Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія вирішення задачі нейромережевої класифікації ірисів»

Виконав: студент 2-го курсу, групи 1КН-23м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Гладченко В. А.
(прізвище та ініціали)

Керівник: к.т.н., проф. каф. КН
_____ Колесницький О.К.
(прізвище та ініціали)
«05» 12 2024 р.

Опонент: к.т.н., професор каф. КСУ
_____ Биков М. М.
(прізвище та ініціали)
«05» 12 2024 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«06» 12 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Д.т.н., проф. Яровий А.А.
11.10 2024 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Гладченку Володимиру Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Інформаційна технологія вирішення задачі нейромережевої класифікації ірисів.
Керівник роботи: к.т.н., проф. кафедри КН Колесницький О. К.
затверджені наказом вищого навчального закладу від "17" 09 2024 року №310

2. Строк подання студентом роботи 11.11 2024 року.

3. Вихідні дані до роботи:
Вхідні дані – кількість класів ірисів – не менше 3, вид класифікатора – нейронна мережа, кількість навчальних зразків ірисів – не менше 100, кількість тестових зразків ірисів – не менше 30, кількість нейронів у мережі – не менше 10, використання об'єктно-орієнтованої мови програмування.

4. Зміст текстової частини:
Вступ, аналіз предметної області класифікації ірисів, розробка інформаційної технології класифікації ірисів, програмна реалізація інформаційної технології класифікації ірисів за допомогою нейронної мережі, тестування та аналіз результатів роботи програми класифікації ірисів на основі нейронної мережі, економічна частина, висновки, перелік використаних джерел, додатки.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
Алгоритм роботи програми, UML діаграма класів, робочі вікна програми, результати тестування програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-4	Колесницький О.К., к.т.н., проф. каф. КН		
5	Лесько О. Й., к. е. н., проф. зав. каф. ЕПВМ		

7. Дата видачі завдання 02.09 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасного рівня інформаційних технологій класифікації ірисів. Постановка задач дослідження	02.09.24 – 15.09.24	
2	Побудова моделей класифікації ірисів	16.09.24 – 29.09.24	
3	Практичне застосування та оцінка ефективності розроблених моделей	30.09.24 – 08.10.24	
4	Підготовка економічної частини	09.10.24 – 26.10.24	
5	Апробація та/або впровадження результатів дослідження	27.10.24 – 03.11.24	
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	04.11.24 – 11.11.24	

Студент _____ Гладченко В. А.
(підпис)

Керівник роботи _____ Колесницький О.К.
(підпис)

АНОТАЦІЯ

УДК 004.8

Гладченко В. А. Інформаційна технологія вирішення задачі нейромережевої класифікації ірисів. Магістерська кваліфікаційна робота зі спеціальності 122 – «Комп'ютерні науки», освітня програма – «Системи штучного інтелекту». Вінниця: ВНТУ, 2024. 89 с.

На укр. мові. Бібліогр.: 21 назв; рис.: 30; табл. 8.

Дана магістерська кваліфікаційна робота присвячена розробці програмного забезпечення для інформаційної технології класифікації ірисів. У роботі було обґрунтовано вибір нейронної мережі багатошаровий персептрон для класифікації ірисів, яка має 4 входи, 2 прихованих шари по 10 нейронів та вихідний шар із 3 нейронів. У прихованих шарах обрано функцію активації ReLU та функцію активації Softmax у вихідному шарі. Для навчання цієї нейромережі використовується метод зворотного поширення помилки. Для програмної реалізаціїі інформаційної технології було використано мову програмування Python та спеціалізовані бібліотеки Keras, NumPy та Pandas. Навчання нейромережі відбувалось з використанням набору даних ірисів Фішера, яка налічує 150 записів. Набір даних було поділено на навчальну (120) та тестову (30) вибірки. Розроблене програмне забезпечення має достовірність класифікації ірисів на тестовій вибірці 96,7%, а найкращий із 6 методів-аналогів має достовірність класифікації на тестовій вибірці 92,6%, тобто достовірність класифікації збільшилась на 4,1%.

Графічна частина складається із 6 плакатів.

У економічному розділі доведено доцільність проведення науково-дослідної роботи за даною темою. Термін окупності становить 0,98 р., що свідчить про комерційну привабливість науково-технічної розробки.

Ключові слова: класифікація, машинне навчання, нейронна мережа, багатошаровий персептрон.

ABSTRACT

Hladchenko V. A. Information technology for solving the problem of neural network classification of irises. Master's thesis in the specialty 122 - «Computer sciences», educational program – «Artificial intelligence systems». Vinnytsia: VNTU, 2024. 89 p.

In Ukrainian language. Bibliogr .: 21 titles; fig . 30; table 8.

This master's qualification work is devoted to the development of software for the information technology of iris classification. The work justified the choice of a multilayer perceptron neural network for iris classification, which has 4 inputs, 2 hidden layers of 10 neurons each and an output layer of 3 neurons. The ReLU activation function was selected in the hidden layers and the Softmax activation function in the output layer. The error backpropagation method was used to train this neural network. The Python programming language and the specialized Keras, NumPy and Pandas libraries were used to create the module. The neural network was trained using a Fisher iris dataset, which has 150 records. The dataset was divided into a training (120) and a test (30) sample. The developed software has an iris classification accuracy of 96.7% on the test sample, and the best of the 6 similar methods has an iris classification accuracy of 92.6% on the test sample, i.e. the classification accuracy has increased by 4.1%.

The graphic part consists of 6 posters.

In the economic section, the expediency of conducting research work on this topic is proven. The payback period is 0.98 years, which indicates the commercial attractiveness of scientific and technical development.

Key words: classification, machine learning, neural network, multilayer perceptron

ЗМІСТ

ВСТУП

Актуальність. На сучасному етапі ми спостерігаємо бурхливий розвиток інтелектуальних інформаційних технологій і систем штучного інтелекту (ШІ). Їх застосовують, зокрема, для класифікації об`єктів різної природи. Класифікація об`єктів – дуже актуальна задача у сучасному світі. У даній роботі ця задача розв’язується на прикладі класифікації квіток ірису.

Задача розпізнавання ірисів є класичною задачею машинного навчання, на якій перевіряється ефективність різних методів та алгоритмів класифікації. Це непроста задача, тому для неї найкраще підходять методи штучного інтелекту. Вона розв`язувалась раніше такими методами як дерева рішень, машина опорних векторів та інші, але цікаво було б подивитись як впорається із цією задачею штучна нейронна мережа.

Одним з напрямків штучного інтелекту, що активно розвиваються, є штучні нейронні мережі, які працюють за принципами біологічних нейронних мереж і являють собою систему взаємодіючих між собою штучних нейронів. Серед основних галузей застосування штучних нейронних мереж можна виділити: класифікацію зображень, обробку звукової та текстової інформації, прийняття рішень, прогнозування, аналіз даних, оптимізація.

Тому актуальною є задача перевірки чи будуть нейронні мережі більш ефективними у розв`язанні задачі класифікації ірисів, ніж класичні методи класифікації на основі машинного навчання.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Розробка прикладних інтелектуальних інформаційних технологій та систем» та плану наукової та навчально-методичної роботи кафедри.

Мета і завдання досліджень. Метою магістерської кваліфікаційної роботи є підвищення достовірності класифікації ірисів за рахунок використання нейронної мережі.

Для досягнення мети розробки необхідно виконати такі задачі:

- провести аналіз предметної області класифікації ірисів;
- розглянути існуючі методи класифікації ірисів та обрати й обґрунтувати вибір методу, який задовольняє мету даної магістерської кваліфікаційної роботи;
- обгрунтувати вибір архітектури нейромережі;
- сформулювати стадії інформаційної технології, розробити структуру та алгоритм роботи програмного засобу;
- виконати програмну реалізацію запропонованої інформаційної технології;
- провести тестування програмного продукту та виконати аналіз отриманих результатів.

Об’єкт дослідження – процес комп’ютеризованої класифікації ірисів на основі нейромережі..

Предмет дослідження – інформаційна технологія та програмні засоби класифікації ірисів за допомогою нейронної мережі та достовірність їх роботи.

Методи дослідження. У роботі використані наступні методи наукових досліджень: системного аналізу, теорії штучних нейронних мереж для реалізації інформаційної технології, методи математичної статистики для розробки процесу обрахунків результатів експериментів із програмним засобом, об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів.

1. Набула подальшого розвитку інформаційна технологія вирішення задачі класифікації ірисів, яка відрізняється використанням нейронної мережі багатошаровий персептрон, що дозволило підвищити достовірність класифікації ірисів.

Практичне значення одержаних результатів полягає в тому, що на основі проведених досліджень розроблено програмне забезпечення класифікації ірисів.

Запропонована інформаційна технологія сприяє підвищенню достовірності класифікації ірисів, зокрема:

- розроблено алгоритм роботи програмного забезпечення класифікації ірисів;
- розроблено програмні засоби для класифікації ірисів.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується коректністю постановки завдання, коректністю використання математичного апарату методів дослідження, експериментальними дослідженнями тестування програмної реалізації інформаційної технології достовірності класифікації ірисів. Адекватність розроблених математичних моделей підтверджується результатами експериментальних досліджень.

Особистий внесок здобувача. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно. У працях, написаних у співавторстві, здобувачу належать: аналіз класифікації ірисів та методів підвищення достовірності програмних засобів класифікації ірисів [1].

Апробація результатів роботи. Результати роботи були апробовані на Міжнародній науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ (МН-2025)», Вінниця, 15 жовтня 2024 року - 15 червня 2025 року» [1].

Публікації. За результатами досліджень опубліковано тези доповіді на науково-технічній конференції [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ КЛАСИФІКАЦІЇ ОБ`ЄКТІВ

1.1 Постановка задачі

У багатьох літературних джерелах для ілюстрації роботи різних алгоритмів класифікації і кластеризації використовується набір даних, що вже став класичним та отримав назву Іриси Фішера [2]. Цей набір даних містить відомості про 150 екземплярів квітів ірису, по 50 екземплярів для кожного з трьох видів - Ірис Щетинистий (Iris setosa), Ірис Різнокольоровий (Iris versicolor) та Ірис Віргінський (Iris virginica) – рисунок 1.1. Для кожного екземпляру рослини задані 4 характеристики:

- Довжина чашолистка в см
- Ширина чашолистка в см
- Довжина пелюстки в см
- Ширина пелюстки в см

Рисунок 1.1 – Вид ірисів Фішера

Представницький зріз вихідного набору даних за двома параметрами дозволяє зробити висновок, що один із класів, а саме Iris setosa є лінійно-відокремленим від двох інших, чого не можна сказати про два інші класи (рис.1.2).

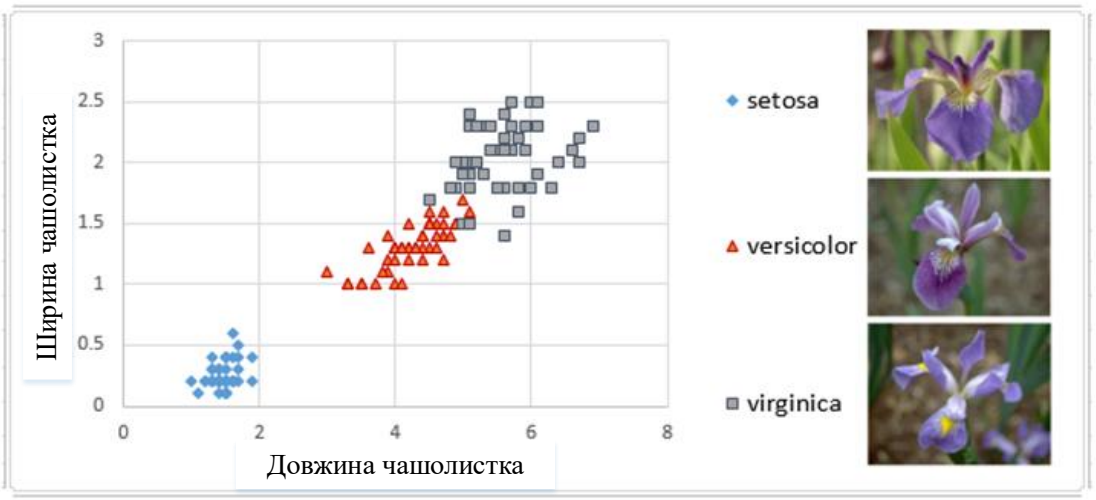

Рисунок 1.2 – Зріз вихідного набору даних за двома параметрами

Мета нейронної мережі, яка створюється, - передбачити клас квітки ірису на основі інших атрибутів. Це означає, що потрібно створити модель, яка буде описувати зв'язок між значеннями атрибутів і класом.

1.2 Огляд відомих методів класифікації об`єктів

Класифікація - це процес віднесення конкретного об`єкта до певного класу об`єктів, причому кількість класів та їх назви наперед відомі, а також відомі параметри об`єктів, що належать цим класам [3,4].

Існує багато різних типів завдань класифікації, які ви можете виконувати. Для кожного завдання часто потрібен свій алгоритм, оскільки кожен з них використовується для вирішення конкретної проблеми.

Для кожної проблеми ви повинні вибрати правильний алгоритм. Ваше питання полягає в тому, як це зробити. Якщо у вас достатньо обчислювальних ресурсів, ви можете протестувати кілька алгоритмів і налаштувань параметрів. При такому підході головне питання полягає в тому, як надійно оцінити і порівняти продуктивність алгоритмів.

Класифікація - це процес розпізнавання, розуміння та групування ідей та об'єктів за заданими категоріями або "підгрупами". Використовуючи

попередньо класифіковані навчальні набори даних, програми машинного навчання застосовують різноманітні алгоритми для класифікації майбутніх наборів даних за категоріями.

Алгоритми класифікації в машинному навчанні використовують вхідні навчальні дані для прогнозування ймовірності того, що наступні дані потраплять в одну із заздалегідь визначених категорій. Одне з найпоширеніших застосувань класифікації - фільтрація електронних листів на "спам" і "неспам".

Коротко кажучи, класифікація - це форма "розпізнавання образів", коли алгоритми класифікації застосовуються до навчальних даних, щоб знайти той самий образ (схожі слова або думки, послідовності чисел тощо) у наступних наборах даних.

Вивчення класифікації в статистиці дуже широке, і існує кілька типів алгоритмів класифікації, які можна використовувати залежно від набору даних, з яким ви працюєте. Нижче наведено найпоширеніші алгоритми класифікації у машинному навчанні [3,4]:

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

Логістична регресія

Логістична регресія - це обчислення, яке використовується для прогнозування бінарного результату: або щось відбувається, або ні. Це може бути представлено як "Так/Ні", "Пройшов/Не пройшов" тощо.

Незалежні змінні аналізуються для визначення бінарного результату, і результати потрапляють в одну з двох категорій. Незалежні змінні можуть бути категоріальними або числовими, але залежна змінна завжди є категоріальною. Записується так:

$$P(Y=1|X) \text{ or } P(Y=0|X) \quad (1.1)$$

Ця формула обчислює ймовірність залежної змінної Y, враховуючи незалежну змінну X. Її можна використовувати для обчислення ймовірності того, що слово має позитивну або негативну конотацію (0, 1 або за шкалою між 0 та 1). Або ж її можна використовувати для визначення об'єкта, зображеного на фотографії (дерево, квітка, трава тощо), при цьому кожному об'єкту присвоюється ймовірність від 0 до 1.

Наївний Байєс

Наївний Байєс обчислює ймовірність того, чи належить точка даних до певної категорії, чи ні. В аналізі тексту його можна використовувати для класифікації слів або фраз як таких, що належать до заданого "тегу" (класифікації) чи ні. Наприклад, як у табл. 1.1.

Таблиця 1.1 – Приклад класифікації слів або фраз як таких, що належать до заданого "тегу"

Текст	Тег
“Хороша гра”	Спорт
“Вибори закінчилися”	Не спорт
“Дуже чистий матч”	Спорт
“Чистий, але незабутній матч”	Спорт
“Це були близькі вибори”	Не спорт

Щоб вирішити, чи варто позначати словосполучення як "спорт", потрібно порахувати за формулою:

$$P(A \mid B)=(P(B \mid A)\times P(A))/(P(B)) \quad (1.2)$$

Або ймовірність A, якщо B істинна, дорівнює ймовірності B, якщо A істинна, помноженій на ймовірність що A істина, і все поділено на ймовірність що B істина.

K-найближчих сусідів

K-найближчих сусідів (k-NN) - це алгоритм розпізнавання образів, який використовує навчальні набори даних для пошуку k найближчих родичів у майбутніх прикладах.

Коли k-NN використовується для класифікації, ви обчислюєте, щоб віднести дані до категорії найближчого сусіда. Якщо k = 1, то дані будуть поміщені в клас, найближчий до 1. K класифікується шляхом множинного опитування своїх сусідів.

Дерево рішень

Дерево рішень - це алгоритм керованого навчання, який ідеально підходить для задач класифікації, оскільки він здатний впорядковувати класи на точному рівні. Він працює як блок-схема, розділяючи точки даних на дві схожі категорії одночасно від "стовбура дерева" до "гілок" і "листя", де категорії стають все більш схожими. Це створює категорії всередині категорій, що дозволяє органічно класифікувати дані з обмеженим людським наглядом.

Продовжуючи приклад зі спортом, дерево рішень працює як показано на рис. 1.3.

Випадковий ліс

Алгоритм випадкового лісу є розширенням дерева рішень, в якому ви спочатку будуєте безліч дерев рішень з навчальними даними, а потім вписуєте нові дані в одне з дерев як "випадковий ліс".

Він, по суті, усереднює ваші дані, щоб пов'язати їх з найближчим деревом на шкалі даних. Моделі випадкових лісів корисні, оскільки вони вирішують проблему дерева рішень, яка полягає в тому, що вони без потреби "примушують" точки даних знаходитися в межах однієї категорії.

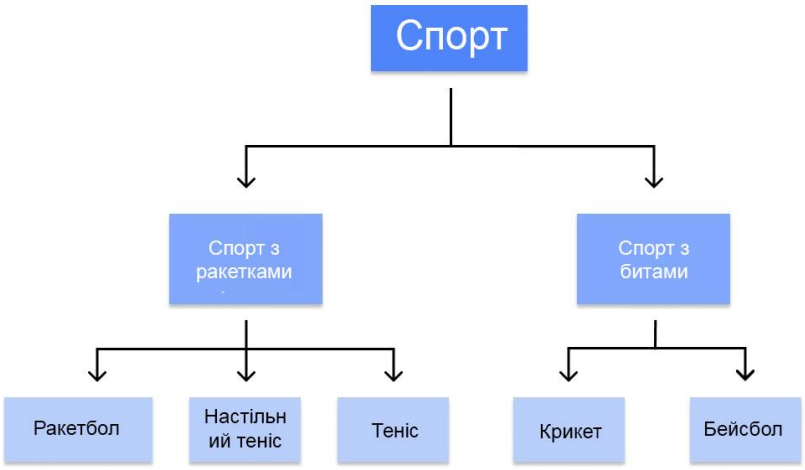

Рисунок 1.3 – Приклад зі спортом, як працює дерево рішень

Машини опорних векторів

Машина опорних векторів (SVM) [4] використовує алгоритми для навчання і класифікації даних у межах ступенів полярності, виводячи їх на рівень, що виходить за межі прогнозування X/Y (рис. 1.4).

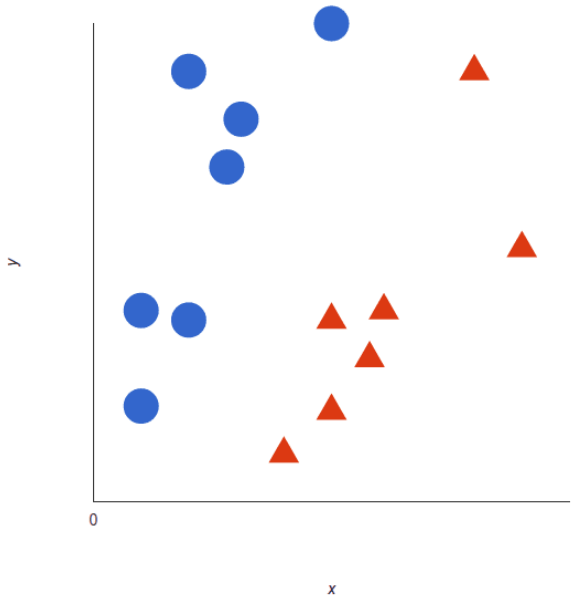

Рисунок 1.4 – Просте візуальне пояснення розподілу даних двох класів (мітки: червона і синя) за двома характеристиками: X та Y

Для простого візуального пояснення використано дві мітки: червону і синю, з двома характеристиками даних: X та Y, а потім навчимо наш класифікатор виводити координату X/Y як червону або синю.

Потім SVM призначає гіперплощину, яка найкраще розділяє мітки (рис. 1.5). У двовимірному просторі це просто лінія. Все, що знаходиться по один бік лінії, є червоним, а все, що по інший бік - синім.

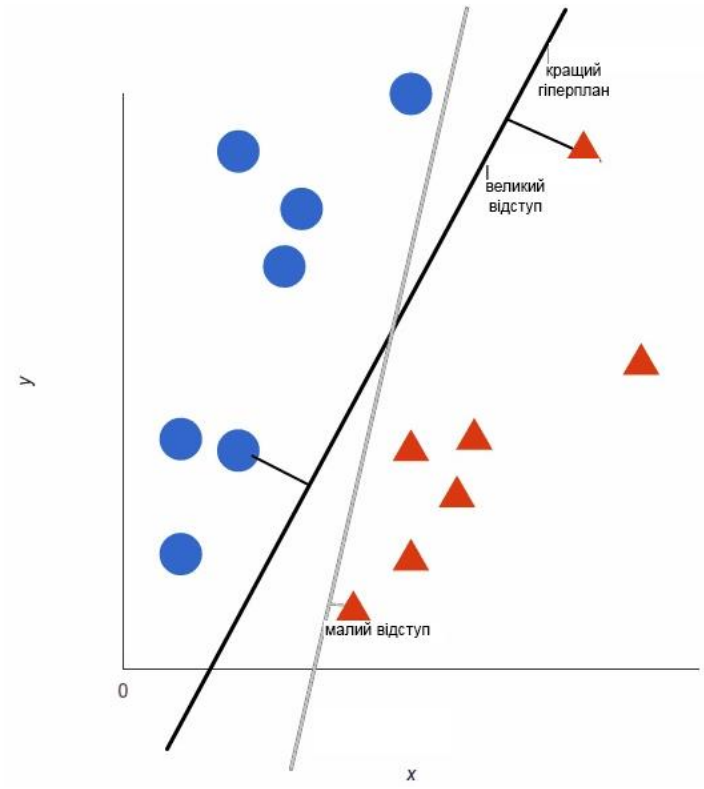

Рисунок 1.5 – SVM призначає гіперплощину (лінію), яка найкраще розділяє класи

Для того, щоб максимізувати машинне навчання, найкраща гіперплощина - це та, що має найбільшу відстань між кожною міткою.

Однак, оскільки набори даних стають складнішими, може виявитися неможливим провести єдину лінію для класифікації даних на два табори. Приклад цього представлено на рис. 1.6.

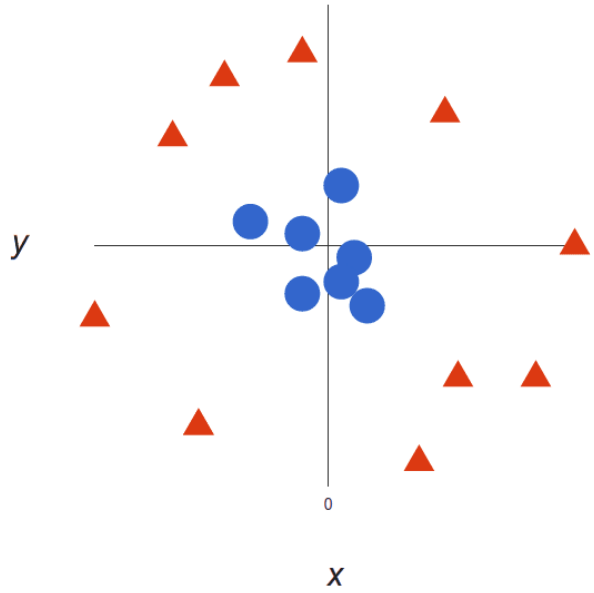

Рисунок 1.6 – Приклад неможливості провести єдину лінію для класифікації даних на два табори

Використовуючи SVM, чим складніші дані, тим точнішим буде предиктор. Уявіть все вищесказане в тривимірному вигляді, додавши вісь z, щоб вийшло коло. У двовимірному зображенні з найкращою гіперплощиною це виглядає так, як показано на рис. 1.7.

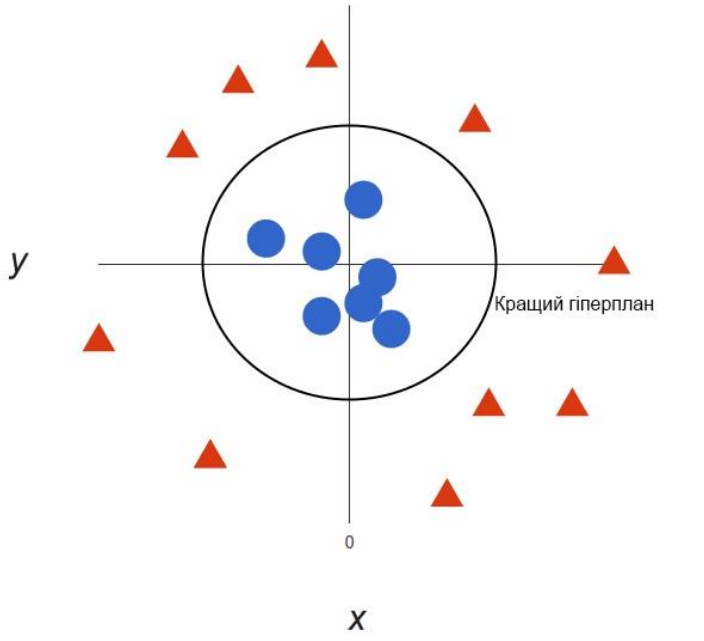

Рисунок 1.7 – Приклад найкращої роздільної лінії (коло)

SVM дозволяє більш точне машинне навчання, оскільки є багатовимірним.

Нейронні мережі.

Нейромережа - це математична модель біологічної нейромережі [5,6]. Вона містить велику кількість простих пристроїв – нейронів, що з'єднані між собою. Кожен нейрон отримує вхідний сигнал, перетворює його, а потім надсилає перетворений сигнал іншим нейронам.

Відмінна особливість нейромереж – їх здатність до навчання для вирішення певного завдання. З точки зору математики, під час навчання відбувається підбір коефіцієнтів міжнейронних зв'язків. Нейромережа здатна у процесі навчання знаходити складні взаємозв'язки між параметрами вхідних та вихідних даних. Також, нейромережі здатні до узагальнення, тобто видавати вірну відповідь для вхідних даних, які не брали участь у процесі навчання.

Метод опорних векторів характеризується великою обчислювальною складністю, і для його використання потрібно мати більший простір ознак, ніж є в даній задачі. Від початку відсутня статистична інформація про вихідні дані ускладнює застосування байєсівського класифікатора.

Нейронні мережі спроможні вирішувати такі завдання класифікації, набори даних у яких є лінійно-нероздільними. До того ж, нейромережі довели здатність до апроксимації довільної функції з будь-якою заданою точністю [7,8]. Через наведені аргументи, використання нейронних мереж для вирішення даної задачі класифікації ірисів є кращим варіантом.

1.3 Обгрунтування вибору аналогу до програмного забезпечення класифікації ірисів

На теперішній час існує багато відомих програмних додатків, що здатні здійснювати класифікацію ірисів. Це є класична задача класифікації, тому

різні дослідники перевіряють на цій задачі свої нові методи та алгоритми класифікації.

Як аналог візьмемо програму Iris-Flower-Classification-Project [9]. Вона є консольною програмою без графічного інтерфейсу користувача та здійснює моделювання задачі класифікації ірисів за шістьмома алгоритмами. Результати роботи цієї програми наведено у табл. 1.2.

Таблиця 1.2 – Результати роботи програми Iris-Flower-Classification-Project по вирішенню задачі класифікації ірисів

№	Назва методу	Достовірність класифікації, %
1.	Логістична регресія (Logistic regression)	90,1
2.	Лінійний дискримінантний аналіз (Linear discriminant Analysis)	92,2
3.	K-найближчих сусідів (K-Nearest Neighbour)	92,4
4.	Дерева класифікації та регресії (Classification and Regression Trees)	89,1
5.	Гаусівський наївний Байєс (Gaussian Naive Bayes)	91,3
6.	Машини опорних векторів (Support Vector Machines)	92,6

Із табл. 1.2 видно, що найкращим є метод машини опорних векторів і його середня достовірність класифікації складає 92,6 %.

До недоліків цієї програми можна віднести невисоку достовірність класифікації ірисів. Інші програмні реалізації класифікації ірисів також мають такий недолік як невисока достовірність. Звідси випливає завдання розробки нового програмного засобу для класифікації ірисів з підвищеною достовірністю роботи.

1.4 Висновок до розділу 1

У розділі описано детальну постановку задачі класифікації ірисів. Також розглядаються різні методи розв'язання задачі класифікації і оцінюється їх застосовність до завдання класифікації ірисів. Серед таких методів машинного навчання як метод логістичної регресії, наївний Байєс, K-найближчих сусідів, дерева рішень, машини опорних векторів та нейромережева класифікація як найбільш перспективний і застосовний до даної задачі було обрано нейромережевий метод. Крім цього, було обгрунтовано вибір аналогу до розроблюваної програми та на основі його недоліків сформульовано мету роботи – підвищення достовірності класифікації ірисів.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КЛАСИФІКАЦІЇ ІРИСІВ НА ОСНОВІ НЕЙРОМЕРЕЖІ

2.1 Обґрунтування вибору типу нейронної мережі для інформаційної технології класифікації ірисів

Для задачі класифікації ірисів, в принципі, можуть бути використані дуже багато типів нейронних мереж, які працюють в режимі класифікації образів. Розглянемо з метою вибору найбільш ефективної для задачі класифікації ірисів такі нейронні мережі [5]:

– багатошаровий персептрон,
– гібридні нечіткі мережі [10],
– нейро-нечіткі мережі Такагі-Сугено-Канг [11].

В основі процесу навчання всіх цих моделей лежить набір навчальних пар (xi, di), де xi це вектор характеристик, а di це вектор кодів класу. Розглянемо їх та оберемо найбільш оптимальну для нашої задачі.

Нейронні мережі – це моделі, що імітують діяльність людського мозку. Вони, подібно до мозку людини, складаються з великого числа однотипних елементів – нейронів, які пов'язані між собою. На рис. 2.1. показано схему нейрона [8].

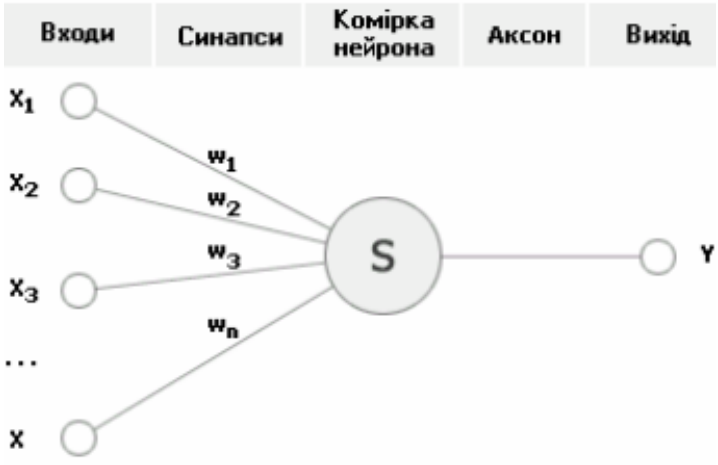

Рисунок 2.1 – Схема нейрона

З рисунку 2.1 видно, що штучний нейрон, подібно до біологічного, складається з синапсів, які зв'язують входи нейрона з ядром, ядра нейрона, яке здійснює обробку вхідних сигналів та аксона, що зв'язує нейрон з нейронами наступного шару. Кожен синапс має вагу, яка визначає наскільки відповідний вхід нейрона впливає на його стан. Стан нейрона визначається за формулою:

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

де n - число входів нейрону, x_i – значення i-го входу нейрона, w_i – вага i-го синапсу. Значення виходу визначається за формулою:

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

де f – певна функція, що називається передатною. Найчастіше в якості передатної функції використовують, так звану, сигмоїду, що має наступний вигляд:

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

Основною перевагою сигмоїди є те, що вона диференційована по всій осі абсцис і має просту похідну:

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

При зменшенні параметра α сигмоїда стає більш похилою, вироджуючись у горизонтальну лінію на рівні 0,5 при a=0. При збільшенні α сигмоїда наближається до функції одиничного стрибка.

2.1.1 Багатошаровий персептрон

Багатошарова нейронна мережа (БНМ) прямого поширення (багатошаровий персептрон) складається з формальних нейронів і характеризується наступними параметрами і властивостями: M – кількість шарів мережі, Nμ – кількість нейронів у μ-му шарі, зв'язки між нейронами у шарі відсутні. Виходи нейронів μ-го шару, μ=1,2,..., M–1, надходять на входи нейронів тільки наступного μ+1-го шару. Зовнішній векторний сигнал x надходить на входи нейронів тільки першого шару, виходи нейронів останнього M-го шару утворюють вектор виходів мережі $y^{(M)}$. Схема мережі показана на рис. 2.2.

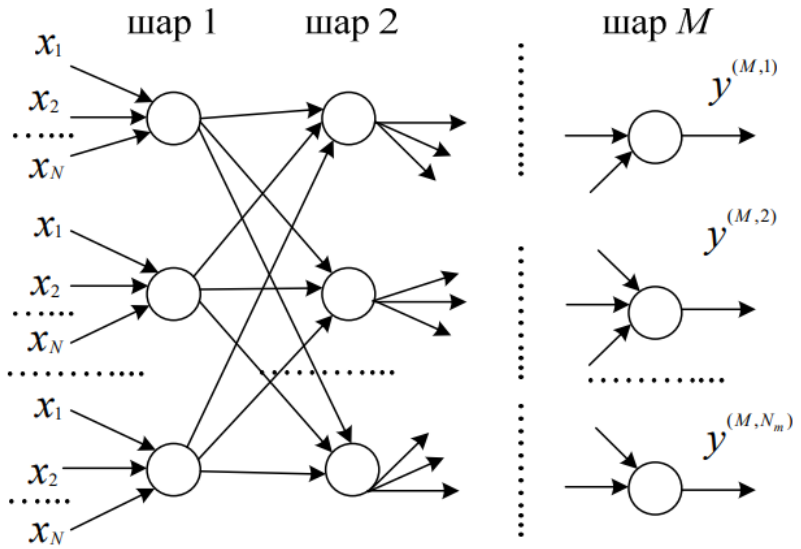

Рисунок 2.2 – Схема багатошарового персептрона

Кожен i-й нейрон μ-го шару ((μ, i)-й нейрон) перетворює вхідний вектор x(μ, i) у вихідну скалярну величину y(μ, i) . Якість рішення, одержуваного БНМ, буде істотно залежати від кількості шарів, кількості нейронів у кожному шарі і кількості зв'язків між шарами. Для різних класів задач, що вирішуються за допомогою НМ, запропоновані евристичні оцінки кількості шарів і нейронів. Відомо, що тришарова БНМ здатна апроксимувати будь-яку обчислювану функцію.

2.1.2 Нечітка нейронна мережа

Нечітка нейромережа (fuzzy-neural networks) функціонує стандартно на основі чітких дійсних чисел. Нечіткою буде тільки інтерпретація вихідних результатів.

Для пояснення суті нечітких нейромереж, розглянемо просту нейромережу, яка має два входи і один нейрон (рис. 2.3).

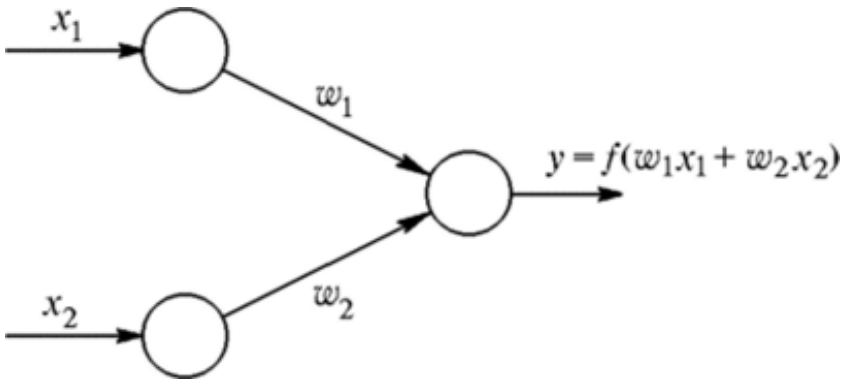

Рисунок 2.3 – Елементарна нейронна мережа

Тут вхідні сигнали x, «взаємодіють» з вагами vr, утворюючи твори:

$$p_i = x_i w_i, \qquad i = 1,2 \tag{2.5}$$

Вихід нейрона утворюється в результаті перетворення входу net активаційною функцією:

$$y = f\left(net\right) = f\left(x_1 w_1 + x_2 w_2\right), \tag{2.6}$$

Наприклад, сигмоїдою:

$$f\left(x\right) = \frac{1}{1+e^{-x}} \cdot \tag{2.7}$$

Наведену одношарову нейромережу, у якій застосовуються операції множення, підсумовування та сигмоїдальна функція активації, будемо називати стандартною нейромережею.

У разі застосування інших операцій, таких як t-норма чи t-конорма, отримаємо нейромережу, яка носить назву нечіткої нейронної мережі.

Нечітка нейронна (гібридна) мережа - це нейромережа з чіткими сигналами, вагами і активаційною функцією, але з об'єднанням x, і vv ,, і p 2 з використанням t-норми, t-конорми або деяких інших неперервних операцій. Входи, виходи і ваги нечіткої нейромережі - речові числа, що належать відрізку [0, 1]. Нечіткою нейромережею зазвичай називають чітку нейромережу, яка побудована на основі багатошарової архітектури із застосуванням «І», «АБО» нейронів.

Нечітка нейромережа зазвичай складається з чотирьох шарів: шар фазифікації вхідних змінних, шар агрегування значень активації умови, шар агрегування нечітких правил, вихідний шар.

Нечітка нейромережа являє собою набір нечітких правил, які описують класи в наявному наборі вхідних даних, а також нечітку систему виведення для їх переробки з метою отримання результату [12]. Фрагмент нечіткої нейромережі представлений на рис. 2.4.

На рис. 2.4 показана нечітка нейромережа з чотирма входами (n = 4). Шари позначені символами від L1 до L5. Елементи, позначені символом П (мультиплікатори), перемножують всі вхідні сигнали, елементи, позначені символом Σ (Суматори) - підсумовують їх. Призначення шарів наступне: перший шар - терми вхідних змінних; другий шар - антецеденти (посилки) нечітких правил; третій шар - нормалізація ступенів виконання правил; четвертий шар - укладення правил; п'ятий шар - агрегування результату, отриманого за різними правилами. Входи мережі в окремий шар не виділяються.

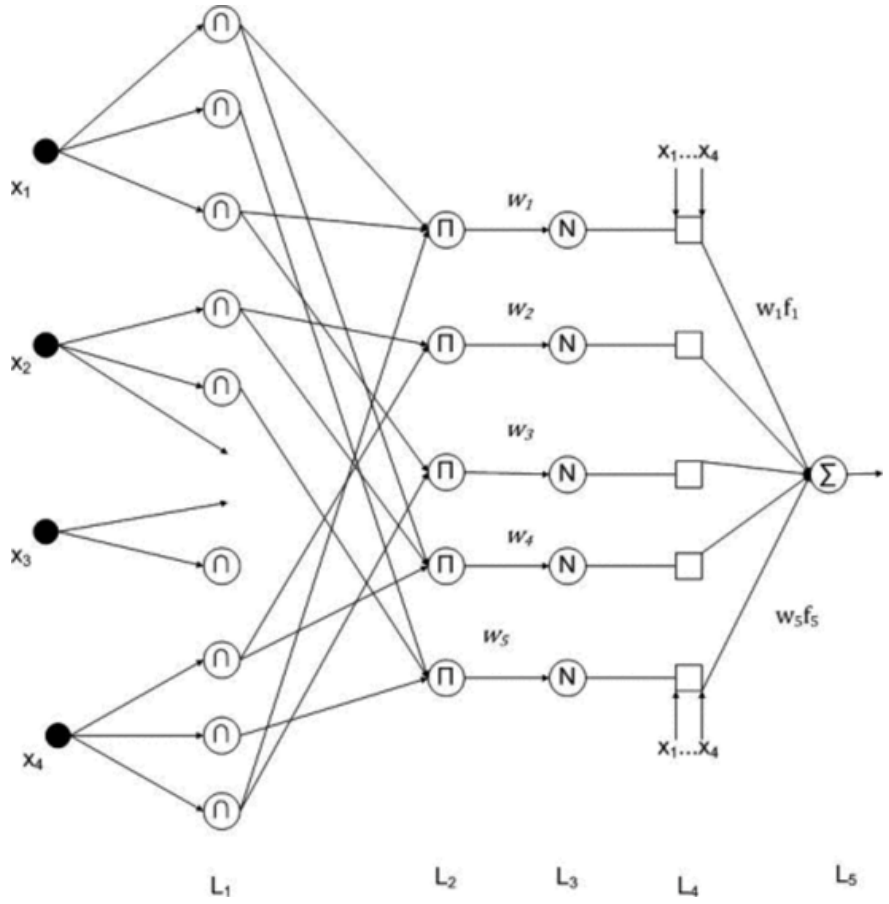

Рисунок 2.4 – Схема нечіткої нейронної мережі

Кожен елемент шару 1 (L1) представляє один терм з функцією приналежності. Входи нейромережі з'єднані тільки зі своїми термами. Кількість вузлів першого шару дорівнює сумі потужностей терм-множин вхідних змінних. Виходом вузла є ступінь приналежності значення вхідної змінної відповідного нечіткого терму. Фактично, в цьому шарі оцінюється ступінь приналежності вхідних даних до відповідних нечітких множин. Функціональна залежність між входом і виходом в вузлах цієї мережі визначається за формулою:

$$\mu_A(x) = \frac{1}{1+\left(\frac{x-c}{\sigma}\right)^{2b}}, \qquad (2.8)$$

Її параметри будуть модифікуватися в процесі навчання, що дозволить поліпшити підбір нечітких множин. Факт фізичної інтерпретації цих

параметрів дозволяє отримати хороше початкове розміщення функції приналежності нечітких множин, а також аналізувати її в процесі навчання.

Кількість вузлів шару 2 (L2) 5. Кожен вузол цього шару відповідає одному нечіткому правилу. Вузол другого шару з'єднаний з тими вузлами першого шару, які формують антецеденти відповідного правила. Отже, кожен вузол другого шару може приймати від 1 до п вхідних сигналів. Виходом вузла є ступінь виконання правила (вага деякого правила):

$$\omega_i = \mu_{A_i}(x) \times \mu_{B_i}(x), \quad i = 1,2,3,4,5. \tag{2.9}$$

Кількість елементів цього шару дорівнює кількості правил N. Кожен вузол пов'язаний з попереднім шаром таким чином, що вузол шару L2, відповідний до правила, з'єднаний з усіма вузлами шару L1, відповідними нечітким множинам суджень цього правила.

Кожен і-й вузол шару 3 (L3) визначає відношення ваги правила до суми ваг всіх правил:

$$\overline{\omega} = \frac{\omega_i}{\omega_1 + \omega_2 + \omega_3 + \omega_4 + \omega_5}, \; i = 1,2,3,4,5 \tag{2.10}$$

Вихідні сигнали 3-го шару називаються нормалізованими вагами.

Кожен вузол шару 4 (L4) з'єднаний з одним вузлом третього шару, а також з усіма входами нейромережі. Вузол четвертого шару розраховує внесок одного нечіткого правила у вихід нейромережі:

$$O_i^4 = \overline{\omega_i} f_i = \overline{\omega_i}\left(p_i x + q_i y + r_i\right) \tag{2.11}$$

Шар 5 (L5) являє собою реалізацію блоку дефазифікації, що реалізує залежність:

$$O_i^5 = \sum_i \overline{\omega}_i f_i = \frac{\sum_i \omega_i f_i}{\sum_i \omega_i} \qquad (2.12)$$

Ваги зв'язків, що доходять до верхнього вузла шару L4 і позначені у, інтерпретуються як центри функцій приналежності нечітких множин. Ці ваги, також як і значення параметрів x в шарі L1, будуть модифікуватися в процесі навчання. На виході шару L5 формується «чітке» (дефазіфіцірованное) вихідне значення модуля управління у. Представлена на рис. 2.4 структура має багато спільного з нейромережами. Вона являє собою багатошарову нейромережу, засновану на ідеї нечіткого виведення. На відміну від «чистих» нейромереж, кожен шар в цілому і окремі складові його елементи, також, як і конфігурація зв'язків, всі параметри і ваги мають фізичну інтерпретацію. Це властивість виявляється надзвичайно важливим, оскільки знання не розподіляються по мережі і можуть бути легко локалізовані і при необхідності відкориговані експертом-спостерігачем.

Швидкі алгоритми навчання та интерпретованість накопичених знань - ці фактори зробили сьогодні нечіткі нейромережі одним з найперспективніших і ефективних інструментів м'яких обчислень. Доведено, що такі мережі є універсальними аппроксіматорами.

2.1.3 Нейро-нечітка мережа Такагі-Сугено-Канг

Структура нечіткої мережі TSK заснована на системі нечіткого виведення Такагі- Сугено-Канга [10,11]. При цьому в якості опції фаззіфікаціі для кожної змінної x_j використовується узагальнена функція Гаусса:

$$\mu_A(x_j) = \frac{1}{1 + \left(\frac{x_j - c_j}{\sigma_j} \right)^{2b_j}} \qquad (2.13)$$

Для агрегації умови *i*-го правила в системі виведення TSK використовується операція алгебраїчного добутку:

$$\mu_A^{(i)}(x) = \prod_{j=1}^{N} \left[\frac{1}{1 + \left(\frac{x_j - c_j^{(i)}}{\sigma_j^{(i)}} \right)^{2b_j^{(i)}}} \right] \qquad (2.14)$$

При M правилах виведення агрегування вихідного результату мережі здійснюється за формулою, яку можна представити у вигляді:

$$y(x) = \frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} w_i y_i(x), \qquad (2.15)$$

Де $y_i(x)$ - агрегація імплікації. При цьому, формулі можна зіставити багатошарову структуру мережі, зображену на рис. 2.5. У такій мережі виділяється п'ять шарів:

Шар 1. Виконує роздільну фазифікацію кожної змінної x_j, j = 1,2 ..., N визначаючи для кожного *i*-го правила виведення значення коефіцієнта приналежності відповідно до застосовуваної функцією фазифікації. Це параметричний шар з параметрами, що підлягають адаптації в процесі навчання.

Шар 2. Виконує агрегування окремих змінних x_j, визначаючи результуюче значення коефіцієнта приналежності для вектора x відповідно до формули. Цей шар непараметричний.

Шар 3. Є генератор функції TSK, що розраховує значення $y_i(x)$. У цьому шарі також відбувається множення сигналів $y_i(x)$ на значення w_i , що сформовані в попередньому шарі. Це параметричний шар, в якому адаптації підлягають лінійні ваги, що визначають функцію слідства моделі TSK.

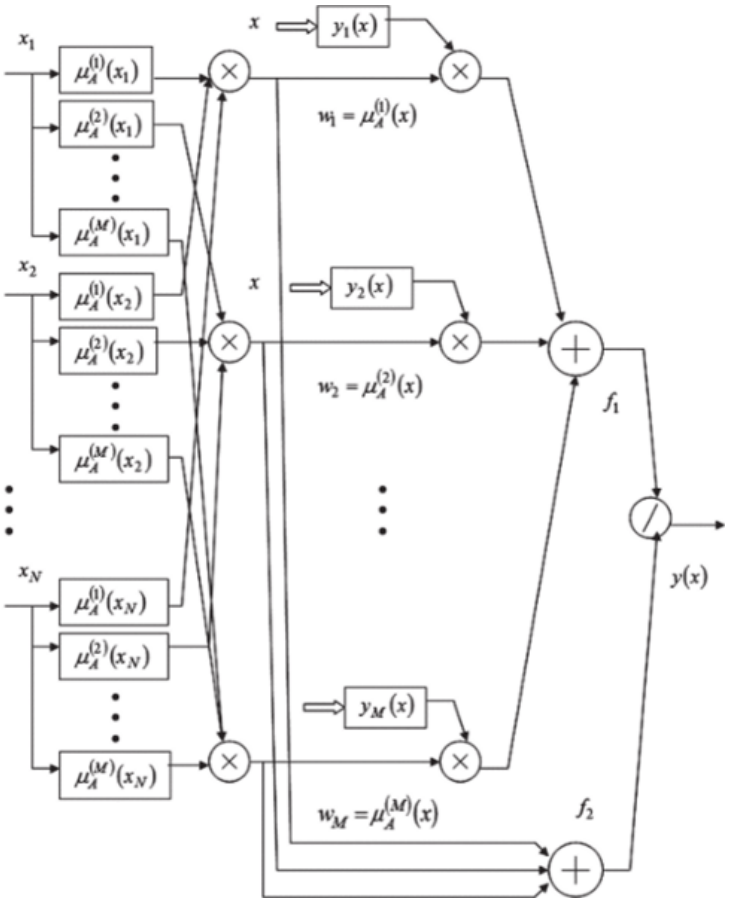

Рисунок 2.5 – Структура нечіткої нейронної мережі TSK

Шар 4. Складають два нейрона-суматора, один з яких розраховує зважену суму сігналів $y_i(x)$, а другий визначає суму ваг w_i. Це непараметрический шар.

Шар 5. Складається з одного вихідного нейрона - це нормалізаційний шар, в якому ваги піддаються нормалізації відповідно до формули. Вихідний сигнал $y_i(x)$ визначається виразом, відповідним залежності:

$$y = \sum_{i=1}^{M} \frac{w_i}{\sum_{i=1}^{M} w_i} \left(p_0 + \sum_{J=1}^{N} p_j x_j \right), \qquad (2.16)$$

яка лінійна щодо всіх вхідних змінних системи x_j, j = 1,2 ..., N. Ваги w_i є нелінійними параметрами функції y, які уточнюються в процесі навчання. Коефіцієнти p_0, p_1, ..., p_N - це ваги, що підбираються в процесі навчання.

$$y(x) = f(x) = \frac{f_1}{f_2}. \qquad (2.17)$$

За формулою (2.17) також непараметричний шар.

При уточненні функціональної залежності для мережі TSK отримуємо:

$$y(x) = \frac{1}{\sum_{i=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(i)}(x_j)\right]} \sum_{i=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(i)}(x_j)\right]\left[p_{i0} + \sum_{j=1}^{N} p_{ij}x_j\right]. \qquad (2.18)$$

Якщо прийняти, що в конкретний момент часу параметри умови зафіксовані, то функція $y(x)$ є лінійною щодо змінних x_j, j = 1,2 ..., N.

При наявності N вхідних змінних кожне правило формує N + 1 змінних p_i лінійної залежності TSK. При M правилах виведення це дає M (N + 1) лінійних параметрів мережі. У свою чергу кожна функція приналежності використовує три параметри, які підлягають адаптації. Так як кожна змінна x_j характеризується власною функцією приналежності, то ми отримаємо 3 MN нелінійних параметрів. В сумі це дає M (4N + 1) лінійних і нелінійних параметрів, значення яких повинні підбиратися в процесі навчання мережі.

У цій роботі пропонується обрати саме багатошаровий персептрон, оскільки він краще підійде для нашої задачі через простоту і легкість навчання, а його узагальнюючої спроможності буде достатньо для нашого набору даних.

2.2 Структура процесів обробки інформації технології класифікації ірисів

Структура процесів обробки інформації технології класифікації ірисів складається з таких етапів:

1. Завантаження набору даних із файла.
2. Підготовка даних для обробки.

3. Створення моделі нейронної мережі.
4. Навчання нейронної мережі.
5. Прогнозування результатів за допомогою моделі нейромережі.
6. Оцінювання точності моделі нейронної мережі.

Процеси навчання та оцінювання мають вирішальне значення для будь-якої штучної нейронної мережі. Ці процеси зазвичай виконуються з використанням двох наборів даних, одного для навчання, а іншого тестового для перевірки точності навченої мережі. У реальному світі ми часто отримуємо лише один набір даних, а потім розділяємо його на два окремі набори даних. Для навчального набору ми зазвичай використовуємо 80% даних і ще 20% використовуємо для оцінки точності моделі нейромережі. У цій програмі набір даних вже поділено на навчальний і тестовий. Тобто є два окремих файли, в одному - навчальний набір даних, а в другому - тестовий набір.

2.3 Архітектура нейронної мережі багатошаровий персептрон

Нейронна мережа багатошаровий персептрон є найпоширенішим типом штучної нейронної мережі - мережа прямого зв'язку зі зворотним розповсюдженням. Прямий зв'язок означає, що сигнал зазвичай рухається через мережу в одному напрямку. Зворотне розповсюдження передбачає, що ми будемо виявляти помилки наприкінці проходження кожного сигналу по мережі і намагатимемося поширювати виправлення цих помилок у зворотному напрямку, особливо значно впливаючи на нейрони, найбільш відповідальні за ці помилки.

Найменшою одиницею штучної нейронної мережі є нейрон. Він зберігає вектор ваг, які є звичайними числами з плаваючою точкою. Нейрону передається вектор вхідних даних, які також є звичайними числами з плаваючою точкою. Нейрон поєднує вхідні дані зі своїми вагами у вигляді скалярного добутку. Потім запускає функцію активації для цього добутку та

видає результат на виході. Можна вважати, що ця дія аналогічна до поведінки справжніх нейронів.

Функція активації - це перетворювач вихідного сигналу нейрона. Вона майже завжди нелінійна, що дозволяє нейронним мережам представляти рішення нелінійних завдань. Якби не було функцій активації, то вся нейронна мережа була б просто лінійним перетворенням. На рис. 2.6 показано роботу одного нейрона.

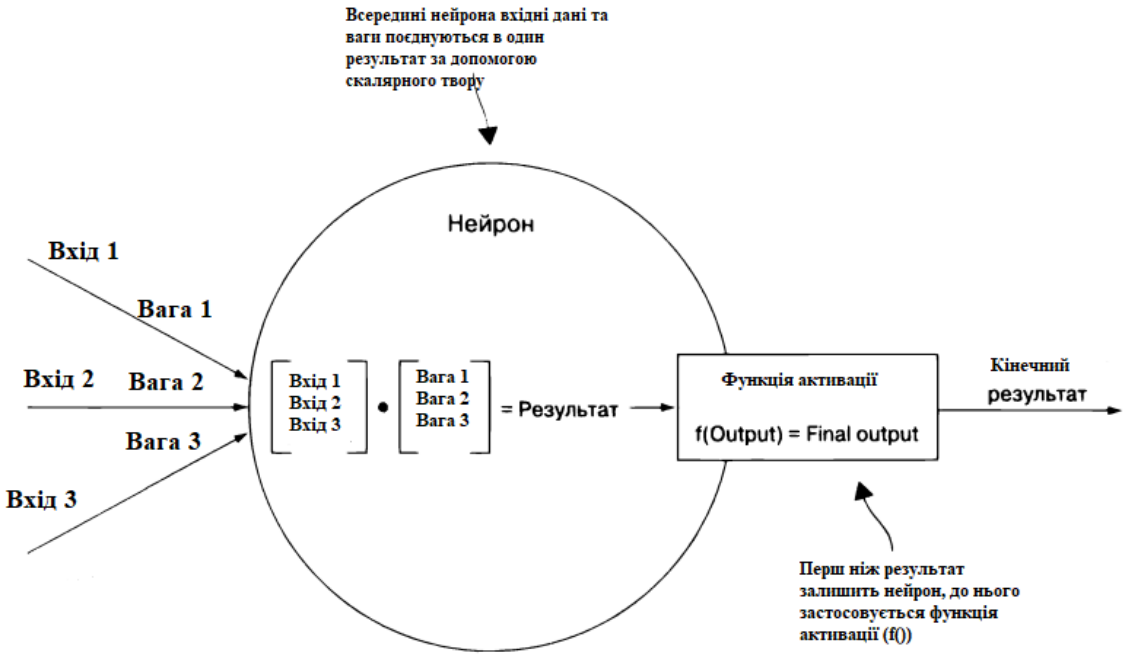

Рисунок 2.6 - У нейроні його ваги поєднуються з вхідними сигналами для отримання результуючого сигналу, що модифікується функцією активації

У типовій штучній нейронній мережі з прямим зв'язком нейрони утворюють шари (рис. 2.7). Кожен шар складається з певної кількості нейронів, збудованих у рядок або стовпець, залежно від діаграми (варіанти еквівалентні). У нейромережі з прямим розповсюдженням, яку ми будемо створювати, сигнали завжди передаються в одному напрямку від одного шару до іншого. Нейрони в кожному шарі надсилають вихідний сигнал, який є вхідним для нейронів наступного шару. Кожен нейрон у кожному шарі пов'язаний із кожним нейроном у наступному шарі.

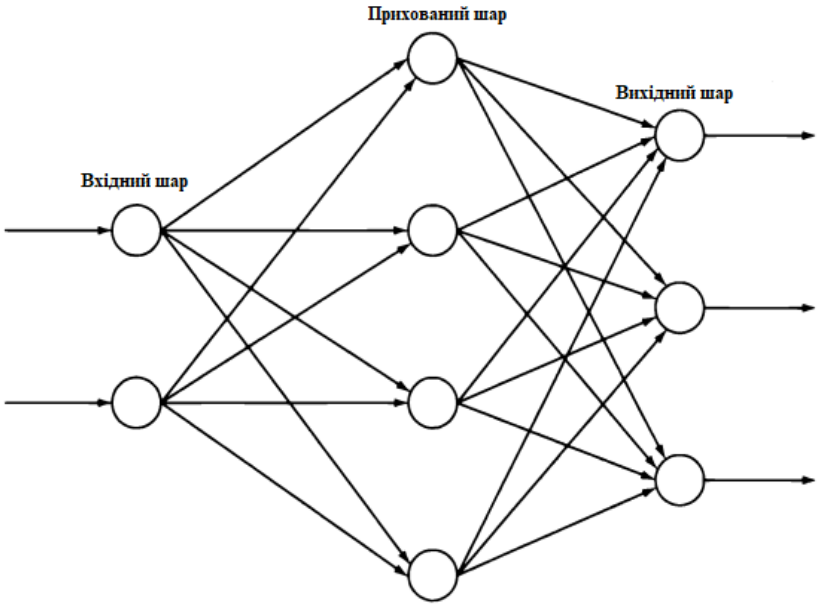

Рисунок 2.7 - Проста нейронна мережа з одним вхідним шаром, що складається з двох нейронів, одним прихованим шаром з чотирьох нейронів та одним вихідним шаром з трьох нейронів. Кількість нейронів у кожному шарі може бути довільна.

Перший шар називається вхідним, він отримує сигнали деякого зовнішнього об'єкта. Останній шар відомий як вихідний, і його вихідні сигнали зазвичай повинен інтерпретувати зовнішній суб'єкт, щоб отримати осмислений результат. Шари, розташовані між вхідним та вихідним шарами, називаються прихованими. У простих нейронних мережах, таких як та, яку ми будуватимемо в цьому розділі, є лише один прихований шар, але в мережах глибокого навчання шарів багато. На рис. 2.7 показані шари, що утворюють просту мережу. Вихідні сигнали одного шару використовуються як вхідні для кожного нейрона наступного шару.

Ці шари просто маніпулюють числами з плаваючою комою. Вхідні сигнали для вхідного шару та вихідні сигнали вихідного шару є числами з плаваючою комою.

Очевидно, що ці числа повинні являти щось значуще. Розробимо нейромережу для класифікації ірисів. Оскільки квітки ірисів характеризуються чотирма параметрами (довжина чашолистка, ширина чашолистка, довжина пелюстки, ширина пелюстки), то вхідний шар нашої нейромережі повинен мати 4 нейрони (рис. 2.8). А кількість класів ірисів, які потрібно прогнозувати, є 3 (Ірис Щетинистий, Ірис Різнокольоровий та Ірис Віргінський), тому вихідний шар нейромережі повинен мати 3 нейрони, що будуть давати ймовірність того, що вхідні параметри квітки ірису відповідають кожному із трьох класів.

Остаточна класифікація може бути визначена вихідним нейроном з найвищим вихідним сигналом з плаваючою комою. Якби вихідні числа дорівнювали 0,24, 0,70, 0,06, то таку квітку ірису можна було б класифікувати як Ірис Різнокольоровий (2-й клас).

У i-тий нейрон першого прихованого шару (рис. 2.8) надходять вхідні значення, наприклад, у нашому випадку 4 параметри розміру частин квітки: x_1, x_2, … x_4 (елементи вхідного вектора) з відповідними вагами $w_{1,i}$, $w_{2,\mathrm{i}}$, … $w_{4,\mathrm{i}}$. Далі, всередині нейрона відбувається обчислення двох операцій, а саме, композиції лінійної та нелінійної функції:

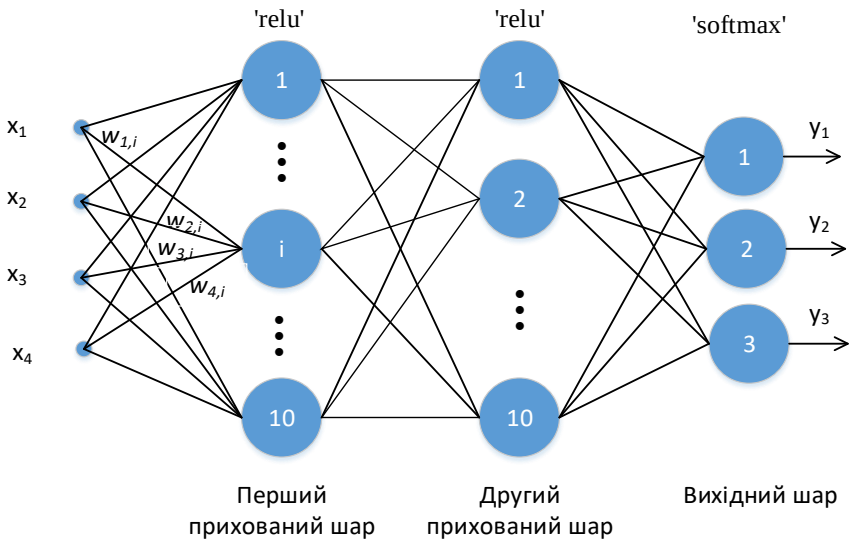

Рисунок 2.8 – Структура багатошарового персептрона для інформаційної технології класифікації ірисів

Було обрано 2 прихованих шари, оскільки відомо [5,6], що кількість прихованих шарів більше 2 не впливає на здатність мережі розділяти класи, обмежені неопуклими багатовимірними областями. Число нейронів у кожному класі було обрано по 10 (підібрано емпірично), оскільки дані не дуже складні і небагаточисельні, то такої кількості вистачає. Функція активації була обрана ReLu [5,13] для обох прихованих шарів, а для вихідного шару – Softmax [5,13].

2.4 Метод навчання нейромережі багатошаровий персептрон

Будемо використовувати метод зворотного поширення помилки (back propagation). Зворотне поширення дозволяє виявити помилку у вихідних даних нейронної мережі та задіяти її для зміни ваги нейронів. Найсильніше змінюються нейрони, найбільше відповідальні за помилку. Помилки виникають на етапі навчання нейронної мережі, коли бажаний вихідний сигнал не збігається з фактичним вихідним сигналом нейрона.

Перш ніж з нейронною мережею можна буде працювати, її, як правило, необхідно навчити. Але ми повинні знати правильні вихідні дані для певних вхідних даних, щоб можна було, використовуючи різницю між очікуваними та фактичними вихідними даними, знаходити помилки та змінювати ваги. Іншими словами, нейронні мережі нічого не знають, поки їм не повідомлять правильні відповіді для певного набору вхідних даних, щоб потім вони могли підготуватися до інших вхідних даних. Зворотне поширення відбувається лише під час навчання.

Оскільки більшість нейронних мереж потребує навчання, то їх відносять до типу контрольованого машинного навчання. А алгоритм k-середніх та інші кластерні алгоритми вважаються формою неконтрольованого машинного навчання, оскільки після їх запуску зовнішнє втручання не потрібне. Існують інші типи нейронних мереж, відмінні від багатошарового

персептрону, які не вимагають попередньої підготовки та вважаються формою неконтрольованого навчання.

Першим кроком при зворотному поширенні є обчислення помилки між вихідним сигналом нейронної мережі деякого вхідного сигналу і очікуваним вихідним сигналом (рис. 2.9). Ця помилка поширюється на всі нейрони у вихідному шарі. (Для кожного нейрона є очікуваний і фактичний вихідний сигнал.)

Потім до того, що було б вихідним сигналом нейрона до того, як була задіяна його функція активації, застосовується похідна функції активації вихідного нейрона (кешуємо результат його функції перед активацією). Цей результат множать на помилку нейрона, щоб знайти його дельту. Формула обчислення дельти використовує часткову похідну. Головне, що ми отримуємо в результаті, те, за скільки помилок відповідає кожен вихідний нейрон. Діаграма обчислення показана на рис. 2.9.

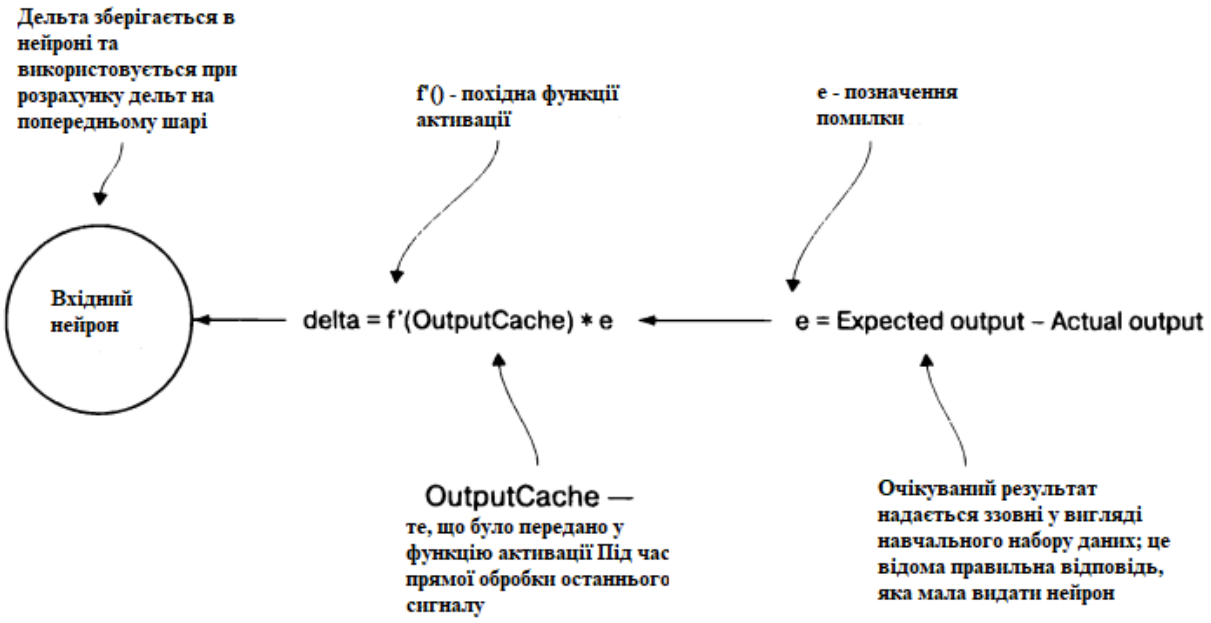

Рисунок 2.9 - Механізм, за допомогою якого обчислюється дельта вихідного нейрона на етапі навчання за допомогою зворотного розповсюдження

Потім необхідно обчислити дельти для всіх нейронів прихованого шару (шарів) даної мережі (рис. 2.10).

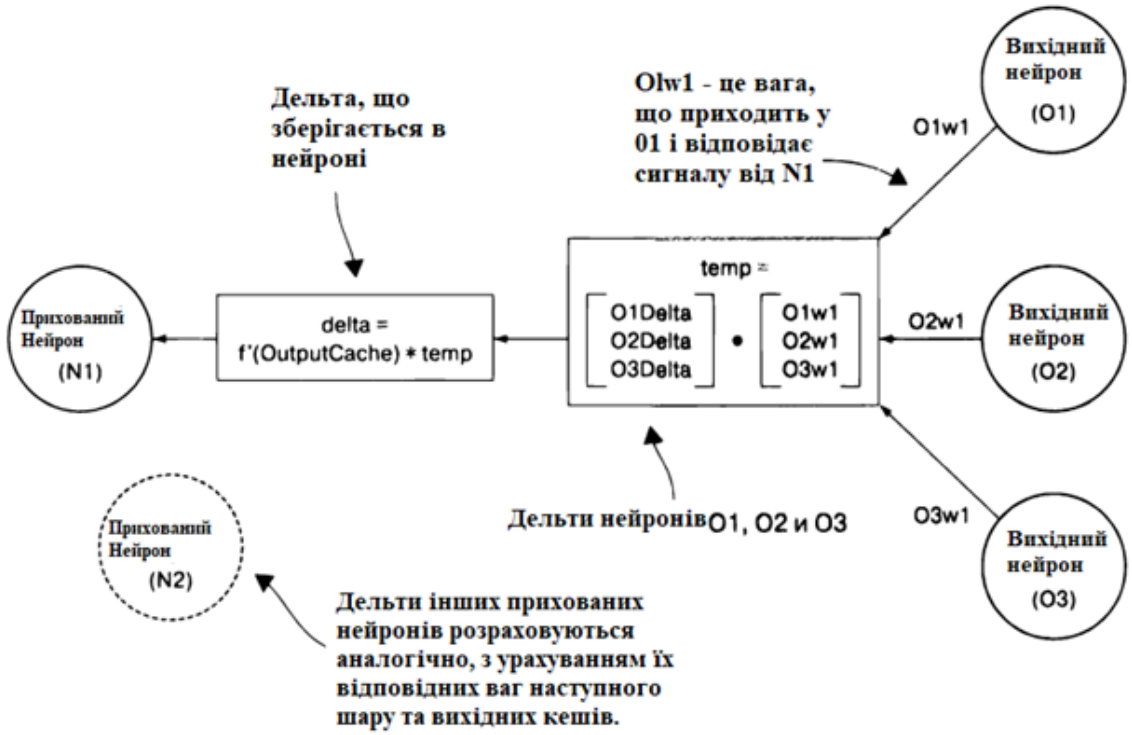

Рисунок 2.10 - Обчислення дельти для нейрона у прихованому шарі

Потрібно визначити, наскільки кожен нейрон є відповідальним за неправильний вихідний сигнал у вихідному шарі. Дельти вихідного шару використовуються для обчислення дельт у попередньому прихованому шарі. Для кожного попереднього шару дельти обчислюються визначенням скалярного добутку ваг наступного шару по відношенню до конкретного нейрона, що розглядається, і вже обчислених дельт в наступному шарі. Щоб отримати дельту нейрона, це значення множиться на похідну від функції активації, яка застосовується до останнього вихідного сигналу нейрона, кешованого перед використанням функції активації. Ця формула також отримана за допомогою окремої похідної.

На рис. 2.10 показаний фактичний розрахунок дельт для нейронів у прихованих шарах. У мережі з декількома прихованими шарами нейрони 01,

02 та 03 можуть бути нейронами наступного прихованого шару, а не нейронами вихідного шару.

І останнє, але найголовніше: всі ваги для кожного нейрона в мережі мають бути оновлені шляхом множення останнього вхідного значення кожної окремої ваги на дельту нейрона і щось, що називається швидкістю навчання, та додавання цього значення до існуючої ваги (рис. 2.11).

Цей метод зміни ваги нейрона називається градієнтним спуском. Він схожий на спуск із пагорба, де пагорб – це графік функції помилки нейрона, до точки мінімальної помилки. Дельта — це напрям, у якому ми хочемо спускатися, а швидкість навчання визначає те, як швидко це робитимемо. Важко визначити хорошу швидкість навчання для невідомого завдання без спроб і помилок. На рис. 2.11 показано, як змінюються ваги у прихованому та вихідному шарах.

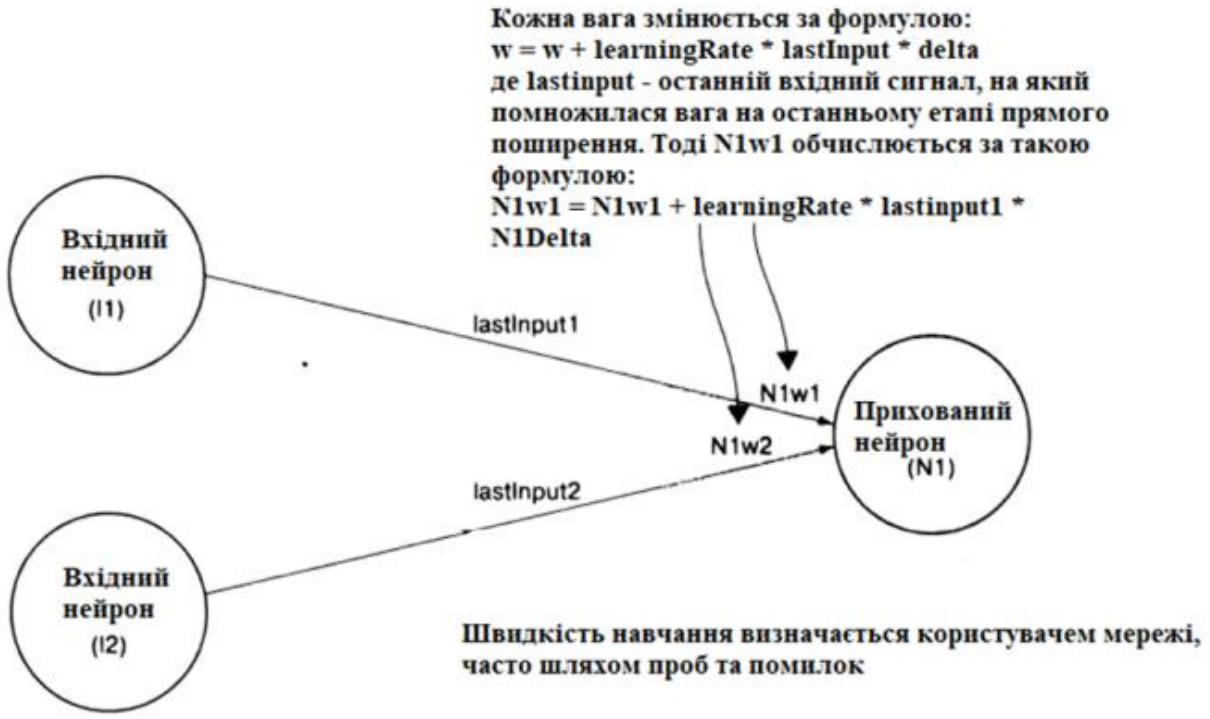

Рисунок 2.11 - Вага кожного нейрона прихованого шару та нейрона вихідного шару оновлюються з використанням дельт, розрахованих на попередніх кроках, попередніх ваг, попередніх вхідних даних та заданої користувачем швидкості навчання

Після зміни ваг нейронна мережа готова до повторного навчання з іншим набором вхідних та очікуваних вихідних даних. Процес повторюється до тих пір, поки користувач нейронної мережі не визнає, що мережа добре навчена. Це можна визначити, перевіривши мережу за вхідними даними з відомими правильними вихідними даними.

Зворотне поширення – складний процес, це спосіб коригування кожної окремої ваги нейрона в мережі відповідно до того, наскільки правильно він визначає неправильні вихідні дані.

Основні властивості мереж прямого зв'язку зі зворотним розповсюдженням.

1. Сигнали (числа з плаваючою комою) проходять через нейрони, розташовані по шарах, в одному напрямку. Кожен нейрон кожного шару пов'язаний з кожним нейроном наступного шару.
2. Кожен нейрон (крім нейронів вхідного шару) обробляє одержувані сигнали, комбінуючи їх з вагами (які також є числами з плаваючою комою) і застосовуючи до них функцію активації.
3. Під час процесу, званого навчанням, результати мережі порівнюються з очікуваними результатами, щоб визначити помилки.
4. Помилки поширюються по мережі у зворотному напрямку (туди, звідки вони прийшли) для зміни ваг, щоб у результаті вони з більшою ймовірністю призводили до вироблення правильних вихідних даних.

2.5 Розробка алгоритму роботи програми нейромережевої класифікації ірисів

Згідно з метою роботи та постановкою задачі було розроблено алгоритм нейромережевої класифікації ірисів, представлений на рис. 2.12.

Першим кроком в програмі нейромережевої класифікації ірисів буде завантаження бібліотек (блок 1). Потім йде завантаження навчального набору

даних (блок 2) та тестового набору даних (блок 3) із параметрами ірисів та їх ідентифікатором класу:

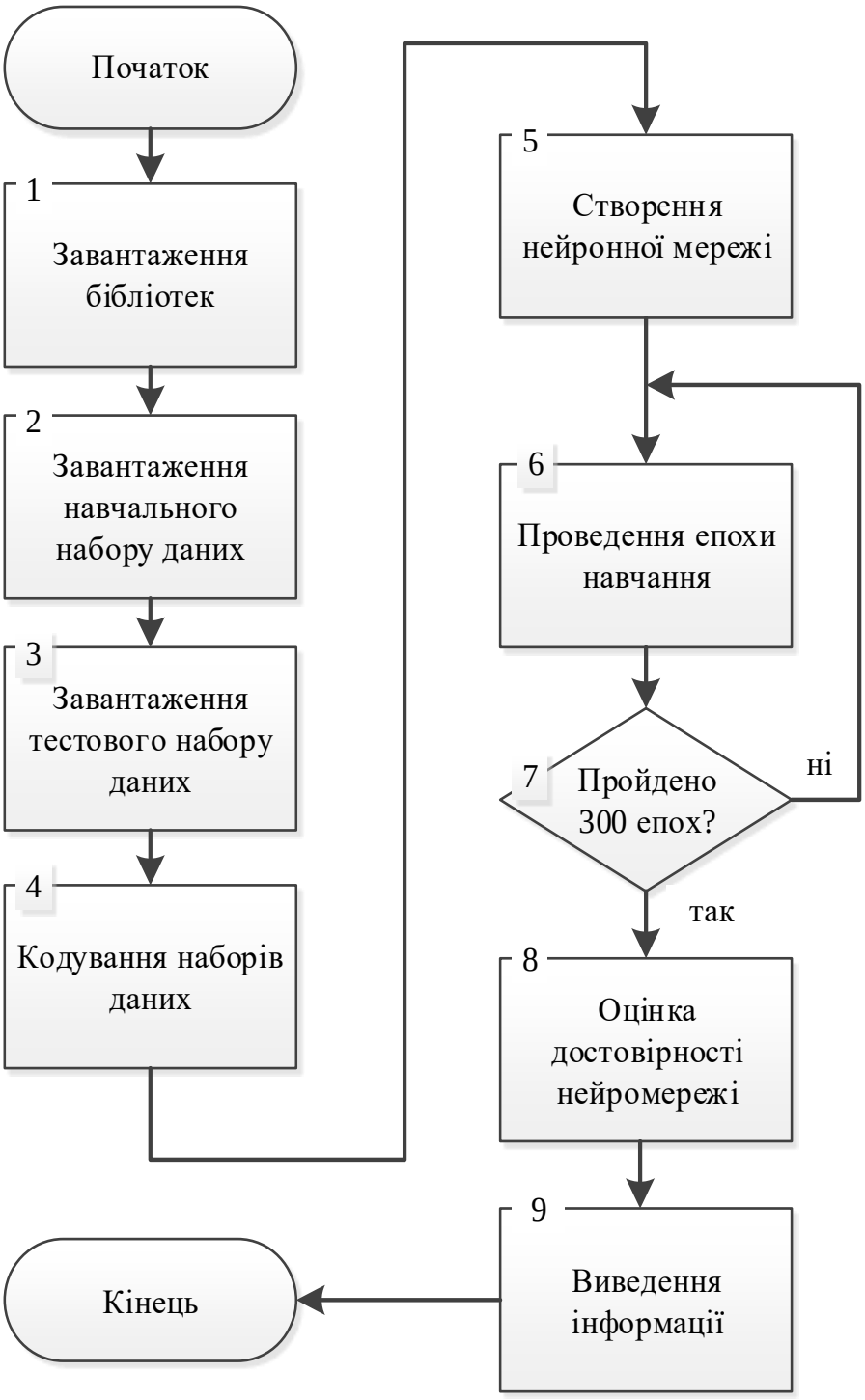

Рисунок 2.12 – Схема алгоритму роботи програми нейромережевої класифікації ірисів

Далі переходимо до кодування наборів даних ірисів (блок 4). Потім проводиться створення нейронної мережі (блок 5). За тим починається процес навчання нейромережі (блок 6), який триває 300 епох (блок 7), після чого проводиться оцінка достовірності роботи нейромережі (блок 8) та виведення достовірності класифікації ірисів на тестовій вибірці (блок 9).

UML-діаграма класів розробленої програми нейромережевої класифікації ірисів представлена на рис. 2.13.

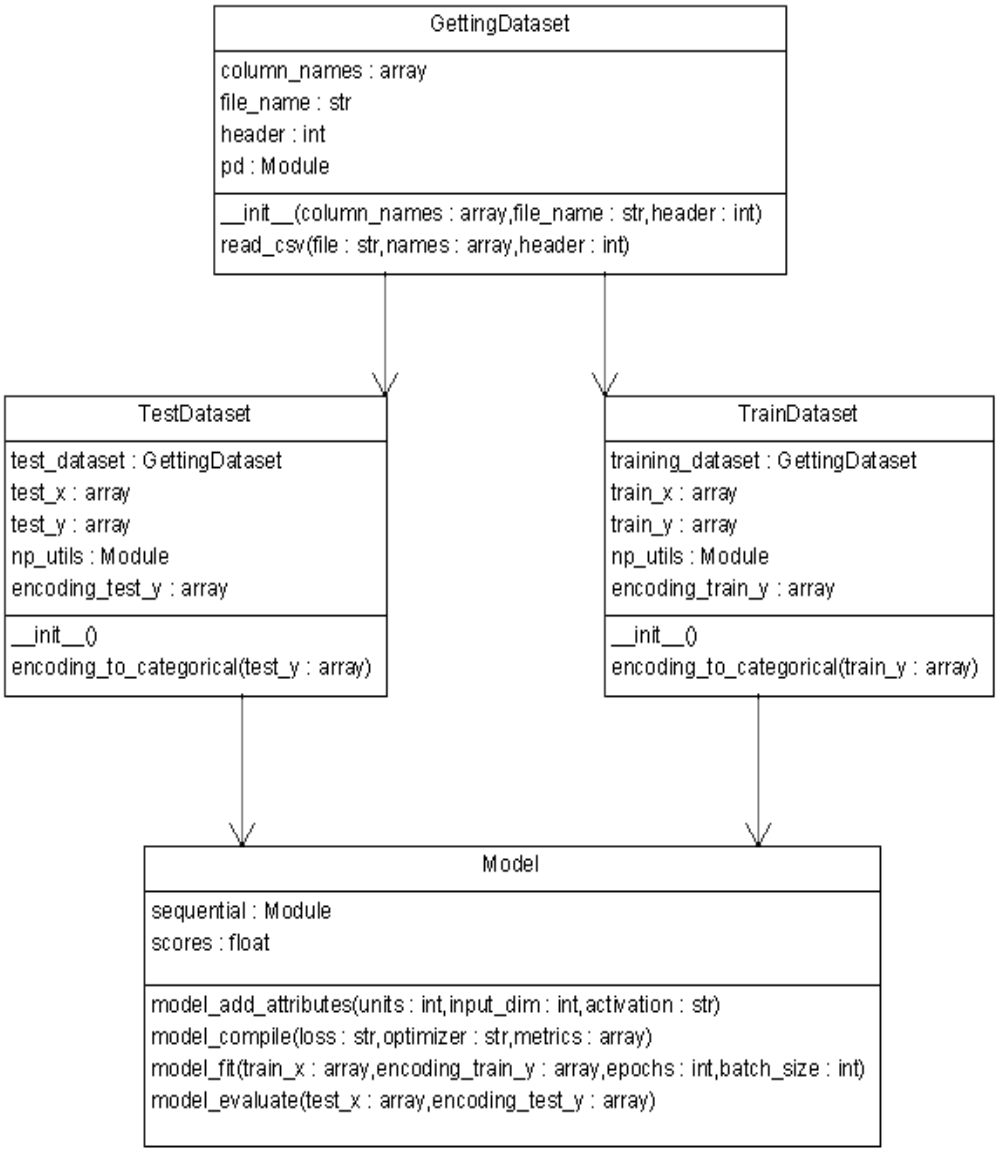

Рисунок 2.13 – UML-діаграма класів програми нейромережевої класифікації ірисів

З отриманої UML-діаграми класів програми нейромережевої класифікації ірисів по рис. 2.13 видно, що вона містить такі класи: GettingDataset для завантаження та аналізу наборів даних, TrainDataset TestDataset та Model для створення, навчання та оцінки достовірності нейронної мережі.

2.6 Висновок до розділу 2

У розділі було обґрунтовано вибір типу нейронної мережі багатошаровий персептрон для інформаційної технології класифікації ірисів. Обрано архітектуру багатошарового персептрона, який має 4 входи (по кількості параметрів квіток ірисів), 2 прихованих шари по 10 нейронів та вихідний шар із 3 нейронів (по кількості класів). Було обрано функцію активації ReLu у двох прихованих шарах та функцію активації Softmax у вихідному шарі. Розроблено структуру процесів обробки інформаційної технології класифікації ірисів. Розроблено алгоритм роботи програми нейромережевої класифікації ірисів та UML діаграму класів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КЛАСИФІКАЦІЇ ІРИСІВ

3.1 Обґрунтування вибору мови та спеціалізованих бібліотек

Для програмної реалізації інформаційної технології класифікації ірисів за допомогою нейронної мережі обрано мову програмування Python та програмне середовище PyCharm.

Python [14,15] (прочитання — «Пайтон») — інтерпретована об'єктно-орієнтована мова програмування високого рівня з суворою динамічною типізацією. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її гідною для швидкої розробки програм. Python підтримує модулі та пакети модулів та сприяє модульності та повторному використанню кодів. Інтерпретатор Python та стандартні бібліотеки доступні і у скомпільованій, і у вихідній формі на усіх основних платформах. У мові Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, функціональна, процедурна та аспектно-орієнтована.

Переваги Python:

- чистий синтаксис (для виділення блоків використовуються відступи);
- переносність програм (це властиве більшості мов);
- стандартний дистрибутив має велику кількість дуже корисних модулів (з модулем розробки графічного інтерфейсу);
- можливість використовувати Python у діалоговому режимі (корисне для експериментування та розв'язання простих завдань);
- стандартний дистрибутив має просте, але досить потужне середовище розробки IDLE і яке написане на Python;

- зручний для розв'язання математичних задач (має засоби роботи з комплексними числами, з цілими числами довільної величини, може використовуватися як потужний калькулятор у діалоговому режимі);
- відкритий код (можливість редагувати код іншими користувачами).

Недоліки:

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

Оскільки ми використовуємо нейронні мережі, то для цієї роботи знадобляться бібліотеки Keras [16], NumPy [17] та Pandas [18].

Keras — відкрита нейромережна бібліотека, написана мовою Python. Вона може працювати поверх TensorFlow, Microsoft Cognitive Toolkit, R, Theano та PlaidML. ЇЇ було спроєктовану для уможливлення швидких експериментів з мережами глибинного навчання, та зосереджено на тому, щоб вона була зручною у використанні, модульною та розширюваною. Keras було створено як частину дослідницьких зусиль проєкту ONEIROS (англ. Open-ended Neuro-Electronic Intelligent Robot Operating System).

2017 року команда TensorFlow Google вирішила підтримувати Keras в основній бібліотеці TensorFlow. Було пояснено, що Keras замислювався радше як інтерфейс, аніж як самостійна система машинного навчання.

NumPy (скорочення від Numerical Python) —фреймворк з відкритим кодом для Python. NumPy має такі можливості:

- підтримка багатовимірних масивів (матриць);
- підтримка високорівневих математичних функцій для роботи з багатовимірними масивами.

Pandas — програмна бібліотека [18], написана для мови програмування Python для маніпулювання даними та їхнього аналізу. Вона, зокрема, пропонує структури даних та операції для маніпулювання чисельними таблицями та часовими рядами. Pandas є вільним програмним забезпеченням, що випускається за трипунктовою ліцензією BSD. Ця назва походить від терміну «панельні дані» (англ. panel data), який в економетрії позначає багатовимірні структуровані набори даних.

Можливості бібліотеки:

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

3.2 Використання спеціалізованої бібліотеки TensorFlow

У TensorFlow все обертається навколо тензорів, які є примітивними одиницями в TensorFlow. TensorFlow використовує тензорну структуру для представлення всіх даних.

У математиці тензори - це геометричні об'єкти, які описують лінійні відношення між іншими геометричними об'єктами. У TesnsorFlow це багатовимірні масиви або дані, тобто матриці. Вся ця концепція тензорів походить з лінійної алгебри.

Ми можемо розглядати тензори як n-вимірні масиви, за допомогою яких легко і ефективно виконуються матричні операції. Наприклад, у коді нижче ми визначили два константні тензори і додали одне значення до іншого:

```
import tensorflow as tf

const1 = tf.constant([[1,2,3], [1,2,3]]);
const2 = tf.constant([[3,4,5], [3,4,5]]);

result = tf.add(const1, const2);

with tf.Session() as sess:
  output = sess.run(result)
  print(output)
```

Константи - це значення, які не змінюються. Однак, TensorFlow має багатий API, який добре задокументований, і з його допомогою ми можемо визначати інші типи даних, наприклад, змінні:

```
import tensorflow as tf

var1 = tf.Variable([[1, 2], [1, 2]], name="variable1")
var2 = tf.Variable([[3, 4], [3, 4]], name="variable2")

result = tf.matmul(var1, var2)

with tf.Session() as sess:
  output = sess.run(result)
  print(output)
```

Окрім тензорів, TensorFlow використовує графи потоків даних. Вузли графа представляють математичні операції, а ребра - тензори, що передаються між ними.

Екосистема TensorFlow.

Звичайно, ми не хочемо просто виконувати прості арифметичні операції, ми хочемо використовувати цю бібліотеку для побудови предикторів, класифікаторів, генеративних моделей, нейронних мереж і так далі. Загалом, при побудові таких рішень ми маємо пройти кілька етапів:

1. Аналіз та попередня обробка даних.
2. Побудова та навчання моделі (модель машинного навчання, нейронна мережа, ...).
3. Оцінювання моделі.
4. Створення нових прогнозів.

Оскільки навчання цих моделей може бути дорогим і тривалим процесом, ми можемо використовувати для цього різні машини. Навчання цих моделей на CPU може зайняти досить багато часу, тому використання GPU завжди є кращим варіантом.

Найшвидшим варіантом для навчання цих моделей є тензорні процесори або TPU. Вони були представлені компанією Google ще в 2016 році і являють собою прикладну інтегральну схему (ASIC) для прискорення ШІ. Однак вони все ще досить дорогі.

Крім того, ми хочемо розгорнути нашу модель на різних платформах, таких як хмара, вбудовані системи (IoT) або інтегрувати її в інші мови. Ось чому екосистема TensorFlow виглядає так, як показано на рис. 3.1.

Ми бачимо, що всі ці основні моменти розробки таких рішень охоплені в рамках цієї екосистеми. Що стосується Python, ми зазвичай аналізуємо та обробляємо дані за допомогою таких бібліотек, як Numpy та Pandas. Потім ми використовуємо ці дані, щоб вставити їх у модель, яку ми побудували.

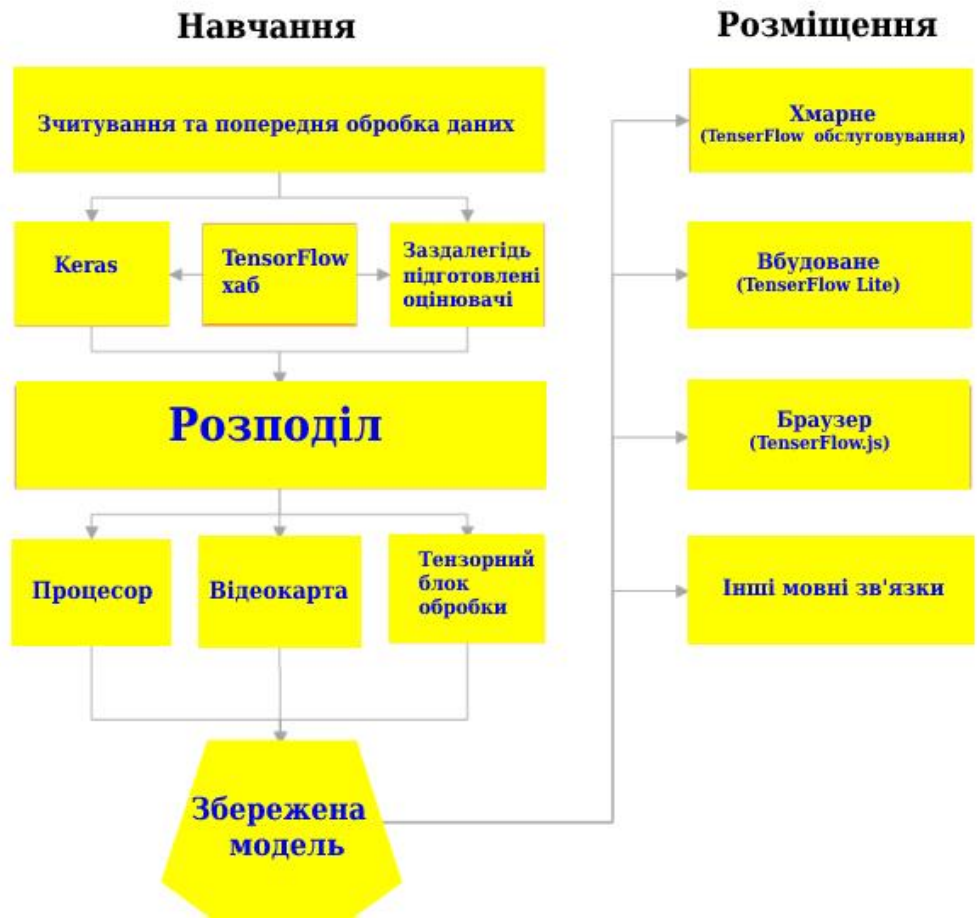

Рисунок 3.1 – Структура екосистеми TensorFlow

Хоча аналіз даних є окремою темою. Однак, TensorFlow надає нам деякі модулі, за допомогою яких ми можемо виконати деяку попередню обробку та розробку функцій. Крім того, він надає набори даних (tensorflow.datasets), які ми можемо використовувати для навчання деяких наших користувацьких рішень і для досліджень в цілому.

Найважливішою частиною TensorFlow є TensorFlow Hub. Там ми можемо знайти численні модулі та низькорівневі API, які ми можемо використовувати. На вершині цих, скажімо так, основних модулів ми можемо знайти високорівневий API - Keras. Можна сказати, що дорога до версії 2.0 була прокладена в TensorFlow 1.10.0, коли Keras був включений як високорівневий API за замовчуванням.

До цього Keras був окремою бібліотекою і для цього використовувався модуль Tensorflow.contrib. У TensorFlow 1.10.0 ми отримали новину, що

модуль Tensorflow.contrib буде скоро вилучено, а його місце займе Keras. І це було одним з основних фокусів TensorFlow 2.0, щоб полегшити використання і очистити API. Насправді, багато API з 1.0 або переміщено, або повністю видалено. Наприклад, tf.app і tf.flags більше не існують, а деякі менш використовувані функції з tf.* переміщені в інші модулі.

Окрім цього високорівневого API, який ми будемо використовувати, є кілька попередньо навчених моделей. Ці моделі навчаються на певному наборі даних і можуть бути налаштовані для нашого рішення.

Такий підхід до розробки рішення для машинного навчання також називається перенесеним навчанням. Перенесене навчання набуває все більшої популярності серед інженерів штучного інтелекту, оскільки воно прискорює процес. Звичайно, ви можете використовувати ці попередньо навчені моделі як готові рішення.

Для TensorFlow також існує кілька варіантів дистрибутиву, тому ми можемо вибрати, на якій платформі ми хочемо навчати наші моделі. Це вирішується під час встановлення фреймворку. Ми можемо вибрати дистрибутив TensorFlow, який працює на CPU, GPU або TPU.

Нарешті, після того, як ми побудували нашу модель, ми можемо зберегти її. Ця модель може бути включена в інші додатки на різних платформах. Тут ми вступаємо у світ розгортання. Важливо зазначити, що побудова моделі - це зовсім інший процес, ніж решта розробки додатку. Загалом, data scientist будує ці моделі і зберігає їх. Пізніше ці моделі викликаються з компонентів бізнес-логіки додатку.

Встановлення та налаштування TensorFlow.

TensorFlow надає API для широкого спектру мов, таких як Python, C++, Java, Go, Haskell та R (у вигляді сторонньої бібліотеки). Крім того, він підтримує різні типи операційних систем. У цій роботі ми будемо використовувати Python на Windows 10, тому розглянемо процес встановлення на цій платформі.

TensorFlow підтримує тільки Python 3.5 і вище, тому треба переконатися, що у нас встановлена одна з цих версій. Для інших операційних систем та мов можна перевірити офіційний посібник з встановлення. Ще одна річ, яку нам потрібно знати - це апаратна конфігурація нашої системи. Існує два варіанти встановлення TensorFlow:

1. TensorFlow з підтримкою лише CPU.
2. TensorFlow з підтримкою GPU.

Якщо наша система має графічний процесор NVIDIA®, ми можемо встановити TensorFlow з підтримкою GPU. Звичайно, версія для GPU працює швидше, але CPU простіше встановлювати і налаштовувати.

Якщо використовується Anaconda, встановлення TensorFlow можна виконати таким чином:

1.Створити середовище conda "tensorflow", виконавши команду:

```
conda create -n tensorflow pip python=3.5
```

2. Активувати створене середовище командою:

```
activate tensorflow
```

3.Викликати команду для встановлення TensorFlow у нашому середовищі. Для процесорної версії запустіть цю команду:

```
pip install --ignore-installed --upgrade tensorflow
```

Для GPU-версії виконайте команду:

```
pip install --ignore-installed --upgrade tensorflow-gpu
```

Звичайно, можна встановити TensorFlow і за допомогою "рідного" pip. Для версії для CPU виконайте:

```
pip3 install --upgrade tensorflow
```

Для GPU-версії TensorFlow виконайте команду:

```
pip3 install --upgrade tensorflow-gpu
```

Тепер TensorFlow встановлений.

3.3 Опис набору даних для навчання та тестування програми класифікації ірисів за допомогою нейронної мережі

Набір даних Iris, поряд з набором даних MNIST, є, мабуть, одним з найвідоміших наборів даних, які можна знайти в літературі з розпізнавання образів. Це свого роду приклад "Hello World" для задач класифікації машинного навчання. Вперше його представив Рональд Фішер у 1936 році. Він був британським статистиком і ботаніком і використав цей приклад у статті "Використання множинних вимірювань у таксономічних задачах", на яку часто посилаються і донині.

У папці проекту міститься файл із даними, розділеними комами (у форматі CSV), який містить набір даних про іриси. Цей набір отримано з репозиторію машинного навчання UCI Каліфорнійського університету: Lichman M. UCI Machine Cearning Repository. Irvine, CA: University of California, School of Information and Computer Science, 2013, [2]. CSV-файл - це просто текстовий файл зі значеннями, розділеними комами. Це загальний формат обміну табличними даними, включаючи електронні таблиці. Фрагмент набору даних ірисів показано на рис. 3.2.

Набір даних містить 3 класи по 50 екземплярів у кожному (всього 150). Кожен клас відноситься до одного типу рослин ірису: Iris Setosa (ідентифікатор 0 в останньому стовпчику рис. 3.2), Iris Virginica (ідентифікатор 1 в останньому стовпчику рис. 3.2) та Iris Versicolor (ідентифікатор 2 в останньому стовпчику рис. 3.2). Перший клас (0) є лінійно відокремленим від двох інших (1 та 2), але два останніх не є лінійно відокремленими один від одного. Кожна пластинка має п'ять ознак:

- Довжина чашолистка в см (SepalLength),
- Ширина чашолистка в см (SepalWidth),

– Довжина пелюстки в см (PetalLength),
– Ширина пелюстки в см (PetalWidth),
– Клас (Species).

1	SepalLength	SepalWidth	PentalLength	PentalWidth	Species
2	6.4	2.8	5.6	2.2	2
3	5.0	2.3	3.3	1.0	1
4	4.9	2.5	4.5	1.7	2
5	4.9	3.1	1.5	0.1	0
6	5.7	3.8	1.7	0.3	0
7	4.4	3.2	1.3	0.2	0
8	5.4	3.4	1.5	0.4	0
9	6.9	3.1	5.1	2.3	2
10	6.7	3.1	4.4	1.4	1
11	5.1	3.7	1.5	0.4	0
12	5.2	2.7	3.9	1.4	1
13	6.9	3.1	4.9	1.5	1
14	5.8	4.0	1.2	0.2	0

Рисунок 3.2 – Фрагмент набору даних ірисів

Зазвичай при навчанні набір розбивають на 2 вибірки: навчальну і тестову рандомно у пропорції приблизно 80 на 20. В нашій програмі ми заздалегідь розбили набір даних на 2 вибірки навчальну (файл **iris_training.csv**) і тестову (файл **iris_test.csv**). Тобто у папці проекту маємо 2 окремих файли. Це зроблено для того, щоб уникнути впливу конкретного розбиття набору даних на точність навчання та тестування при заданні різних параметрів нейромережі.

Мета нейронної мережі, яку ми збираємося створити, - передбачити клас квітки ірису на основі інших атрибутів. Це означає, що потрібно створити модель, яка буде описувати зв'язок між значеннями атрибутів і класом.

3.4 Програмна реалізація нейромережевої класифікації ірисів

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

Першим кроком в програмі нейромережевої класифікації ірисів буде завантаження бібліотек (рис. 3.3):

```
1  # Importing libraries
2  from keras.models import Sequential
3  from keras.layers import Dense
4  from keras.utils import np_utils
5  import numpy
6  import pandas as pd
```

Рисунок 3.3 – Завантаження бібліотек

Далі ми визначаємо назви колонок набору даних (рис. 3.4).

А потім як показано на рис. 3.4 проводимо завантаження навчального набору даних та його кодування, а за тим - завантаження тестового набору даних та його кодування. Як бачите, спочатку ми використали функцію read_csv для імпорту набору даних у локальні змінні, а потім розділили вхідні дані (train_x, test_x) та очікувані результати (train_y, test_y), створивши чотири окремі матриці.

Ми підготували дані, які будуть використані для навчання та тестування.

```
19    # Defining column names for datasets
20    COLUMN_NAMES = [
21            'SepalLength',
22            'SepalWidth',
23            'PetalLength',
24            'PetalWidth',
25            'Species'
26            ]
27
28    # Import training dataset
29    training_dataset = pd.read_csv('iris_training.csv', names=COLUMN_NAMES, header=0)
30    train_x = training_dataset.iloc[:, 0:4].values
31    train_y = training_dataset.iloc[:, 4].values
32
33    # Encoding training dataset
34    encoding_train_y = np_utils.to_categorical(train_y)
35
36    # Import testing dataset
37    test_dataset = pd.read_csv('iris_test.csv', names=COLUMN_NAMES, header=0)
38    test_x = test_dataset.iloc[:, 0:4].values
39    test_y = test_dataset.iloc[:, 4].values
40
41    # Encoding training dataset
42    encoding_test_y = np_utils.to_categorical(test_y)
```

Рисунок 3.4 – Завантаження навчального та тестового наборів даних та їх кодування

Тепер нам потрібно вибрати модель, яку ми будемо використовувати. У нашій задачі ми намагаємося передбачити клас квітки ірису на основі даних атрибутів. Тому ми обрали багатошаровий персептрон (мережа з послідовним слідуванням шарів – **Sequential**) – рис. 3.5.

```
# Creating a model
model = Sequential()
model.add(Dense(10, input_dim=4, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

Рисунок 3.5 – Створення моделі нейромережі

Із рис. 3.5 видно, що ми додали два приховані шари з десятьма нейронами в кожному (**Dense(10)**), причому входів у першого шару нашої нейромережі 4 (**input_dim=4**). Кількість класів ірисів, які потрібно прогнозувати, є 3 (Ірис Щетинистий, Ірис Різнокольоровий та Ірис Віргінський), тому вихідний шар нейромережі має 3 нейрони (**Dense(3)**), що будуть давати ймовірність того, що вхідні параметри квітки ірису відповідають кожному із трьох класів. А імовірність виходить тому, що для вихідного шару була обрана функція активації **softmax**. Для обох прихованих шарів була обрана функція активації **relu**.

Після цього ми компілюємо нашу модель, де визначаємо функцію втрат (**loss**) і оптимізатор (**optimizer**).

```
# Compiling model
model.compile(loss='categorical_crossentropy',
optimizer='backprop', metrics=['accuracy'])
```

У цьому випадку ми використаємо алгоритм оптимізації зворотного розповсюдження ('backprop').

Потім, ми навчаємо нашу мережу:

```
# Training a model
model.fit(train_x, encoding_train_y, epochs=300,
batch_size=10)
```

Бачимо, що обрано тривалість навчання в 300 епох та розмір пакету навчальних прикладів – 10.

Нарешті, ми викликаємо функцію evaluate, яка оцінить нашу нейронну мережу і поверне нам точність мережі.

```
# evaluate the model
scores = model.evaluate(test_x, encoding_test_y)
print("\nAccuracy: %.2f%%" % (scores[1]*100))
```

Під точністю мережі тут розуміється достовірність розпізнавання на тестовій вибірці.

3.5 Висновок до розділу 3

У розділі було обґрунтовано вибір мови програмування Python, середовища програмування PyCharm та спеціалізованих бібліотек Keras, NumPy та Pandas для програмної реалізації інформаційної технології нейромережевої класифікації ірисів. Проведено аналіз набору даних ірисів для навчання та тестування програми. Описано основні етапи програмної реалізація та функціонування програми нейромережевої класифікації ірисів, що складається з завантаження бібліотек, завантаження набору даних із параметрами ірисів та їх ідентифікатором класу, аналізу цих параметрів, створення, компіляції, тренування, оцінки точності моделі нейромережі та виведення результатів.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КЛАСИФІКАЦІЇ ІРИСІВ

4.1 Тестування програмного забезпечення нейромережевої класифікації ірисів

Результатом роботи є програмний застосунок, що здійснює класифікацію ірисів. Для того щоб впевнитися, що програма працює правильно, проведемо її тестування.

Робота програми складається з таких етапів:

1. Аналіз та попередня обробка даних
2. Побудова та навчання моделі
3. Оцінка моделі
4. Створення нових прогнозів

Аналіз та попередня обробка даних.

Аналіз даних зазвичай має такі основні кроки:

- Одновимірний аналіз - аналіз типів і природи кожної функції.
- Обробка відсутніх даних - виявлення відсутніх даних та розробка стратегії щодо них.
- Кореляційний аналіз - порівняння ознак між собою.
- Розділення даних - оскільки ми маємо один набір інформації, нам потрібно створити окремий набір даних для навчання нейронної мережі та набір даних для оцінки нейронної мережі.

Використовуючи інформацію, яку ми збираємо під час цього аналізу, ми можемо зробити відповідні дії під час створення самої моделі. По-перше, ми імпортуємо дані:

```
COLUMN_NAMES = [
        'SepalLength',
        'SepalWidth',
        'PetalLength',
        'PetalWidth',
        'Species'
        ]

data = pd.read_csv('iris_data.csv', names=COLUMN_NAMES, header=0)
data.head()
```

Як ви можете бачити, ми використовуємо для цього бібліотеку Pandas, а також виводимо перші п'ять рядків даних. Це виглядає як показано на рис. 4.1.

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
0	6.4	2.8	5.6	2.2	2
1	5.0	2.3	3.3	1.0	1
2	4.9	2.5	4.5	1.7	2
3	4.9	3.1	1.5	0.1	0
4	5.7	3.8	1.7	0.3	0

Рисунок 4.1 – Виведені перші п'ять рядків даних

Коли це зроблено, ми хочемо побачити, яка природа кожної функції. Для цього ми також можемо використовувати Pandas:

```
data.dtypes
```

Вихідні дані виглядають так:

```
SepalLength    float64
SepalWidth     float64
PetalLength    float64
PetalWidth     float64
Species          int64
dtype: object
```

Як ми бачимо, вид або вивід має тип int64. Однак ми розуміємо, що це не те, що нам потрібно. Ми хочемо, щоб ця функція була категоріальною змінною. Це означає, що нам потрібно трохи модифікувати ці дані, знову ж таки за допомогою Pandas:

```
data['Species'] = data['Species'].astype("category")
data.dtypes
```

Після цього ми перевіряємо, чи немає в нашому наборі даних пропущених даних. Це робиться за допомогою цієї функції:

```
print(data.isnull().sum())
```

Результатом цього виклику буде:

```
SepalLength    0
SepalWidth     0
PetalLength    0
PetalWidth     0
Species        0
dtype: int64
```

Відсутність даних може стати проблемою для нашої нейронної мережі. Якщо в нашому наборі даних є пропущені дані, нам потрібно визначити стратегію, як з ними впоратися. Деякі з підходів полягають у тому, що відсутні значення замінюються середнім значенням ознаки або її максимальним значенням.

Однак, не існує універсального рішення, і іноді різні стратегії дають кращі результати, ніж інші. Далі переходимо до кореляційного аналізу. На цьому кроці ми перевіряємо, як ознаки співвідносяться одна з одною. Використовуючи модулі Pandas та Seaborn, ми отримали зображення, яке показує матрицю з рівнями залежності між деякими ознаками - кореляційну матрицю:

```
corrMatt = data[["SepalLength","SepalWidth","PetalLength","PetalWidth","Species"]].corr()
mask = np.array(corrMatt)
mask[np.tril_indices_from(mask)] = False
fig,ax= plt.subplots()
fig.set_size_inches(20,10)
sn.heatmap(corrMatt, mask=mask,vmax=.8, square=True,annot=True)
```

Ця матриця виглядає як показано на рис. 4.2.

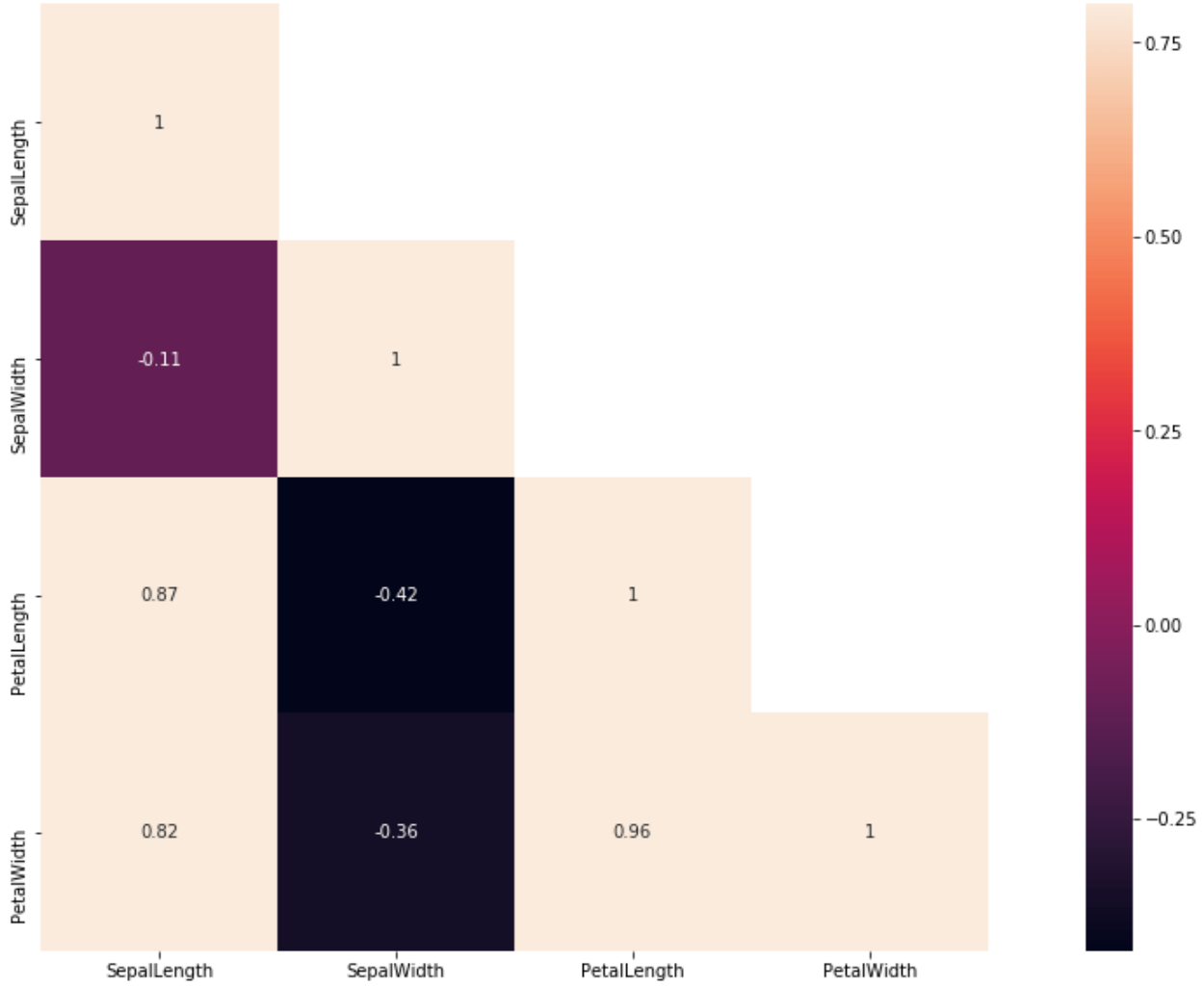

Рисунок 4.2 – Кореляційна матриця (матриця з рівнями залежності між ознаками квіток ірисів)

За допомогою цієї кореляційної матриці ми хотіли знайти зв'язок між типами ірисів (спеціями) та деякими ознаками. Значення, як ви можете бачити на рис. 4.2, знаходяться між -1 і 1. Ми шукаємо ті, що мають значення, близькі до 1 або -1, що означає, що ці ознаки мають занадто багато спільного, тобто занадто сильно впливають одна на одну.

Якщо ми маємо таку ситуацію, пропонується надати моделі лише одну з цих ознак. Таким чином ми уникнемо ситуації, коли наша модель дає надто оптимістичні (або просто неправильні) прогнози. Однак у цьому наборі даних у нас так чи інакше мало інформації, тому, якщо ми видалимо всі залежності, у нас не залишиться жодних даних.

Нарешті, розділимо дані на навчальні та тестові. Оскільки клієнт зазвичай надає нам один великий шматок даних, ми повинні залишити частину даних для тестування. Зазвичай це співвідношення становить 80:20. У цій роботі ми використовуємо 80:20, просто для експерименту. Для цього ми використаємо функцію з бібліотеки SciKit Learn:

```
output_data = data["Species"]
input_data = data.drop("Species",axis=1)
X_train, X_test, y_train, y_test = train_test_split(input_data, output_data, test_size=0.3, random
```

У кінці, ми маємо чотири змінні, які містять вхідні дані для навчання та тестування, а також вихідні дані для навчання та тестування. Тепер ми можемо будувати модель нейромережі.

Після запуску, перше, що виконує програма, це імпортує набір даних і аналізує його. Для цього ми використовуємо ще одну бібліотеку Python - Pandas. Створюється чотири окремі матриці тренувальних та тестових даних як показано на рис. 4.3 та 4.4.

Рисунок 4.3 – Дві матриці із навчальним набором даних

Ми використовуємо функцію read_csv, щоб імпортувати набір даних у локальні змінні, а потім розділили вхідні дані (train_x, test_x) і очікувані виходи (train_y, test_y),.

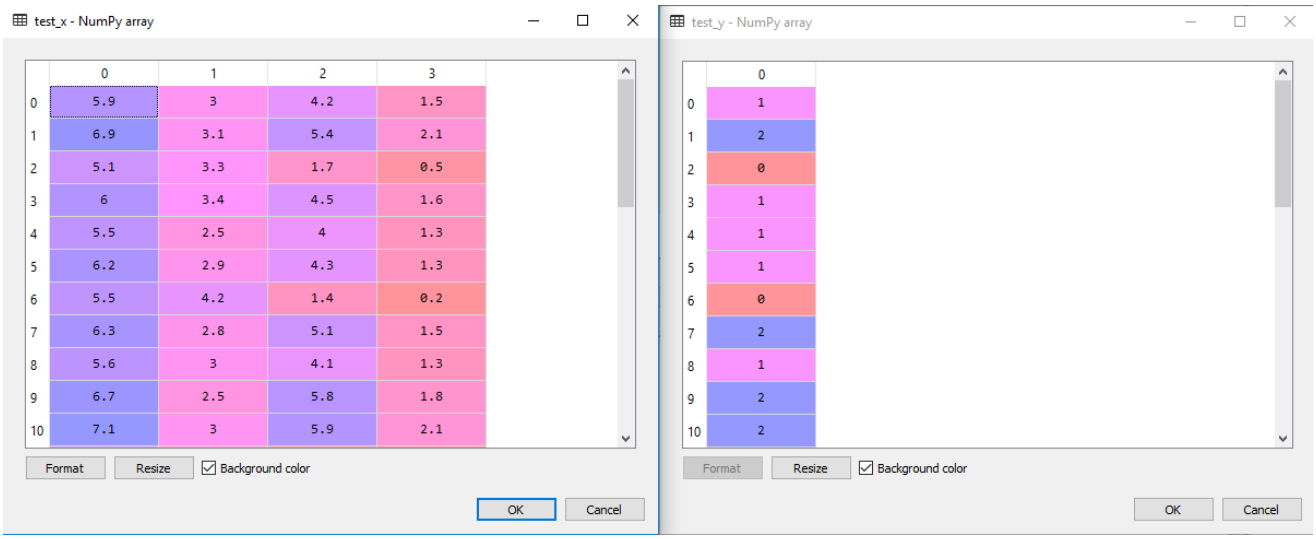

Рисунок 4.4 – Дві матриці із тестовим набором даних

Проте, наші дані ще не підготовлені до обробки. Якщо ми подивимося на наші очікувані вихідні значення, то побачимо, що у нас є три значення: 0, 1 і 2. Значення 0 використовується для представлення Iris Setosa, значення 1 для представлення Iris Versicolor і значення 2 для представлення Iris Virginica. Що ми хочемо зробити, це змінити очікуваний результат із вектора, який містить значення для кожного значення класу, на матрицю з 3 стовпчиків (рис. 4.5) з логічним значенням для кожного ідентифікатора класу.

Рисунок 4.5 – Перетворення вектора, який містить значення для кожного класу, на матрицю з 3 стовпчиків

Це називається кодуванням «один із n». Після цього в програмі створюється та навчається нейронна мережа. Навчимо нашу нейронну мережу на даних, які ми вибрали з навчального набору даних. В останньому шарі нейромережі в якості функції активації використовується softmax. Це означає, що ми отримаємо вихід у вигляді ймовірності належності вхідного обєкту (за його параметрами) до кожного із 3 класів. І, нарешті, викликається функція evaluate, яка оцінює нашу нейронну мережу і повертає нам точність (достовірність) класифікації мережі.

4.2 Аналіз результатів роботи програмного забезпечення класифікації ірисів

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

```
Epoch 1/300
12/12 [==============================] - 1s 2ms/step - loss: 2.7296 - accuracy: 0.3000
Epoch 2/300
12/12 [==============================] - 0s 2ms/step - loss: 1.9793 - accuracy: 0.3000
Epoch 3/300
12/12 [==============================] - 0s 1ms/step - loss: 1.5587 - accuracy: 0.3750
Epoch 4/300
12/12 [==============================] - 0s 932us/step - loss: 1.4221 - accuracy: 0.3583
…

Epoch 148/300
12/12 [==============================] - 0s 914us/step - loss: 0.1110 - accuracy: 0.9750
Epoch 149/300
12/12 [==============================] - 0s 918us/step - loss: 0.1110 - accuracy: 0.9750
Epoch 150/300
12/12 [==============================] - 0s 913us/step - loss: 0.1137 - accuracy: 0.9750
Epoch 151/300
12/12 [==============================] - 0s 913us/step - loss: 0.1080 - accuracy: 0.9667
…

Epoch 297/300
12/12 [==============================] - 0s 1ms/step - loss: 0.0725 - accuracy: 0.9750
Epoch 298/300
12/12 [==============================] - 0s 0s/step - loss: 0.0738 - accuracy: 0.9750
```

```
Epoch 299/300
12/12 [==============================] - 0s 2ms/step - loss: 0.0740 - accuracy: 0.9833
Epoch 300/300
12/12 [==============================] - 0s 1ms/step - loss: 0.0720 - accuracy: 0.9750
1/1 [==============================] - 0s 123ms/step - loss: 0.0589 - accuracy: 0.9667

Accuracy: 96.67%
```

Тобто виводиться результат кожної із 300 епох навчання у вигляді: номер епохи/всього епох/тривалість епохи/похибка навчання на цій епосі (loss)/достовірність класифікації на цій епосі (accuracy). Як бачимо, навчання тривало десь хвилину.

Отже, ми отримали достовірність 96,67% класифікації ірисів для розробленої програми нейромережевої класифікації ірисів, що є досить непоганим показником. Після цього ми повинні порівняти наші результати із програмами-аналогами. Чисельні результати роботи розробленого програмного забезпечення у порівнянні із кращим методом-аналогом із табл. 1.1, яким є метод опорних векторів, занесено до табл. 4.1.

Таблиця 4.1 – Порівняння достовірності різних методів класифікації ірисів

Метод класифікації	Кількість записів у навчальній вибірці	Кількість записів у тестовій вибірці	Достовірність класифікації на тестовій вибірці
Метод машини опорних векторів	120	30	92,6%
Нейромережевий метод (розроблена програма)	120	30	96,7%

Із табл. 4.1 видно, що розроблена програма має достовірність класифікації ірисів на тестовій вибірці 96,7%, а найкращий із 6-х методів, модельованих програмою-аналогом, має достовірність класифікації на тестовій вибірці 92,6%.

Таким чином, можна зробити висновок, що розроблена програма має порівняно з методами-аналогами збільшену на 4,1% достовірність

класифікації ірисів. Тобто мета роботи досягнута – достовірність класифікації ірисів підвищена.

4.3 Висновок до розділу 4

У розділі в результаті тестування програми було доведено її повну працездатність та відповідність поставленому завданню. Розроблена програма має достовірність класифікації ірисів на тестовій вибірці 96,7%, а найкращий із 6-х методів, модельованих програмою-аналогом, має достовірність класифікації на тестовій вибірці 92,6%. Таким чином, розроблена програма має порівняно з методами-аналогами збільшену на 4,1% достовірність класифікації ірисів. Тобто мета роботи досягнута – достовірність класифікації ірисів підвищена.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Проведення комерційного та технологічного аудиту інформаційної технології вирішення задачі нейромережевої класифікації ірисів.

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності, тобто під час виконання магістерської кваліфікаційної роботи.

Для проведення комерційного та технологічного аудиту залучаємо 3-х незалежних експертів, якими є провідні викладачі випускової або спорідненої кафедри.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснюємо із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями [19,20], а результати зводимо до таблиці 5.1.

Таблиця 5.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу інформаційної технології вирішення задачі нейромережевої класифікації ірисів.

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Технічна здійсненність концепції	3	2	3
Ринкові переваги (наявність аналогів)	2	3	2
Ринкові переваги (ціна продукту)	3	2	3
Ринкові переваги (технічні властивості)	3	2	3
Ринкові переваги (експлуатаційні витрати)	2	3	3
Ринкові перспективи (розмір ринку)	3	3	2
Ринкові перспективи (конкуренція)	2	3	3
Практична здійсненність (наявність фахівців)	3	2	3

Продовження Таблиця 5.1

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Практична здійсненність (наявність фінансів)	2	3	3
Практична здійсненність (необхідність нових матеріалів)	3	3	3
Практична здійсненність (термін реалізації)	3	2	3
Практична здійсненність (розробка документів)	3	2	2
Сума балів	32	30	31
Середньоарифметична сума балів, СБ	31		

За результатами розрахунків, наведених в таблиці 5.1 робимо висновок про те, що науково-технічний рівень та комерційний потенціал інформаційної технології вирішення задачі нейромережевої класифікації ірисів – вище середнього.

5.2 Розрахунок витрат на здійснення науково-дослідної розробки інформаційної технології вирішення задачі нейромережевої класифікації ірисів

Витрати на оплату праці. Належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці, також

будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці» [20].

Основна заробітна плата дослідників. Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{к} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (5.1)$$

де к – кількість посад дослідників, залучених до процесу дослідження; $М_{ni}$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.; $Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (21…23) дні; t_i – число робочих днів роботи розробника (дослідника).

Зроблені розрахунки зводимо до таблиці 5.2.

Таблиця 5.2 – Витрати на заробітну плату дослідників

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	10000	476	5	2380
Розробник	8000	381	10	3810
Консультанти	8000	381	8	3048
Всього:	9238			

Основна заробітна плата робітників. Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (5.2)$$

де $С_i$ – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год; t_i – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду С і можна визначити за формулою:

$$C_i = \frac{М_м \cdot К_i \cdot К_c}{Т_р \cdot t_{зм}}, \quad (5.3)$$

де $М_м$ – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), у 2024 році $М_м$=8000 грн; $К_i$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду; $К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати; $Т_р$ – середня кількість робочих днів в місяці, приблизно $Т_р$ = 21…23 дні; $t_{зм}$ – тривалість зміни, год.

Таблиця 5.3 – Витрати на заробітну плату робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Аналіз предметної області	23	5	61,8	1,36	1422
Моделювання інформаційної технології	113	5	61,8	1,36	6983
Створення основної структури проєкту	18	6	65,9	1,45	1186
Створення та наповнення бази даних	79	6	65,9	1,45	5206
Тестування інформаційної технології	71	5	61,8	1,36	4388
Всього					19185

Додаткова заробітна плата. Додаткова заробітна плата $З_д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи,

розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = 0{,}1 \cdot (З_о + З_р) = 0{,}1 \cdot (9238 + 19185) = 2842{,}3 \text{ грн.}$$

Відрахування на соціальні заходи. Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{зп} = \beta \cdot (З_о + З_р + З_д) =$$
$$= 0{,}22 \cdot (9238 + 19185 + 2842{,}3) = 6878{,}4 \text{ грн.}$$

де $З_о$ – основна заробітна плата розробників, грн.; $З_р$ – основна заробітна плата робітників, грн.; $З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.; β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % (приймаємо для 1-го класу професійності ризику 22%).

Амортизація обладнання. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час (чи для) виконання даного етапу роботи.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц_б}{Т_в} \cdot \frac{t}{12}, \quad (5.4)$$

де $Ц_б$ – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.; t – термін використання основного фонду, місяці; $Т_в$ – термін корисного використання основного фонду, роки.

Таблиця 5.4 – Амортизаційні відрахування за видами основних фондів

Найменування	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців	Сума амортизації, грн.
Ноутбук	30000	2	2	2500
Стіл	7000	5	2	233
Стілець	3000	5	2	100
Мишка	1300	2	2	108
Клавіатура	3000	2	2	250
Настільна лампа	1000	3	2	56
Всього	3247			

Витрати на електроенергію для науково-виробничих цілей. Витрати на силову електроенергію $В_е$, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою:

Таблиця 5.5 – Витрати на електроенергію

Найменування обладнання	Потужність, кВт	Тривалість годин роботи
Ноутбук	0,25	320
Освітлення	0,031	140

$$Ве = \sum \frac{W_i \cdot t_i \cdot Ц_е \cdot К_{впі}}{ККД}$$

$$= \frac{0{,}25 \cdot 320 \cdot 7{,}5 \cdot 0{,}95}{0{,}95} + \frac{0{,}031 \cdot 140 \cdot 7{,}5 \cdot 0{,}95}{0{,}95} = 633 \text{ грн.},$$

W_i – встановлена потужність обладнання, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год.; Це – вартість 1 кВт електроенергії, грн.; $К_{впі}$ – коефіцієнт використання потужності; ККД – коефіцієнт корисної дії обладнання.

Інші витрати. До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$\text{Ів} = \left(З_о + З_р\right) \cdot \frac{Н_{ів}}{100\%} = (9238 + 19185) \cdot \frac{50}{100} = 14212 \text{ грн.},$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

Накладні (загальновиробничі) витрати. До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$\text{Внзв} = \left(З_о + З_р\right) \cdot \frac{Н_{нзв}}{100\%} = (9238 + 19185) \cdot \frac{100}{100} = 28423 \text{ грн.},$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Витрати на проведення науково-дослідної роботи. Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + К_в + В_{спец} + В_{прг} + А_{обл} + В_е +$$

$$+I_в + В_{нзв} =$$
$$= 9238 + 19185 + 2842{,}3 + 6878{,}4 + 3247 + 633 + 14212 +$$
$$28423 = 84659 \text{ грн.}$$

Загальні витрати. Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} = \frac{84659}{0{,}9} = 94065 \text{ грн.},$$

де η – коефіцієнт, що характеризує етап виконання науково-дослідної роботи. Оскільки, якщо науково-технічна розробка знаходиться на стадії впровадження, то η=0,9.

5.3 Розрахунок економічної ефективності науково-технічної розробки інформаційної технології вирішення задачі нейромережевої класифікації ірисів за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

В даному випадку відбувається розробка засобу, тому основу майбутнього економічного ефекту буде формувати: ΔN – збільшення кількості споживачів, яким надається відповідна інформаційна послуга в аналізовані періоди часу; N – кількість споживачів, яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки; $Ц_б$ – вартість послуги у році до впровадження інформаційної системи; $\pm\Delta Ц_о$ – зміна вартості послуги (зростання чи

зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Можливе збільшення чистого прибутку у потенційного інвестора ΔПі для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П = (\pm \Delta Ц_о \cdot N + Ц_о \cdot \Delta N_i)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right), \quad (5.5)$$

де ±ΔЦ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай, таким показником може бути зміна ціни реалізації одиниці нової розробки в аналізованому році (відносно року до впровадження цієї розробки); $\pm\Delta Ц_о$ може мати як додатне, так і від'ємне значення (від'ємне – при зниженні ціни відносно року до впровадження цієї розробки, додатне – при зростанні ціни); N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки; $Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році; $Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів; ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки); λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість становить 20%, а коефіцієнт λ =0,8333; ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги).

Рекомендується брати ρ =0,2...0,5; ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році ϑ =18%.

Очікуваний термін життєвого циклу розробки 1 рік, тому:

$$\Delta П = ((1700 - 1300) \cdot 1000 - (200 - 700) \cdot 1000) \cdot 0{,}8333 \cdot 0{,}3 \cdot \left(1 - \frac{18}{100}\right) = 216480 \text{ грн.}$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} = \frac{216480}{(1+0{,}1)^1} = 196800 \text{ грн.},$$

де ΔП – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн.; Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки (приймаємо Т=1 рік); τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15; t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} \cdot ЗВ = 2 \cdot 94065 = 188130 \text{ грн.}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу,

маркетингові заходи тощо; зазвичай кінв=2...5, але може бути і більшим; ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

Тоді абсолютний економічний ефект $Е_{абс}$ або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$Е_{абс} = ПП - PV = 196800 - 188130 = 8670 \text{ грн.},$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн.; PV – теперішня вартість початкових інвестицій, грн.

Оскільки $Е_{абс}>0$, то можемо припустити про потенційну зацікавленість інвесторів у розробці.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність Ев або показник внутрішньої норми дохідності вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Внутрішня економічна дохідність інвестицій Ев, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Еабс}{PV}} - 1 = \sqrt[1]{1 + \frac{8670}{188130}} - 1 = 1{,}02,$$

де $Т_ж$ – життєвий цикл розробки, роки.

Далі розраховуємо період окупності інвестицій $Т_о$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію

науково-технічної розробки інформаційної технології вирішення задачі нейромережевої класифікації ірисів:

$$T_о = \frac{1}{E_в} = \frac{1}{1{,}02} = 0{,}98 \text{ року.}$$

Оскільки $T_о$<1…3-х років, то це свідчить про комерційну привабливість науково-технічної розробки інформаційної технології вирішення задачі нейромережевої класифікації ірисів і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

5.4 Висновок до розділу 5

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія вирішення задачі нейромережевої класифікації ірисів» становить 31,0 бал, що вказує на комерційну вагомість проведення даних досліджень (рівень комерційного потенціалу розробки середній). Термін окупності становить 0,98 р., що менше 3-х років, що говорить про комерційну привабливість науково-технічної розробки і може спонукати потенційних інвесторів на фінансування впровадження цієї розробки та виведення її на ринок. Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія вирішення задачі нейромережевої класифікації ірисів».

ВИСНОВКИ

У роботі було розглянуто задачу класифікації ірисів на основі нейронної мережі багатошаровий персептрон. В ході аналізу предметної області було описано детальну постановку задачі класифікації ірисів. Також розглянуто різні методи розв'язання задачі класифікації і оцінено їх застосовність до завдання класифікації ірисів. Серед таких методів машинного навчання як метод логістичної регресії, наївний Байєс, K-найближчих сусідів, дерева рішень, машини опорних векторів та нейромережева класифікація як найбільш перспективний і застосовний до даної задачі було обрано нейромережевий метод. Крім цього, було обгрунтовано вибір програми-аналогу до розроблюваної програми та на основі її недоліків сформульовано мету роботи – підвищення достовірності класифікації ірисів.

Було обґрунтовано вибір типу нейронної мережі багатошаровий персептрон для інформаційної технології класифікації ірисів. Обрано архітектуру багатошарового персептрона, який має 4 входи, 2 прихованих шари по 10 нейронів та вихідний шар із 3 нейронів. Було обрано функцію активації ReLu у двох прихованих шарах та функцію активації Softmax у вихідному шарі. Розроблено структуру процесів обробки інформації технології класифікації ірисів. Розроблено алгоритм роботи програми класифікації ірисів та UML діаграму класів.

Також було обґрунтовано вибір мови програмування Python та спеціалізованих бібліотек Keras, NumPy та Pandas для програмної реалізації інформаційної технології нейромережевої класифікації ірисів. Проведено аналіз набору даних ірисів для навчання та тестування програми. Описано основні етапи програмної реалізація та функціонування інформаційної технології нейромережевої класифікації ірисів, що складається з завантаження бібліотек, завантаження набору даних із параметрами ірисів та

їх ідентифікатором класу, аналізу цих параметрів, створення, компіляції, тренування, оцінки точності моделі нейромережі та виведення результатів.

У результаті тестування програми було доведено її повну працездатність та відповідність поставленому завданню. Розроблена програма має достовірність класифікації ірисів на тестовій вибірці 96,7%, а найкращий із 6-х методів, модельованих програмою-аналогом, має достовірність класифікації на тестовій вибірці 92,6%. Таким чином, розроблена програмна реалізація інформаційної технології нейромережевої класифікації ірисів має порівняно з методами-аналогами збільшену на 4,1% достовірність класифікації ірисів. Тобто мета роботи досягнута – достовірність класифікації ірисів підвищена.

У економічному розділі було визначено рівень комерційного потенціалу розробки за темою «Інформаційна технологія вирішення задачі нейромережевої класифікації ірисів» становить 31,0 бал, що вказує на комерційну вагомість проведення даних досліджень (рівень комерційного потенціалу розробки середній). Термін окупності становить 0,98 р., що менше 3-х років, що говорить про комерційну привабливість науково-технічної розробки і може спонукати потенційних інвесторів на фінансування впровадження цієї розробки та виведення її на ринок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. В. А. Гладченко, О. К. Колесницький Інформаційна технологія вирішення задачі нейромережевої класифікації ірисів / Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)», Вінниця, 2024, [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/22767/18770

2. The iris data set [Електронний ресурс] – Режим доступу: http://archive.ics.uci.edu/ml

3. Іванчук Я. В., Месюра В. І., Яровий А. А., Манжілевський О. Д. Інтелектуальний аналіз даних та машинне навчання. Частина 1. Базові методи та засоби аналізу даних : навчальний посібник, Вінниця : ВНТУ, 2021. – 69 с., https://iq.vntu.edu.ua/card.php?id=6030

4. Machine Learning with PyTorch and Scikit-Learn . Book Code Repository. Paperback: 770 pages. Publisher: Packt Publishing. Language: English. ISBN-10: 1801819319 I.SBN-13: 978-1801819312

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

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

7. Субботін С. О. Нейронні мережі : теорія та практика: навч. посіб. / С. О. Субботін. – Житомир : Вид. О. О. Євенок, 2020. – 184 с. ISBN 978-966-995-189-2

8. М.А. Новотарський, Б.Б. Нестеренко. Штучні нейронні мережі: обчислення // Праці Інституту математики НАН України. – Т50. – Київ: Ін-т математики НАН України, 2004. – 408 с.

9. Iris-Flower-Classification-Project [Електронний ресурс] – Режим доступу: https://github.com/Swarupa567/Iris-Flower-Classification-Project

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

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

12. Pasko, R., & Terenchuk, S., (2020). The Use of Neuro-Fuzzy Models in Expert Support Systems for Forensic Building Technical Expertise. ScienceRise, 2(67), 10 – 18, doi: http://doi.org/10.21303/2313-8416.2020.001278/

13. Кононова К. Ю. Машинне навчання: методи та моделі: підручник для бакалаврів, магістрів та докторів філософії спеціальності 051 «Економіка» / К. Ю. Кононова. – Харків: ХНУ імені В. Н. Каразіна, 2020. – 301 с.

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

15. Python [Електронний ресурс] – режим доступу: https://uk.wikipedia.org/wiki/Python

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

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

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

19. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

20. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

21. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальності 122 «Комп’ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. К. Колесницький. – Вінниця : ВНТУ, 2023. – (58 с.)

Додаток А (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія вирішення задачі нейромережевої класифікації ірисів

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

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

Показники звіту подібності Turnitin

Оригінальність 82,00% Схожість 18,00%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

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

Автор роботи _____ Гладченко В. А.

Керівник роботи _____ Колесницький О.К.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку _____ Озеранський В.С.

Додаток Б (обов'язковий)

Лістинг програми

```
# Importing libraries

from keras.models import Sequential

from keras.layers import Dense

from keras.utils import np_utils

import numpy

import pandas as pd


# fix random seed for reproducibility

numpy.random.seed(7)


# Defining column names for datasets

COLUMN_NAMES = [

        'SepalLength',

        'SepalWidth',

        'PetalLength',

        'PetalWidth',

        'Species'

        ]


# Import training dataset

training_dataset = pd.read_csv('iris_training.csv', names=COLUMN_NAMES,
header=0)

train_x = training_dataset.iloc[:, 0:4].values

train_y = training_dataset.iloc[:, 4].values


# Encoding training dataset

encoding_train_y = np_utils.to_categorical(train_y)


# Import testing dataset
```

```
test_dataset = pd.read_csv('iris_test.csv', names=COLUMN_NAMES, header=0)

test_x = test_dataset.iloc[:, 0:4].values

test_y = test_dataset.iloc[:, 4].values


# Encoding training dataset

encoding_test_y = np_utils.to_categorical(test_y)


# Creating a model

model = Sequential()

model.add(Dense(10, input_dim=4, activation='relu'))

model.add(Dense(10, activation='relu'))

model.add(Dense(3, activation='softmax'))


# Compiling model

model.compile(loss='categorical_crossentropy', optimizer='backprop',
metrics=['accuracy'])


# Training a model

model.fit(train_x, encoding_train_y, epochs=300, batch_size=10)


# evaluate the model

scores = model.evaluate(test_x, encoding_test_y)

print("\nAccuracy: %.2f%%" % (scores[1]*100))
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ВИРІШЕННЯ ЗАДАЧІ
НЕЙРОМЕРЕЖЕВОЇ КЛАСИФІКАЦІЇ ІРИСІВ

Виконав: студент 2-го курсу,
групи 1КН-23м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Гладченко В. А.
(прізвище та ініціали)

Керівник: к.т.н., проф. каф. КН
_____ Колесницький О.К.
(прізвище та ініціали)
«05» 12 2024 р.

Вінниця ВНТУ - 2024 рік

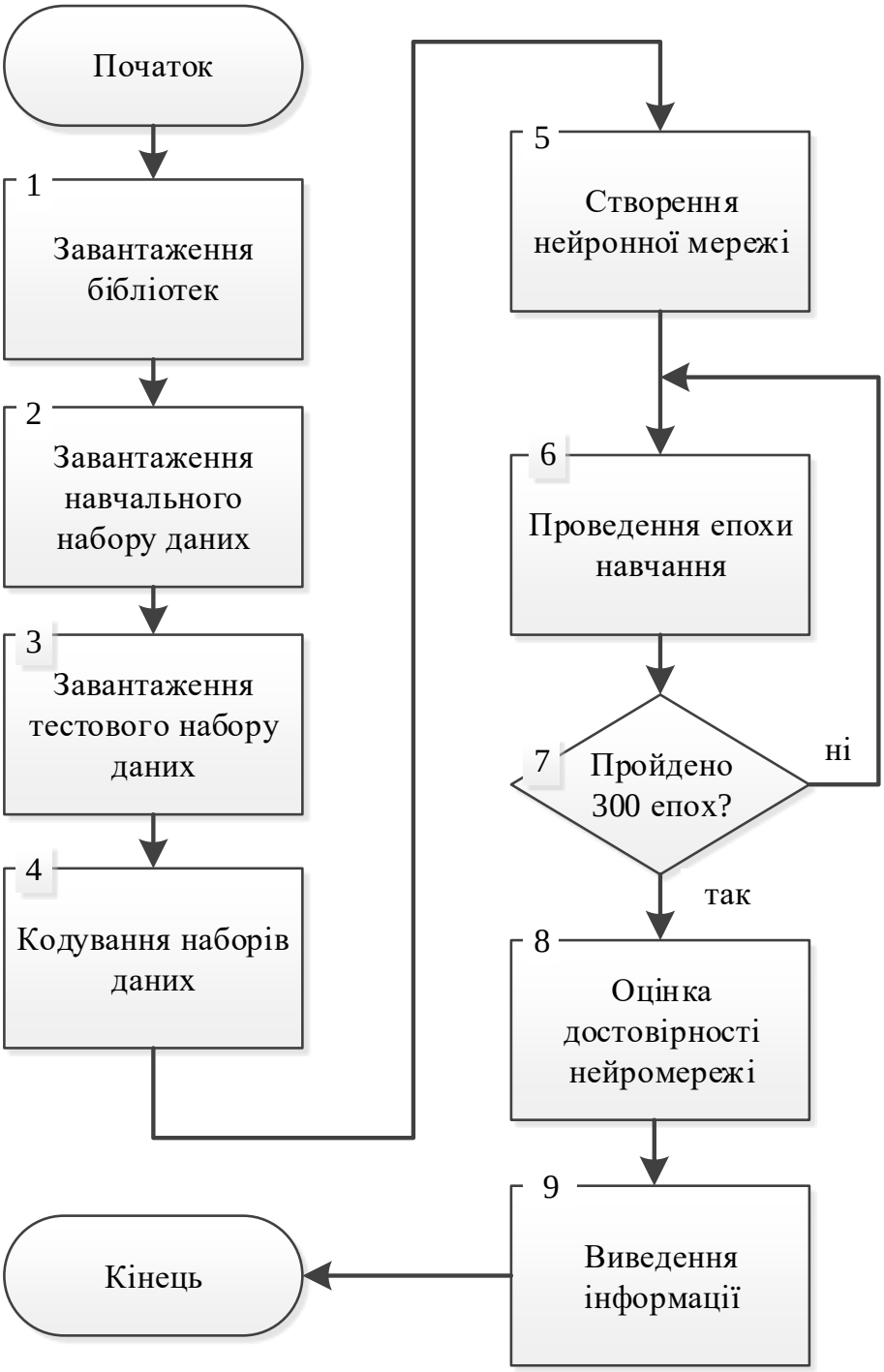

Рисунок В.1 – Схема алгоритму роботи програми вирішення задачі нейромережевої класифікації ірисів

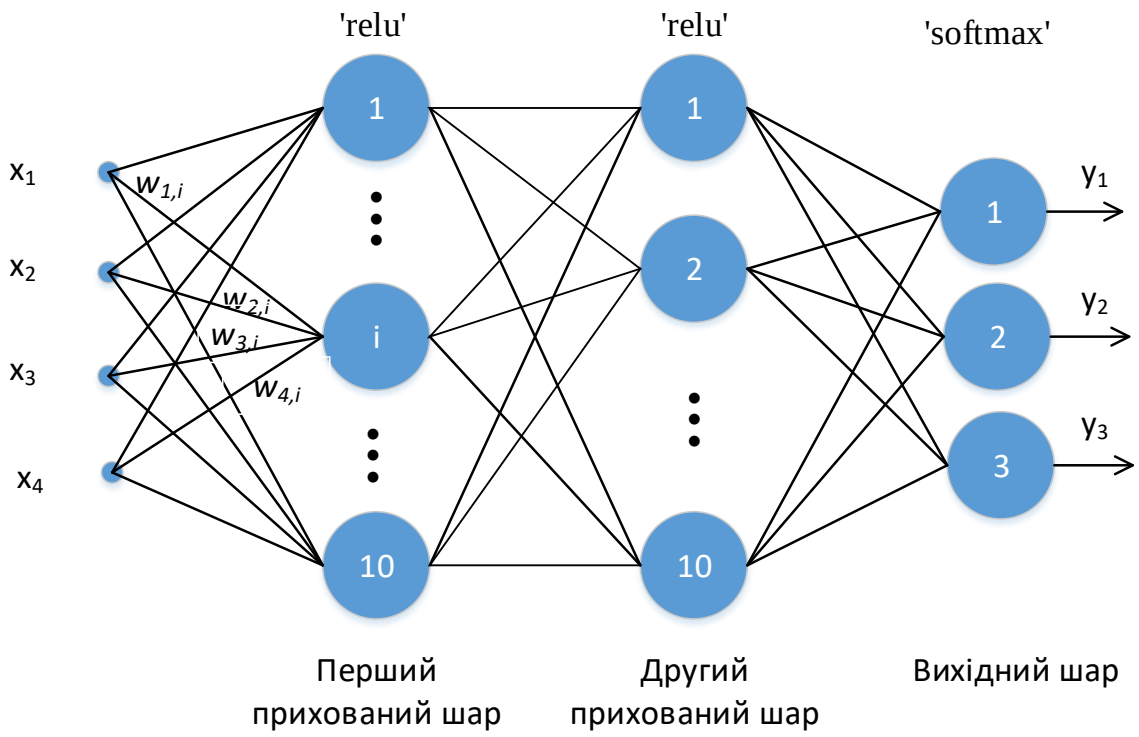

Рисунок В.2 – Структура багатошарового персептрона для інформаційної технології класифікації ірисів

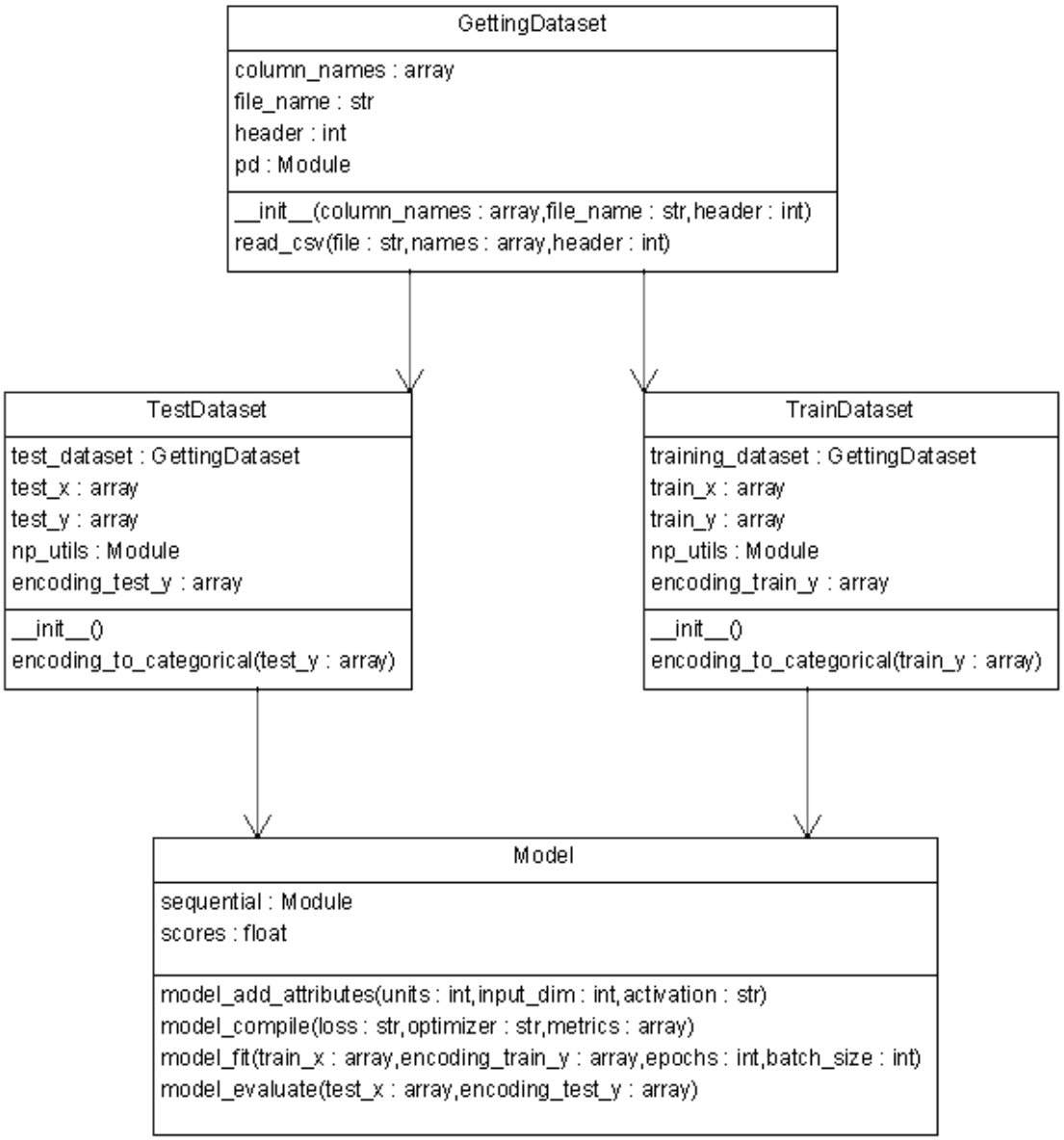

Рисунок В.3 – UML-діаграма класів програми нейромережевої класифікації ірисів

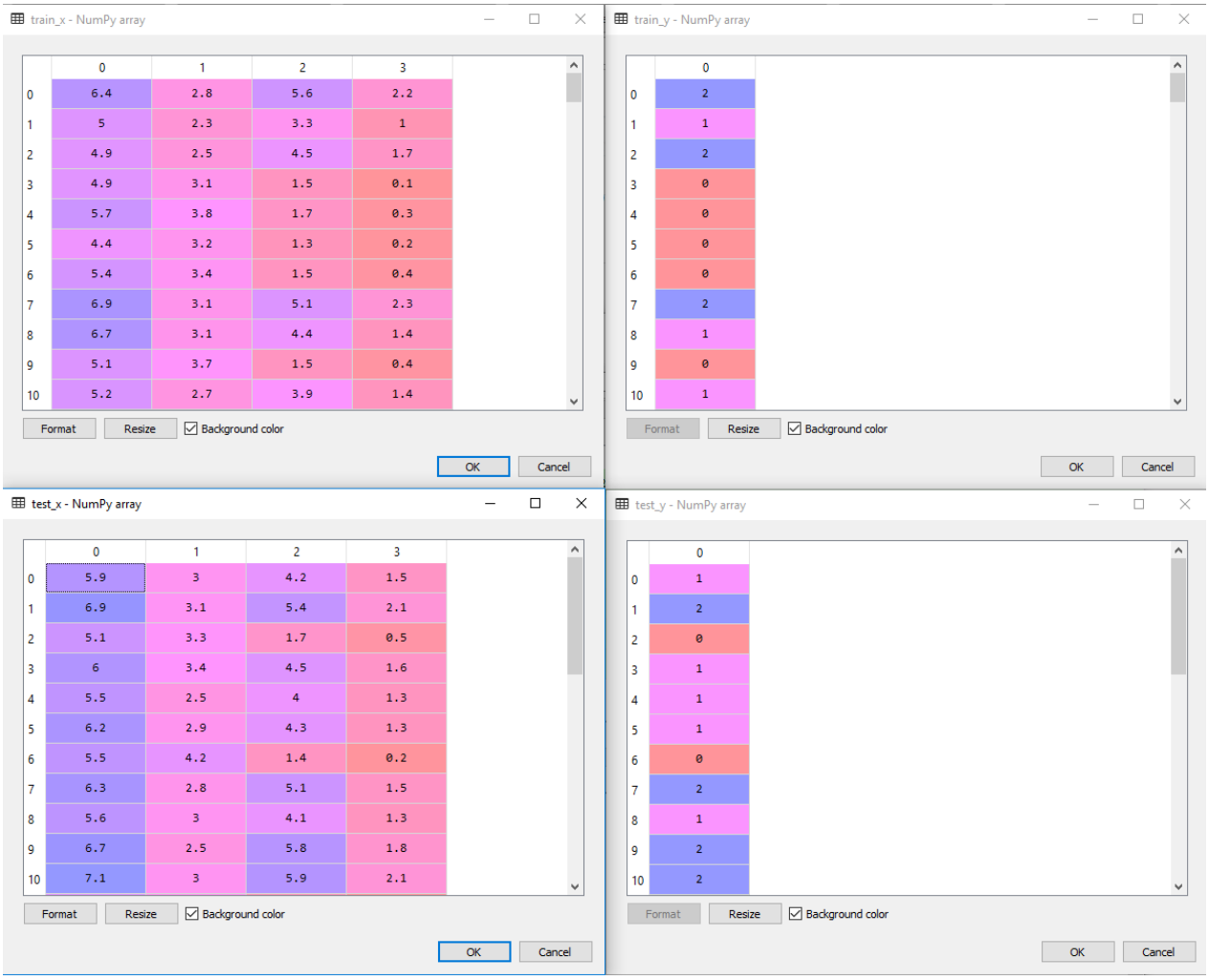

Рисунок В.4 – Чотири матриці із навчальним та тестовим наборами даних

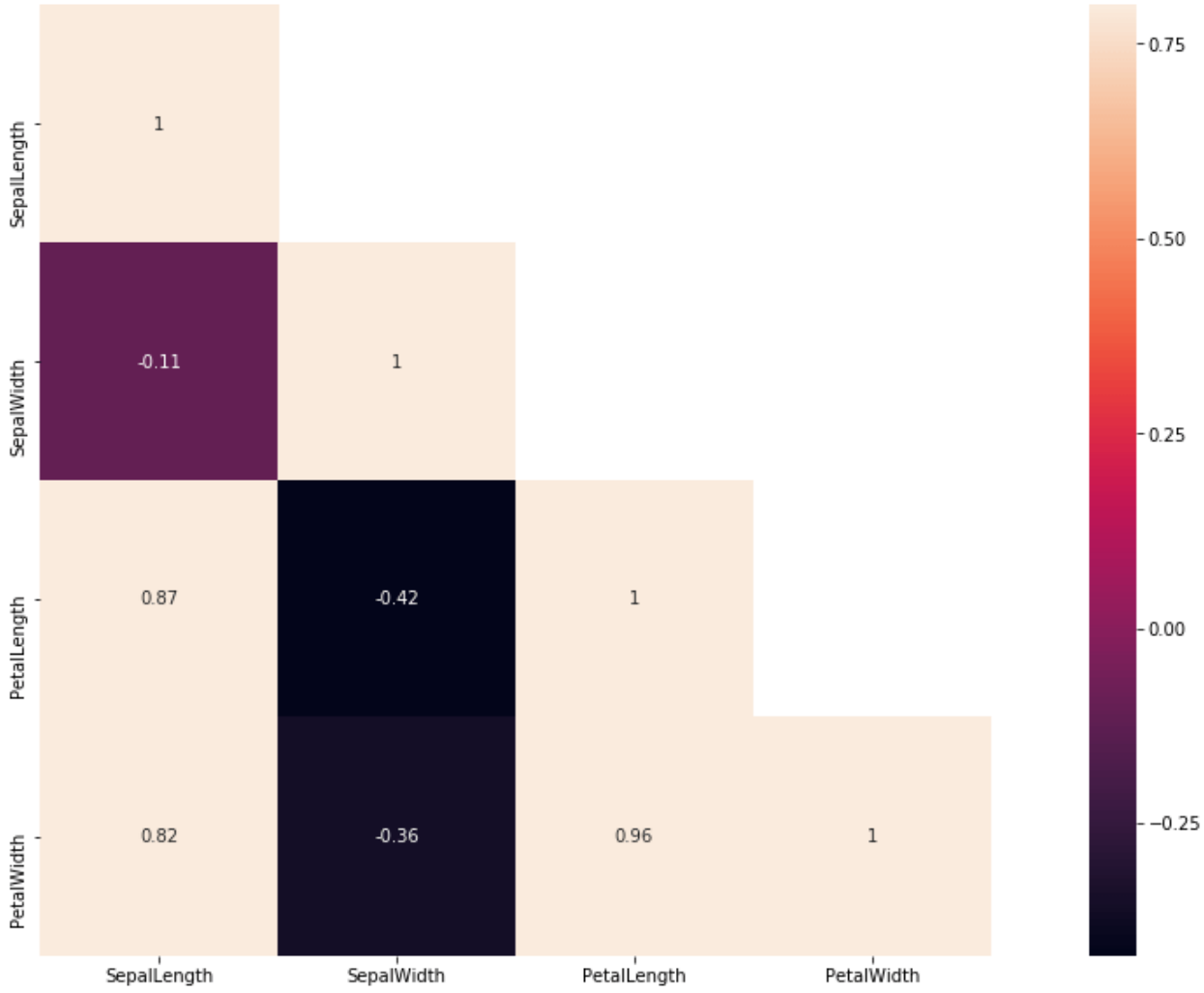

Рисунок В.5 – Кореляційна матриця (матриця з рівнями залежності між ознаками квіток ірисів)

```
Epoch 1/300
12/12 [==============================] - 1s 2ms/step - loss: 2.7296 - accuracy: 0.3000
Epoch 2/300
12/12 [==============================] - 0s 2ms/step - loss: 1.9793 - accuracy: 0.3000
Epoch 3/300
12/12 [==============================] - 0s 1ms/step - loss: 1.5587 - accuracy: 0.3750
Epoch 4/300
12/12 [==============================] - 0s 932us/step - loss: 1.4221 - accuracy: 0.3583
…

Epoch 148/300
12/12 [==============================] - 0s 914us/step - loss: 0.1110 - accuracy: 0.9750
Epoch 149/300
12/12 [==============================] - 0s 918us/step - loss: 0.1110 - accuracy: 0.9750
Epoch 150/300
12/12 [==============================] - 0s 913us/step - loss: 0.1137 - accuracy: 0.9750
Epoch 151/300
12/12 [==============================] - 0s 913us/step - loss: 0.1080 - accuracy: 0.9667
…

Epoch 297/300
12/12 [==============================] - 0s 1ms/step - loss: 0.0725 - accuracy: 0.9750
Epoch 298/300
12/12 [==============================] - 0s 0s/step - loss: 0.0738 - accuracy: 0.9750
Epoch 299/300
12/12 [==============================] - 0s 2ms/step - loss: 0.0740 - accuracy: 0.9833
Epoch 300/300
12/12 [==============================] - 0s 1ms/step - loss: 0.0720 - accuracy: 0.9750
1/1 [==============================] - 0s 123ms/step - loss: 0.0589 - accuracy: 0.9667

Accuracy: 96.67%
```

Рисунок В.6 – Приклад результату роботи програми