п	Назва	Точність, %	Значення ROC-кривої	Час, с
1	2	3	4	5
1	Нейронна мережа	78,78	0,811	117
2	Нейронна мережі + ГА	89,3	0,884	99
3	Нейронна мережі + перша модифікація ГА	91,3	0,9	101
4	Нейронна мережі + друга модифікація ГА	89,97	0,899	102
5	Нейронна мережі + PSO	90,82	0,893	97

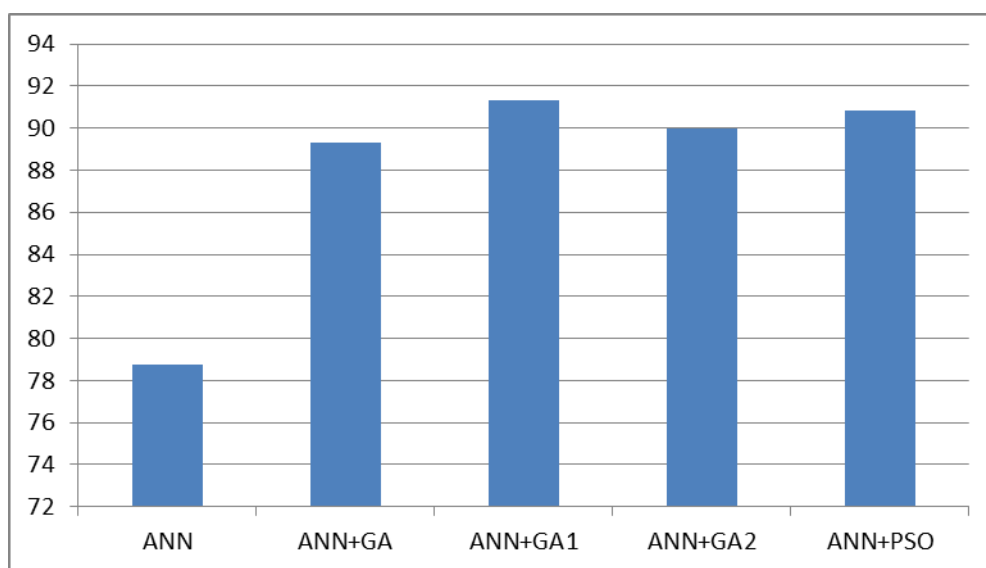


Рисунок 6.16 – Гістограма порівняння точності тестованих моделей

У результатів проведених експериментів можна зробити висновок про те, що навчання мережі за допомогою еволюційних алгоритмів показує значне покращення порівняно зі стандартними методами. Використання розглянутих методів показує приблизно однаковий результат за точністю діагностування, а час, що було затрачено на роботу є доволі незначним'.

З рис. 6.16 видно, що найкращу точність показав повнозв'язний персептрон, що навчався за допомогою генетичного алгоритму із запропонованим оператором відбору та першого підходу до оператору мутації, точність якого склала 91,3%. Повнозв'язний персептрон, що навчався за допомогою методу рою часток показав точність у 90,82%. Трохи гірші результати має повнозв'язний персептрон, що навчався за допомогою генетичного алгоритму із запропонованим оператором відбору та другого підходу до оператору мутації, точність якого склала 89,97%. Повнозв'язний персептрон, що навчався за допомогою генетичного алгоритму показав точність у 89,3%. Найгірший результат за точністю має повнозв'язний персептрон, що навчався за допомогою градієнтного методу, його точність складала 78,78%.

6.2 Висновки за розділом

У розділі наведено методику проведення експериментів. Зроблено опис метрик, за якими оцінюється якість навчання нейронної мережі, а саме точність, значення функції втрат та ROC-кривої. Також враховується час, за який відбувається робота моделі, але його значення для всіх способів не сильно різняться. Всі результати експериментів наведено у чисельному та графічному вигляді. Зроблено порівняння значення точності всіх експериментів і відображено на гістограмі (рис. 6.16). за результатами, що зображені на гістограмі видно, що навчання нейронної мережі за допомогою еволюційних методів суттєво збільшують їх точність як класифікатора.

7 ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ РОБОТИ

7.1 Опис ідеї

Перед початком створення будь-якого програмного продукту важливо провести аналіз ринку, за допомогою якого можна зрозуміти чи є попит на розроблюваний продукт, визначити потенційних клієнтів. Окрім того, попередній аналіз дозволить обчислити кошторис витрат на розробку проєкту.

Розроблено проєкт на тему «Дослідження та програмна реалізація інтелектуальних методів для медичного діагностування».

В результаті дослідження розроблено інтелектуальну систему підтримки прийняття рішень для діагностування цукрового діабету, яка базується на моделях машинного навчання. Реалізовано модель параметричної оптимізації нейронної мережі за рахунок використання генетичного алгоритму та методу рою часток. Модифікація простого генетичного алгоритму, яка реалізована в рамках виконання проєкту, дозволяє пришвидшити виконання підбору параметрів навчання нейронних мереж, та підвищити результуючий показник точності порівняно з базовою версією простого генетичного алгоритму, за рахунок модифікації оператора мутації, а також, зміненого підходу до вибору особин для схрещення.

Розроблена модель призначена для застосування у сфері медичного обслуговування та дозволяє з певною точністю визначати наявність ризику захворюваності пацієнта на цукровий діабет за клінічними показниками стану здоров'я. Результатом застосування даного програмного продукту є зменшення імовірності помилки лікаря, підвищення впевненості лікаря у прийнятому рішенні при постановці діагнозу та більша кількість врятованих життів, за рахунок постановки вірного та своєчасного діагнозу.

Даний проєкт визначення ризику виникнення цукрового діабету реалізовано за допомогою бібліотеки Keras, що є безкоштовним дослідницьким та навчальним проєктом

Найближчими аналогами є проєкти Watson Health, Deepmind Health, Sensely.

На сьогодні, вирішення проблеми діагностики діабету є важливою оскільки хвороба є хронічною та може мати серйозний вплив на весь організм людини. Тож створення програмного продукту, який би міг враховувати анамнез пацієнта та дані його клінічних обстежень є актуальною.

7.2 Попередня характеристика потенційного ринку

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проєкту, та ринкових загроз, які можуть перешкодити реалізації проєкту, дозволяє спланувати напрями розвитку проєкту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проєктів конкурентів.

Сьогодні алгоритми штучного інтелекту знаходяться у зоні підвищеного інтересу як інвесторів, так і медичних працівників, через швидкість опрацювання даних та точність результатів. Програмні продукти, що ґрунтуються на алгоритмах ШІ спрямовані не на заміну лікарів, а на їх більш ефективну роботу: опрацювання даних пацієнтів, постановку діагнозу, надання рекомендацій для лікування чи додаткових обстежень тощо. Тож динаміку ринку можна визначити як зростаючу.

Прикладами програмних застосунків, які аналізують стан здоров'я людини за показниками є Deepmind Health, Watson Health, Sensely, результатом роботи яких є постановка діагнозу. В свою чергу перераховані програмні системи є сильними конкурентами, що зайняли своє місце на ринку, оскільки це повноцінні продукти, які можуть обчислювати великі об'єми даних, враховують їх структурованість та наявні пропуски. Проте є доволі дорогими, є складними, та через роботу з великими об'ємами даних є порівняно повільними в роботі.

Результати проведеного аналізу потенційного ринку наведені в таблиці 7.1.

Таблиця 7.1 – Попередня характеристика потенційного ринку

№	Показники стану ринку (найменування)	Характеристика
1	Головні конкуренти	Deepmind Health
		Watson Health
		Sensely
2	Динаміка ринку (якісна оцінка)	Зростає
3	Наявність обмежень для входу (вказати характер обмежень)	Надвеликі об'єми даних, наявність пропусків у даних, необхідність налаштування програмного продукту під конкретну галузь, проблема неструктурованих даних.
4	Специфічні вимоги до стандартизації та специфікації	Відсутні

Формування ринку обумовлено потребою точної та швидкої постановки діагнозу шляхом визначення взаємозв'язків, що містять дані. Програмний продукт спрямований на вирішення цих потреб та допомоги лікарю при прийнятті рішення щодо подальшого діагностування (за потреби) та постановки діагнозу.

Цільовою аудиторією є заклади медичного обслуговування. Даний продукт може зацікавити їх високою точністю та швидкістю обчислення

даних пацієнта. Попередня характеристика потенційних клієнтів узагальнена в таблиці 7.2.

Таблиця 7.2 – Попередня характеристика потенційних клієнтів

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів (користувачів)
Потреба точної та швидкої постановки діагнозу пацієнту, що ґрунтується на ряді персональних даних, медичних показників та результатів клінічних досліджень.	Державні та приватні медичні установи	Система має бути доступною для користувача як за ціною так і за зрозумілістю використання.	Користувачі очікують, що програмний продукт буде проводити обробку та аналіз даних швидко та якісно, забезпечуватиме результат високої точності.

В таблиці 7.3 наведено SWOT-аналіз розробленого застосування. Описано сильні сторони, такі як впровадження модифікованого генетичного алгоритму, за рахунок якого підвищується точність та швидкість роботи програмного продукту, та недоліки, а саме нездатність обробляти великі об'єми даних та швидкий темп розвитку галузі. Основними можливостями є впровадження інноваційних рішень та спрямованість на медичну діагностику взагалі. До загроз можна віднести появу сильних конкурентів на ринку.

Таблиця 7.3– SWOT – аналіз

Сильні сторони	Слабкі сторони
- Іноваційність - Унікальність - Швидкість роботи алгоритму - Висока точність	- Нездатність працювати з великими об'ємами даних - Швидкий темп розвитку галузі
Можливості	Загрози
- Можливість впровадження проекту для вирішення медичної діагностики в цілому - Використання нейроалгоритмів	- Поява сильних конкурентів

7.3 Розроблення ринкової стратегії проекту

Потенційними клієнтами розроблюваного програмного продукту є медичні працівники, а саме лікарі ендокринологи, що здійснюють свою медичну в закладах медичного обслуговування приватної та державної форми власності. Оскільки продукт надає можливість лікарю швидше провести діагностування та визначити схильність пацієнта до виникнення діабету, то спеціалісти мають певну зацікавленість у використанні даного продукту у своїй діяльності. Але обмеженням для цього є спосіб фінансування закладів медичного обслуговування. Державним закладам для придбання продукту необхідно підготувати документи та пройти певну процедуру і часто гроші виділяються, якщо у бюджет було закладено суму на придбання додаткового програмного забезпечення. У приватних закладах, як правило, є резервні фонди, з яких вони можуть дозволити собі здійснювати придбання додаткової продукції. Відповідно до цього оцінюється готовність споживачів сприйняти продукт.

Конкуренцію в сегментів складають як вітчизняні, так і закордонні компанії-розробники програмного забезпечення. Зазвичай, в Україні, незалежно від форми власності медичних закладів, використовують аналоги з меншою ціною, тож інтенсивність конкуренції оцінена з точки зору цінової політики. Відповідно до проведеного аналізу простота входу у сегмент визначена як середня.

Розроблення ринкової стратегії запропоновано в таблиці 7.4.

Таблиця 7.4 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегментів	Простота входу у сегмент
1	Державні медичні заклади	Середня	Помірний	Низька	Середня
2	Приватні медичні заклади	Висока	Помірний	Середня	Середня
Які цільові групи обрано: державні та приватні медичні заклади					

7.4 Калькуляція витрат на реалізацію проєкту

Для визначення витрат на розробку програми складається калькуляція кошторисної вартості робіт, яка включає:

- розрахунок витрат на заробітну плату працівникам;
- відрахування на єдиний соціальних внесок;

- розрахунок матеріальних витрат;
- витрати на спеціальне обладнання;
- накладні витрати.

7.4.1 Визначення складу, чисельності та фонду заробітної плати працівників

Для реалізації проєкту необхідними учасниками є інженер-програміст та консультант. Розробка розпочинається 31 серпня і повинна бути виконана до 13 листопада 2020 року.

Ефективний фонд робочого часу визначається шляхом обчислення різниці між тривалістю календарного часу, кількості вихідних та святкових днів та кількості днів невиходу за причиною (7.5).

$$\Phi_{pc} = T_K - T_{Bux} - T_{Невих}, \quad (7.1)$$

де T_K – календарний час;

T_{Bux} – вихідні і святкові дні;

$T_{Невих}$ – невиходи за причиною.

Тривалість календарного часу складає 75 днів. За період розробки маємо 1 святковий день та 20 вихідних. Отже, ефективний фонд робочого часу складає:

$$\Phi_{pc} = 75 - (20 + 1) = 56 \text{ днів}.$$

Розрахунок заробітної плати за виконання проєкту здійснюється на основі календарного плану робіт (табл. 7.5), в якому визначається трудомісткість, тривалість робіт, кількість виконавців.

Таблиця 7.5 – Етапи розробки продукту

Назва роботи	Трудовісткість		Виконавці	Тривалість, днів
	люд.– дні	% до підсумку		
1. Розробка календарного плану	1	1,64	Консультант	1
2. Аналіз предметної області	3	4,92	Програміст	3
3. Визначення вимог до програмного продукту та постановка завдання	2	3.28	Програміст	2
3. Розробка і затвердження технічного завдання	2	3.28	Програміст, консультант	1
3. Аналіз принципів оптимізації нейронних мереж	4	6.55	Програміст	4
4. Дослідження методів оптимізації	5	8.2	Програміст	5
5. Побудова моделі системи	6	9.84	Програміст	6
6. Створення програмного продукту	25	40.98	Програміст	25
7. Тестування і налагодження програмного продукту	5	8.2	Програміст	5
8. Складання програмної документації	8	13.11	Програміст, консультант	4
Разом	61	100		56

Відповідно до табл. 7.5 програміст задіяний на проєкті протягом 55 днів, консультант – 6 днів. Робочий день складає 8 годин.

Отже, тарифний заробіток кожного зі спеціалістів розраховується за формулою (7.2):

$$T_3 = T_C * PD * \Phi_{PЧС} , \quad (7.2)$$

де T_C – тарифна ставка за розрядом;

P_D – тривалість робочого дня (8 годин);

$\Phi_{PЧC}$ – фонд робочого часу спеціаліста, тобто скільки днів спеціаліст задіяний на проєкті.

Відповідно до (7.2) тарифний заробіток програміста складає:

$$T_3 = 120 * 8 * 55 = 52800 \text{ грн.}$$

Тарифний заробіток консультанта складає:

$$T_3 = 70 * 8 * 6 = 3360 \text{ грн.}$$

Розрахунок річного фонду заробітної плати виконується на основі чисельності працівників, тарифної ставки, розміру преміальних виплат та додаткових виплат. Обчислення річного фонду оплати праці здійснюється за допомогою формули (7.3).

$$\Phi_{3П} = \sum_{i=1}^n (T_C + П) * P_{Г}, \quad (7.3)$$

де T_C – тарифна ставка за розрядом;

$П$ – розмір премії;

$P_{Г}$ – кількість робочих годин на рік;

n – кількість працівників.

Відповідно до (7.3) річного фонду заробітної плати становить:

$$\Phi_{3П} = ((120 + 24) * 2002) + (70 + 14) * 2002 = 456456 \text{ грн}$$

Відрахування на єдиний соціальний внесок становлять 22% і беруться від основної та додаткової заробітної плати. Вони розраховуються за формулою 7.4:

$$ECB = (T_z + П) * 0,22, \quad (7.4)$$

де T_z – тарифний заробіток;

$П$ – розмір премії.

ЄСВ для програміста складає:

$$ECB = (52800 + 10560) * 0,22 = 13939,2 \text{ грн.}$$

ЄСВ для консультанта складає:

$$ECB = (3360 + 672) * 0,22 = 887.02 \text{ грн.}$$

Загалом додаткові соціальні відрахування становлять 14826,22 грн.

Витрати на заробітну плату працівників, премій, відрахування ЄСВ проводиться на підставі даних таблиці 7.5. Для наочного відображення всі проведені обчислення зведені до таблиці 7.6.

Таблиця 7.6 – Склад, чисельність та фонд заробітної плати працівників

Категорії працівників	Наявна чисельність осіб	Тарифна ставка за розрядом виконуваних робіт, грн/год	Ефективний фонд робочого часу, днів	Тарифний заробіток, грн	Преміальний відсоток до тарифного заробітку	Розмір премії, грн	Річний фонд заробітної плати, грн	ЄСВ, грн
Інженер-програміст	1	120	55	52800	20	10560	2882888	19939,2
Консультант	1	70	6	3360	20	672	168168	887,02
Всього	2	X	X	X	X	11232	456456	14826,22

7.4.2 Визначення витрат на матеріали та вартості спожитих послуг

У загальну вартість реалізації проєкту необхідно включити вартість використаних матеріалів та спожитих послуг. Ціну кожного матеріалу та відповідних послуг визначають за прейскурантами чи по іншим довідковими даними.

Витрати на матеріали розраховуються за формулою:

$$M = \sum_{i=1}^n (C_i N_i (1 + K_{m.z.}) - C_{i0} N_{i0}), \quad (7.5)$$

де M – витрати на матеріали, покупні напівфабрикати та комплектуючі (грн.);

$K_{m.з.}$ – коефіцієнт, що враховує транспортно–заготівельні витрати;

C_i – ціна i –го найменування чи комплектуючого (грн.);

N_i – потреба в i –му матеріалі, напівфабрикаті, комплектуючих;

C_{io} – ціна зворотних відходів i –го найменування матеріалу (грн.);

N_{io} – кількість зворотних відходів i –го найменування;

n – кількість найменувань матеріалів, напівфабрикатів та комплектуючих.

Витрати на матеріали для розробки наведені в табл. 7.7:

Таблиця 7.7 – Розрахунок матеріальних витрат

Матеріальні витрати, найменування	Одиниця виміру	Ціна за одиницю, грн	Кількість використаних одиниць	Сума, грн
Папір офісний А4	пачка	85	1	85
Ручка	шт.	5	2	10
Блокнот	шт.	20	1	20
USB-флеш накопичувач 8Gb	шт.	130	1	130
Картридж принтера чорний	шт.	410	1	410
Разом				655

З урахуванням можливих транспортних витрат (5% від вартості матеріалів), сумарна вартість матеріалів становить 687.75 грн.

$$M = 1,05 \cdot 655 = 687.75 \text{ грн.}$$

При виконанні роботи електроенергія споживалась комп'ютерною технікою (комп'ютер з периферійними пристроями) і настільною освітлювальною лампою; розрахуємо їх щоденні витрати.

Для розрахунку вартості експлуатації обладнання використаємо формулу:

$$C_E = N_H \Phi_{ef} K_{zm} K_{zn} C_e, \quad (7.6)$$

де N_H – номінальна потужність ЕОМ, кВт;

Φ_{ef} – річний ефективний фонд часу роботи ЕОМ;

K_{zm} – середні коефіцієнт завантаження за часом;

K_{zn} – коефіцієнт завантаження за потужністю;

C_e – ціна одного кВт*год. Електроенергії, грн./(кВт*год.).

Для оцінки собівартості годин машинного часу необхідно визначити дійсний фонд робочого часу ЕОМ. Дійсний фонд робочого часу ЕОМ обчислюється як число фактично відпрацьованих годин спеціаліста, за винятком часу на профілактику та ремонт ЕОМ. Час профілактики: щомісячно – 5 годин.

$$\Phi_{ef} = \Phi_{pq} - \chi_{pp}, \quad (7.7)$$

де, Φ_{pq} – фонд робочого часу працівника (448 години);

χ_{pp} – часу на профілактику та ремонт ЕОМ.

Отже, річний фонд часу роботи ЕОМ становить 436 години. Середні коефіцієнти завантаження за часом та за потужністю становлять 0.6 та 0.9.

Відповідно, до формули 7.6 вартість експлуатації ЕОМ складає:

$$C_E = 0,3 * 436 * 0,8 * 1,68 = 175,8 \text{ грн.}$$

Також при роботі використовувалась настільна освітлювальна лампа, яка використовувалася протягом 2 год/день (112 год протягом усього проєкту). Отже, вартість її експлуатації становить:

$$C_E = 0,1 * 112 * 0,8 * 1,68 = 15,05 \text{ грн.}$$

Загалом на електропостачання витрачено 190,85 грн.

Витрати на водопостачання визначаються з розрахунку на 1 особу $4\text{м}^3/\text{місяць}$. Оскільки розробка ведеться протягом 2,5 місяців то споживання води приймаємо у розмірі $12\text{м}^3/\text{місяць}$ з урахуванням того що програміст працює 55 днів, а консультант 6 днів. Вартість 1м^3 становить 20,77 грн. Отже витрати на водопостачання становлять 249,24 грн.

Витрати на теплопостачання обчислюються з урахуванням площі приміщення (42 м^2) та дати початку опалення (02.11.2020) та дати завершення проєкту (13.11.2020). Вартість опалення 1м^2 становить 34,56 грн/місяць. Отже, вартість опалення у період з 02.11 до 13.11 з врахуванням площі приміщення становить 580,6 грн.

Розрахунок витрат на спожиті послуги зведено до таблиці 7.8.

Таблиця 7.8 – Розрахунок вартості спожитих послуг

Вид послуг	Норма витрат	Ціна, грн	Сума, грн
Електропостачання	0,4 кВт/год	1,68	190,85
Водопостачання	$4\text{м}^3/\text{ос}*\text{міс}$	20,77	249,24
Теплопостачання	1м^2	34,56	580,6
Разом			1020,69

7.4.3 Розрахунок суми амортизації

Суму амортизаційних відрахувань від балансової вартості універсального обладнання, апаратів і приладів, які відносяться до основних фондів, розраховується відповідно до їх зайнятості в даній роботі. Для використаного комп'ютерного обладнання сума амортизації становить 25% на рік від його балансової вартості:

$$A = \Phi_{\phi} * H_a, \quad (7.8)$$

де Φ_{ϕ} – балансова вартість обчислювальної техніки, грн.;

H_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, %.

Підставивши у формулу 7.8 значення балансової вартості використовуваних EOM DELL Inspiron 15, маніпулятору типу миша та принтеру HP Deskjet 2710 (разом 17 500 грн.) отримаємо:

$$A = 17500 * 0,25 = 4375 \text{ грн.}$$

Амортизаційні відрахування становлять 4375 грн

Результати розрахунків зводимо в таблицю 7.9.

Таблиця 7.9 – Розрахунок суми амортизаційних відрахувань

Обладнання	Балансова вартість, грн	Норма амортизації, %	Сума амортизації, грн
Комп'ютерна техніка	17500	25	4375

7.4.4 Калькуляція кошторису витрат

До накладних витрат при виконанні даної роботи відносяться витрати за інтернет-послуги, оренду приміщення та інші витрати, які складають 40% від основної заробітної плати розробника і становлять 22464 грн.

Результати визначення витрат на розробку програми у вигляді калькуляції кошторисної вартості робіт наведено в таблиці 7.10. У таблиці розрахована підсумкова сума вартості робіт, а так само питома вага кожної статті витрат у відсотках від підсумкової суми.

Таблиця 7.10 – Кошторис витрат

Калькуляційні статті	Витрати за період виконання проєкту, грн	Питома вага від підсумку, %
Електропостачання	190,85	0,19
Водопостачання	249,24	0,25
Теплопостачання	580,6	0,58
Заробітна плата	56160	56,43
ЄСВ	14826,22	14,9
Амортизація універсального обладнання	4375	4,4
Витрати на матеріали	655	0,65
Накладні витрати	22464	22,6
Всього	99500,91	100

7.5 Висновки за розділом

Розробка програмного забезпечення проводиться за рахунок гранту в 100 тисяч гривень. Оскільки витрати на розробку програмного забезпечення є нижчими, ніж сума гранту, розробка є економічно обґрунтованою.

8 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

8.1 Аналіз потенційних небезпек

Використання програмного додатку, що було розроблено в ході виконання проєкту передбачає використання обчислювальної техніки (персональної ЕОМ, периферійних пристроїв, тощо) та, як і будь-яка виробнича діяльність, пов'язане з впливом небезпечних і шкідливих виробничих чинників. Небезпечні і шкідливі чинники визначені в ДСанПіН «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу» №0472-14 [51].

На основі детального аналізу потенційних небезпек, що виникають під час виробничої діяльності пов'язаної з використанням розробленого програмного додатку, було здійснено планування і впровадження низки заходів направлених на забезпечення безпечних умов праці.

Усі потенційні небезпеки, що виникають при роботі в офісі або обчислювальному центрі, при роботі з обчислювальною технікою можна поділити на такі групи:

- а) потенційні небезпеки, фізичного характеру;
- б) потенційні небезпеки психофізіологічного характеру;
- в) потенційні небезпеки санітарно-гігієнічного характеру;
- г) потенційні небезпеки, пов'язані з порушеннями правил пожежної безпеки;
- г) потенційні небезпеки, пов'язані з проявом наслідків надзвичайних ситуацій.

До потенційних небезпек фізичного характеру належать механічні травмування, що можуть статися в наслідок невідповідного планування робочого простору виробничого приміщення (розташування електричних або комунікаційних дротів на підлозі в зонах з високою прохідністю, несправні офісні меблі, погано закріплені полиці та шафи, тощо), можливі ураження

електричним струмом під час роботи за комп'ютером або використанні периферійної комп'ютерної техніки (в наслідок порушення правил безпеки під час роботи або використанні несправного обладнання), тощо. Відомо, що робота за комп'ютером супроводжується значним статичним навантаженням на опорно-руховий апарат та органи зору. Професійним захворюванням фахівців, чия праця пов'язана з довготривалим використанням ЕОМ (інженери-програмісти, оператори ЕОМ, тощо) є синдром зап'ястного каналу, або тунельний синдром зап'ястя, який може бути наслідком хронічної травми. Причиною цього захворювання є тривала робота з мишею: постійна напруга і здавлювання приводить до мікротравм, здавлювання нерва прилеглими оточуючими тканинами, через що виникає набряк.

Потенційні небезпеки психофізіологічного характеру пов'язані з незадовільною організацією робочого місця (недостатнє врахуванням вимог ергономіки робочого місця, зокрема нераціональне розташування комп'ютерного та периферійного обладнання, захаращеність робочої зони, неправильно підібрані столи та стільці відносно параметрів людини) або виробничого процесу (надлишкові комунікації під час робіт, що потребують зосередженості та значних розумових зусиль, тощо).

До потенційних небезпек санітарно-гігієнічного характеру можна віднести ті, що виникають в наслідок порушення вимог санітарно-гігієнічних норм до приміщення, в якому безпосередньо здійснюється виробничий процес (наприклад, недостатня або підвищена освітленість робочої зони, недостатній повітрообмін, високий рівень шумів внаслідок роботи комп'ютерних складових тощо).

До групи небезпек, що пов'язані з порушенням правил пожежної безпеки, можна віднести потенційні небезпеки, що пов'язані з недотриманням або неналежним дотриманням правил пожежної безпеки при плануванні виробничого приміщення, виробничого процесу або під час використання пожежонебезпечного обладнання чи устаткування (комп'ютерної техніки, периферійних пристроїв, тощо).

Нарешті, до потенційних небезпек, що пов'язані з проявами наслідків надзвичайних ситуацій, слід віднести ті, що пов'язані з невідповідністю персоналу в умовах надзвичайних ситуацій (наприклад, під час пожежі або стихійного лиха).

8.2 Заходи із забезпечення безпеки

Для запобігання виникненню потенційних небезпек фізичного, психофізіологічного, санітарно-гігієнічного характеру та небезпек, що пов'язані з порушеннями правил пожежної безпеки, та з проявом наслідків надзвичайних ситуацій, і мінімізації впливу шкідливих виробничих факторів передбачено низку заходів по забезпеченню техніки безпеки. При плануванні виробничого процесу і розробці заходів щодо забезпечення техніки безпеки враховано норми і вимоги визначені у «Правилах охорони праці під час експлуатації електронно-обчислювальних машин» затверджених наказом начальника відділу організації державного нагляду в АПК та соціально-культурній сфері департаменту організації державного нагляду №65 від 26.03.2010 року [52], ГОСТ 12.1.019-79 «Электробезопасность. Общие требования и номенклатура видов защиты» [53], «ПУЕ» («Правила устрою електроустановок»), ДСанПІН 3.3.2.007 – 98 «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [51], СНиП 2.04.05-91*У «Отопление, вентиляция и кондиционирование» [54], ГОСТ 12.1.005-88 (1991) «ССБТ. Общие санитарно – гигиенические требования к воздуху рабочей зоны» [55], а також норми зазначені у ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» [56] та інші.

Передбачені заходи можна поділити на організаційні та технічні.

Для запобігання травмуванню персоналу проведено планування ергономіки приміщення, розташування офісних меблів і виробничого обладнання у приміщенні. Розводка крученої пари виконана згідно стандарту

EIA/TIA 568 (Американський стандарт проводки у комерційних приміщеннях), а також міжнародному стандарту ISO 11801 [57]. Приміщення оснащено офісними меблями, що відповідають ДСТУ EN 527-2:2007 «Меблі для адміністративних приміщень. Робочі столи. Частина 2. Вимоги механічної безпеки» [58].

З метою запобігання ураження під час виконання службових обов'язків персоналу електричним струмом передбачено низку заходів технічного характеру згідно ГОСТ 12.1.019-79 [53] і «ПУЕ» [59], відповідно яких захисне заземлення наявного електроустаткування не перевищує 4 Ом.

З організаційних заходів, передбачене проведення навчання з правил електробезпеки, перевірка знань та атестація персоналу на кваліфікаційну групу з електробезпеки, згідно НПАОП 0.00-4.12-05 «Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці» [60], з метою попередження можливих травмувань, проведення періодичних інструктажів з правил електробезпеки.

Для профілактики синдрому зап'ястного каналу у операторів ОЕМ, що використовують розроблений під час виконання проєкту програмний додаток проведено відповідний інструктаж з персоналом і внесено відповідний пункт до правил внутрішньої трудової дисципліни, а саме необхідність перерв протягом декількох хвилин після кожних 45 хвилин безперервної роботи за комп'ютером.

Для запобігання впливу факторів, що обумовлюють психофізіологічні небезпеки впроваджено низку заходів організаційного характеру, а саме:

- оптимізація і обмеження системи комунікацій (вертикальних і горизонтальних) у відповідності до потреб існуючих бізнес процесів організації;
- введення системи тайм-менеджменту;
- проведення корпоративних заходів розважального характеру для співробітників організації;

– введення системи контролю за виконанням вимог внутрішнього трудового розпорядку.

8.3 Заходи із забезпечення виробничої санітарії та гігієни праці

До санітарно-гігієнічних умов відносять:

- мікрокліматичні умови та їх забезпечення (відповідно до вимог ДСН 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень» та ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування»);
- освітленість робочої зони та її забезпечення (відповідно до вимог ДБН В.2.5-28:2018 «Природне та штучне освітлення»);
- рівень шуму в робочій зоні (відповідно до вимог ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку»);
- виробничі вібрації (відповідно до ДСН 3.3.6.039-99 «Державні санітарні норми виробничої загальної локальної вібрації» та ДСТУ ГОСТ 12.1.012:2008 «ССБП. Вібраційна безпека. Загальні вимоги»);
- електромагнітні поля та випромінювання (відповідно до ДСанНіП 3.3.6.096-2002 «Державними санітарними нормами і правилами і правилами при роботі з джерелами електромагнітних полів», ДСН 239-96 «Державні санітарні норми і правила захисту населення від впливу електромагнітних випромінювань» і ГОСТ 12.1.045-84 «ССБТ. Электростатические поля. Допустимые уровни на рабочих местах и требования к проведению контроля»);
- ергономіка робочого місця (відповідно до ДСанПіН 3.3.2.007-98 "Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин").

При організації роботи операторів ЕОМ було враховано вимоги ДСанПіН 3.3.2.007-98 «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин». Було розроблено і узгоджено інструкції щодо режиму роботи співробітників

організації з урахуванням специфіки роботи, що регламентують час початку і час закінчення робочого дня, тривалість обідньої перерви, наявність і тривалість додаткових перерв, тощо.

Відповідно до ДСН 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень» робота з розробки програмного забезпечення відноситься до легкої категорії фізичних робіт (категорія Ia). Тож норми мікроклімату повинні відповідати параметрам визначених в ГОСТ 12.1.005-88 (1991) «ССБТ. Общие санитарно-гигиенические требования к воздуху рабочей зоны» [55] (табл. 8.1).

Для забезпечення оптимального рівня параметрів повітряного середовища у приміщенні передбачена наявність системи водяного опалення приміщення, що забезпечує необхідну температуру повітря в холодний період року відповідно СНиП 2.04.05-91*У «Отопление, вентиляция и кондиционирование».

Для забезпечення оптимального температурного режиму приміщення у теплий період року встановлено систему кондиціювання повітря.

Таблиця 8.1 – Рівень параметрів повітряного середовища виробничого приміщення

Період року	Температура, °С	Відносна вологість, %	Швидкість руху, м/с
Холодний	22-24	40-60	0,1-0,3
Теплий	23-25	40-60	0,1-0,4

Необхідний повітрообмін у приміщенні визначається за формулою:

$$L = \frac{Q}{C * y * (t - t_{np})}, \quad (8.1)$$

де Q – сумарна кількість теплоти, що утворюється в приміщенні, кДж/год;

C – питома теплоємність повітря, що дорівнює $C = 1 \text{ кДж}/(\text{кг} \cdot ^\circ\text{C})$;

γ – густина зовнішнього повітря, що дорівнює $\gamma = \frac{353}{(273+t_{\text{пр}})}$, кг/м³;

t – температура повітря, що видаляється, $^\circ\text{C}$;

$t_{\text{пр}}$ – температура приточного повітря.

Сумарна кількість теплоти, що утворюється у приміщенні за годину, очікується на рівні 5500 кДж, а температура повітря, що видаляється, 27°C .

Встановлена система кондиціювання повітря забезпечує температуру приточного повітря на рівні 19°C .

Таким чином, необхідний повітрообмін у приміщенні дорівнюватиме:

$$L = \frac{5500}{1 \cdot \frac{353}{273+19} - (27-19)} = 568.8 \text{ м}^3 / \text{год}.$$

Отже для асиміляції надлишкового тепла у приміщенні в теплий період року необхідне встановлення системи кондиціювання повітря, що забезпечить повітрообмін на рівні 600 м³ за годину. На виконання цієї вимоги було встановлено мультизональний кондиціонер Mitsubishi Heavy FDTC45KXE6 з касетним внутрішнім блоком і потужністю (на охолодження) 4500 Вт.

Штучне освітлення в приміщеннях з робочими місцями, обладнаними ВДТ ЕОМ і ПЕОМ повинно здійснюватися системою загального рівномірного освітлення. Освітленість на поверхні столу в зоні розміщення документів згідно розряду зорового навантаження від 5мм клас VI освітлення повинно складати 200-300 лк згідно ДБН В.2.5-28:2018 "Природне і штучне освітлення". Якщо ці значення освітленості неможливо забезпечити системою загального освітлення, допускається використати місцеві

освітлення. При цьому місцеве освітлення встановлювати так, щоб не створювати відблисків на поверхні екрану, а освітленість екрану більше 250 лк.

Передбачається, що в приміщенні не буде використовуватися устаткування, під час роботи якого, за умови справного стану, можливе перевищення допустимих норм шуму (згідно ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку»). Враховуючи те, що потенційним джерелом шуму може виявитись друкувальний пристрій (принтер), його місце знаходження обрано таким чином, щоб він знаходився якомога далі від робочих місць персоналу.

Електромагнітне випромінювання згідно ДСН 239-96 "Державні санітарні норми і правила захисту населення від впливу електромагнітних випромінювань" має наступні показники (табл. 8.2):

Таблиця 8.2 – Електромагнітне випромінювання дисплеїв і персональних комп'ютерів.

Вид електромагнітного випромінювання	Діапазон випромінювання	
	5...2000 Гц	2000...400000 Гц
Напруженість змінного електричного поля, В/м	25	2,5
Щільність магнітного потоку, нТл	100	25

У разі невідповідності електромагнітного випромінювання санітарним нормам передбачено: фільтри антивідблисків, контрастуючих на екранах дисплеїв, які можуть одночасно захищати від електростатичного потенціалу і певною мірою, – від електричної складової змінного електромагнітного випромінювання. Також випромінювання можна ослабити за допомогою зовнішнього захисного фільтру, обов'язково з провідниковим покриттям, що

заземляється. Згідно ДСанПІН 3.3.2.007-98 "Гігієнічних вимог до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин".

Враховуючи наявність в приміщенні встановленого захисного заземлення наявного електроустаткування і відповідність комп'ютерної техніки сучасним стандартам безпеки і енергосбереження, не передбачається перевищення норм електромагнітного випромінювання зазначених у ДСанПІН 3.3.2.007–98 «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

Усі комп'ютерні монітори на робочих місцях відповідають стандарту безпеки і енергозбереження ТСО'99 і, відповідно, не викликають перевищення рівнів електромагнітного і іонізуючого випромінювання (згідно ДСанПІН 3.3.2.007–98 «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»).

Для забезпечення ергономіки робочого місця передбачено проведення інструктажу по ДСанПіН 3.3.2.007-98 "Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин".

Робоче місце має наступні показники:

- відстань між бічними поверхнями сусідніх дисплеїв – 1.2 м;
- відстань від тильної поверхні одного дисплея до екрану іншого дисплея – 2.5 м;
- прохід між рядами робочих місць має бути не менше 1 м;
- висота робочої поверхні робочого столу – 680-800 мм;
- ширина робочої поверхні робочого столу – 600-1400 мм;
- глибина робочої поверхні робочого столу – 800-1000 мм;
- простір для ніг: заввишки – 600 мм, ширина – 500 мм, глибина – 450 мм;

Конструкція робочого місця користувача ВДТ повинна забезпечувати підтримку оптимальної робочої пози з такими ергономічними характеристиками:

- ступні ніг – на підставці для ніг або на підлозі;
- стегна – в горизонтальній площині;
- передпліччя – вертикально;
- лікті – під кутом 70-90 градусів до вертикальної площини;
- зап'ястки – зігнуті під кутом не більше 20 градусів відносно горизонтальної площини;
- нахил голови – 15-20 градусів відносно вертикальної площини.

Це вимоги згідно ГОСТ 12.2.032-99 ССБТ "Система стандартів безпеки праці. Робоче місце при виконанні робіт сидячи. Загальні ергономічні вимоги".

8.4 Заходи безпеки у надзвичайних ситуаціях

8.4.1 Заходи з пожежної безпеки

Розробка програмного забезпечення супроводжується використанням не лише комп'ютерної техніки та необхідних меблів, а також веденням відповідної документації для чого в приміщенні додатково встановлюється необхідне електрообладнання (принтери) та організовується зберігання паперів. Також за необхідності може бути встановлене і додаткове електрообладнання (проектори, кухонні прилади тощо). Вся техніка, що може знаходитися в приміщенні, меблі, а також супутні матеріали, при неправильній організації пожежної безпеки можуть спричинити пожежу. Для мінімізації ризику виникнення пожежі в обчислювальному центрі маємо необхідність впровадження правил та заходів з пожежної безпеки.

Планування та впровадження заходів пожежної безпеки здійснено у відповідності до Закону України Про пожежну безпеку, Правил пожежної безпеки в Україні та вимогам відповідних нормативних актів.

Згідно НАПБ.03.001-2004 "Типові норми належності вогнегасників" тип і необхідну кількість вогнегасників визначають згідно з наступними параметрами:

- граничну захисну площу;
- клас пожежі горючих речовин і матеріалів;
- категорію приміщень і будівель за вибухопожежною і пожежною небезпекам.

При плануванні заходів з пожежної безпеки враховано, що обчислювальний центр, згідно НАПБ Б.03.002-2007 «Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою» відноситься до категорії «Д», а клас можливої пожежі, відповідно до ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги», визначається як «А» (горіння твердих речовин) та «Е» (горіння електроустановок, що перебувають під напругою). Враховуючи площу приміщення, що складає 42 м², згідно НАПБ Б.03.001-2004 «Типові норми належності вогнегасників», приміщення обладнане двома порошковими вогнегасниками ВП-5.

Розроблено графік перевірки вогнегасників і призначено відповідальну особу. Вогнегасники в приміщенні розташовані таким чином, щоб відстань від найвіддаленішого місця приміщення до найближчого вогнегасника не перевищувала 20 метрів.

Окрім цього, згідно вимог ДБН В.2.5-56:2014 «Системи протипожежного захисту», приміщення обладнане пожежною сигналізацією, що використовує комбіновані теплові мультисенсорні датчики та димові іонізаційні датчики, підключені до сигналізаційної димової пожежної установки типу СДПУ-1.

З метою запобігання пожеж в обчислювальному центрі обладнаному ПК з ВДТ передбачені наступні протипожежні заходи. На силовому обладнанні, силових та освітлювальних колах, згідно вимог пункту 3.1

«ПУЕ», встановлені захисні пристрої, що вимикають джерело живлення від ділянки електричного кола, у якій виникло коротке замикання.

Меблі та устаткування розташоване таким чином, щоб забезпечити вільний прохід у разі термінової евакуації завширшки не менш 1 метру.

Для обробки стін, підлоги та стелі приміщення використано негорючі полімерні матеріали, що зменшує ризики швидкого розповсюдження вогню у разі виникнення пожежі.

З організаційних заходів щодо забезпечення пожежної безпеки відповідно до вимог ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» розроблено план евакуації персоналу в разі виникнення пожежі і розміщено в доступному для перегляду місці, проведено інструктаж з персоналом щодо дотримання правил пожежної безпеки у приміщенні і вміння користуватися наявними засобами пожежогасіння, розроблено план навчань персоналу (з використанням учбових пожежних тривог).

8.4.2 Заходи з цивільного захисту. Єдина державна система цивільного захисту, її складові та режими функціонування

Єдина державна система цивільного захисту – це сукупність органів управління, сил і засобів центральних та місцевих органів виконавчої влади, Ради міністрів Автономної Республіки Крим, виконавчих органів рад, підприємств, установ та організацій, які забезпечують реалізацію державної політики у сфері цивільного захисту.

Відповідно до Кодексу Цивільного захисту України цивільний захист забезпечується з урахуванням особливостей, визначених Законом України "Про основи національної безпеки України", суб'єктами, уповноваженими захищати населення, території, навколишнє природне середовище і майно, згідно з вимогами цього Кодексу – у мирний час, а також в особливий період – у межах реалізації заходів держави щодо оборони України.

Координацію діяльності органів виконавчої влади у сфері цивільного захисту у межах своїх повноважень здійснюють:

- Рада національної безпеки і оборони України;
- Кабінет Міністрів України.

Для координації діяльності центральних і місцевих органів виконавчої влади, підприємств, установ та організацій, пов'язаної з техногенно-екологічною безпекою, захистом населення і територій, запобіганням і реагуванням на надзвичайні ситуації:

- Кабінетом Міністрів України утворюється Державна комісія з питань техногенно-екологічної безпеки та надзвичайних ситуацій;
- Радою міністрів Автономної Республіки Крим, обласними, Київською та Севастопольською міськими державними адміністраціями утворюються регіональні комісії з питань техногенно-екологічної безпеки та надзвичайних ситуацій;
- районними державними адміністраціями, виконавчими органами міських рад, районними у містах та селищними радами утворюються місцеві комісії з питань техногенно-екологічної безпеки та надзвичайних ситуацій;
- керівними органами підприємств, установ та організацій утворюються комісії з питань надзвичайних ситуацій.

Для координації робіт з ліквідації конкретної надзвичайної ситуації та її наслідків на державному, регіональному, місцевому та об'єктовому рівнях утворюються спеціальні комісії з ліквідації надзвичайної ситуації.

Основні завдання єдиної державної системи цивільного захисту:

- забезпечення готовності міністерств та інших центральних та місцевих органів виконавчої влади, органів місцевого самоврядування, підпорядкованих їм сил і засобів до дій, спрямованих на запобігання і реагування на надзвичайні ситуації;
- забезпечення реалізації заходів щодо запобігання виникненню надзвичайних ситуацій;

- навчання населення щодо поведінки та дій у разі виникнення надзвичайної ситуації;
- виконання державних цільових програм, спрямованих на запобігання надзвичайним ситуаціям, забезпечення сталого функціонування підприємств, установ та організацій, зменшення можливих матеріальних втрат;
- опрацювання інформації про надзвичайні ситуації, видання інформаційних матеріалів з питань захисту населення і територій від наслідків надзвичайних ситуацій;
- прогнозування і оцінка соціально-економічних наслідків надзвичайних ситуацій, визначення на основі прогнозу потреби в силах, засобах, матеріальних та фінансових ресурсах;
- створення, раціональне збереження і використання резерву матеріальних та фінансових ресурсів, необхідних для запобігання і реагування на надзвичайні ситуації;
- оповіщення населення про загрозу та виникнення надзвичайних ситуацій, своєчасне та достовірне інформування про фактичну обстановку і вжиті заходи;
- захист населення у разі виникнення надзвичайних ситуацій;
- проведення рятувальних та інших невідкладних робіт щодо ліквідації наслідків надзвичайних ситуацій, організація життєзабезпечення постраждалого населення;
- пом'якшення можливих наслідків надзвичайних ситуацій у разі їх виникнення;
- здійснення заходів щодо соціального захисту постраждалого населення;
- реалізація визначених законом прав у сфері захисту населення від наслідків надзвичайних ситуацій, в тому числі осіб (чи їх сімей), що брали безпосередню участь у ліквідації цих ситуацій;
- інші завдання, визначені законом.

Єдина державна система цивільного захисту залежно від масштабів і особливостей надзвичайної ситуації, що прогнозується або виникла, може функціонувати в наступних режимах:

- повсякденного функціонування;
- підвищеної готовності;
- надзвичайної ситуації;
- надзвичайного стану.

Режим повсякденного функціонування – встановлюється за умов нормальної виробничо-промислової, радіаційної, хімічної, сейсмічної, гідрогеологічної, гідрометеорологічної, техногенної та пожежної обстановки та за відсутності епідемій, епізоотій, епіфітотій.

Режим підвищеної готовності – встановлюється тимчасово, у разі загрози виникнення надзвичайної ситуації, у повному обсязі або частково для окремих територіальних підсистем (ОТП) ЄДСЦЗ (єдиної державної системи цивільного захисту) за рішенням відповідно Кабінету Міністрів України, Ради міністрів Автономної Республіки Крим, обласних, Київської чи Севастопольської міських державних адміністрацій.

Режим надзвичайної ситуації – встановлюється тимчасово, у разі виникнення надзвичайної ситуації, у повному обсязі або частково для ОТП ЄДСЦЗ (окремих територіальних підсистем єдиної державної системи цивільного захисту) за рішенням відповідно Кабінету Міністрів України, Ради міністрів Автономної Республіки Крим, обласних, Київської чи Севастопольської міських державних адміністрацій.

Режим надзвичайного стану – встановлюється тимчасово, у повному обсязі або частково, у межах території, на якій введено правовий режим надзвичайного стану для ОТП ЄДСЦЗ відповідно до Закону України «Про правовий режим надзвичайного стану».

ВИСНОВКИ

Під час виконання проєкту було досліджено предметну область, вивчено проблематику поставленої задачі, наведено її актуальність доцільність.

Було проаналізовано класичні методи, що можуть бути застосовані для задачі класифікації, а саме Байєсова мережа, NNge, Random Forest, HMM, RBFNetwork, AdaBoost та ENORA. Визначено, що вони не видають задовільної точності при класифікації даних. Найкращим чином себе показує Байєсова мережа та алгоритм Random Forest. Також для порівняння було проаналізовано багатошаровий перцептрон, згорткову нейронну мережу, мережу LSTM та повнозв'язну НМ. З яких найкращу точність показала мережа LSTM, але для задачі класифікації має доволі складну структуру.

Проаналізовано сучасні літературні джерела щодо підходів до модифікації та застосування програмних засобів для прогнозування цукрового діабету. В результаті чого виконано постановку завдання.

В результаті виконання роботи запропоновано спосіб навчання нейронної мережі за допомогою генетичного алгоритму, для якого запропоновано спосіб вибору осіб до батьківських пар на основі їх неспорідненості та два способи мутації генів хромосом. Запропоновано два підходи до оператора мутації. При мутації значення обраних генів обчислюються на основі їх середніх значень після чого додається, віднімається або обнуляється значення.

Під час створення і тренування моделі на основі мережі довгої короткочасної пам'яті було досліджено можливість використання методу рою часток і генетичного алгоритму для налаштування ваг мережі. За результатами тестування та проведення експериментального дослідження визначено, що еволюційні методи значно підвищують точність класифікації. Точність класифікації повнозв'язного перцептрона склала 78.78%. Точність

класифікації повнозв'язного персептрона, що навчався за допомогою генетичного алгоритму склала 89.3%, за допомогою PSO – 90.82%. Повнозв'язний персептрон, що навчався за допомогою генетичного алгоритму з запропонованими операторами вибору та першого підходу до мутації показав точність в 91.3%, а з другим підходом до виконання мутації точність класифікатора склала 89.97%. За результатами досліджень запропонованого генетичного алгоритму можна сказати, що спостерігається поліпшення за точністю класифікації, а саме у першому випадку в порівнянні з класичним ГА на 2%, в другому випадку на 0,67%.

Було проаналізовано економічний аспект проєкту, розраховано витрати на розробку проєкту, обґрунтовано його економічну доцільність.

Розроблені рекомендації щодо охорони праці та безпеки у надзвичайних ситуаціях.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Краткая медицинская энциклопедия: диагностика [Электронный ресурс]. – Режим доступа: <http://www.golkom.ru/kme/05/1-408-3-1.html>
2. Експертні системи в медицині: навчальний посібник для студентів вищих навчальних закладів / Ю. С. Синєкоп, А. М. Продеус, Є. Я. Кісельов, М. М. Баран. – Запоріжжя: видавництво ЗДІА, 2014. – 332 с.
3. Диабет: основные факты [Электронный ресурс]. – Режим доступа: <https://www.who.int/ru/news-room/fact-sheets/detail/diabetes>.
4. Тарасенко Н.А. Сахарный диабет: действительность, прогнозы, профилактика / Н.А. Тарасенко // Современные проблемы науки и образования. – 2017. – № 6. – С. 9.
5. Non-nested generalization (NNGE) algorithm for efficient and early detection of diabetes / [D. E. Gbenga, J. Hemanth, H. Chiroma та ін.]. // Information technology and intelligent transportation system: proceedings of the 3rd international conference on information technology and intelligent transportation system. – 2018. – №314. – P. 233–242.
6. Penentuan harga jual property secara otomatis menggunakan metode probabilistic neural network [Electronic resource]. – Access mode: <https://docplayer.info/176898-Penentuan-harga-jual-properti-secara-otomatis-menggunakan-metode-probabilistic-neural-network.html>.
7. Sushilkumar R. Successful assessment of categorization od Indian news JRip and NNge Algorithm / R. Kalmegh Sushilkumar // International journal of emerging technology and advanced engeneering. – 2014. – №4. – P. 395–402.
8. Sushilkumar. R. Kalmegh Comparative analisis of Weka data mining algorithm RandomForest, RandomTree and LADTree for classification of indigenous news data / R. Kalmegh Sushilkumar // International journal of emerging technology and advanced engeneering. – 2015. – №5. – P. 507-517.
9. Random Forest classifier – Machine [Electronic resource]. – Access mode: <https://www.globalsoftwaresupport.com/random-forest-classifier/>.

10. Гефке Д. А. Применение скрытых марковских моделей для распознавания звуковых последовательностей / Д. А. Гефке, П. М. Зацепин. // Управление, вычислительная техника и информатика. – 2012. – №2. – С. 72–76.
11. Севергин С.В. Распознавание частей речи слов при помощи скрытой марковской модели и открытого корпуса русского языка / С. В. Севергин. – ТГУ: Томск. – 2014. – 35 с.
12. Mark J. L . Orr Introduction to radial basis function networks // Center for cognitive science / Mark J. L . Orr. – Edinburgh: University of Edinburgh, 1996. – 67 p.
13. John A. Bullinaria Radial basis function network: Algorithms / John A. Bullinaria // Neural Computation: lecture 14. – 2015. – №1. – P. 18.
14. Robert E. Schapine Explaining AdaBoost / Robert E. Schapine. – Princeton: Princeton University. – 2012. – 16 p.
15. Webb I. G. MultiBoosting: A technique for combining boosting and wagging / I. G. Webb // Machine Learning. – 2000. – № 40. – P. 156-196.
16. Panda M. Combining mulri-objective evolutionary algorithm with averaged one-dependence estimators for big data analytics / M. Panda // International journal computation intelligence studies. – 2018. – №7. – P. 1-18.
17. Поцелуев П. А. Реализация игрового ИИ для аэрохоккея с помощью нейронных сетей / П. А. Поцелуев. – Санкт-Петербург: СПбПУ, 2014. – 10 с.
18. Субботін С. О. Нейронні мережі / С. О. Субботін, А. О. Олійник. – Запоріжжя: ЗНТУ. – 2014. – 131 с.
19. Вакуленко С. А. Практический курс по нейронным сетям / С.А. Вакуленко, А. А. Жихарева. – Санкт-Петербург: Университет ИТМО. – 2018. – 71 с.
20. Liu T. Implementation of training convolutional neural networks / T. Liu, S. Fang, Y. Zhao, P. Wang, J. Zhang. – Beijing: University of Chinese Academy of Science. – 2015. – P. 10.

21. Dual temporal scale convolutional neural network for micro-expression recognition [Electronic resource]. – Access mode: <https://www.frontiersin.org/articles/10.3389/fpsyg.2017.01745/full>.
22. Hemanth D. Jude A modified deep convolutional neural network for abnormal brain image classification / D. Jude Hemanth, J. Anitha, A. Naaji, O. Geman, D. E. Popescu, L. H. Son // IEEE Access. – 2019. – №7. – P. 4275-4283.
23. Джулли А. Библиотека Keras –инструмент глубокого обучения. Реализация нейронных сетей с помощью библиотек Theano и TensorFlow / А. Джулли, С. Пал. – Москва: ДМК Пресс. – 2018. – 293 с.
24. Долгая краткосрочная память [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Долгая_краткосрочная_память.
25. Farre P. Dense neural networks for predicting chromatin conformation / P. Farre, A. Heurteau, O. Cuvier, E. Emberly // BMC Bioinformatics 19. – 2018. – №372. – P. 17.
26. Jahani M. Comparison of predictive models for the early diagnosis of diabetes / M. Jahani, M. Mahdi // Healthcare informatics research. – 2016. – P. 95-100.
27. Parte R. S. Non-invasive method for diabetes detection using CNN and SVM classifier / R. S. Parte, A. Patil, A. Patil, A. Kad, S. Kharat // International journal of research in engineering, science and management. – 2019. – №2. – P. 659-661.
28. Lekha S. A Novel 1-D convolution neural network with SVM architecture for real-time detection applications / S. Lekha, M. Suchetha // School of electronics and communication. – 2020. – P. 20.
29. Lekha S. Real-time non-invasive detection and classification of diabetes using modified convolution neural network / S. Lekha, M. Suchetha // IEEE journal of biomedical and health informatics. – 2019. – №22. – P. 8.
30. Optimal predictive analytics of Pima diabetics using deep learning / [S. Ramesh, H. Balaji, N. Iyengar та ил.]. // International journal of databases theory and applications. – 2017. – №10. – P. 47–62.

31. Predictive models for diabetes mellitus using machine learning techniques / [H. Lai, H. Huang, K. Keshavjee та ін.]. // BMC endocrine disorders. – 2019. – №19. – P. 9.

32. Nimmagadda S. P. Improved diabetes prediction model for predicting Type-II diabetes / S. P. Nimmagadda, S. Yeruva, R. Siempu // International journal of innovative technology and exploring engineering. – 2019. – №8. – P. 230-235.

33. H2RM: A hybrid rough set reasoning model for prediction and management of diabetes mellitus / [R. Ali, J. Hussain, M. H. Siddiqi. // Sensors. – 2015. – №15. – P. 32.

34. Pavate A. Risk prediction of disease complications in type 2 diabetes patients using soft computing techniques / A. Pavate, N. Ansari // First international conference on advances in computing and communications. – 2015. – P. 370-375.

35. Pulse-wave-pattern classification with a convolutional neural network / [G. Li, K. Watanabe, H. Anzai, X. Song] // Science reports. – 2019. – №9. – P. 11.

36. Jlang Han-Jing Predicting drug-disease association via sigmoid kernel-based convolutional neural networks / Han-Jing Jlang, Zhu-Hong You, Yu-An Huang //Jiang et al. J Transl Med. – 2019. – №17. – P. 11.

37. Al-Azzawi D. S. Human age and gender prediction using deep multi-task convolutional neural network / D. S. Al-Azzawi // ISSN. – 2019. – №54. – P. 11.

38. Уніфікований клінічний протокол первинної та вторинної (спеціалізованої) медичної допомоги [Електронний ресурс]. – Режим доступу: <https://zakon.rada.gov.ua/rada/show/v1118282-12#n29>

39. Рейтинг языков программирования для ИИ и машинного обучения [Электронный ресурс]. – Режим доступа: <http://digitrode.ru/articles/1643-yazyki-programmirovaniya-dlya-iskusstvennogo-intellekta.html>

40. R – язык статистической обработки данных. Часть 1/3 [Электронный ресурс]. – Режим доступа: <https://medium.com/nuances-of->

programming/r-язык-для-статистической-обработки-данных-часть-1-3-6e5fcaada586.

41. Сузи Р. А. Язык программирования Python / Р. А. Сузи. – Nerezus'a. – 206 с.

42. McGrath M. Python in easy steps / M. McGrath. – Leamingthon. – 2013. – 194 p.

43. Дьяконов А. Г. Анализ данных, обучение по прецедентам, логических игры, системы WEKA, RapidMiner и Matlab: уч.пособие / А. Г. Дьяконов. – М: МГУ. – 2010. – 278 с.

44. Мюллер А. Введение в машинное обучение с помощью Python / А. Мюллер. С. Гвидо. – Москва: Диалектика. – 2016-2017. – 393 с.

45. Atienza R. Advanced deep learning with Keras / R. Atienza, – Birmingham. – Mumbai: Packt Publishing. – 2018. – 369 p.

46. Введение в Scikit-learn [Электронный ресурс]. – Режим доступа: <https://neurohive.io/ru/osnovy-data-science/vvedenie-v-scikit-learn/>

47. Microsoft Visual Studio [Electronic resource]. – Access mode: <https://open-file.ru/programs/microsoft-visual-studio>

48. Олійник А. О Еволюційні обчислення та програмування: навчальний посібник / А. О. Олійник, С. О. Субботін, О. О. Олійник. – ЗНТУ. – 2010. – 324 с.

49. Родственное спаривание [Электронный ресурс]. – Режим доступа: http://www.kgau.ru/distance/zif_03/razvedenie-110401/06_02.html

50. Метод рою частиц [Электронный ресурс]. – Режим доступа: <https://jenyay.net/Programming/ParticleSwarm>

51. Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу : Державні санітарні норми та правила від 30.05.2014 р. № z0472-14. Редакція від: 30.05.2014. [Електронний ресурс]. – Режим доступу: <http://zakon2.rada.gov.ua/laws/show/z0472-14>

52. Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин. Наказ начальника відділу організації державного нагляду в АПК та соціально-культурній сфері департаменту організації державного нагляду №65 від 26.03.2010 року [Електронний ресурс]. – Режим доступу: <http://zakon5.rada.gov.ua/laws/show/z0293-10>.

53. ГОСТ 12.1.019-79* ССБТ. Электробезопасность. Общие требования и номенклатура видов защиты [Текст]. – Москва: ИПК Издательство стандартов № 1996, ИПК Издательство стандартов № 2001. – 18 с.

54. СНиП 2.04.05-91*У Отопление, вентиляция и кондиционирование. Строительные нормы отопление, вентиляция и кондиционирование, 1996. [Электронный ресурс]. – Режим доступа: https://dnaop.com/html/1671/doc-СНиП_2.04.05-91.

55. ГОСТ 12.1.005-88 ССБТ. Общие санитарно-гигиенические требования к воздуху рабочей зоны, 1998. [Электронный ресурс]. – Режим доступа: https://znaytovar.ru/gost/2/GOST_12100588_SSBT_Obshhie_san.html.

56. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку, 1999. [Электронный ресурс]. – Режим доступа: <http://arm.te.ua/docs/DSN-3.3.6.037-99.pdf>.

57. ISO 11801 Стандарт телекомунікаційної інфраструктури комерційних будівель [Электронный ресурс]. – Режим доступа: <https://inkabel.ru/assets/files/ISO-IEC-11801.pdf>.

58. ДСТУ EN 527-2:2007 Меблі для адміністративних приміщень. Робочі столи. Частина 2. Вимоги механічної безпеки [Текст] – К.: Держспоживстандарт України. – 2007. – 8 с.

59. Правила улаштування електроустановок [Текст] : ПУЕ-2017. – На заміну ПУЕ-86 ; чинний з 2017-08-21. – К. : Міненерговугілля України, 2017. – 617 с.

60. Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці НПАОП 0.00-4.12-05. – На заміну

ДНАОП 0.00-4.12-99, ДНАОП 0.00-8.01-93 ; чинний від 2005-02-26. – К. : Держнаглядохоронпраці України. – 2005. [Электронный ресурс]. – Режим доступа: <http://zakon.rada.gov.ua/laws/show/z0231-05>.

ДОДАТОК А
Технічне завдання

Вступ

У роботі розглядається галузь інженерії програмного забезпечення, пов'язана з параметричним синтезом нейронних мереж на основі еволюційного підходу та її застосуванням при діагностування цукрового діабету.

A.1 Підстави для розробки

Підставою для розробки є завдання на магістерську дипломну роботу «Дослідження та програмна реалізація інтелектуальних методів для медичного діагностування» відповідно до наказу №301 по Національному університету «Запорізька політехніка».

A.2 Призначення розробки

Призначенням розробки є програмний продукт, який би реалізував такі методи для параметричного синтезу нейронних мереж, як еволюційні методи. Програмний продукт повинен ефективно реалізовувати представлені методи і дозволяти користувачеві ефективно розв'язувати задачу аналізу даних для медичного діагностування, зокрема цукрового діабету.

A.3 Основні вимоги до програми, що розроблюється

A.3.1 Вимоги до функціональних характеристик

Розроблюване програмне забезпечення повинно відповідати таким вимогам:

- відповідність темі проєкту;
- коректність роботи;
- можливість завантаження даних з файлу;

- можливість роботи у консольному режимі;
- можливість виводу вихідних даних у графічному та текстовому вигляді;
- можливість збереження графічної інформації.

A.3.2 Вимоги до надійності

Програма повинна забезпечувати достатній рівень надійності. В процесі роботи можуть виникати збійні ситуації, що мають відповідні повідомлення, що описані в керівництві користувача та програміста. Програмний продукт повинен забезпечувати досягнення відносно якісного результату за умов слідування користувачем правил керівництва.

A.3.3 Умови експлуатації

Для роботи з програмою користувач має володіти достатніми знаннями для коректного вводу інформації. Програма може бути записана як на гнучкому так і на магнітному диску. Експлуатація здійснюється відповідно до експлуатаційної документації на розроблену програму, що відповідає стандартам, а також містить інформацію, необхідну для освоєння і зручної експлуатації програмного продукту.

A.3.4 Вимоги до складу та параметрів технічних засобів

Для експлуатації програмного такі мінімальні програмно-апаратні засоби як: IBM-сумісний комп'ютер під керуванням ОС сімейства Windows з процесором Intel Core 3, об'єм оперативної пам'яті – 4 Гб.

Для роботи з програмним продуктом попередньо необхідно встановити наступні залежності:

- мова програмування Python 3.7;

- нейромережева бібліотека Keras;
- бібліотека машинного навчання Scikit-learn;
- бібліотека для побудови графіків та візуалізації даних Matplotlib;
- бібліотека, що дозволяє ефективно працювати з матрицями, таблицями та числовими даними NumPy.

A.3.5 Вимоги до програмної документації

Програмний продукт має поставлятися з технічним завданням та керівництвами програміста та користувача.

A.4 Стадії та етапи розробки

Програмний продукт повинен бути виконаний у 12-тижневий строк з моменту отримання завдання. Проєкт складається з таких етапів:

- аналіз предметної області;
- розробка технічного завдання;
- проєктування моделі програмного продукту;
- реалізація програмного продукту;
- тестування та експериментальне дослідження.

A.5 Порядок контролю та приймання

Магістерська робота допускається до захисту керівником, консультантом та нормо контролером, затверджується завідувачем кафедри та захищається у присутності членів екзаменаційної комісії.

ДОДАТОК Б
Код програми

Б.1 Файл «GA_standart»

```

import numpy as np
from copy import deepcopy
from model.root import BaseAlgo

class StandartGA(BaseAlgo):
    ID_FITNESS = 1

    def __init__(self, base_algo_param=None, ga_param = None):
        BaseAlgo.__init__(self, base_algo_param)
        self.epoch = ga_param["epoch"]
        self.pop_size = ga_param["pop_size"]
        self.pc = ga_param["pc"]
        self.probmute = ga_param["probmute"]

    ### Selection
    def _get_index_selection__(self, list_fitness, summ_fitness):
        r = np.random.uniform(low=0, high=summ_fitness)
        for idk, fit in enumerate(list_fitness):
            r = r + fit
            if r > summ_fitness:
                return idk

    ### Crossover
    def _crossover__(self, dad1=None, dad2=None):
        r = np.random.uniform()
        # g1 = g2 when r = 0.5
        g1 = np.multiply(r, dad1) + np.multiply((1 - r), dad2)
        g2 = np.multiply(r, dad2) + np.multiply((1 - r), dad1)
        return g1, g2

    ### Mutation
    def _mutation__(self, parent, index):
        g = deepcopy(parent)
        g[index] = np.random.uniform(self.domain_range[0], self.domain_range[1])
        return g

    def _create_next_popul__(self, popul):
        next_popul = []
        list_fitness = [popul[i][self.ID_FITNESS][self.ID_ERROR] for i in
range(self.pop_size)]
        fitness_summ = sum(list_fitness)
        while (len(next_popul) < self.pop_size):
            ### Selection
            ch1 = popul[self._get_index_selection__(list_fitness, fitness_summ)]
            ch2 = popul[self._get_index_selection__(list_fitness, fitness_summ)]

            g1, g2 = deepcopy(ch1[0]), deepcopy(ch2[0])
            ### Crossover
            if np.random.uniform() < self.pc:
                g1, g2 = self._crossover__(ch1[0], ch2[0])
            ### Mutation
            for id in range(0, self.problem_size):
                if np.random.uniform() < self.probmute:
                    g1 = self._mutation__(g1, id)
                if np.random.uniform() < self.probmute:
                    g2 = self._mutation__(g2, id)
            ch1_new = [g1, self._fitness_model__(g1, minmax=1)]
            ch2_new = [g2, self._fitness_model__(g2, minmax=1)]
            next_popul.append(ch1_new)
            next_popul.append(ch2_new)
        return next_popul

    def _train__(self):
        best_train = [None, [-1, -1]]
        popul = [ self._create_solution__(minmax=1) for _ in range(self.pop_size) ]

        for j in range(0, self.epoch):
            # Next generations

```

```

        pop = deepcopy(self._create_next_popul__(pop))
        current_best = self._get_global_best__(pop=pop,
id_fitness=self.ID_FITNESS, id_best=self.ID_MAX_PROBLEM)
        if current_best[self.ID_FITNESS][self.ID_ERROR] >
best_train[self.ID_FITNESS][self.ID_ERROR]:
            best_train = current_best
        if self.print_train:
            print("> Epoch {0}: Best fitness {1}".format(j + 1, 1.0 /
best_train[self.ID_FITNESS][self.ID_ERROR]))
            self.loss_train.append(np.power(best_train[self.ID_FITNESS], -1))

    return best_train[0], self.loss_train

```

Б.2 Файл «PSO»

```

import numpy as np
from copy import deepcopy
from model.root import BaseAlgo

class StandartPSO(BaseAlgo):
    ID_CURRENT_POSITION = 0
    ID_PAST_POSITION_BEST = 1
    ID_VECTOR_V = 2
    ID_CURRENT_FITNESS = 3
    ID_PAST_FITNESS = 4

    def __init__(self, base_algo_param=None, pso_param = None):
        BaseAlgo.__init__(self, base_algo_param)
        self.epoch = pso_param["epoch"]
        self.pop_size = pso_param["pop_size"]
        self.c1 = pso_param["c_minmax"][0]
        self.c2 = pso_param["c_minmax"][1]
        self.w_min = pso_param["w_minmax"][0]
        self.w_max = pso_param["w_minmax"][1]

    def _create_solution__(self, minmax=0):
        x = np.random.uniform(self.domain_range[0], self.domain_range[1],
(self.problem_size, 1))
        x_past_best = deepcopy(x)
        v = np.zeros((len(x), 1))
        x_fitness = self._fitness_model__(model=x, minmax=minmax)
        x_past_fitness = deepcopy(x_fitness)
        return [x, x_past_best, v, x_fitness, x_past_fitness]

    def _train__(self):
        pop = [self._create_solution__(minmax=0) for _ in range(self.pop_size)]
        gbest = self._get_global_best__(pop=pop, id_fitness=self.ID_CURRENT_FITNESS,
id_best=self.ID_MIN_PROBLEM)

        for i in range(self.epoch):
            w = (self.epoch - i) / self.epoch * (self.w_max - self.w_min) + self.w_min
            for j in range(self.pop_size):
                r1 = np.random.random_sample()
                r2 = np.random.random_sample()
                v_sau = w * pop[j][self.ID_VECTOR_V] + self.c1 * r1 * \
                    (pop[j][self.ID_PAST_POSITION_BEST] -
pop[j][self.ID_CURRENT_POSITION]) \
                    + self.c2 * r2 * (gbest[self.ID_CURRENT_POSITION] -
pop[j][self.ID_CURRENT_POSITION])

                x_sau = pop[j][self.ID_CURRENT_POSITION] + v_sau #
X(sau) = X(truoc) + V(sau) * deltaT (deltaT = 1)
                fitn_sau = self._fitness_model__(model=x_sau, minmax=0)
                fitn_truoc = pop[j][self.ID_PAST_FITNESS]
                pop[j][self.ID_CURRENT_POSITION] = deepcopy(x_sau)
                pop[j][self.ID_VECTOR_V] = deepcopy(v_sau)
                pop[j][self.ID_CURRENT_FITNESS] = fitn_sau

```

```

        if fitn_sau[self.ID_ERROR] < fitn_truoc[self.ID_ERROR]:
            pop[j][self.ID_PAST_POSITION_BEST] = deepcopy(x_sau)
            pop[j][self.ID_PAST_FITNESS] = fitn_sau
        gbest = self._get_global_best__(pop=pop,
id_fitness=self.ID_CURRENT_FITNESS, id_best=self.ID_MIN_PROBLEM)

        self.loss_train.append(gbest[self.ID_CURRENT_FITNESS])

        if self.print_train:

            print("Epoch: {0}, [MSE, MAE]: {1}".format(i+1,
gbest[self.ID_CURRENT_FITNESS]))

        return gbest[self.ID_CURRENT_POSITION], self.loss_train

```

Б.3 Файл «GA_first_modification»

```

import numpy as np
from copy import deepcopy
from model.root import BaseAlgo

class GA_first(BaseAlgo):
    ID_FITNESS = 1

    def __init__(self, base_algo_param=None, ga_param = None):
        BaseAlgo.__init__(self, base_algo_param)
        self.epoch = ga_param["epoch"]
        self.pop_size = ga_param["pop_size"]
        self.pc = ga_param["pc"]
        self.probmute = ga_param["probmute"]

    ### Selection
    def _get_index_selection__(self, list_fitness, summ_fitness):
        r = np.random.uniform(low=0, high=summ_fitness)
        for idx, fit in enumerate(list_fitness):
            r = r + fit
            if r > summ_fitness:
                return idx

    ### Crossover
    def _crossover__(self, dad1=None, dad2=None):
        r = np.random.uniform() # g1 = g2 when r =0.5
        g1 = np.multiply(r, dad1) + np.multiply((1 - r), dad2)
        g2 = np.multiply(r, dad2) + np.multiply((1 - r), dad1)
        return g1, g2

    ### Mutation
    def _first_mutation__(self, parent, index):
        w = deepcopy(parent)
        g1[index] = np.random.uniform(self.domain_range[0], self.domain_range[1])
        g2[index] = np.random.uniform(self.domain_range[0], self.domain_range[1])
        average_value_gen = (g1+g2)/2;
        g1 = average_value_gen + random.randrange(0, 1, 0.1);
        g2 = average_value_gen - random.randrange(0, 1, 0.1);
        return g1, g1

    def _create_next_popul__(self, pop):
        next_popul= []

        list_fitness = [pop[i][self.ID_FITNESS][self.ID_ERROR] for i in
range(self.pop_size)]
        fitness_summ = sum(list_fitness)
        while (len(next_popul) < self.pop_size):
            ### Selection
            ch1 = pop[self._get_index_selection__(list_fitness, fitness_summ)]
            ch2 = pop[self._get_index_selection__(list_fitness, fitness_summ)]

            g1, g2 = deepcopy(ch1[0]), deepcopy(ch2[0])

```

```

    ### Crossover
    if np.random.uniform() < self.pc:
        g1, g2 = self._crossover__(ch1[0], ch2[0])
    ### Mutation
    g = deepcopy(parent)
    g1[index] = np.random.uniform(self.domain_range[0],
    self.domain_range[1])
    g2[index] = np.random.uniform(self.domain_range[0],
    self.domain_range[1])
    avarage_value_gen = (g1+g2)/2;
    g1 = avarage_value_gen + random.randrange(0, 1, 0.1);
    g2 = avarage_value_gen - random.randrange(0, 1, 0.1);
    return g
    for id in range(0, self.problem_size):
        if np.random.uniform() < self.probmute:
            g1 = self._fisrt_mutation__(g1, id)
        if np.random.uniform() < self.probmute:
            g2 = self._fisrt_mutation__(g2, id)

    ch1_new = [g1, self._fitness_model__(g1, minmax=1)]
    ch2_new = [g2, self._fitness_model__(g2, minmax=1)]
    next_popul.append(ch1_new)
    next_popul.append(ch2_new)
    return next_popul

def _train__(self):
    best_train = [None, [-1, -1] ]
    pop = [ self._create_solution__(minmax=1) for _ in range(self.pop_size) ]

    for j in range(0, self.epoch):
        # Next generations
        pop = deepcopy(self._create_next_popul__(pop))
        current_best = self._get_global_best__(pop=pop,
        id_fitness=self.ID_FITNESS, id_best=self.ID_MAX_PROBLEM)
        if current_best[self.ID_FITNESS][self.ID_ERROR] >
        best_train[self.ID_FITNESS][self.ID_ERROR]:
            best_train = current_best
        if self.print_train:
            print("> Epoch {0}: Best training fitness {1}".format(j + 1, 1.0 /
            best_train[self.ID_FITNESS][self.ID_ERROR]))
            self.loss_train.append(np.power(best_train[self.ID_FITNESS], -1))

    return best_train[0], self.loss_train

```

Б.4 Файл «GA_second_modification»

```

import numpy as np
from copy import deepcopy
from model.root import BaseAlgo

class GA_second(BaseAlgo):
    ID_FITNESS = 1

    def __init__(self, base_algo_param=None, ga_param = None):
        BaseAlgo.__init__(self, base_algo_param)
        self.epoch = ga_param["epoch"]
        self.pop_size = ga_param["pop_size"]
        self.pc = ga_param["pc"]
        self.probmute = ga_param["probmute"]

    ### Selection
    def _get_index_selection__(self, list_fitness, summ_fitness):
        r = np.random.uniform(low=0, high=summ_fitness)
        for idk, fit in enumerate(list_fitness):
            r = r + fit
            if r > summ_fitness:
                return idk

```

```

    ### Crossover
    def _crossover__(self, dad1=None, dad2=None):
        r = np.random.uniform() # g1 = g2 when r =0.5
        g1 = np.multiply(r, dad1) + np.multiply((1 - r), dad2)
        g2 = np.multiply(r, dad2) + np.multiply((1 - r), dad1)
        return g1, g2

    ### Mutation
    def _second_mutation__(self, parent, index):
        g = deepcopy(parent)
        g1[index] = np.random.uniform(self.domain_range[0], self.domain_range[1])
        g2[index] = np.random.uniform(self.domain_range[0], self.domain_range[1])
        avarage_value_gen = (g1+g2)/2;
        g1 = avarage_value_gen + random.randrange(0, 1, 0.1);
        g2 = 0;
        return g1, g2

    def _create_next_popul__(self, popul):
        next_popul= []

        list_fitness = [popul[i][self.ID_FITNESS][self.ID_ERROR] for i in
range(self.pop_size)]
        fitness_summ = sum(list_fitness)
        while (len(next_popul) < self.pop_size):
            ### Selection
            ch1 = popul[self._get_index_selection__(list_fitness, fitness_summ)]
            ch2 = popul[self._get_index_selection__(list_fitness, fitness_summ)]

            g1, g2 = deepcopy(ch1[0]), deepcopy(ch2[0])
            ### Crossover
            if np.random.uniform() < self.pc:
                g1, g2 = self._crossover__(ch1[0], ch2[0])

            ### Mutation
            for id in range(0, self.problem_size):
                if np.random.uniform() < self.probmute:
                    g1 = self._second_mutation__(g1, id)
                if np.random.uniform() < self.probmute:
                    g2 = self._second_mutation__(g2, id)

            ch1_new = [g1, self._fitness_model__(g1, minmax=1)]
            ch2_new = [g2, self._fitness_model__(g2, minmax=1)]
            next_popul.append(ch1_new)
            next_popul.append(ch2_new)
        return next_popul

    def _train__(self):
        best_train = [None, [-1, -1] ]
        popul = [ self._create_solution__(minmax=1) for _ in range(self.pop_size) ]

        for j in range(0, self.epoch):
            # Next generations
            popul = deepcopy(self._create_next_popul__(popul))
            current_best = self._get_global_best__(popul=popul,
id_fitness=self.ID_FITNESS, id_best=self.ID_MAX_PROBLEM)
            if current_best[self.ID_FITNESS][self.ID_ERROR] >
best_train[self.ID_FITNESS][self.ID_ERROR]:
                best_train = current_best
            if self.print_train:
                print("> Epoch {0}: Best training fitness {1}".format(j + 1, 1.0 /
best_train[self.ID_FITNESS][self.ID_ERROR]))
                self.loss_train.append(np.power(best_train[self.ID_FITNESS], -1))

        return best_train[0], self.loss_train

```

Б.5 Файл «ANN»

```

from keras.models import Sequential
from keras.layers import Dense

```

```

import numpy
from matplotlib import pyplot as plt
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from model.optimizer import GA_standart,
GA_first_modified_mutation, GA_second_modified_mutation, PSO

numpy.random.seed(2)

dataset = numpy.loadtxt("../dataset/diabetes.csv", delimiter=",")
# розбиття датасету на вхідні та вихідні параметри
X, Y = dataset[:,0:8], dataset[:,8]

# модель
model = Sequential()
model.add(Dense(8, input_dim=8, activation='relu'))
model.add(Dense(15, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss="binary_crossentropy", optimizer="adam", metrics=['accuracy'])

# звичайне навчання моделі
history = model.fit(X, Y, validation_split=0.05, epochs = 200, batch_size=10)

print(history.history.keys())
# 'Graphical Reporting'
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc = 'upper left')
plt.savefig('accuracy.png')
plt.show()

# 'Outputs plotting'
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc = 'upper left')
plt.savefig('loss.png')
plt.show()
# 'model saving'
#####model.save('pima_indian.model')

# 'Curva ROC Curve'
probs = model.predict_proba(X)
probs = probs[:,0]
auc = roc_auc_score(Y, probs)
print('AUC: %.3f' % auc)
fpr, tpr, thresholds = roc_curve(Y, probs)
plt.plot([0, 1], [0, 1], linestyle = '--')
plt.plot(fpr, tpr, marker = '.')
plt.title('ROC-curve')
plt.ylabel('True positive rate')
plt.xlabel('False positive rate')
plt.savefig('roc.png')
plt.show()
scores = model.evaluate(X, Y)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»

кафедра програмних засобів

Дипломна робота магістра
на тему

**ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
ІНТЕЛЕКТУАЛЬНИХ МЕТОДІВ ДЛЯ МЕДИЧНОГО
ДІАГНОСТУВАННЯ**

Виконала
Ст. гр. КНТ-119м
Керівник
к.т.н., доцент

Саман М. В.

Колпакова Т. О.

2020 р.

Рисунок В.1 – Слайд №1

ОБ'ЄКТ, ПРЕДМЕТ ТА МЕТА ДОСЛІДЖЕННЯ

- Об'єкт дослідження – процес прогнозування та виявлення цукрового діабету за клінічними показниками стану здоров'я людини
- Предмет дослідження – методи для діагностування захворюваності
- Мета роботи – дослідження, модифікація та програмна реалізація методів діагностування ризику виникнення цукрового діабету.

2

Рисунок В.2 – Слайд №2

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

- Медичне діагностування - це розділ клінічної медицини, що вивчає склад та методи процесу розпізнавання захворювань з метою встановлення правильного діагнозу.
- Сьогодні лікарі у своїй роботі часто використовують програмне забезпечення, що здатне автоматизувати роботу та здатне встановлювати попередній діагноз. Такими системами є МІС – медичні інформаційні системи.

3

Рисунок В.3 – Слайд №3

АНАЛІЗ МЕТОДІВ ДЛЯ ПРОВЕДЕННЯ ДІАГНОСТУВАННЯ

№ п/п	Назва методу/алгоритму	Параметри			
		Cross-validation (10 folds)		Training/Test percentage split (80%)	
		Значення, %	Час роботи алгоритму, хв	Значення, %	Час роботи алгоритму, хв
1	Байєсова мережа	75.25	0,02	76.30	0,01
2	NNge	72.84	0,06	74.67	0,01
3	Random Forest	76.10	0,3	76.69	0,03
4	HMM	65.11	0,02	65.10	0.002
5	RBFNetwork	75.94	0,56	75.59	0,05
6	AdaBoost	73.38	0,03	73.77	0,01
7	ENORA	74.57	19,7	75.20	1,46

4

Рисунок В.4 – Слайд №4

АНАЛІЗ МЕТОДІВ ДЛЯ ПРОВЕДЕННЯ ДІАГНОСТУВАННЯ

№ п/п	Назва нейронної мережі	Параметри	
		Значення точності, %	Час роботи алгоритму, хв
1	MultiLayerPerceptron (2 прихованих шари)	78.00	0,04
2	Згорткова мережа(CNN)	77,92	1,9
3	LSTM	89,6	~5
4	Dense neural network	88,67	~3

5

Рисунок В.5 – Слайд №5

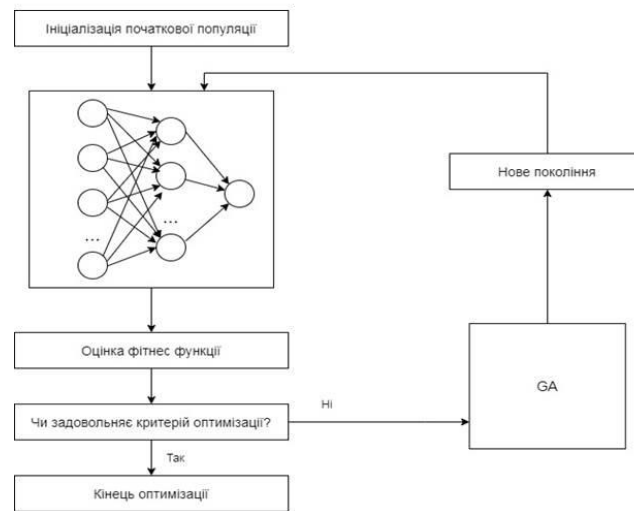
ВИБІР МОВИ ПРОГРАМУВАННЯ

Критерії порівняння	Мова програмування			
	C++	R	Python	Matlab
Легкість вивчення	-	+	+	-
Складність	+	-	-	+
Безкоштовність	+	+	+	-
Платформонезалежність	+	+	+	+
Наявність бібліотек машинного навчання	+	+	+	+
Висока продуктивність	+	-	+	-
Швидкість роботи	+	-	+	-
Швидкість розробки	-	+	+	-
Велике співтовариство	+	-	+	-
Робота з пам'яттю комп'ютера	+	-	+	+

6

Рисунок В.6 – Слайд №6

ОПТИМІЗАЦІЯ НЕЙРОННОЇ МЕРЕЖІ ЗА ДОПОМОГОЮ ГЕНЕТИЧНОГО АЛГОРИТМУ



7

Рисунок В.7 – Слайд №7

ЗАПРОПОНОВАНІ РІШЕННЯ

- Ступінь спорідненості:

$$d_{ij} = \sum_{k=1}^p |x_{ik} - x_{jk}|$$

- Оператор мутації:

$$H_A = \frac{h_{ij} + h_{ik}}{2}$$

обчислення значень генів після мутації

перший спосіб

$$h_{ij}^* = H_A + \text{rand}(0,1)$$

$$h_{ik}^* = H_A - \text{rand}(0,1)$$

другий спосіб

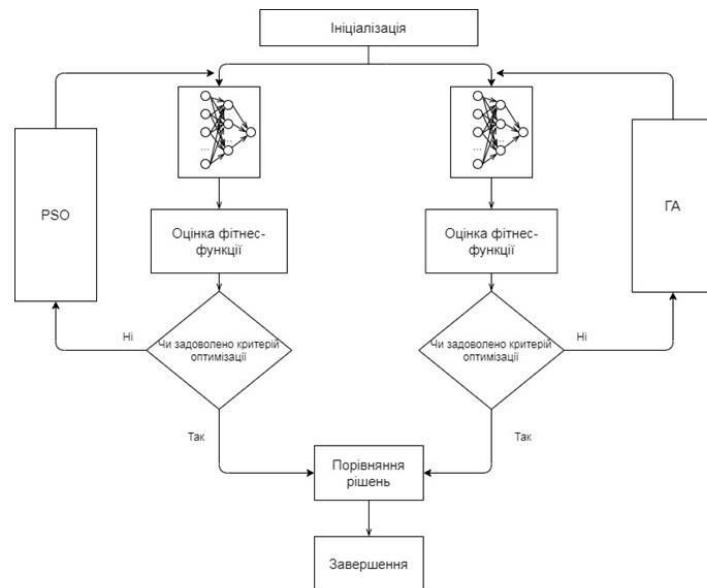
$$h_{ij}^* = H_A + \text{rand}(0,1)$$

$$h_{ik}^* = 0$$

8

Рисунок В.8 – Слайд №8

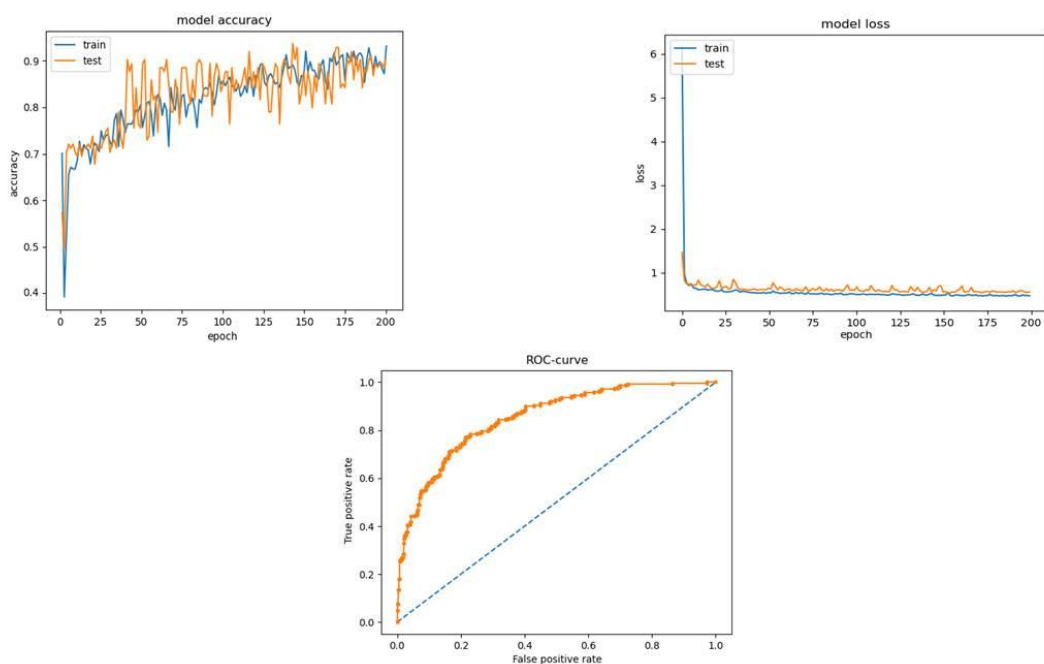
ОПТИМІЗАЦІЯ НЕЙРОННОЇ МЕРЕЖІ ЗА ДОПОМОГОЮ ГЕНЕТИЧНОГО АЛГОРИТМУ ТА МЕТОДУ PSO



9

Рисунок В.9 – Слайд №9

РЕЗУЛЬТАТИ



10

Рисунок В.10 – Слайд №10

РЕЗУЛЬТАТИ

№ п/п	Назва	Точність, %	Значення ROC- кривої	Час, с
1	Нейронна мережа	78,78	0,811	117
2	Нейронна мережі + ГА	89,3	0,884	99
3	Нейронна мережі + перша модифікація ГА	91,3	0,9	101
4	Нейронна мережі + друга модифікація ГА	89,97	0,899	102
5	Нейронна мережі + PSO	90,82	0,893	97

11

Рисунок В.11 – Слайд №11

ВИСНОВКИ

Під час виконання дипломної роботи було:

- проаналізовано предметну область та методи, які можна використовувати для діагностування захворювань;
- проведено аналіз засобів розробки;
- наведено опис основних рішень, щодо реалізації системи.
- Запропоновано спосіб вибору осіб до батьківських пар за ступенем неспорідненості та два оператори мутації, за яких значення генів обчислюються на основі їх середніх значень після чого додається, віднімається певне число або обнуляється значення.
- Проведено експериментальне дослідження, відповідно до якого запропонована модифікація ГА показала кращу точність порівняно з класичним ГА та методом PSO.

12

Рисунок В.12 – Слайд №12

***ДЯКУЮ ЗА
УВАГУ!***

13

Рисунок В.13 – Слайд №13