Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

Пояснювальна записка

до магістерської кваліфікаційної роботи
магістр
(ступінь вищої освіти)

на тему: «Розробка нейромережевих методів і моделей прогнозування ефективності використання інвестицій»

Студента 2 курсу, групи 1ПІ-18м (д/ф)
Спеціальності
121 «Інженерія програмного забезпечення»
Задорожного В. М.
(прізвище та ініціали)
Керівник к.т.н., доц. Ракитянська Г. Б.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент к.т.н., доц. Крилик Л. В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

ВНТУ– 2019

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Ступінь вищої освіти – магістр
Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
"____"_____20___року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ
СТУДЕНТУ

Задорожному Віталію Миколайовичу

1. Тема роботи: «Розробка нейромережевих методів і моделей прогнозування ефективності використання інвестицій»
керівник роботи: к.т.н., доцент кафедри ПЗ Ракитянська Г. Б. затверджені наказом вищого навчального закладу від "___"_____20__року №__

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи:
Мова програмування: Python;
Нейромережева бібліотека: Keras;
Вид нейромережі: радіально-базисна мережа;
Метод навчання: глибинне навчання.

4.Зміст розрахунково-пояснювальної записки: вступ; аналіз стану питання та постановка задачі; розробка методів та моделей нейронної мережі; програмна реалізація нейронної мережі; тестування розробленої нейронної мережі; економічна частина; висновки; перелік посилань; додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): мета роботи, об'єкт та предмет дослідження; основні задачі дослідження; наукова новизна, практичне значення; моделі нейронної мережі; приклад вигляду розробленої програми; висновки.

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видано	завдання прийнято
1–4	к.т.н. Ракитянська Г. Б., доцент кафедри ПЗ		
5	к.е.н. Бальзан М. В., доцент кафедри ЕПВМ		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Техніко-економічне обґрунтування доцільності розробки нейронної мережі прогнозування	30.09.2019 – 13.10.2019	Вик.
2	Розробка методів та моделей нейронної мережі прогнозування	14.10.2019 – 27.10.2019	Вик.
3	Програмна реалізація нейронної мережі	27.10.2019 – 17.11.2019	Вик.
4	Тестування роботи нейронної мережі	18.11.2019 – 24.11.2019	Вик.
5	Економічне обґрунтування розробки програмного продукту	25.11.2019 – 01.12.2019	Вик.
6	Оформлення матеріалів до захисту МКР	2.12.2019 – 07.12.2019	Вик.

Студент _____ **Задорожний В. М.**

(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Ракитянська Г. Б.**

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У магістерській кваліфікаційній роботі проведено детальний аналіз процесу прогнозування ефективності використання інвестицій.

Запропоновано метод навчання при побудові нейромережі для засобів прогнозування, який характеризується динамічним збільшенням шарової архітектури мережі та корегуванням значенння вагів впливу кожного шару в процесі навчання, що дозволяє підвищити точність розрахунків.

Запропоновано метод прогнозування ефективності використання інвестицій, який відрізняється від існуючих використанням діапазону дискретних точок оцінювання інвестицій з урахуванням умов зміни інвестиційного процесу, що дозволяє врахувати динамічні критерії впливу на ефективність використання інвестицій.

Для реалізації нейронних мереж та обрано мову програмування Python з використанням бібліотеки Keras. Обрано радіально-базисну нейромережу з глибинним навчанням.

Розроблена нейронна мережа проводить прогнозування ефективності використання інвестицій, яка орієнтована на прогнозування інвестування зі зменшенням ризиків, що дозволяє користувачеві слідкувати за можливим станом інвестицій до моменту залучення коштів.

ABSTRACT

In the master's qualification work, a detailed analysis of the process of forecasting the efficiency of investment utilization was carried out.

The method of training in the construction of a neural network for forecasting tools is proposed, which differs from the existing dynamic increase in the network layer architecture and adjusts the values of the impact weights of each layer in the learning process, which improves the accuracy of calculations.

The method of forecasting the efficiency of investment utilization is proposed, which differs from the existing ones using the range of discrete points of investment valuation taking into account the conditions of change of the investment process, which allows to take into account dynamic criteria of influence on the efficiency of investment utilization.

Python programming language using Keras library is selected for neural networks implementation. Radial-base neural network with deep learning is selected.

The developed neural network conducts investment forecasting, which is focused on forecasting investment, with reduced risk, which allows the user to monitor the possible state of investment until the moment of raising funds.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. В епоху інформаційних технологій динамічне зростання обсягу даних викликало стрімкий розвиток наукового напрямку штучний інтелект. Ведуться роботи зі створення машини, яка зможе емулювати нейронну структуру, подібну мозку, і мати можливість формувати уявлення про знання, зберігати і обробляти отримані дані, а також самонавчатися. Але виникає проблема в тому, що немає повного уявлення механізму біологічного інтелекту, тому здійснити перехід від природнього інтелекту до штучного не вдалося. Проте під інтелектом в межах цієї науки розуміється тільки обчислювальна складова, котра забезпечує автоматизацію і швидкість обробки даних в релевантну інформацію.

Існує багато напрямків науки штучного інтелекту і найбільш обширними є машинне навчання та інтелектуальний аналіз даних (Data Mining). У рамках цих наук вирішується доволі багато завдань, а саме таких як регресія, класифікація, кластерування, прогнозування та інші. Найбільш ефективні методи реалізовані на основі штучних нейронних мереж [1].

Задача прогнозування, мабуть, може вважатися однією з найбільш складних задач інтелектуального аналізу даних, вона вимагає ретельного дослідження вихідного набору даних і методів, придатних для аналізу.

Задачі прогнозування вирішуються в найрізноманітніших галузях людської діяльності, таких як наука, економіка, виробництво і безліч інших сфер. Прогнозування є важливим елементом організації управління як окремими господарюючими суб'єктами, так і економікою в цілому.

Розвиток методів прогнозування безпосередньо пов'язаний з розвитком інформаційних технологій, зокрема, із зростанням обсягів даних, що зберігаються і ускладненням методів і алгоритмів прогнозування.

Задача прогнозування фінансових часових рядів була і залишається актуальною, оскільки передбачення є необхідним елементом будь-якої інвестиційної діяльності, адже сама ідея інвестування – вкладення грошей з

метою отримання доходу в майбутньому – грунтується на ідеї прогнозування майбутнього.

Мета та завдання дослідження. Метою дослідження є підвищення продуктивності методів прогнозування в системах оцінки інвестиційної привабливості за рахунок використання багатошарових радіально-базисних нейронних мереж.

Об'єктом дослідження є процес оцінювання інвестиційної привабливості з використанням методів прогнозування.

Предметом дослідження є методи та засоби програмної реалізації процесу прогнозування інвестиційної привабливості.

Для досягнення поставленої мети слід розв'язати такі **задачі**:

- аналіз особливостей створення та навчання нейронних мереж;
- розробка моделі та методу побудови нейронної мережі прогнозування;
- розробка методу прогнозування ефективності використання інвестицій;
- аналіз програмних засобів для створення нейронної мережі;
- програмна реалізація розроблених методів для нейронної мережі;
- навчання розробленої нейронної мережі;
- тестування розробленого програмного продукту.

Методи дослідження:

- теорія нейронних мереж для створення методів оцінки ефективності використання інвестицій;
- методи навчання нейронних мереж для розробки методу навчання нейромережі прогнозування;
- методи проектування програмного продукту для розробки системи прогнозування ефективності використання інвестицій;
- комп'ютерне моделювання для аналізу і перевірки достовірності отриманих теоретичних результатів.

Наукова новизна одержаних результатів:

1. Дістав подальшого розвитку метод побудови нейромережі для засобів прогнозування, який відрізняється від існуючих динамічним збільшенням

шарової архітектури мережі та корегуванням значенння вагів впливу кожного шару в процесі навчання, що дозволяє підвищити точність розрахунків.

2. Дістав подальшого розвитку метод прогнозування ефективності використання інвестицій, який відрізняється від існуючих використанням діапазону дискретних точок оцінювання інвестицій з урахуванням умов зміни інвестиційного процесу, що дозволяє врахувати динамічні критерії впливу на ефективність використання інвестицій.

Практичне значення одержаних результатів. Практична цінність одержаних результатів полягає в тому, що на основі проведених теоретичних досліджень і отриманих наукових результатів розроблено нейрону мережу прогнозування ефективності використання інвестицій, яка орієнтована на прогнозування інвестування зі зменшенням ризиків, що дозволяє користувачеві слідкувати за можливим станом інвестицій до моменту залучення коштів.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася відповідно до плану науково-дослідних робіт кафедри програмного забезпечення.

Особистий внесок здобувача. В публікації [2] автором розроблено метод побудови вхідних шарів та навчання нейромережі прогнозування. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто.

Достовірність теоретичних положень підтверджена результатами тестування розробленої нейронної мережі.

Апробація матеріалів магістерської кваліфікаційної роботи. Результати роботи доповідалися на науково-технічній конференції – на Всеукраїнській науково-практичній Інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи - 2019».

Публікації. Результати роботи опубліковані в науковій публікації: «Розробка нейромережевих методів прогнозування ефективності використання інвестицій» [2].

Структура та обсяг роботи. Магістерська кваліфікаційна робота

складається зі вступу, п’яти розділів, висновків, списку літератури, що містить 30 найменувань, 3 додатків. Робота містить 29 ілюстрацій, 4 таблиці.

Робота складається з 5 розділів. У першому розділі проаналізовано сучасний стан питання теми розробки, проведено порівняння аналогів і обґрунтовано вибір вибір типу нейромережі для розробки, сформульовано задачі дослідження.

Другий розділ містить аналіз методів навчання, розробку моделі та методу навчання нейронної мережі прогнозування та розробку методу та моделі прогнозування використання інвестицій на базі нейронних мереж.

У третьому розділі проведено варіантний аналіз засобів реалізації програмного продукту, розроблено нейронну мережу прогнозування та метод прийняття рішень і проведено навчання нейронної мережі.

Четвертий розділ показує результати тестування розробленої нейронної мережі прогнозування.

У п’ятому розділі розраховуються витрати на розробку програмного забезпечення, експлуатаційні витрати, обсяг роботи, пов'язаної з використанням програмного забезпечення, та економічний ефект від впровадження нового програмного продукту.

У висновках наведені основні результати дослідження.

У додатках міститься технічне завдання, лістинг коду основних частин програми та ілюстративний матеріал до захисту магістерської кваліфікаційної роботи.

1 АНАЛІЗ МОЖЛИВОСТЕЙ ВИКОРИСТАННЯ НЕЙРОННИХ МЕРЕЖ В ЗАДАЧАХ ПРОГНОЗУВАННЯ

1.1 Аналіз особливостей використання нейромереж

Концепція обробки великої кількості даних, їх аналітика, обробка, пошук інформації по ним розвивається протягом багатьох років. Більшість організацій в даний час розуміють, що якщо вони будуть збирати масово інформацію в сфері свого бізнесу, вони можуть застосувати аналітику і отримати значну віддачу від нього. Але навіть в 1950-і роки, перш ніж хтось винайшов такий термін як «великі дані», підприємства вже використовували аналітику здебільшого це були таблиці, обстеженні вручну, щоб розкрити ідеї і тенденції [3].

Нові переваги, які приносить аналіз великих даних – це швидкість і ефективність. У той час, як кілька років тому бізнес збирав би інформацію, запускав аналітику і розкопував інформацію, яка могла бути використана для прийняття майбутніх рішень, сьогодні бізнес може ідентифікувати інформацію для швидкої обробки. Здатність працювати швидше і залишатися гнучкими – надає організаціям конкурентну перевагу, що у них не було раніше. Високоефективна аналітика даних дозволяє робити речі, недоступні раніше, оскільки обсяги даних були занадто великими. Наприклад, можливість отримати своєчасні ідеї для прийняття рішень при швидкоплинних можливостях та отримати точні відповіді на важко вирішувані проблеми і відкрити нові можливості для зростання ефективності, також це дає можливість для використання ресурсів та часу більш ефективно. Поява величезної кількості мобільних телефонів, планшетних ПК, фліп-комп'ютерів, а також зростання хмарних обчислень, поряд з іншими джерелами даних, таких як датчики і Інтернет, прискорили великі маси даних, ніж хто-небудь колись бачив. Тільки за останні два роки, більше даних були створено, ніж у всій історії. Обробка, аналіз та пошук великого масиву даних, як правило, відносяться до будь-якого великій кількості необроблених даних, які можуть бути зібрані, збережені і аналізовані за допомогою різних засобів, що розкривають закономірність або тенденцію, що

стосуються поведінки, особливо у споживачів, які можуть бути використані, щоб максимізувати потенціал власного бізнесу, також слід більше зосередити увагу на ефективному використанні великих обсягів даних за рахунок використання візуалізації даних, тому як візуалізація сприяє кращому сприйняттю даних.

Візуалізація даних є ключем для аналітиків для розуміння трендів ринку та аналізу даних. Збільшуючи використання візуалізацій, бізнес знайшов цінність, яку він шукав. Створення більш інфографіки, і візуальних елементів, дозволив їм отримати більше інформації та швидше. Це дозволяє розуміти та сприймати більшу частину інформації, яка була очевидна і раніше. Візуалізація даних дозволяє збільшити функціональність відділу аналітиків, щоб вони могли розуміти краще тренди їх систем, візуалізація дозволяє створити зв'язок між точками даних, які, здавалося б, не мають будь-яких посилань на всіх, та дає змогу знайти хороші і погані дані, і при цьому аналітики можуть збільшити свою продуктивність, а також вартість інформації, яку вони зібрали [4].

1.2 Опис штучної нейронної мережі та її складових

Наш мозок являє собою складну біологічну нейронну мережу, яка приймає інформацію від органів почуттів і якимось чином її обробляє (впізнавання осіб, виникнення відчуттів і т.д.). Наш мозок, як і будь-яка біологічна нейронна мережа, складається із сукупності нейронів.

Біологічний нейрон – надзвичайно складна система. Багато в чому це пояснюється тим, що нейрон, крім обробки сигналу (основне його призначення), змушений ще виконувати купу інших функцій, що підтримують його життя. Більш того, сам механізм передачі сигналу від нейрона до нейрона теж дуже складний з біологічної та хімічної точки зору [5].

Штучний нейрон це структура, яка приймає сигнал, перетворює його (приблизно так, як це роблять справжні нейрони), і передає іншим нейронам (які роблять те ж саме).

Біологічні нейронні мережі являють собою сукупність біологічних нейронів. Однак в таких мережах теж багато непотрібних для обробки сигналу аспектів. Плюс до всього нейронів в біологічної нейромережі дуже багато.

Штучні нейронні мережі являють собою систему з'єднаних між собою простих обробників (штучних нейронів), які взаємодіють. Такі обробники зазвичай є доволі простими. Кожен обробник подібної мережі має справу лише з сигналами, які він періодично отримує, і сигналами, які він періодично надсилає іншим обробникам, такі локально прості обробники разом здатні виконувати доволі складні завдання [6]. Синапсами називають зв'язки, за якими вихідні сигнали одних нейронів надходять на вхід інших. Кожен зв'язок характеризується своєю вагою. Зв'язки з позитивним вагою називаються збудливими, а з негативним - гальмуючими. Вихід нейрона називається аксон. У ШНМ штучний нейрон - це деяка нелінійна функція, аргументом якої є лінійна комбінація всіх вхідних сигналів. Така функція називається функцією активації. Результат функції активації надсилається на вихід нейрона.

Функція активації нейрона характеризує залежність сигналу на виході нейрона від суми сигналів на його входах. Зазвичай функція є монотонно зростаючею і знаходиться в області значень [-1,1] (гіперболічний тангенс) і [0,1] (сигмоїда). Для деяких алгоритмів навчання необхідно, щоб функція активації була безперервно диференціючею по всій числовій осі. Штучний нейрон характеризується своєї активаційною функцією.

Основні функції активації [7]:

– порогова функція активації (функція Хевісайда), не використовується в алгоритмах зворотного поширення помилки (вираз (1.1)):

–

$$f(x) = \begin{cases} 1\, if\ \ x \geq -w_0 x_0 \\ 0\, else \end{cases}, \qquad (1.1)$$

де $-w_0 x_0$ - зсув функції активації відносно горизонтальної осі,

x - зважена суму сигналів на входах нейрона.

– сигмоідальна функція активації (1.2):

$$\sigma(x) = \frac{1}{1+e^{-x}}, \tag{1.2}$$

– гіперболічний тангенс (1.3):

$$\tanh(Ax) = \frac{e^{Ax} - e^{-Ax}}{e^{Ax} + e^{-Ax}}. \tag{1.3}$$

В основі перцептрона лежить математична модель сприйняття інформації мозком. Різні дослідники по-різному його визначають. У найзагальнішому своєму вигляді (як його описував Розенблатт) перцептрон представляє систему з елементів трьох різних типів: сенсорів, асоціативних елементів і реагуючих елементів. Архітектура нейронної мережі перцептрона наведена на рис. 1.1.

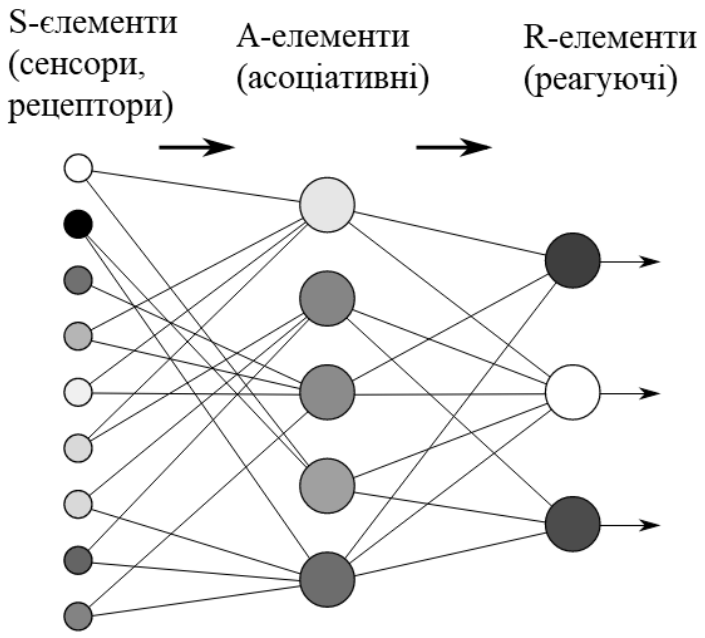

Рисунок 1.1 – Архітектура нейронної мережі перцептрон

Принцип роботи перцептрона:

Першими в роботу включаються *S*-елементи. Вони можуть перебувати в стані покою (сигнал дорівнює 0), або в стані збудження (сигнал дорівнює 1).

Далі сигнали від *S*-елементів передаються *A*-елементам по так званим *S*-*A* зв'язкам. Ці зв'язки можуть мати ваги, рівні тільки -1, 0 або 1.

Потім сигнали від сенсорних елементів, які пройшли по *S*-*A* зв'язкам потрапляють в *A*-елементи, які ще називають асоціативними елементами. Варто зауважити, що одному *A*-елементу може відповідати кілька *S*-елементів. Якщо сигнали, що надійшли на *A*-елемент, в сукупності перевищують деякий його поріг θ, то цей *A*-елемент збуджується і видає сигнал, рівний 1. В іншому випадку (сигнал від *S*-елементів не перевищив порогу *A*-елемента) генерується нульовий сигнал.

Персептрон (Perceptron) - найпростіший вид нейронних мереж. В основі лежить математична модель сприйняття інформації мозком, що складається з сенсорів, асоціативних і реагують елементів [8].

Одношаровий перцептрон (перцептрон Розенблатта) - одношарова нейронна мережа, всі нейрони якої мають жорстку порогову функцію активації.

Одношаровий перцептрон має простий алгоритм навчання і здатний вирішувати лише найпростіші задачі. Ця модель викликала до себе великий інтерес на початку 1960-х років і стала поштовхом до розвитку штучних нейронних мереж.

Класичний приклад такої нейронної мережі - одношаровий перцептрон наведен на рис. 1.2.

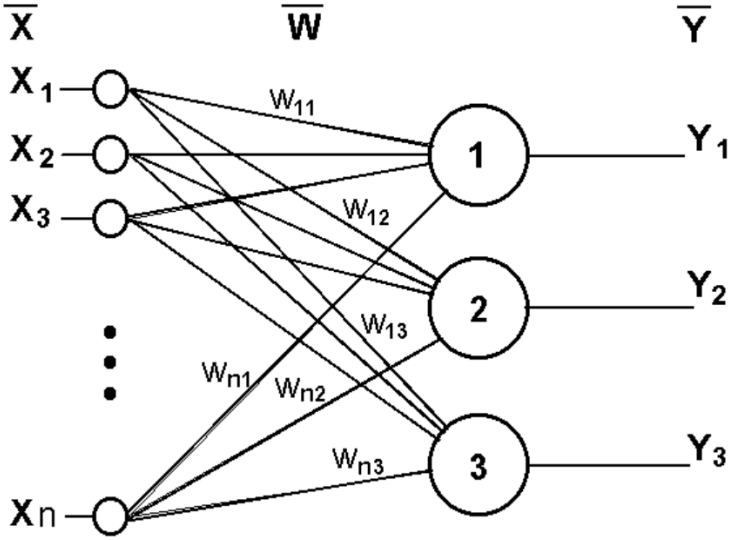

Рисунок 1.2 – Архітектура одношарового перцептрону

Мережа зображена на малюнку, має *n*-входів, на які надходять сигнали, що йдуть по синапсах на 3 нейрона. Ці три нейрона утворюють єдиний шар даної мережі і мають три вихдних сигнала.

Багатошаровий перцептрон (MLP) - нейронна мережа прямого поширення сигналу (без зворотних зв'язків), в якій вхідний сигнал перетвориться в вихідний, проходячи послідовно через кілька шарів. Перший з таких шарів називають вхідним, останній - вихідним. Ці шари містять так звані вироджені нейрони і іноді в кількості шарів не враховуються. Крім вхідного і вихідного шарів, в багатошаровому перцептроні є один або кілька проміжних шарів, які називають прихованими. У цій моделі перцептрона повинен бути хоча б один прихований шар. Присутність декількох таких шарів виправдано лише в разі використання нелінійних функцій активації. Приклад двошарового перцептрона наведен на рис. 1.3.

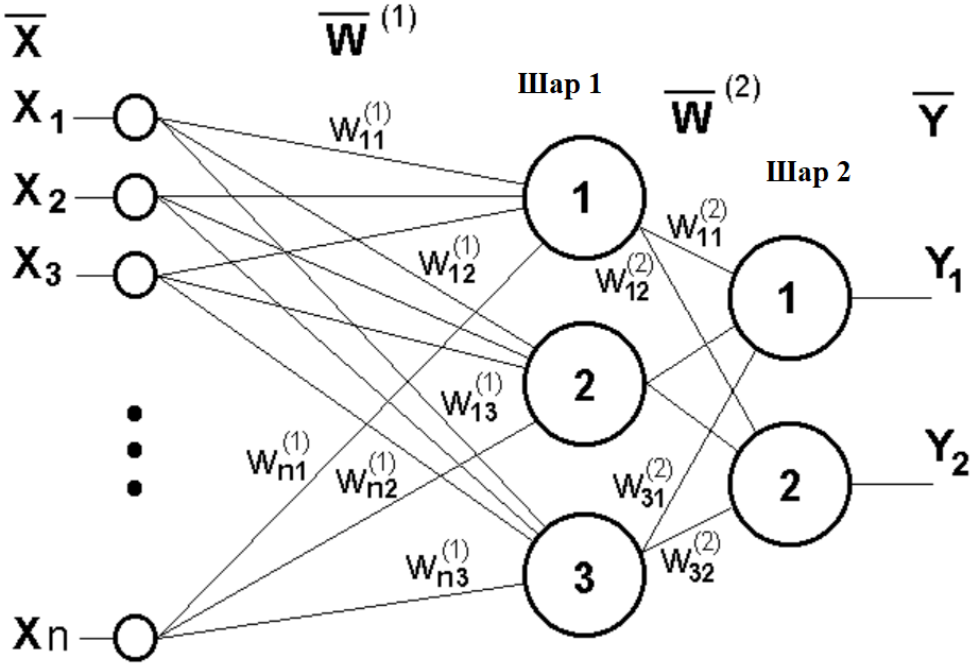

Рисунок 1.3 – Архітектура двошарового перцептрону

Мережа, зображена на малюнку, має *n* входів. На них надходять сигнали, що йдуть далі по синапсах на 3 нейрона, які утворюють перший шар. Вихідні сигнали першого шару передаються двом нейронам другого шару. Останні, в свою чергу, видають два вихідних сигнали.

Мережа радіально-базисних функцій у математичному моделюванні – це штучна нейронна мережа, яка використовує радіальні базисні функції у якості функції активації. Виходом мережі є лінійна комбінація радіальних базисних функцій входу та параметрів нейрона. Мережі радіальних базисних функцій мають багато застосувань, зокрема, такі як апроксимацію функції, прогнозування часових рядів, задачі класифікації та керування системою.

Мережі радіально базисних функцій (РБФ) зазвичай мають три шари: вхідний шар, прихований шар з нелінійною РБФ функцією активації та лінійний вихідний рівень. Базисні функції в першому шарі виробляють локалізовану реакцію на вхідний стимул. Вихідні вузли мережі формують зважену лінійну комбінацію з базисних функцій, обчислених вузлами першого шару.

Завдяки гнучким умовам на форму функції активації, РБФ мережі є універсальними апроксиматорами на компактному просторі. Це означає, що

мережа буде з достатньою кількістю прихованих нейронів може апроксимувати будь-яку неперервну функцію на замкненій обмеженій множині з довільною точністю.

Для реалізації краще обрати радіально-базисну нейронну мережу через, те що вона більше пристосована до прогнозування складних числових рядів [9].

1.3 Аналіз аналогів нейронних мереж прогнозування

Завдяки своїй можливості виявляти нелінійні математичні закономірності часових рядів, швидко адаптуватися до змін ринкових тенденцій, нейронні мережі є на даний момент одним з найбільш перспективних інструментів прогнозування фінансових часових рядів.

Зараз існує чимала кількість розроблених нейронних мереж прогнозування певних числових рядів. Однією з прикладів є програма Anfis.

Anfis – це програма прогнозування цін. Вона розроблена за допомогою MATLAB та має можливість підключення додаткових бібліотек даних [10]. Редактор складається з чотирьох панелей: для даних (Load data); для генерування мережі (Generate FIS); для тренування (Train FIS); для тестування (Test FIS).

Для навчання нейромереж в пакеті AnfisEdit передбачено два алгоритма навчання – зворотнього розповсюдження та гібридний. При гібридному способі навчання мережі проходить за два-три прохода даних, що дозволяє швидко отримати результат. Приклад роботи зображено на рис. 1.4.

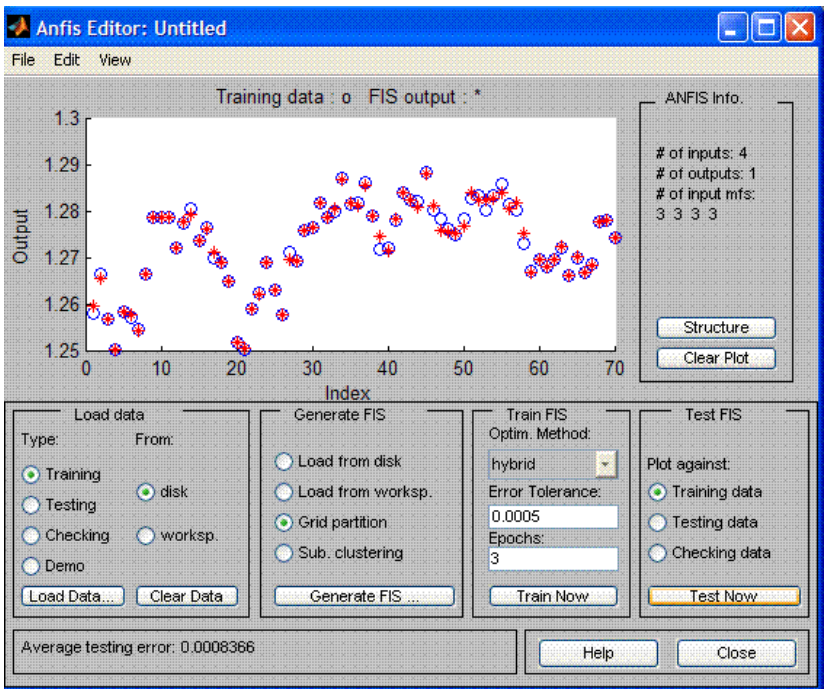

Рисунок 1.4 – Приклад роботи програми Anfis

Але дана нейронна мережа не має змоги прогнозувати довготермінові числові ряди, що є суттєвим мінусом.

Ще одним прикладом є нейронна мережа прогнозування змін на фондовому ринку – MICEX. MICEX – це нейромережа побудована на базі багатошарової нейронної мережі, що видає непоганий результат та має швидкий період навчання [11].

Також є змога підтянути статистичні дані у форматі “.csv”, що пришвидшує процес навчання.

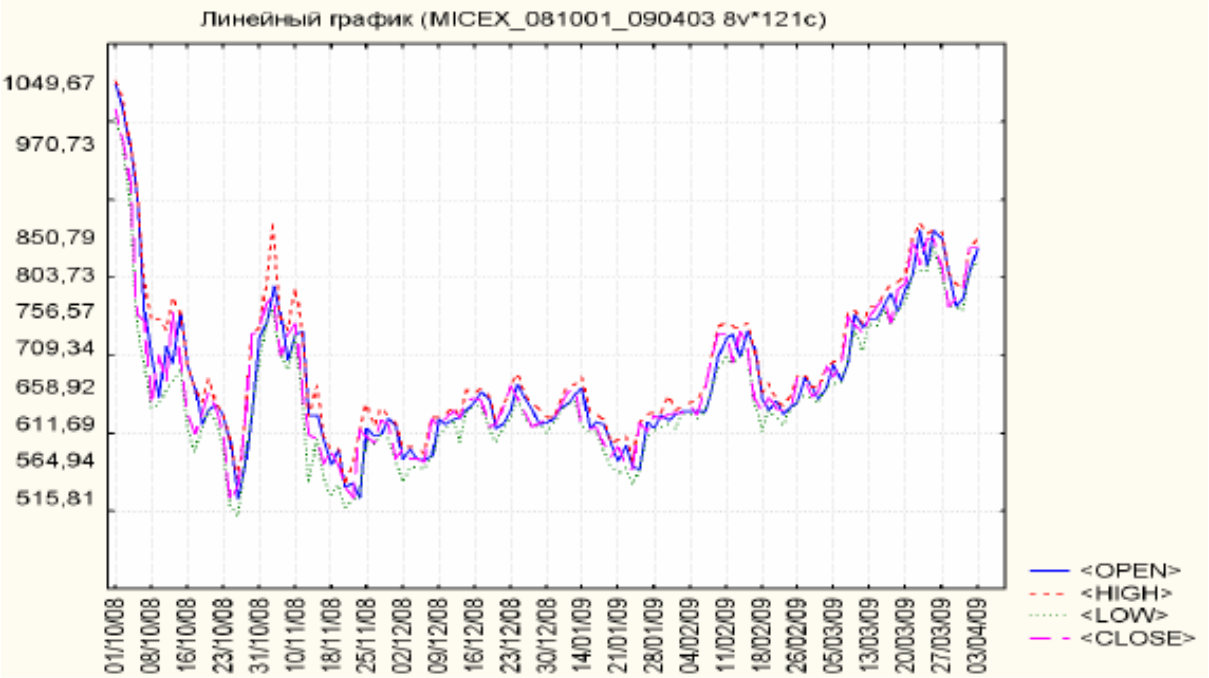

Рисунок 1.4 – Приклад роботи програми MICEX

Проте використання даної архітектури не дає високої точності результатів через свою незміну кількість вузлів нейронної мереж. Це означає, що похибка буде збільшуватись зі збільшенням періоду прогнозування суттєво, так як вона не враховує зміни у часі.

Данні аналоги мають чимало мінусів, тому розробка власної нейронної мережі є доцільною, адже в наш час програмні продукти повинні працювати швидко і давати результати з високою ймовірністю точності.

1.4 Постановка задач дослідження

На основі проведеного аналізу стану питання та можливостей нейронних мереж в задачах прогнозування числових рядів у магістерській кваліфікаційній роботі потрібно виконати такі задачі:

- проаналізувати особливості створення та навчання нейронних мереж;
- розробити модель та метод побудови нейронної мережі прогнозування;

- розробити метод прогнозування ефективності використання інвестицій;
- провести аналіз програмних засобів для створення нейронної мережі;
- здійснити програмну реалізацію розроблених методів для нейронної мережі;
- провести навчання нейронної мережі;
- провести тестування створеного програмного продукту.

1.5 Висновки

У першому розділі магістерської кваліфікаційної роботи проведено аналіз предметної області, також порівняно аналоги нейронних мереж на прикладі перцептронів. Для реалізації було обрано радіально-базисну нейронну мережу. Нейромережа буде призначена для прогнозування ефективності використання інвестицій. Серед аналогів нейромереж для прогнозування числових рядів наведено Anfis та MICEX, наведено їх основні переваги та недоліки. Проаналізовані аналоги дозволили сформувати основі задачі дослідження до нейроної мережі, яка орієнтована на досягнення відповідного результату.

2 РОЗРОБКА НЕЙРОМЕРЕЖЕВИХ МЕТОДІВ І МОДЕЛЕЙ В ЗАДАЧАХ ПРОГНОЗУВАННЯ

2.1 Визначення принципів навчання нейромереж

Основною властивістю біологічного мозку є його здатність до навчання, а оскільки штучна нейронна мережа є моделлю мозку, тому поняття «навчання» є ключовим в теорії штучної нейронної мережі. Тип і характер навчання визначаються на основі розв'язуваної задачі, властивостей даних, які надходять на вхід мережі із зовнішнього середовища у вигляді навчальної вибірки образів або прикладів. Відомо дві основні парадигми навчання: з учителем і без вчителя [12].

Розглянемо їх для аналізу та вибору кращого для реалізації поставлених задач.

Навчання з учителем є більш простим і очевидним. Завідомо відома інформація про зовнішнє середовище, яка задана у вигляді послідовності або пакета вхідних векторів x. Також відомий навчальний сигнал d. Реакція ненавченої мережі, відрізняється від «правильної» реакції вчителя, в результаті чого виникає помилка. У процесі навчання необхідно так налаштувати параметри ШНМ, щоб деяка скалярна функція помилки (критерій якості) досягла свого мінімального значення. Навченою вважається мережа, яка в деякому, як правило, статистичному сенсі повторює реакцію вчителя. Оскільки інформація про зовнішнє середовище зазвичай має нестаціонарний характер, процес навчання йде безперервно, для чого використовуються ті чи інші рекурентні процедури. Відповідно оцінити обраний алгоритм і обчислити критерій якості неможливо, тому крім навчання мережі, необхідно визначати параметр по якому буде оцінено правило навчання. На рис. 2.1 запропонована схема навчання з учителем.

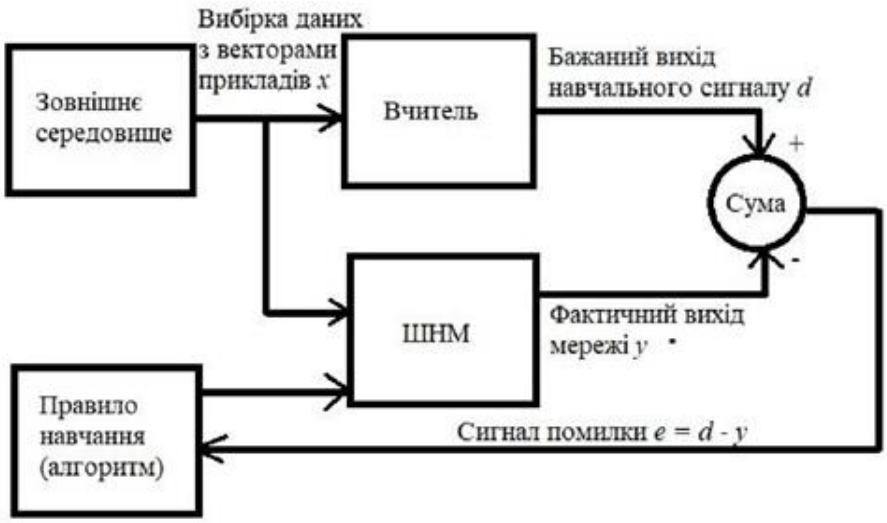

Рисунок 2.1 – Схема навчання з учителем

Парадигмою навчання без учителя, або самонавчання, називають навчання, коли правильна реакція на сигнали зовнішнього середовища невідома. Процес самонавчання схематично представлений на рис. 2.2. Мережі, які реалізують парадигму самонавчання, призначені, як правило, для аналізу внутрішньої латентної структури вхідної інформації і вирішують завдання автоматичної класифікації, кластеризації, факторного аналізу, компресії даних. Ці задачі є більш складними виходячи з математичної постановки, але вирішують безліч реальних проблем.

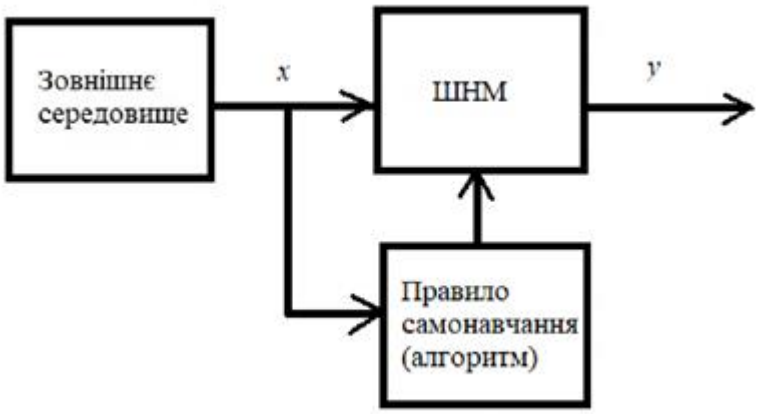

Рисунок 2.2 – Схема навчання без учителя

Використання данних методів має свою плюси і мінуси при розробці систем прогнозування. Виділити якийсь один який чітко підійде для даної розробки важко, тому є сенс об'єднати ці два методи [13].

Своєрідним компромісом між двома цими парадигмами є навчання з підкріпленням, при якому доступна лише непряма інформації про правильну реакції на вхідний сигнал. Нейронна мережа виробляє відображення вхідної інформації у вихідний вектор, проте, оскільки бажаний навчальний сигнал в явному вигляді не заданий, неможливо отримати помилку, на підставі якої відбувається навчання. Передбачається, що є деякі апріорні знання, що дозволяють зв'язати евристичний сигнал підкріплення з бажаним виходом за допомогою деякої функції. Зазвичай ця функція враховує зв'язок вихідних сигналів мережі з подіями у зовнішньому середовищі, для чого в схему навчання вводиться додатковий блок – «критик», що відображає поведінку мережі. Далі обчислюється евристична помилка, на основі якої і реалізується процес навчання.

Досить широке поширення набула також парадигма змішаного навчання, коли частина параметрів мережі налаштовується за допомогою навчання з учителем, а інша частина або архітектура в цілому – за допомогою самонавчання. Цей підхід набув найбільшого поширення при навчанні радіально-базисних ШНМ [14].

З введеними парадигмами тісно пов'язані правила навчання, що лежать в основі конкретних алгоритмів. У конкурентному навчанні, можуть бути реалізовані всі описані парадигми, при цьому його відмінною рисою є процес «змагання» нейронів вихідного шару. Найбільш яскравими прикладами мереж, що використовують це правило, є мережі адаптивного резонансу і самоорганізовані мапи (SOM).

Безліч реальних задач характеризується тим, що дані надходять в online режимі, їх обробка повинна проводитися в темпі функціонування об'єкта, а сам об'єкт є нестаціонарним. Зрозуміло, що класичний багатошаровий персептрон, що є універсальним апроксіматором, в даному випадку не може бути

використано, а в якості альтернативи йому в ряді випадків можуть бути використані радіально-базисні нейронні мережі (РБНМ) [9]. Ці мережі також є універсальними апроксіматорамі, а оскільки їх виходовий сигнал лінійно залежить від параметрів, що налаштовуються, тобто синаптичних ваг, для їх навчання в реальному часі може бути використаний рекурентний метод найменших квадратів або його модифікації, що є, по суті, алгоритмами оптимізації другого порядку, що забезпечують квадратичну збіжність до оптимального рішенням [15].

Поєднує між собою ці методи навчання саме глибинне навчання. Саме його буде використовувати нейронна мережа прогнозування.

Глибинне навчання – це техніка навчання нейромережі, яка використовує безліч шарів для вирішення складних проблем за допомогою шаблонів.

Алгоритми глибинного навчання перетворюють свої входи крізь більшу кількість шарів, ніж алгоритми поверхневого навчання. На кожному шарі сигнал перетворюється блоком обробки, таким як штучний нейрон, параметри якого «навчаються» шляхом тренування. Ланцюг перетворень від входу до виходу є шляхом передачі довіри. Вони описують потенційно причинні зв'язки між входом та виходом, і можуть мати змінну довжину. Для нейронної мережі прямого поширення довжина шляхів передачі довіри, і відтак глибина цієї мережі, є числом прихованих шарів.

Багато алгоритмів глибинного навчання застосовуються до задач спонтанного навчання. Це є важливою перевагою, оскільки немічені дані зазвичай є багатшими за мічені. Прикладами глибинних структур, які можуть тренуватися спонтанним чином, є нейронні стискачі історії та глибинні мережі переконань [16-17].

2.2 Розробка методу та моделі навчання нейромереж для задач прогнозування

В якості алгоритму навчання було обрано метод навчання з вчителем для радіально-базисної нейронної мережі – метод зворотного розповсюдження помилки.

Це ітеративний градієнтний алгоритм, який використовується з метою мінімізації помилки роботи та отримання результату на виході. Основна ідея цього методу полягає в поширенні сигналів помилки від виходів мережі до її входів, в напрямку, зворотному прямому поширенню сигналів у звичайному режимі роботи [18].

У мережі є множина входів $x_1 \ldots x_n$. Перенумеруємо всі вузли (включаючи входи і виходи) числами від 1 до N (наскрізна нумерація, незалежно від топології шарів). Позначимо через ω_{ij} вагу зв'язку, що з'єднує i-й і j-й вузли, а через o_i— вихід i-го вузла. Якщо нам відомий навчальний приклад (правильні відповіді мережі t_k, $k \in Outputs$) то функція помилки, отримана за методом найменших квадратів, виглядає так (2.1):

$$E = (\{w_{ij}\}) = \frac{1}{2} \sum_{k \in Outputs} (t_k - o_k)^2, \tag{2.1}$$

Для модифікації вагів нейронів використовується стохастичний градієнтний спуск, тобто будемо корегувати ваги після кожного навчального прикладу і, таким чином, «рухатися» в багатовимірному просторі ваг. Щоб «дістатися» до мінімуму помилки, нам потрібно «рухатися» в сторону, протилежну градієнту, тобто, на підставі кожної групи правильних відповідей, додавати до кожної ваги w_{ij} (2.2):

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}, \tag{2.2}$$

Де $0 < \eta < 1$ - множник, що задає швидкість «руху».

Похідна розраховується таким чином (2.3). Нехай спочатку $j \in Outputs$ тобто вага, яка нас цікавить, входить в нейрон останнього рівня. Спочатку

зазначимо, що ω_{ij} впливає на вихід мережі лише як частина суми, що береться по входах j-го вузла.

Тому:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial S_j}\frac{\partial S_j}{\partial w_{ij}} = x_i \frac{\partial E}{\partial S_j}\ , \tag{2.3}$$

Аналогічно, S_j впливає на загальну помилку тільки в рамках виходу j-го вузла o_j (2.4). Тому:

$$\frac{\partial E}{\partial S_j} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial S_j} = \left(\frac{\partial}{\partial o_j}\frac{1}{2}\sum_{k\in Outputs}(t_k - o_k)^2\right)\left(\frac{\partial \sigma(S_j)}{\partial S_j}\right), \tag{2.4}$$

де $\sigma()$ — це функція активації , у даному випадку (стосовно обчислення похідної) являє собою Експоненційну сигмоїду (2.5). Якщо ж j-й вузол — не на останньому рівні, то у нього є виходи; позначимо їх через Children (j). У цьому випадку:

$$\frac{\partial E}{\partial S_j} = \sum_{k\in Children(j)}\frac{\partial E}{\partial o_j}\frac{\partial S_k}{\partial S_j}, \tag{2.5}$$

Це анологічна поправка, але обчислена для вузла наступного рівня (будемо позначати її через δ_k від Δ_k вона відрізняється відсутністю множника $(-\eta x_{ij})$.

Оскільки проводиться обчислення поправки для вузлів останнього рівня і вираження поправки для вузла нижчого рівня через поправки більш високого, можна вже створювати алгоритм навчання. Саме через цю особливість обчислення поправок цей алгоритм називається алгоритмом зворотного поширення помилки [19].

На вхід алгоритму, крім зазначених параметрів, потрібно також подавати форматі структуру мережі. На практиці дуже гарні результати показують мережі досить простої структури, що складаються з двох рівнів нейронів — прихованого рівня (hidden units) і нейронів-виходів (output units), кожен вхід мережі з'єднаний з усіма прихованими нейронами, а результат роботи кожного прихованого нейрона подається на вхід кожному з нейронів-виходів. У такому випадку досить подавати на вхід кількість нейронів прихованого рівня [20].

Зважаючи на обрану технологію, загальний алгоритм можна описати блок-схемою, поданою на рис. 2.3.

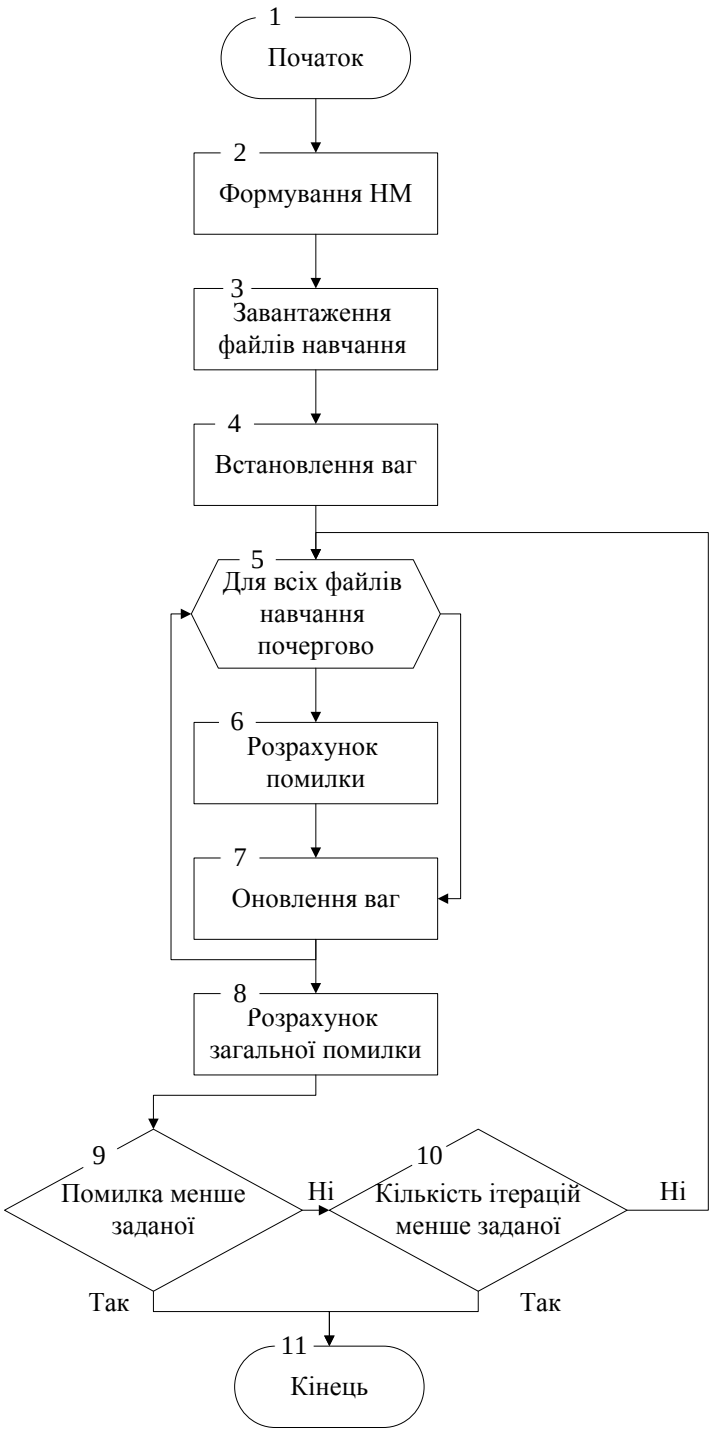

Рисунок 2.3 – Блок-схема моделі навчання нейромережі для прогнозування

Це дозволяє при побудові нейромережі для засобів прогнозування, динамічно збільшувати шарову архітектуру мережі, збільшуючи кількість вузлів,

та корегувати значення вагів впливу кожного шару в процесі навчання, що дозволяє підвищити точність розрахунків.

2.3 Розробка методу та моделі прогнозування використання інвестицій на базі нейронних мереж.

Найчастіше в літературі пропонується використовувати в якості моделей прогнозування середнє арифметичне значення прогнозів всіх мереж-експертів [21]. Але цю ідею можна істотно розвинути. Розглянемо кілька можливих способів організації прогнозуючих нейромереж.

Можна, наприклад, навчити ще одну нейронну мережу («керівника комітету»), входами якої будуть прогнози всіх нейро-експертів, а виходом - підсумковий прогноз комітету.

Іншим підходом може бути введення поняття «спеціалізації» експертів. З цією метою пропонується провести попередню кластеризацію вхідних образів навчальної вибірки, тобто розбити вихідну вибірку на кілька (2-5) груп схожих вхідних наборів. Наприклад, в деякі групи даних можуть потрапити навчальні набори, які характеризуються зростаючим трендом, в інші - спадним і т.п. Для такої кластеризації може бути використана система, що самоорганізується, нейронна мережа, яка називається картою Когонена. Далі, для кожного кластера виділяється, як мінімум, два нейроексперта, які навчаються тільки на даних, які потрапили в цей кластер. Таким чином, створюються підкомітети нейроекспертів, що спеціалізуються на прогнозуванні в умовах тієї чи іншої ситуації, ринкової ситуації. У режимі функціонування комітету вхідний образ спочатку аналізується картою Когонена, щоб визначити, до якого з наявних кластерів він відноситься. Потім підсумковий прогноз здійснюється тим підкомітетом, спеціалізацією якого є даний кластер. Нейромережева мапа спеціалізованих експертів може використовуватися не тільки в процесі прогнозування, а й для аналітичних цілей. Зокрема, на основі мапи в міру надходження запитів можна зробити висновок про проблемні областях комітету

(наприклад, виявити кластери експерти яких відрізняються гіршою якістю прогнозу). Це дає нову стратегію навчання і поповнення комітету новими моделями.

Після завершення етапу навчання всі вхідні дані навчальної вибірки кластеризуються, як і при використанні попереднього підходу [22]. А потім для кожного нейроексперта визначається коефіцієнт його компетентності на даних кожного. В процесі функціонування мережі коефіцієнти компетентності корегуються в залежності від величини помилок прогнозу нейроексперта на даних цього кластера. Об'єднання експертів в ансамбль при вирішенні завдання підсумкового прогнозування проводиться з вагами, відповідними коефіцієнтами компетентності нейроекспертов для того кластера, в який потрапляє аналізований вхідний вектор. Тобто в якості значення комітету береться зважене середнє арифметичне значення прогнозів всіх експертів (рис. 2.4)

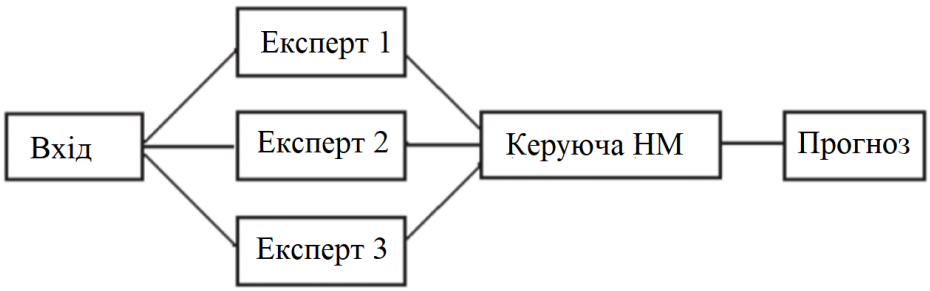

Рисунок 2.4 – Архітектура моделі прогнозування

У останньому етапі прогнозування, як відповідь комітету використовується прогноз тієї з P наявних в розпорядженні мереж, яка в даний момент t є найкращою в тому сенсі, що для неї сума квадратів помилок прогнозів за k періодів (2.6):

$$\sum_{s=1}^{k}(y_i(t-s)-y(t-s))^2 \to \min_{1\le i\le P}. \qquad (2.6)$$

де $y_i(t-s)$ - прогноз мережі з номером і в момент часу $(t-s)$;

$y(t-s)$ - справжнє значення ряду в цей момент.

Величина k задає швидкість забування передісторії і, як показують чисельні експерименти, її найкращі значення знаходяться в проміжку $k \in [3,5]$.

Необхідно відзначити, що етап оптимізації моделі і вибір даних проходить в індивідуальному порядку щодо кожного вхідного образу. В результаті, з'являється шанс модифікувати архітектуру моделі, складність її алгоритму та чинники, які необхідно враховувати для вибору даних, що зберігає послідовність та враховує поточну ситуацію. Можна враховувати характерну якість і кількість даних, відповідні обмеження, стан і положення об'єкта. Отже, оновлені дані, що надходять на обробку в почерговому порядку, далі зважуються і зберігаються, а попередні по закінченню часу забуваються, так як їх вплив на результат може бути негативним. Таким чином проводиться аналіз діапазону дискретних точок оцінювання інвестицій з урахуванням умов зміни процесу обрахунку, що дозволяє врахувати динамічні критерії впливу на ефективність використання інвестицій

2.4 Висновки

У другому розділі магістерської кваліфікаційної роботи проведено аналіз принципів навчання нейронних мереж таких як: навчання з вчителем чи без нього, навчання з підкріпленням та смішане навчання. Наведено їх переваги та недоліки, так як для розробки було обрано радіально-базисних нейронних мереж, яка може поєднати в собі кілька методів, буде обрано метод глибиного навчання. Який дозволяє реалізувати завдання з найменшою похибкою достовірності результатів. Розроблено метод навчання нейронної мереж побудовану на принципі зворотного розповсюдження помилки, що дозволяє динамічно збільшувати шарову архітектуру мережі та корегувати значення вагів впливу кожного шару в процесі навчання. Розроблено метод та модель прогнозування використання інвестицій на базі нейронних мереж, х використанням карт Когонена, що дозволяє провести аналіз діапазону дискретних точок оцінювання інвестицій з урахуванням умов зміни процесу обрахунку, що дозволяє врахувати динамічні критерії впливу на ефективність використання інвестицій.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ВИЗНАЧЕННЯ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ ІНВЕСТИЦІЙ

3.1 Варіантний аналіз і обгрунтування вибору засобів реалізації програмного продукту

Для розробки нейронних мереж потрібна об'єктно-орієнтована мова програмування високого рівня. Найчастіше використовують: Python, C++ та C#.

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована [23].

Серед основних її переваг можна назвати такі:

- чистий синтаксис (для виділення блоків слід використовувати відступи);
- переносність програм (що властиве більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);
- стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане на мові Python;
- зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);

- відкритий код (можливість редагувати його іншими користувачами).

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

C++ – мова програмування високого рівня з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної [24].

C ++ широко використовується для розробки програмного забезпечення, будучи одним з найпопулярніших мов програмування. Область його застосування включає створення операційних систем, різноманітних прикладних програм, драйверів пристроїв, додатків для вбудованих систем, високопродуктивних серверів, а також ігор. Існує безліч реалізацій мови C ++, як безкоштовних, так і комерційних і для різних платформ.

C ++ містить засоби розробки програм контрольованої ефективності для широкого спектра задач, від низькорівневих утиліт і драйверів до вельми складних програмних комплексів. Зокрема:

- Обчислювальна продуктивність: мова спроектований так, щоб дати програмісту максимальний контроль над усіма аспектами структури і порядку виконання програми.
- Підтримка різних стилів програмування: традиційне імперативне програмування (структурний, об'єктно-орієнтоване), узагальнене програмування, функціональне програмування, що породжує метапрограмування.
- Автоматичний виклик деструкторів об'єктів в адекватному порядку (зворотному виклику конструкторів) спрощує і підвищує надійність управління пам'яттю і іншими ресурсами (відкритими файлами, мережевими з'єднаннями, сполуками з базами даних і т. П.).
- Перевантаження операторів дозволяє коротко і ємко записувати вирази над користувацькими типами у природному алгебраїчній формі.

- Є можливість управління константністю об'єктів (модифікатори const, mutable, volatile). Використання константних об'єктів підвищує надійність і служить підказкою для оптимізації.
- Шаблони C ++ дають можливість побудови узагальнених контейнерів і алгоритмів для різних типів даних. Попутно шаблони дають можливість виробляти обчислення на етапі компіляції.
- Можливість вбудовування предметно-орієнтованих мов програмування в основний код. Такий підхід використовує, наприклад бібліотека Boost.Spirit, що дозволяє задавати EBNF-граматику парсеров прямо в коді C ++.
- Доступність. Для C ++ існує величезна кількість навчальної літератури, перекладеної на всілякі мови. Мова має високий поріг входження, але серед всіх мов такого роду має найбільш широкі можливості.

До числа зазвичай згадуються недоліків мови можна віднести:

- Відсутність системи модулів. C ++ успадкував від Сі підключення заголовних файлів за допомогою препроцесора. Це змушує дублювати опису об'єктів, породжує неочевидні вимоги до коду і збільшує його обсяг, а значить і час компіляції.
- Наявність більш ніж одного механізму для виконання одних і тих же завдань, що ускладнює мову і призводить до неоптимальному і небезпечному кодування.
- Відсутність вбудованих механізмів статичної валідації часу життя об'єктів, що приводить до раптового краху програм через звернення до знищеної змінної, або через неправильну багатопотокової роботи з об'єктами.
- Шаблони породжують об'ємний і не завжди оптимальний код. Часткове визначення шаблонів ускладнює як сама мова, так і програми, де воно використовується.
- Множинне (в тому числі віртуальне) спадкування призводить до створення громіздких ієрархій класів, які при будь-якій зміні вимог до програми можуть зажадати серйозного перегляду.

- Складний синтаксис і об'ємна специфікація мови ускладнюють його вивчення.
- Складна і постійно розростається стандартна бібліотека, яка утрудняє вивчення мови.

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників – мов C++, Delphi, Модула і Smalltalk – C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++) [25].

C# безпечніший в порівнянні з C++. Єдиними неявними перетвореннями за умовчуванням є ті, які вважаються безпечними, наприклад, розширення цілих чисел. Це застосовується під час компіляції, під час JIT і, в деяких випадках, під час виконання. Не відбувається неявних перетворень між булевими і цілими числами, а також між членами перерахування і цілими числами (крім літерала 0, який може бути неявно перетворений в будь-який нумерований тип). Будь-яке призначене для користувача перетворення повинно бути явно позначене як явне або неявне, на відміну від конструкторів копіювання C++ і операторів перетворення, які за умовчанням є неявними.

C# має явну підтримку коваріації та контраваріантності в родових типах, на відміну від C++, яка має певний рівень підтримки контраваріантності просто через семантику типів, що повертаються, на віртуальні методи.

Мова C # не допускає глобальних змінних або функцій. Всі методи і члени повинні бути оголошені всередині класів. Статичні члени відкритих класів можуть замінювати глобальні змінні та функції.

C# надає властивості як синтаксичного цукру для загального шаблону, в якому пара методів, accessor (getter) і mutator (setter) інкапсулює операції по

одному атрибуту класу. Не потрібно писати надлишкові сигнатури методів для реалізацій гетера/сетера і до цієї властивості можна отримати доступ, використовуючи синтаксис атрибутів, а не більш докладні виклики методів.

Беручи до уваги властивості кожної з розглянутих мов програмування, для розробки було обрано мову програмування Python через свою швидкодію, зручність написання коду та можливість підключення додакових бібліотек які покращують роботу нейромереж.

3.2 Програмне моделювання нейромережі

Програма розрозблена згідно з наданою на рис. 3.1 діаграмою класів.

Розглянемо складові діаграми, щоб детальніше зрозуміти роботу програми.

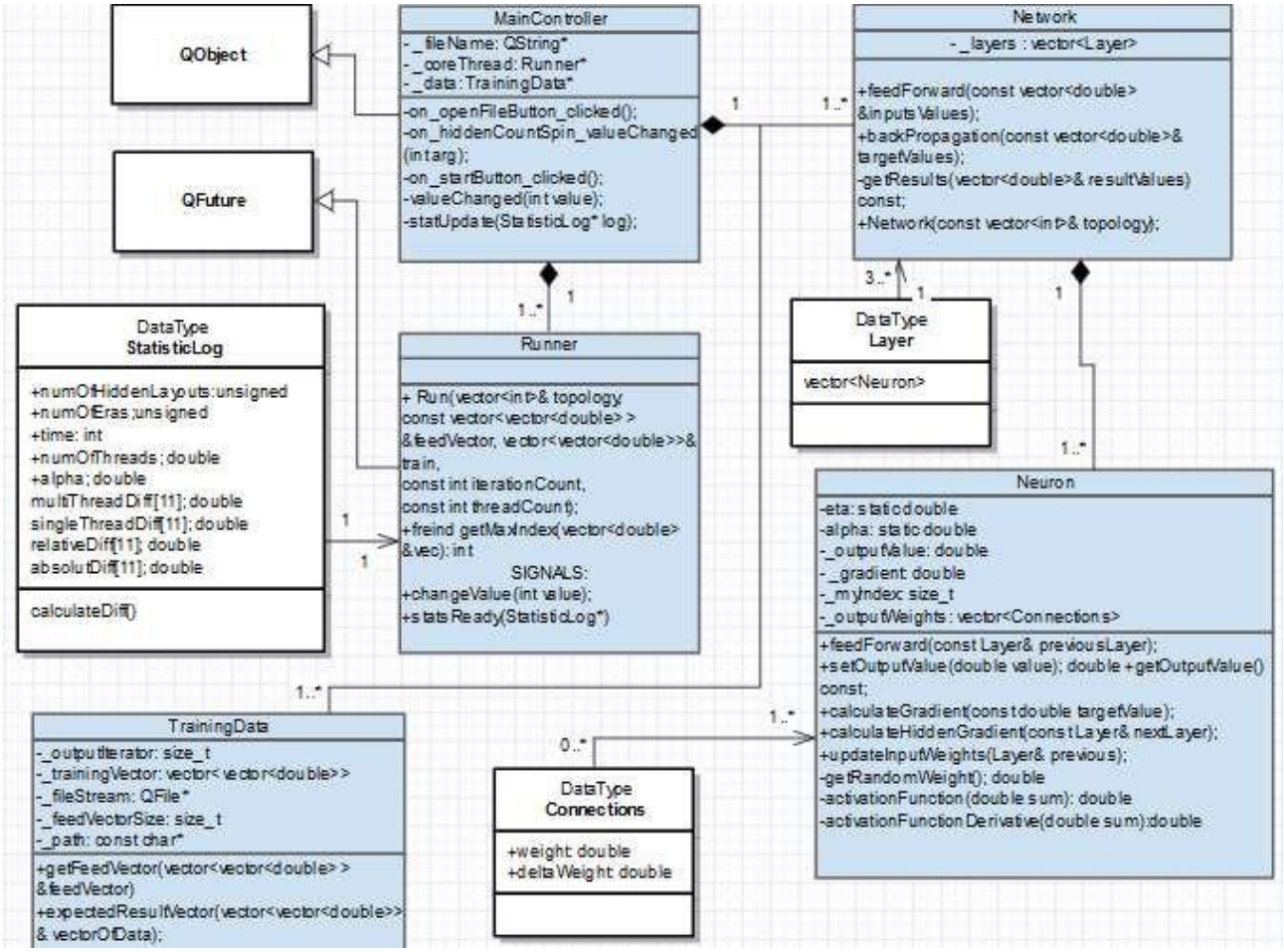

Рисунок 3.1 – UML діаграма класів програми

Клас MainController унаслідований від QObject відповідає за виконання основного функціоналу програми, а також застосовує параметри та функції які викликаються із графічного інтерфейсу користувача. Відповідає за запуск нового потоку в якому будуть проводиться обчислення, не залежно від інтерфейсу, а також приймає та відображає статистичну та графічну статистику після завершення навчання мережі.

TrainingData – реалізує зчитування навчального набору даних, у форматі Semion dataset – бази даних зразків рукописного написання цифр.

Runner – клас, що реалізує високорівневе API QTConcurrent / QFuture, що дозволяє винести основну функціональність в окремий потік, що в свою чергу запобігає зависанню інтерфейсу користувача. Саме в функції Run цього класу і відбувається основне навчання. Ця функція виконується двічі- в послідовному та паралельному режимах. Також після виконання функції Run формується StatisticLog – службовий клас для збереження інформації про часові затрати даного раунду навчання, далі він використовується для представлення статистики в текстовому, або графічному режимах.

Клас Network реалізує навчання нейронної мережі методом зворотного розповсюдження помилки. Даний клас забезпечує такий функціонал: ініціалізацію мережі та встановлення вагів в конструкторі класу, ініціалізацію входів мережі згідно навчального вектору за допомогою функції feedVector. Сам алгоритм зворотнього розповсюдження помилки реалізований функцією backpropagation.

За розрахунок дельт на нейронах та їх складання відповідає клас Neuron. Також в ньому реалізовані функція активації та її похідна, в даному випадку в якості функції активації було обрано гіперболічний тангенс, а в якості похідної його апроксимацію $1 - x^2$. Також в даному класі зберігаються змінні що відповідають за швидкість навчання мережі. $\eta(eta)$ – коефіцієнт швидкості навчання мережі та α – множник вагів попередніг вагів. Експерементально було визначено значення цих змінних 0.35 та 0.05 відповідно [26].

3.3 Навчання нейронної мережі прогнозування

У сучасному світі, починаючи з охорони здоров'я і закінчуючи мануфактурним виробництвом, використовують глибинне навчання. Компанії звертаються до цих технологій для вирішення складних проблем, таких як розпізнавання мов і об'єктів, машинного перекладу і так далі.

Глибинне навчання – це метод навчання, який дозволяє навчати ШНМ для прогнозування виходів з урахуванням набору вхідних даних. Для цих цілей можна використовувати як кероване, так і не кероване навчання [27].

Для глибинного навчання використовуються додаткові файли, що аналізує сама нейронна мережа. Ці файли містять в собі інформацію про різні компанії, зміну їх прибутку ти ціни їх активів протягом всього періоду їх існування.

Архітектура глибинної нейронної мережі зображена на рис. 3.2.

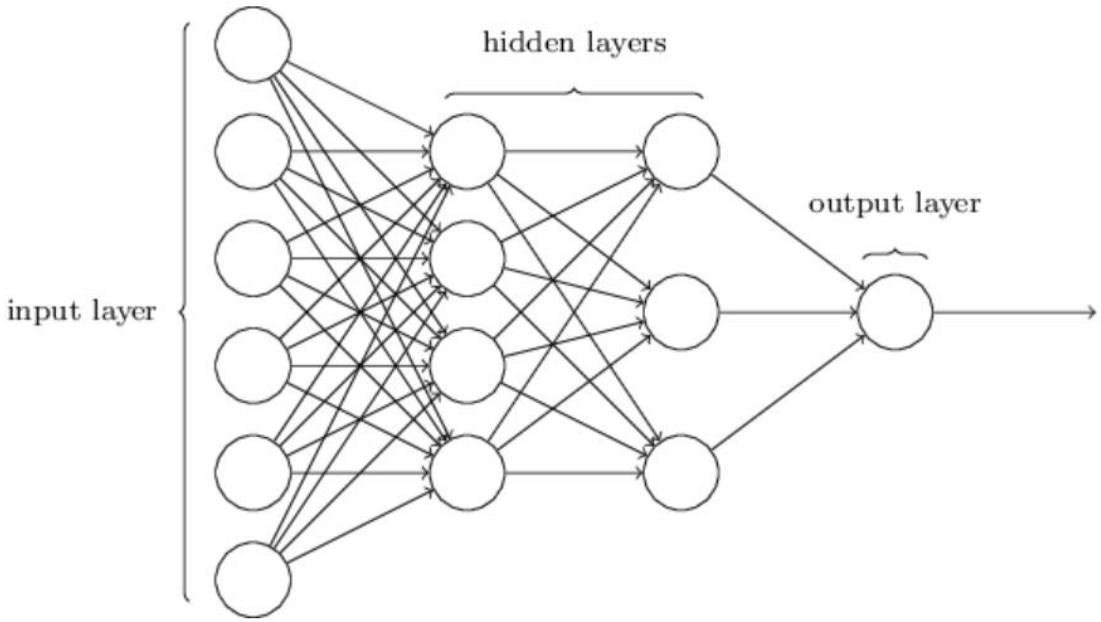

Рисунок 3.2 – Архітектура глибинної нейронної мережі

Нейрони згруповані в три різних типи шарів:

1. Вхідний шар.
2. Прихований(і) шар(и).
3. Вихідний шар.

Вхідний шар приймає спочатку надані дані. У даному випадку тут є чотири нейрона: початковий аеропорт, аеропорт призначення, дата відбуття і авіакомпанія. Вхідний шар передає входи в перший прихований шар.

Приховані шари виконують математичні обчислення на входах. Однією з проблем при створенні нейронних мереж є визначення кількості прихованих шарів, а також кількості нейронів для кожного такого шару. «Глибина» відноситься до наявності більш ніж одного прихованого шару.

Вихідний шар повертає результат (вихідні дані).

Звійсно, що глибинне навчання ще далеко від досконалості, але воно вже близьке до того, щоб приносити комерційну користь. Наприклад, самокеруючі машини. Такі відомі компанії, як Google, Tesla і Uber вже намагаються впровадити автономні автомобілі на вулиці міста.

Кожне з'єднання між нейронами пов'язано з вагою (рис. 3.3). Ця вага визначає важливість вхідного значення. Вихідна вага задається випадковим чином.

При прогнозуванні ціни на активи, прибуток фірми є одним з найбільш важких факторів. Отже, з'єднання нейронів, які відповідають за розрахунок значення прибутку матимуть велику вагу [28].

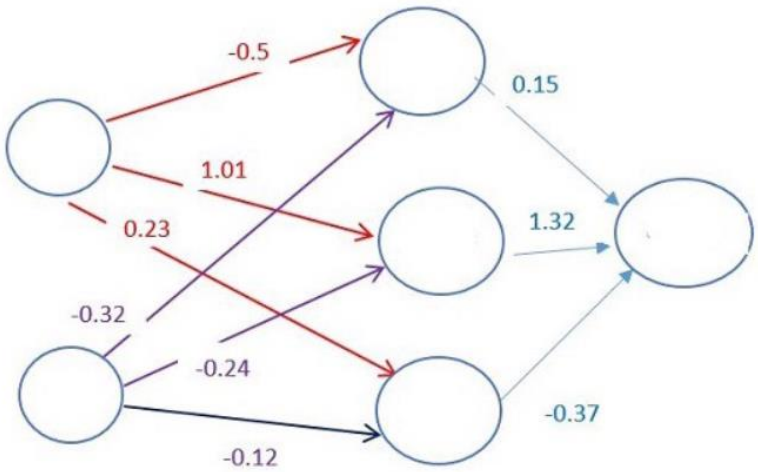

Рисунок 3.3 – Архітуектура нейронної мережі зі значеннями синаптичних ваг

Для досягнення зменшення впливу функції прибутку фірми на точність результату, необхідно використовувати метод градієнтного спуск. Метод градієнтного спуску (рис. 3.4) – це метод, який дозволяє знайти мінімум функції. Він працює, трохи змінюючи вагу після кожної ітерації для набору даних. Обчислюючи похідну (або градієнт) функції прибутку при певному наборі ваги, спостерігаємо в якому напрямку знаходиться мінімум:

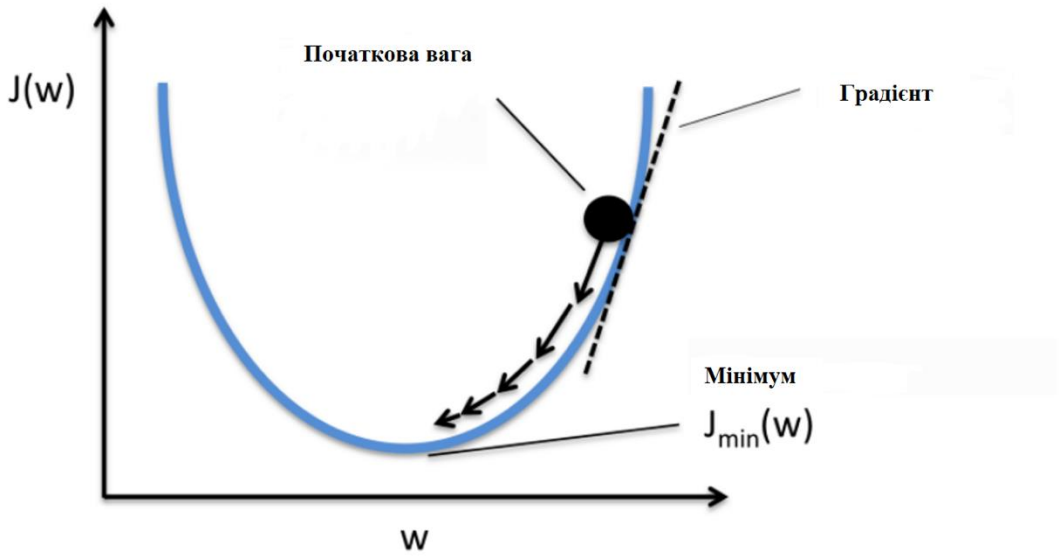

Рисунок 3.4 – Візуалізація методу градієнтного спуску

Щоб підвищити точність отриманих результатів, доведеться багаторазово перебирати дані. Ось чому потрібна велика обчислювальна потужність. Оновлення ваги за допомогою градієнтного спуску виконується автоматично. Саме це є перевагами глибокого навчання.

Після проведення процесу навчання мережі можна запускати моделювання самої системи прогнозування ефективності використання інвестицій.

3.4 Розробка моделей прийняття рішень

Для створення основних критеріїв прийняття рішень будуть використовуватися дані, що описують зміни ціни акцій різних компаній. Тобто під якими чинниками відбувалися розвиток і збільшення вартості компанії чи

навпаки, здешевлення під впливом різних умов. Нейронна мережа записує ці чинники у власні вузли, чим більше використовувати таких файлів, тим більше буде точність результатів.

Їх можна завантажити на Yahoo Finance в форматі .csv. приклад коду який робить зчитування даного файлу наведений на рис. 3.5.

```
data = pd.read_csv('./data/AAPL.csv')[::-1]
close_price = data.ix[:, 'Adj Close'].tolist()
plt.plot(close_price)
plt.show()
```

Рисунок 3.5 – Приклад коду зчитування файлів

Для обробки таких даних використовується бібліотека Keras.

Keras – відкрита нейромережева бібліотека, написана мовою Python. Вона здатна працювати поверх Deeplearning, TensorFlow та Theano. Спроектовану для уможливлення швидких експериментів з мережами глибинного навчання, її зосереджено на тому, щоби вона була мінімальною, модульною та розширюваною. Ця бібліотека містить численні реалізації широко вживаних будівельних блоків нейронних мереж, таких як шари, цільові та передавальні функції, оптимізатори, та безліч інструментів для спрощення роботи із зображеннями та текстом [29].

Для задачі прогнозування в кінці параметр активації повинен бути лінійним. Далі визначаються функції помилки і алгоритм оптимізації. Довжина кроку градієнтного спуску є 0.001. Так як кожен вузол має свій вагу впливу на різні фактори, він задається числовим значенням, приклад опису вхідного вузла для градієнтного спуску зображено на рис. 3.6.

```
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.9, patience=5, min_lr=0.000001, verb
ose=1)
model.compile(optimizer=opt,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Рисунок 3.6 – Приклад коду задання параметрів вхідного вузла

Фрагмент коду початку навчання нейронної мережі зображений на рис. 3.7.

```
history = model.fit(X_train, Y_train,
          nb_epoch = 50,
          batch_size = 128,
          verbose=1,
          validation_data=(X_test, Y_test),
          shuffle=True,
          callbacks=[reduce_lr])
```

Рисунок 3.7 – Приклад коду початку навчання

Якщо з'являється ефект перенавчання, потрібно додати регуляризацію. Під час перенавчання будується модель, яка "запам'ятовує" тренувальні дані і не дозволяє узагальнити знання на нові дані. В процесі регуляризації накладаються певні обмеження на ваги нейронної мережі, щоб не було великого розкиду в значеннях і не зважаючі на велику кількість параметрів (тобто ваг мережі), частина з них перетворюється на нуль для спрощення. Почнемо з самого поширеного способу – додавання до функції похибки додаткової складової з L2 нормою за сумою ваг, в Keras це робиться за допомогою keras.regularizers.activity_regularizer. Приклад коду для регуляризації вагів нейронної мережі наведено на рис. 3.8.

```
model = Sequential()
model.add(Dense(64, input_dim=30,
                activity_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(16,
                activity_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(2))
model.add(Activation('softmax'))
```

Рисунок 3.8 – Приклад коду для регуляризації вагів нейронної мережі

Після того як навчання відбудеться результат буде виводитись у вигляді графіків залежності за допомогою коду наведеного на рис. 3.9.

```
plt.figure()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='best')
plt.show()

plt.figure()
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('acc')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='best')
plt.show()
```

Рисунок 3.9 – Приклад коду виводу результату

Таким чином будується основний метод прийняття рішень для нейромережі прогнозування.

3.5 Висновки

У третьому розділі магістерської кваліфікаційної роботи проведено варіантний аналіз і обгрунтування вибору засобів реалізації програмного

продукту, а саме порівняння високорівневих мов програмування: Python, C++ та C#. Наведено їх переваги та недоліки. Для розробки було обрано мову програмування Python. Обгрунтовано основний принцип навчання нейромережі, побудований на принципі глибокого навчання. Детально описано систему його роботи при навчанні нейронної мережі. Розроблено модель прийняття рішень методу прогнозування ефективності використання інвестицій з наведенням її UML-даіграми класів та фрагментами коду реалізації, таких як: завантаження файлів, створення вулів мережі, початок навчання, регуляризація вагів нейронної мережі та вивід результату.

4 ТЕСТУВАННЯ РОБОТИ ПРОГРАМИ

4.1 Аналіз принципів тестування

Тестування готових програмних продуктів відбувається за методом чорного то бігого ящиків.

Тестування чорного ящика або поведінковий тестування - стратегія (метод) тестування функціонального поведінки об'єкта (програми, системи) з точки зору зовнішнього світу, при якому не використовується знання про внутрішній устрій тестованого об'єкта. Під стратегією розуміються систематичні методи відбору і створення тестів для тестового набору. Стратегія поведінкового тесту виходить з технічних вимог і їх специфікацій

Під «чорним ящиком» розуміється об'єкт дослідження, внутрішній устрій якого невідомо. Поняття «чорний ящик» запропоновано У. Р. Ешбі. У кібернетиці воно дозволяє вивчати поведінку систем, тобто їх реакцій на різноманітні зовнішні впливи і в той же час абстрагуватися від їх внутрішнього устрою.

Маніпулюючи тільки зі входами і виходами, можна проводити певні дослідження. На практиці завжди виникає питання, наскільки гомоморфізм «чорного» ящика відображає адекватність його досліджуваної моделі, тобто наскільки повно в моделі відображаються основні властивості оригіналу.

Опис будь-якої системи управління в часі характеризується картиною послідовності її станів в процесі руху до стоїть перед нею мети. Перетворення в системі управління може бути або взаємно-однозначним і тоді воно називається ізоморфним, або тільки однозначним, в одну сторону. В такому випадку перетворення називають гомоморфним.

«Чорний» ящик являє собою складну гомоморфності модель кібернетичної системи, в якій дотримується різноманітність. Він тільки тоді є задовільною моделлю системи, коли містить таку кількість інформації, яке відображає різноманітність системи. Можна припустити, що чим більше число збурень діє на входи моделі системи, тим більшу різноманітність повинен мати регулятор.

В даний час відомі два види «чорних» ящиків. До першого виду відносять будь-який «чорний» ящик, який може розглядатися як автомат, званий кінцевим або нескінченним. Поведінка таких «чорних» ящиків відомо. До другого виду відносяться такі «чорні» ящики, поведінка яких може бути наблюдаемо тільки в експерименті. В такому випадку в явній або неявній формі висловлюється гіпотеза про передбачуваність поведінки «чорного» ящика в імовірнісному сенсі. Без попередньої гіпотези неможливо будь-яке узагальнення, або, як кажуть, неможливо зробити індуктивне висновок на основі експериментів з «чорним» ящиком. Для позначення моделі «чорного» ящика Н. Вінером запропоновано поняття «білого» ящика. «Білий» ящик складається з відомих компонентів, тобто відомих X, Y, δ, λ. Його вміст спеціально підбирається для реалізації тієї ж залежності виходу від входу, що і у відповідного «чорного» ящика. У процесі проведених досліджень і при узагальненнях, висунення гіпотез і встановлення закономірностей виникає необхідність коригування організації «білого» ящика і зміни моделей. У зв'язку з цим при моделюванні дослідник повинен обов'язково багаторазово звертатися до схеми відносин «чорний» - «білий» ящик.

Тестування білого ящика – тестування, яке враховує внутрішні механізми системи або компонента.

Зазвичай включає тестування гілок, маршрутів, операторів. При тестуванні вибирають входи для виконання різних частин коду і визначають очікувані результати.

Традиційно тестування білого ящика виконується на рівні модулів, проте воно використовується для тестування інтеграції систем і системного тестування, тестування всередині пристрою і шляхів між пристроями. Цей метод тестування не може виявити невиконані частини специфікації, відсутність вимог або створення не того типу додатка.

Саме тому для тестування розробленого програмного додатку використовується метод чорного ящика [30].

4.2 Тестування роботи нейронної мережі

Для тестування розробленої нейронної мережі було використано один з завантажених данних для навчання мережі. Для прикладу були взяті ціни акцій компанії Apple з 2005 по сьогоднішній день, які завантажено в нейронну мережу та виведено на екран (рис. 4.1).

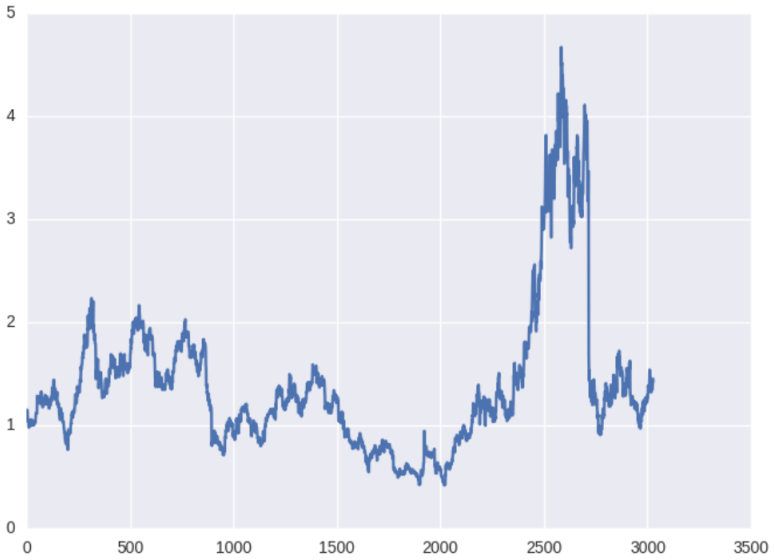

Рисунок 4.1 – Графік змінення цін акцій компанії Apple р 2005 року по сьогоднішній день

Результати графіків змін значень функції похибки та точності навчання нейронної мережі залежно від часу авчання наведені на рис. 4.2 і 4.3.

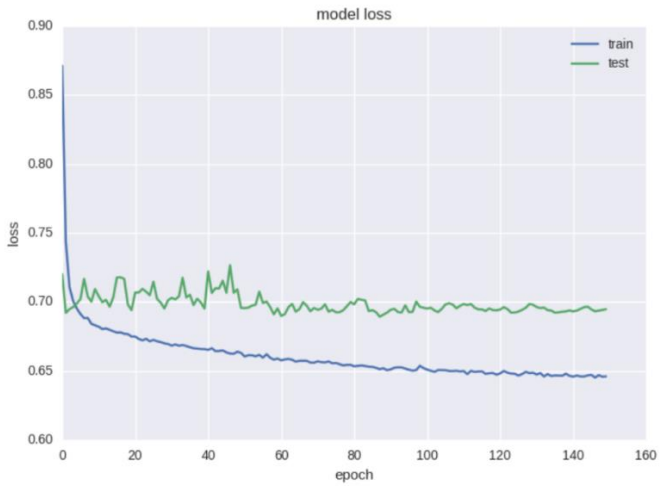

Рисунок 4.2 – Графік зміни значень функції похибки начання нейронної мережі

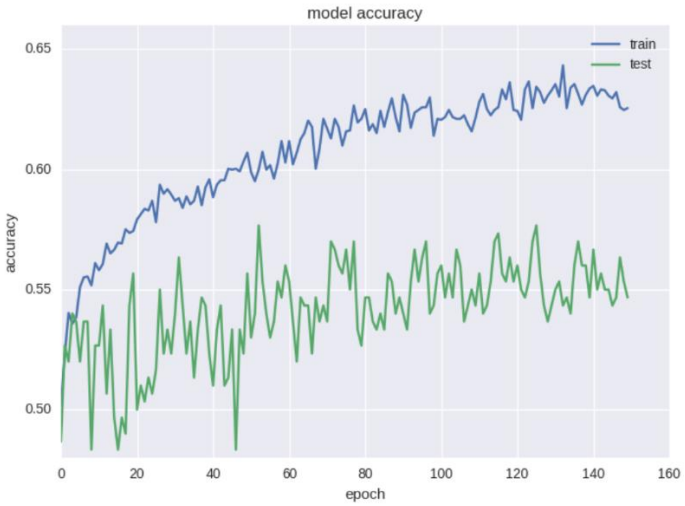

Рисунок 4.3 – Графік зміни значень точності навчання нейронної мережі

Як видно з рисунків 4.2 і 4.3, похибка і точність для тестової вибірки весь час залишається на приблизно одному значенні. При навчанні на тренувальній виборці, похибка падає, а точність зростає, що говорить про перенавчання за допомогою розробленого методу навчання нейронних мереж. Для порівняння візьмемо глибшу модель з двома шарами, результати навчання такої мережі на рисунках 4.4 і 4.5.

Рисунок 4.4 – Графік зміни значень функції похибки начання нейронної мережі з двома шарами

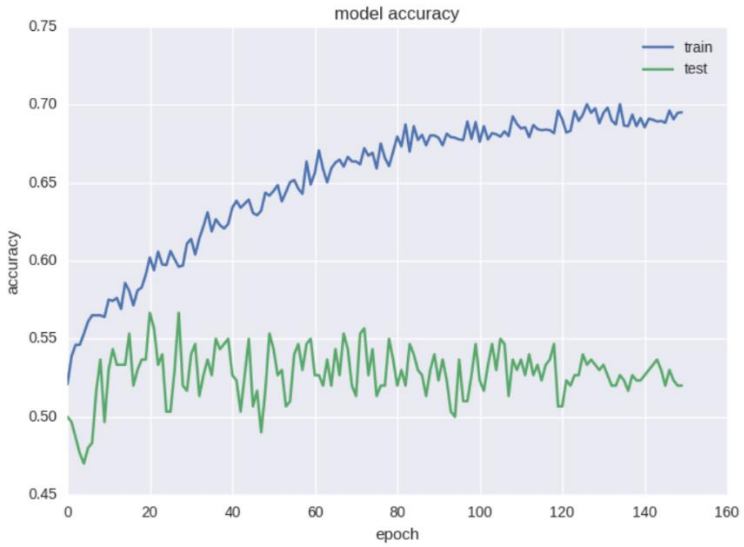

Рисунок 4.5 – Графік зміни значень точності навчання нейронної мережі з двома шарами

Для порівняння результатів, можна оцінити якість навчання на графіках зміни значень функції похибки та точності навчання нейронної мережі, які зображено на рисунках 4.6 і 4.7.

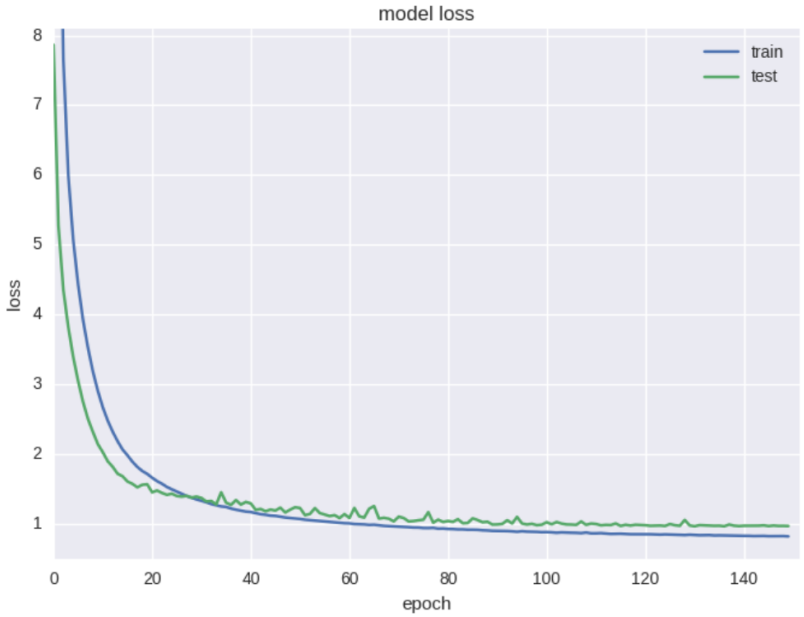

Рисунок 4.6 – Графік зміни значень функції похибки начання нейронної мережі з функцією регуляризації вагів нейронної мережі

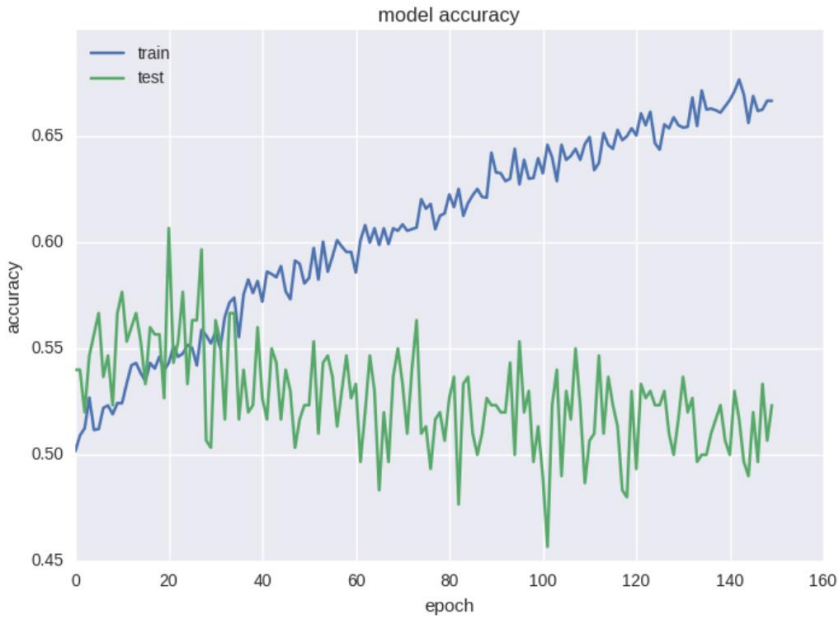

Рисунок 4.7 – Графік зміни значень точності навчання нейронної мережі з функцією регуляризації вагів нейронної мережі

Як показує тестування, графіки помилки і точності мають покращення результатів, якщо зупинити навчання мережі раніше, можно отримати меншеточності передбачення руху ціни.

Основний момент прогнозування фінансових часових рядів полягає в тому, що коливання при короткому терміні навчання має випадкову природу.

Спрогнозувавши рух ціни активів компанії після довгого періоду навчання, збільшенням вхідних чинників та збільшенням кількості шарів нейронної мережі, програма видає результати, що зображено на рисунках 4.8 та 4.9.

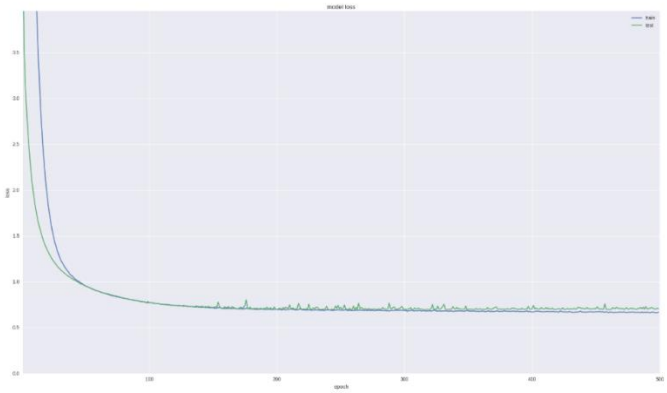

Рисунок 4.8 – Графік зміни значень функції похибки начання нейронної мережі

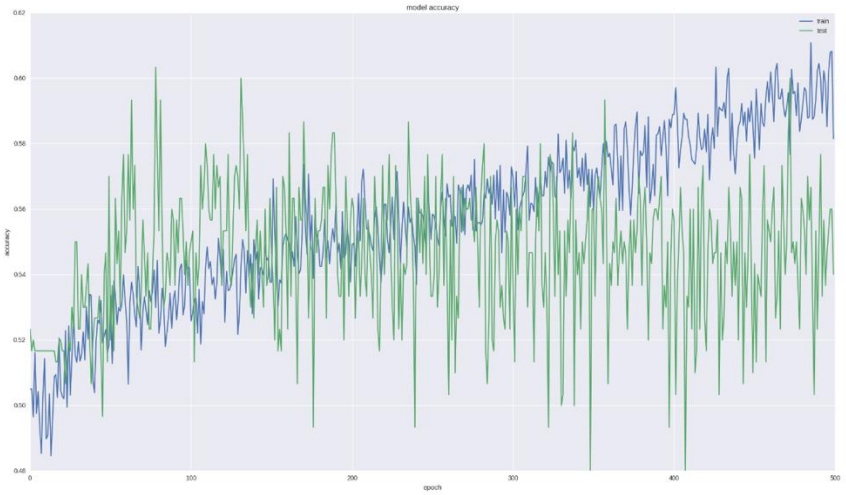

Рисунок 4.9 – Графік зміни значень точності навчання нейронної мережі

Якщо провести довгий період навчання, то можна отримати 90% точності, що є чудовим результатом.

Для задачі прогнозування візьмемо останню успішну архітектуру для класифікації і навчимо на більшій кількості ітерацій. Також в даному випадку можна дивитися вже не тільки на значення похибки, а й візуально оцінити якість прогнозування (рис. 4.10):

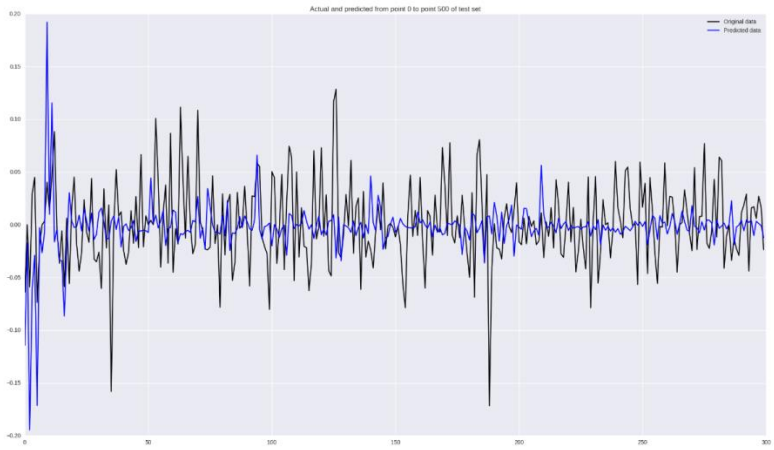

Рисунок 4.10 – Результати навчання нейронної мережі

Результати тестування підтвердили коректність роботи нейронної мережі прогнозування використання інвестицій.

4.3 Висновки

У розділі проаналізовано існуючі методи тестування таких як «білий ящик» та «чорний ящик». Тестування нейромережевої системи проведено функціональни методом, тобто текст програми не доступний, вона розглядається як «чорний ящик».

Тестування підтвердило ефективність та правильність функціонування розробленої нейронної мережі прогнозування ефективності використання інвестицій. Програмний продукт простий у використанні, має привабливий та зрозумілий інтерфейс.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть Ракитянська Г. Б. та Войтко В. В.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.1.

Таблиця 5.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Войтко В. В.	2. Ракитянська Г. Б.
	Бали, виставлені експертами:	
1	4	4
2	3	3
3	3	4
4	4	3
5	3	4
6	4	4
7	3	3
8	4	4
9	4	3
10	4	3
11	3	4
12	3	4
Сума балів	$СБ_1 = 43$	$СБ_2 = 43$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 43$	

Отже, з отриманих даних таблиці 5.1 видно, що нова розробка має високий рівень комерційного потенціалу.

5.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (5.1):

$$З_о = \frac{М}{Т_р} \cdot t, \quad (5.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р$ = 22 дні;

t - число днів роботи розробника, t = 50 днів.

Розрахунки заробітних плат для керівника і програміста наведені в табл.5.2.

Таблиця 5.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	6000	272,72	5	1363,5
Інженер-програміст	3400	154,54	50	7727
Всього:				9090,5

Розрахуємо додаткову заробітну плату:

$$З_{дод} = 0,1 \cdot 9090,5 = 909,05 \text{ (грн.)}$$

Нарахування на заробітну плату операторів $Н_{ЗП}$ розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати (вираз 5.2):

$$Н_{зп} = (З_о + З_р) \cdot \frac{\beta}{100}, \quad (5.2)$$

$$Н_{зп}= (9090,5+909,05)\cdot \frac{36,3}{100} = 3629,83 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою (5.3):

$$А=\frac{Ц\cdot Н_а}{100}\cdot \frac{Т}{12}, \quad (5.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 5.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний комп’ютер	10000	25	3	625
Всього:				625

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою (5.4):

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (5.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 5.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	180	1	180
Пачка паперу	уп.	130	1	130
Ручка	шт.	10	1	10
Всього з урахуванням транспортних витрат				352

Витрати на силову електроенергію розраховуються за формулою (5.5):

$$В_е=В\cdot П\cdot Ф\cdot К_п\ ; \quad (5.5)$$

де В – вартість 1кВт-години електроенергії (В=1,7 грн/кВт);

П – установлена потужність комп'ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=200 год.);

$К_п$ – коефіцієнт використання потужності ($К_п< 1$, $К_п = 0,7$).

$$В_е= 1,7 \cdot 0,6\cdot 200\cdot 0,7 = 142,8 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто (5.6):

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (5.6)$$

Отже, розрахуємо інші витрати:

$В_{ін} = 1 * (9090,5 + 909,05) = 9999,55$ (грн).

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи (5.7):

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в \quad (5.7)$$

В = 9090,5+909,05+3629,83+625+352+142,8+9999,55= 24748,73 (грн.)

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою (5.8):

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (5.8)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{24748,73}{1} = 24748,73$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою (5.9):

$$ЗВ = \frac{В_{заг}}{\beta} \quad (5.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{24748,73}{0,9} = 27498,58 \text{ (грн.)}$$

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

Спрогнозуємо отримання прибутку від реалізації результатів даної розробки. Зростання чистого прибутку можна оцінити у теперішній вартості

грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою (5.10):

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \qquad (5.10)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

Δ N – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 25 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 25 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 150 користувачів, протягом другого року – на 125 користувачів, протягом третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 500 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 400 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 25 \cdot 500 + (400 + 25) \cdot 150 = 76250 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 25 \cdot 500 + (400 + 25) \cdot (150 + 125) = 129375 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 25 \cdot 500 + (400 + 25) \cdot (150 + 125 + 100) = 171875 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$ вкладених інвестицій розраховується за формулою (5.11):

$$Е_{абс} = (ПП - PV), \quad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 5.1.

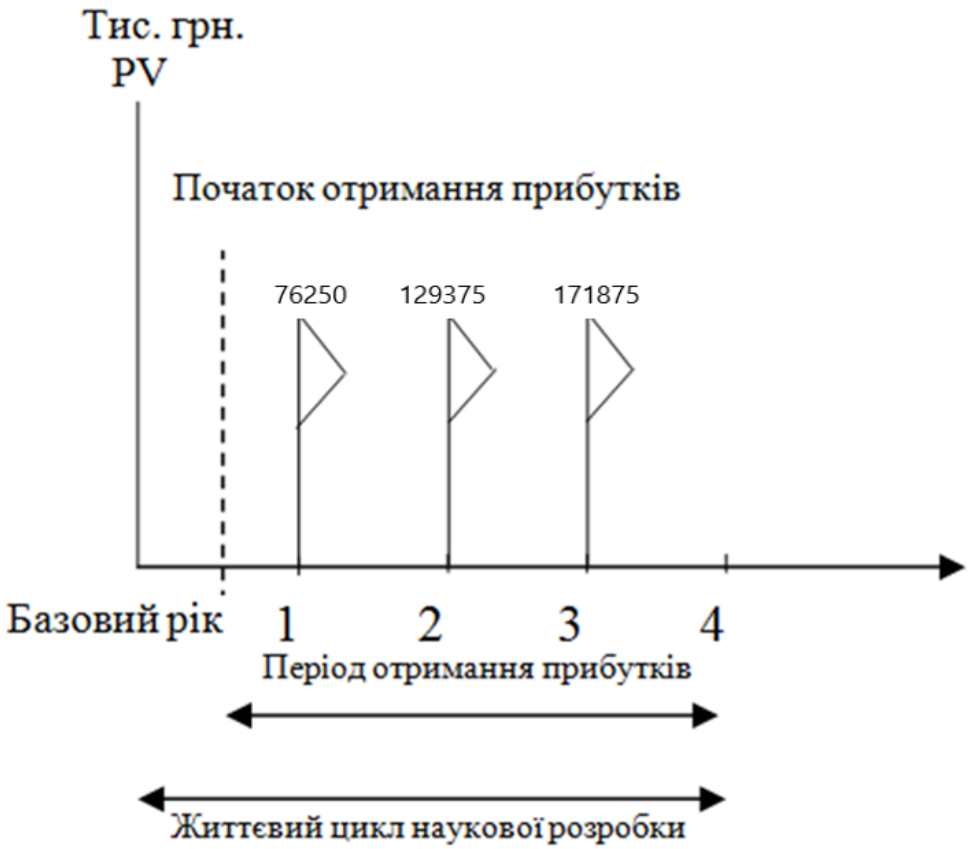

Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою (5.12):

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \quad (5.12)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{27498,58}{(1+0,1)^0} + \frac{76250}{(1+0,1)^2} + \frac{129375}{(1+0,1)^3} + \frac{171875}{(1+0,1)^4} = 305109,38 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 305109,38 - 27498,58 = 277610,8 \text{ грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою (5.13):

$$Е_в = \sqrt[Т]{1 + \frac{Е_{абс}}{PV}} - 1 \qquad (5.13)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$Е_в = \sqrt[3]{1 + \frac{277610,8}{27498,58}} - 1 = 1,23 \text{ або } 123\ \%$$

Далі, розраховану величина $Е_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою (5.14):

$$\tau = d + f, \qquad (5.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3$$

Оскільки $E_в$ = 123% > $\tau_{мін}$ = 0,3 = 30%, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою (5.15):

$$T_{ок} = \frac{1}{E_в} \quad (5.15)$$

$$T_{ок} = \frac{1}{1{,}23} = 0{,}81 \text{ року}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

5.5 Висновки

У п’ятому розділі було проведено оцінювання комерційного потенціалу розробки, виконано технологічний аудит для оцінювання комерційного потенціалу розробки. Залучено 2-х незалежних експертів. Спрогнозовано витрати на виконання науково-дослідної роботи та конструкторсько-технологічної роботи.

На вісі часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР показано життєвий цикл наукової розробки, початок отримання прибутків та його період.

Вартість чистого прибутку складає 305109,38 (грн.). Абсолютна ефективність вкладених інвестицій: 277610,8 грн. Відносна ефективність вкладених інвестицій: 1,23 або 123 %. Термін окупності: 0,81 року. Обрахувавши вартість чистого прибутку, абсолютну і відносну ефективність вкладених інвестицій та термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

ВИСНОВКИ

У магістерській кваліфікаційній роботі розроблено нейронну мережу прогнозування ефективності використання інвестицій. Було проаналізовано стан даної проблеми на сьогоднішній день. Вирішено, що розробка є актуальною, через постійне збільшення інвестиційних вкладів.

Для того, щоб розробити нейромережу було проаналізовано популярні аналоги нейромереж прогнозування, визначено їх переваги та недоліки. Проведено варіантний аналіз засобів реалізації нейронних мереж та обрано мову програмування Python з використанням бібліотеки Keras.

Були проаналізовані принципи розробки та навчання нейромереж і обрано радіально-базисну нейромережу з глибинним навчанням. Здійснено постановку задач дослідження.

Розроблено метод та модель навчання нейромереж для задач прогнозування.

Розроблено метод та модель прогнозування використання інвестицій на базі нейронних мереж.

Проведене тестування підтвердило коректність роботи нейронної мережі прогнозування використання інвестицій.

Проведено оцінювання комерційного потенціалу розробки та проведено розрахунки, які підтверджують економічну доцільність розробки програмного продукту.

Продукт вирішує проблему підвищення продуктивності методів прогнозування в системах оцінки інвестиційної привабливості за рахунок використання багатошарових радіально-базисних нейронних мереж.

Розроблена нейронна мережа проводить прогнозування ефективності використання інвестицій, яка орієнтована на прогнозування інвестування, зі зменшенням ризиків, що дозволяє користувачеві слідкувати за можливим станом інвестицій до моменту залучення коштів.

За результатами дослідження було зроблено наукову публікацію.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Штучна нейронна мережа [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Штучна_нейронна_мережа

2. Войтко В.В. Розробка нейромережевих методів і моделей прогнозування ефективності використання інвестицій / В.В. Войтко, С.В. Бевз. С.М. Бурбело, Л.М. Круподьорова, В.М. Задорожний // матеріали Всеукраїнської науково-практичної Інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи - 2019» [Електронний ресурс]. – Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2020/schedConf/presentations

3. Simon Haykin . Neural Network a comprehensive foundation(2nd edition) / Simon Haykin - Prentice Hall, 842 pages, 2013

4. Борисов Ю.И. Нейросетевые методы обработки информации и средства их программно-аппаратной поддержки / Борисов Ю.И, Кашкаров В.М., Сорокин С.А. // Открытые системы. – 1997.– № 4. – С. 38 – 40.

5. Горбань А.Н., Россиев Д.А. Нейронные сети на персональном компьютере. / А.Н. Горбань, Д.А. Россиев – Н.: Наука, 2006. – 276с.

6. Горбань А.Н. Нейронные сети на персональном компьютере / А.Н. Горбань, Д.А. Россиев. – Новосибирск: Наука, 2006. – 230 с.

7. Комашинский В.И. Смирнов Д.А. Внедрение в нейро-информационные технологии. / В.И. Комашинский, Д.А. Смирнов - СПб., 1999.

8. Hong S.G., Kim S.W. and Lee J.J., 2015. The Minimum Cost Path Finding Algorithm Using a Hopfield Type Neural Network, Proceedings IEEE International Conference on Fuzzy Systems 4, 719–726.

9. Лисе А.А., Степанов М.В. Нейронные сети и нейрокомпьютеры. / А.А. Лисе, М.В. Степанов // Учеб. пособие. ГЭТУ. - СПб., 2009. 64 с.

10. Anfis [Електронний ресурс]. – Режим доступу: https://infostart.ru/public/649065/

11. MICEX [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/396505/

12. Тархов Д.А. Нейронные сети. Модели и алгоритмы. – М.: Радиотехника, 2010. – 82 с.

13. Царегородцев В.Г. Перспективы программ нейросетевого анализа и обработки данных / В.Г. Царегородцев // Материалы III Всерос. конф. «Математика, информатика, управление – 2008». – Иркутск, 2014. – С. 110-117.

14. Метод решения проблемы переобучения в нейронных сетях [Електронний ресурс]. – Режим доступу: https://habr.com/ru/company/wunderfund/blog/330814/.

15. Галушкин А.И. Теория нейронных сетей. / А.И. Галушкин - М.: ИПРЖР, 2010. – 348 с.

16. Боголюбов Д. П., Чанкин А. А., Стемиковская К. В. Реализация алгоритма обучения самоорганизующихся карт Кохонена на графических процессорах// Промышленные АСУ и контроллеры. 2012. № 10. С. 30-35

17. Barbara Chapman, Gabriele Jost, Ruud van der Pas. Using OpenMP: portable shared memory parallel programming (Scientific and Engineering Computation). Cambridge, Massachusetts: The MIT Press., 2008. - 353 pp.

18. Нейромережеве відображення дійсності. [Електронний ресурс]. – Режим доступу: http://studies.in.ua/mpd_seminar/1313-neyrmerezh.html Моделі нейронних мереж. [Електронний ресурс]. – Режим доступу:: https://studme.com.ua/1246122010028/neural/models.htm

19. R. Kruse, C. Borgelt, F. Klawonn, C. Moewes, M. Steinbrecher, P.Held, P. Berlin Computational Intelligence. A Methodological Introduction. Springer, 2013.

20. R.J. Schilling, J.J. Carrol, A.F. Al-Ajlouni. Approximation of nonlinear systems with radial basis function neural networks. IEEE Trans. on Neural Networks. – 2001.

21. R.J. Schilling, J.J. Carrol, A.F. Al-Ajlouni. Approximation of nonlinear systems with radial basis function neural networks. IEEE Trans. on Neural Networks. – 2001. – 12.

22. Ye. Bodyanskiy, N. Teslenko. Generalized regression neuro-fuzzy network. Information Research and Applications i.TECH-2007: Proc. Fifth Int. Conf.- Varna, 2007. – V.1.

23. Mladen Dalto, Jadranko Matusko, Mario Vasak Deep neural networks for time series prediction with applications in ultra-short-term wind forecasting. — IEEE Internatiol Industrial Technology, Seville, Spain, March 17–19 th, 2015.

24. Martin L¨angkvist, Lars Karlsson, Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. Pattern Recognition Letters, V. 42, 1 June 2014.

25. А. А. Ежов, С. А. Шумский. Нейрокомпьютинг и его применения в экономике и бизнесе /— М.: МИФИ, 1998.

26. С. А. Терехов. Гениальные комитеты умных машин. IX Всероссийская научно-техническая конференция «Нейроинформатика-2007»: Лекции по нейроинформатике. Часть 2, М., МИФИ, 2007.

27. Mladen Dalto, Jadranko Matusko, Mario Vasak Deep neural networks for time series prediction with applications in ultra-short-term wind forecasting. — IEEE Internatiol Industrial Technology, Seville, Spain, March 17–19 th, 2015.

28. Martin L¨angkvist, Lars Karlsson, Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. Pattern Recognition Letters, V. 42, 1 June 2014.

29. Keras: The Python Deep Learning library [Електронний ресурс]. – Режим доступу: https://keras.io/.

30. Принципы тестирования программного обеспечения. [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/330746/.

ДОДАТКИ

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
"____ "_____2019 р.

Технічне завдання
на магістерську кваліфікаційну роботу «Розробка нейромережевих методів і моделей прогнозування ефективності використання інвестицій»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
_____ к.т.н., доц. Г. Б. Ракитянська
"____"_____2019 р.
Виконав:
_____студент гр.1ПІ-18м В. М. Задорожний
"____"_____2019 р.

Вінниця – 2019 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка нейромережевих методів і моделей прогнозування ефективності використання інвестицій».

Галузь застосування – системи грогнозування числових рядів.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення продуктивності методів прогнозування в системах оцінки інвестиційної привабливості за рахунок використання багатошарових нейронних мереж.

Призначення роботи – прогнозування інвестування, зі зменшенням ризиків, що дозволяє користувачеві слідкувати за можливим станом інвестицій до моменту залучення коштів до моменту залучення коштів.

3 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Борисов Ю.И. Нейросетевые методы обработки информации и средства их программно-аппаратной поддержки / Борисов Ю.И, Кашкаров В.М., Сорокин С.А. // Открытые системы. – 1997.– № 4. – С. 38 – 40.
2. Тархов Д.А. Нейронные сети. Модели и алгоритмы. – М.: Радиотехника, 2010. – 82 с.
3. Martin L¨angkvist, Lars Karlsson, Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. Pattern Recognition Letters, V. 42, 1 June 2014.

4. Технічні вимоги

Мова програмування: Python;

Нейромережева бібліотека: Keras;

Вид нейромережі: радіально-базисна мережа;

Метод навчання: глибинне навчання.

5. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Техніко-економічне обґрунтування доцільності розробки нейронної мережі прогнозування	30.09.2019 – 13.10.2019
2	Розробка методів та моделей нейронної мережі прогнозування	14.10.2019 – 27.10.2019
3	Програмна реалізація нейронної мережі	27.10.2019 – 17.11.2019
4	Тестування роботи нейронної мережі	18.11.2019 – 24.11.2019
5	Економічне обґрунтування розробки програмного продукту	25.11.2019 – 01.12.2019

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б. Лістинг програми

Login.java

```
import matplotlib.pylab as plt
import numpy as np
import pandas as pd
# обучающая выборка входных и выходных данных
X = np.array([[3,5],[5,1],[10,2]])
y = np.array([[75, 82, 93]]).T   # T-транспонирование матрицы
# нормализация данных - делим на максимальное из них число
X = X / np.amax(X, axis = 0)
y = y / 100
# инициализация случайных весов (от -1 до +1) синапсов
np.random.seed(1)
synapses_hidden = 2 * np.random.random((2,3)) - 1  # 2*3 - hidden слой
synapses_output = 2 * np.random.random((3,1)) - 1  # 3*1 - output слой
# обучение сети - цикл из 10000 повторений
for j in range(10000):
    # Входной слой ( 2 входа )
    l0 = X
    # Скрытый слой ( 3 скрытых нейрона )
    l1 = 1 / (1 + np.exp(-(l0.dot(synapses_hidden))))
    # Выходной слой ( 1 выходной нейрон )
    l2 = 1 / (1 + np.exp(-(l1.dot(synapses_output))))
    # вычисляем ошибку (используем дельта-правило)
    l2_delta = (y - l2) * (l2 * (1 - l2))
    # получаем ошибку на скрытом слое (используем дельта-правило)
    l1_delta = l2_delta.dot(synapses_output.T) * (l1 * (1 - l1))
    # корректируем веса от скрытых нейронов к выходу
    synapses_output += l1.T.dot(l2_delta)
    # корректируем веса от входов к скрытым нейронам
    synapses_hidden += l0.T.dot(l1_delta)
# Печать сигналов на выходе после последнего цикла обучения
# Сигналы на выходе умножаем на коэффициент нормализации (100)
print(l2 * 100));
from numpy import exp, array, random, dot
training_set_inputs = array([[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]])
training_set_outputs = array([[0, 1, 1, 0]]).T
random.seed(1)
synaptic_weights = 2 * random.random((3, 1)) - 1
for iteration in range(10000):
    output = 1 / (1 + exp(-(dot(training_set_inputs, synaptic_weights))))
    synaptic_weights += dot(training_set_inputs.T, (training_set_outputs - output) * output * (1 - output))
print (1 / (1 + exp(-(dot(array([1, 0, 0]), synaptic_weights)))))
from numpy import exp, array, random, dot
data = pd.read_csv('./data/AAPL.csv')[::-1]
close_price = data.ix[:, 'Adj Close'].tolist()
plt.plot(close_price)
plt.show()
model = Sequential()
model.add(Dense(64, input_dim=30))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(2))
model.add(Activation('softmax'))
```

```
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.9, patience=5, min_lr=0.000001, verbose=1)
model.compile(optimizer=opt,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
class NeuralNetwork():
    def __init__(self):
        random.seed(1)
        self.synaptic_weights = 2 * random.random((3, 1)) - 1
    def __sigmoid(self, x):
        return 1 / (1 + exp(-x))

    def __sigmoid_derivative(self, x):
        return x * (1 - x)

    def train(self, training_set_inputs, training_set_outputs, number_of_training_iterations):
        for iteration in range(number_of_training_iterations):
            output = self.think(training_set_inputs)
            error = training_set_outputs - output
            adjustment = dot(training_set_inputs.T, error * self.__sigmoid_derivative(output))
            self.synaptic_weights += adjustment

    def think(self, inputs):
        return self.__sigmoid(dot(inputs, self.synaptic_weights))

if __name__ == "__main__":

    #Intialise a single neuron neural network.
    neural_network = NeuralNetwork()

    print ("Random starting synaptic weights: ")
    print (neural_network.synaptic_weights)

    training_set_inputs = array([[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]])
    training_set_outputs = array([[0, 1, 1, 0]]).T

    # Train the neural network using a training set. Do it 10,000 times.
    neural_network.train(training_set_inputs, training_set_outputs, 10000)

    print ("New synaptic weights after training: ")
    print (neural_network.synaptic_weights)

    # Test the neural network with a new situation.
    print ("Considering new situation [1, 0, 0] -> ?: ")
    print (neural_network.think(array([1, 0, 0])))
import numpy as np

# sigmoid function
def nonlin(x,deriv=False):
    if(deriv==True):
        return x*(1-x)
    return 1/(1+np.exp(-x))

# input dataset
X = np.array([ [0,0,1],
               [0,1,1],
               [1,0,1],
               [1,1,1] ])
```

```
# output dataset
y = np.array([[0,0,1,1]]).T

np.random.seed(1)
syn0 = 2*np.random.random((3,1)) - 1

for i in range(10000):
    l0 = X
    l1 = nonlin(np.dot(l0,syn0))
    l1_error = y - l1
    l1_delta = l1_error * nonlin(l1,True)
    # update weights
    syn0 += np.dot(l0.T,l1_delta)

print ("Output After Training:")
print (l1)
import numpy as np

def nonlin(x,deriv=False):
    if(deriv==True):
        return x*(1-x)
    return 1/(1+np.exp(-x))

X = np.array([[0,0,1],
              [0,1,1],
              [1,0,1],
              [1,1,1]])
y = np.array([[0],
              [1],
              [1],
              [0]])

np.random.seed(1)
syn0 = 2*np.random.random((3,4)) - 1
syn1 = 2*np.random.random((4,1)) - 1

for j in range(60000):
    l0 = X
    l1 = nonlin(np.dot(l0,syn0))
    l2 = nonlin(np.dot(l1,syn1))
    l2_error = y - l2
    if (j% 10000) == 0:
        print ("Error:" + str(np.mean(np.abs(l2_error))))
    l2_delta = l2_error*nonlin(l2,deriv=True)
    l1_error = l2_delta.dot(syn1.T)
    l1_delta = l1_error * nonlin(l1,deriv=True)

    syn1 += l1.T.dot(l2_delta)
    syn0 += l0.T.dot(l1_delta)
function iwebdc_team_post_type()
{
        $labels = array(
                'name'              => _x( 'Teams', 'post type general name', 'iwebdc' ),
                'singular_name'     => _x( 'Team', 'post type singular name', 'iwebdc' ),
                'menu_name'         => _x( 'Teams', 'admin menu', 'iwebdc' ),
                'name_admin_bar'    => _x( 'Team', 'add new on admin bar', 'iwebdc' ),
                'add_new'           => _x( 'Add New', 'team', 'iwebdc' ),
```

```
		'add_new_item'      => __( 'Add New Team', 'iwebdc' ),
		'new_item'          => __( 'New Team', 'iwebdc' ),
		'edit_item'         => __( 'Edit Team', 'iwebdc' ),
		'view_item'         => __( 'View Team', 'iwebdc' ),
		'all_items'         => __( 'All Teams', 'iwebdc' ),
		'search_items'      => __( 'Search Teams', 'iwebdc' ),
		'parent_item_colon' => __( 'Parent Teams:', 'iwebdc' ),
		'not_found'         => __( 'No teams found.', 'iwebdc' ),
		'not_found_in_trash' => __( 'No teams found in Trash.', 'iwebdc' )
	);

	$args = array(
		'labels'            => $labels,
		'public'            => false,
		'exclude_from_search'=> true,
		'show_ui'           => true,
		'show_in_menu'      => true,
		'show_in_nav_menus' => false,
		'query_var'         => true,
		'rewrite'           => array( 'slug' => 'team' ),
		'capability_type'   => 'post',
		'has_archive'       => false,
		'hierarchical'      => false,
		'menu_position'     => null,
		'supports'          => array('title', 'excerpt', 'editor', 'thumbnail', 'page-attributes')
	);

	register_post_type( 'wdc_team' , $args );

	$labels = array(
		'name'              => _x( 'Categories', 'taxonomy general name', 'iwebdc' ),
		'singular_name'     => _x( 'Category', 'taxonomy singular name', 'iwebdc' ),
		'search_items'      => __( 'Search Categories', 'iwebdc' ),
		'all_items'         => __( 'All Categories', 'iwebdc' ),
		'parent_item'       => __( 'Parent Category', 'iwebdc' ),
		'parent_item_colon' => __( 'Parent Category:', 'iwebdc' ),
		'edit_item'         => __( 'Edit Category', 'iwebdc' ),
		'update_item'       => __( 'Update Category', 'iwebdc' ),
		'add_new_item'      => __( 'Add New Category', 'iwebdc' ),
		'new_item_name'     => __( 'New Category Name', 'iwebdc' ),
		'menu_name'         => __( 'Categories', 'iwebdc' ),
	);

	$args = array(
		'public'            => false,
		'hierarchical'      => false,
		'labels'            => $labels,
		'show_ui'           => true,
		'show_admin_column' => true,
		'show_in_nav_menus' => false,
		'query_var'         => false,
		'rewrite'           => array( 'slug' => 'team-category' ),
	);

	register_taxonomy( 'wdc_team_cat', array( 'wdc_team' ), $args );
}
add_action('init', 'iwebdc_team_post_type');
```

```
function iwebdc_team_edit_columns($columns)
{
	$newcolumns = array(
		'cb' => '<input type="checkbox" />',
		'title' => 'Title',
		'team-thumb' => esc_html__('Thumbnail', 'iwebdc'),
		'team-position' => esc_html__('Position', 'iwebdc')
	);

	$columns= array_merge($newcolumns, $columns);

	return $columns;
}
add_filter('manage_edit-wdc_team_columns', 'iwebdc_team_edit_columns');

function iwebdc_team_custom_columns($column)
{
	global $post;

	switch ($column) {
		case 'team-thumb':
			if ( has_post_thumbnail() ) {
				the_post_thumbnail('thumbnail', array('style' => '
			} else {
				echo '<span aria-hidden="true">&#8212;</span>';
			}
		break;
		case 'team-position':
			$team_position = get_post_meta($post->ID, '_wdc_team_position', TRUE);
			if ( !empty($team_position) ) {
				echo '<span>'. esc_attr( $team_position ) .'</span>';
			}
			else {
				echo '<span aria-hidden="true">&#8212;</span>';
			}
		break;
	}
}
add_action('manage_posts_custom_column', 'iwebdc_team_custom_columns', 10, 2);
function iwebdc_team_metaboxes( array $meta_boxes ) {

	// Start with an underscore to hide fields from custom fields list
	$prefix = '_wdc_team_';

	$meta_boxes[] = array(
		'id'         => 'team_metabox',
		'title'      => 'Team Attributes',
		'pages'      => array( 'wdc_team' ), // Post type
		'context'    => 'normal',
		'priority'   => 'default',
		'show_names' => true, // Show field names on the left
		'fields'     => array(
			array(
				'name' => esc_html__('Position', 'iwebdc'),
				'desc' => esc_html__('Enter position name in your company.', 'iwebdc'),
				'id'   => $prefix .'position',
```

```
                                'type' => 'text_medium',
                        ),
                        array(
                                'name' => esc_html__('Phone', 'iwebdc'),
                                'desc' => '',
                                'id'   => $prefix .'phone',
                                'type' => 'text_medium',
                        ),
                        array(
                                'name' => esc_html__('Fax', 'iwebdc'),
                                'desc' => '',
                                'id'   => $prefix .'fax',
                                'type' => 'text_medium',
                        ),
                        array(
                                'name' => esc_html__('Email Address', 'iwebdc'),
                                'desc' => '',
                                'id'   => $prefix .'email',
                                'type' => 'text_medium',
                        ),
                        array(
                                'name' => esc_html__('Social Icons', 'iwebdc'),
                                'desc' => '',
                                'id'   => $prefix .'social',
                                'type' => 'wdc_social_icons',
                        ),
                ),
        );

        return $meta_boxes;
}
add_filter( 'cmb_meta_boxes', 'iwebdc_team_metaboxes' );
history = model.fit(X_train, Y_train,
        nb_epoch = 50,
        batch_size = 128,
        verbose=1,
        validation_data=(X_test, Y_test),
        shuffle=True,
        callbacks=[reduce_lr])
plt.figure()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='best')
plt.show()

plt.figure()
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel(‘acc’)
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='best')
plt.show()
```

Додаток В. Ілюстративний матеріал

ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДО ЗАХИСТУ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Завідувач кафедри ПЗ, д. т. н., професор _____ О. Н. Романюк

Науковий керівник, к. т. н., доцент кафедри ПЗ_____ Г. Б. Ракитянська

Рецензент, к.т.н, доцент кафедри КН___ _____ Л. В. Крилик

Нормоконтроль, д. т. н., проф. кафедри ПЗ _____ Г. Б. Ракитянська

Виконавець, студент групи 1ПІ-18м _____ В. М. Задорожний