МІНІСТЕРСТВО ОСВІТИ НАУКИ УКРАЇНИ

СТРУКТУРНИЙ ПІДРОЗДІЛ «ФАХОВИЙ КОЛЕДЖ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРАТ «ПВНЗ «ЗІЕІТ»

Циклова комісія з інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА

Голова циклової комісії,

спеціаліст в/к

_____ С.О. Сабанов

ВИПУСКНА РОБОТА МОЛОШОГО СПЕЦІАЛІСТА

ТЕМА «ОБЛІКОВА СИСТЕМА ФІНАНСОВОЇ ЗВІТНОСТІ НА МОВІ PYTHON»

Виконав

ст. гр. ІПЗ – 118к9 _____ Д.В. Булавін

Керівник

Викладач _____ Д.О. Костерной

Запоріжжя

2022

СТРУКТУРНИЙ ПІДРОЗДІЛ «ФАХОВИЙ КОЛЕДЖ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРАТ «ПВНЗ «ЗІЕІТ»

Циклова комісія з інформаційних технологій

ЗАТВЕРДЖУЮ

Голова циклової комісії, спеціаліст в/к

_____ С.О. Сабанов

17 січня 2022 р.

ЗАВДАННЯ

ВИПУСКНОЇ РОБОТИ МОЛОДШОГО СПЕЦІАЛІСТА

Студента гр. *ІПЗ-118К9*

Спеціальності: *121 – Інженерія програмного забезпечення*

Булавіну Дмитру Віталійовичу

(прізвище, ім'я, по батькові)

1. Тема: «*Облікова система фінансової звітності на мові Python»*

затверджена наказом по інституту: № 09.2 від 04 березня 2022 року

2. Термін здачі студентом закінченої роботи: 18 червня 2022 року

3. Перелік питань, що підлягають розробці:

1. Зібрати літературу та документацію присвячену тематиці випускної

роботи;

2. Провести бесіду з керівництвом підприємства з приводу

випускної роботи та впровадження як розробку;

3. Розглянути та проаналізувати аналоги;

4. Зробити аналіз апаратного обладнання;

5. Виконати всі поставлені задачі випускної роботи;

6. Протестувати розробку та впровадити перед керівником підприємства;

7. Оформити звіт за результатами роботи

4. Календарний графік підготовки випускної роботи молодшого спеціаліста

№ етапу	Зміст	Терміни виконання	Готовність по графіку %, підпис керівника	Підпис керівника про повну готовність етапу, дата
1	Формулювання (корегування) теми випускної роботи молодшого спеціаліста та збір практичного матеріалу за темою випускної роботи	17.01.22-26.02.22		
2	**І атестація** І розділ випускної роботи молодшого спеціаліста	28.03.22-02.04.22		
3	**ІІ атестація** ІІ розділ випускної роботи молодшого спеціаліста	10.05.22-14.05.22		
4	**ІІІ атестація** ІІІ розділ випускної роботи молодшого спеціаліста, висновки та рекомендації, додатки, реферат	30.05.22-04.06.22		
5	Перевірка випускної роботи молодшого спеціаліста програмою «Антиплагіат»	30.05.22-18.06.22		
6	**Доопрацювання випускної роботи молодшого спеціаліста, підготовка презентації, отримання відгуку керівника і рецензії**	06.06.22-11.06.22		
7	**Попередній захист випускної роботи молодшого спеціаліста**	**14.06.22-18.06.22**		
8	Подача випускної роботи молодшого спеціаліста на кафедру	**за 3 дні до захисту**		
9	**Захист випускної роботи молодшого спеціаліста**	**20.06.22-25.06.22**		

Керівник _____ Д.О. Костерной
(підпис) (ініціали, прізвище)

«____» _____ 2022 р.

Завдання отримав до виконання _____ Д.В. Булавін
(підпис студента) (ініціали, прізвище)

«____» _____ 2022 р.

РЕФЕРАТ

Випускна робота молодшого спеціаліста: 62 сторінка, 30 рисунків, 1 додаток, 28 джерел, 7 формул, 3 лістинга.

Об'єкт розробки: нейронна мережа для прогнозування часових рядів.

Мета роботи полягає в розробці моделі нейронної мережі для аналізу фінансових результатів. Згідно отриманих результатів, а також згідно алгоритму навчання, на основі цих даних виродження фінансової звітності.

Завдання, які треба вирішити в роботі:

1. Виконати огляд мов програмування та середовищ розробки для створення нейронної мережі;
2. Проаналізувати функціональні можливості нейронної мережі;
3. Створити нейронну мережу для прогнозування часових рядів;
4. Тестування нейронної мережі на точність прогнозування;
5. Написати випускну роботу молодшого спеціаліста по даній темі та зробити висновок.

Результатом роботи є створення нейронної мережі для прогнозування часових рядів.

MONOGODB, PYTHON, TENSORFLOW, PYCHARM.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Скорочення	Словосполучення	Пояснення/переклад
БД	База даних	
ООП	Об’єктно – орієнтоване програмування	
RNN	Recurrent neural network	Рекурентна нейронна мережа
CNN	Convolutional neuralnetwork	Згорткова нейронна мережа
DB	Database	
JSON	JavaScript Object Notation	Текстовий формат обміну даними, що базується на JavaScript
SQL	Standartized Query Language	Декларативна мова програмування
UML	Unified Modeling Language	Уніфікована мова моделювання
URI	Uniform Resource Identifier	Уніфікований ідентифікатор ресурсу
CSS	Cascading Style Sheets	Каскадні таблиці стилів

ЗМІСТ

ВСТУП

Актуальність теми

Тема фінансових показників завжди буде актуальною, зокрема, на біржовому та фондовому ринці, оскільки саме ці показники відображають доцільність інвестування для покупки цінних бумаг. Часові ряди використовуються у багатьох сферах життя людини. Виходячи з цього - є велика потрібність та актуальність в аналізу цих самих показників. Оскільки мова йде про часові ряди, тобто ряди які постійно змінюються з часом, то для постійного (динамічного) аналізу цих рядів, зокрема, фінансових показників, є актуальним використання нейронних мереж, адже з розвитком технології - головною проривною технологією став штучний інтелект, який набирає свою популярність во всіх сферах інформаційного життя людини у тому числі у галузі «data science», де потреба в аналізу даних тільки зростає. С подальшим розвитком штучного інтелекту - технологія стане невід'ємною частинної життя кожної людини, тому є актуальним використання нейронних мереж для створення фінансової звітності по часовим рядам для подальшого аналізу цих даних, тобто створення обліку.

Мета і задачі розробки

Мета роботи – розробити модель факторного аналізу фінансових результатів та проаналізувати існуючі нейронні мережі, їх можливість створення за технології «TensorFlow», а також проаналізувати можливість кореляції фактичних даних часових рядів за певний період зі прогнозуючими даними, таким чином виявити рентабельність цінних бумаг компаній, для подальшого інвестування.

Методи, засоби та технології розробки

Для створення нейронної мережі, яка аналізує дані, та на основі свого аналізу зможе прогнозувати дані – використано технологію «Ten-

sorFlow» для створення графу нейронної мережі та її тренування, середа розробки «PyCharm» для написання алгоритму тренування нейронної мережі на мові програмування «Python», база даних «MongoDB» для зберігання тренувальних даних для нейронної мережі.

Практичне значення одержаних результатів

Результатом виконання дипломної роботи є програма, яка використовує нейронну мережу для прогнозування часові рядів для подальшого аналізу цих рядів.

РОЗДІЛ 1
ТЕОРІТИЧНІ ВІДОМОСТІ

1.1 Дослідження основних понять

1.1.1 Часові ряди

Часовий ряд або ряд динаміки – це підібраний у різні моменти часу статистичний матеріал про значення будь-яких параметрів ряду. У тимчасовому ряді для кожної одиниці статистичного матеріалу має бути вказано час виміру або номер виміру по порядку. Тимчасовий ряд суттєво різниться від простої вибірки даних, оскільки при аналізі враховується взаємозв'язок статистичного матеріалу з часом, а не лише статистичне розмаїття та статистичні характеристики вибірки[2].

Аналіз часових рядів

Аналіз часових рядів - це сукупність математико-статистичних методів аналізу, призначених для виявлення структури часових рядів та їх прогнозування. До цього належить метод регресійного аналізу, тобто статистичного взаємозв'язку між однією залежною кількісною залежною змінною від однієї або кількох незалежних кількісних змінних, а також методи згладжування, - методи часових рядів призначених для акомодації до змін даних у часі, наприклад метод ковзного середнього, який створює ряд середніх шляхом взяття середніх значень у часовому ряду в межах заданих періодів. Визначення структури часового ряду необхідне у тому для того, щоб побудувати математичну модель аналізованого часового ряду. Часові ряди складаються з двох елементів: періоду часу, за який або станом на який наводяться числові значення та числових значень тієї чи іншої показника, званих рівнями низки.

Є дві головні цілі аналізу часових рядів: визначення природи ряду та прогнозування, тобто передбачення майбутніх значень тимчасового ряду за минулими значеннями.

Часові ряди класифікують:

1. За кількістю показників, тобто одновимірні, двомірні та багатовимірні часові ряди;

2. По наявності значень, бувають повні та неповні часові ряди;

3. За характером тимчасового параметра: моменті та інтервальні тимчасові ряди;

4. Залежно від наявності основної тенденції виділяють стаціонарні ряди (не змінної характеристикою з часом), у яких середнє значення та міра розкиду значень випадкової величини постійні, та нестаціонарні, що містять основну тенденцію розвитку.

У свою чергу прогнозування часових рядів це додання до даних минулих періодів, прогнозуючих даних, де ці значення ще доступні. Типовим прикладом часового ряду - біржовий курс, під час аналізу якого намагаються спрогнозувати основний напрямок розвитку курсу[1].

1.1.2 Нейронна мережа

Нейронна мережа – це парцептрон, що є спрощеним відображенням роботи біологічної мережі, яка складається з нейронів, з'єднаних між собою дендритами[5].

Класичним прикладом полнозв’язаної мережі прямого розповсюдження є мережа, яка складеться з декількох шарів, нейрон попереднього шару пов'язаний з кожним нейроном наступного шару. А сигнал поширюється від вхідного шару до вихідного. Між нейронами є зв’язок, який має певну «вагу». При проходженні по лініям – сигнал міняє своє значення відповідно до цієї ваги.

Сам собою нейрон з математичної суті – це суматор вхідних сигналів, який, потім, пропускає суму через активаційну функцію. Вихідне значення цієї функції є вихідне значення нейрона.

Активаційна функція є невід'ємною частиною при побудові нейронної мережі. В залежності від типу поставленої задачі, використовують різні активаційні функції, існують такі типи:

1. Порогова функція;
2. Лінійна функція;
3. Функція ReLU;
4. Сигмоїдна логістична функція;
5. Гіперболічний тангенс.

Порогова функція - це функції, яка пропускає сигнал через формулу:

$$f(x) = \begin{cases} 1, x \geq a \\ 0, x < a, \end{cases} \quad (1.1)$$

Ця функція є найпростішою, але має ряд недоліків, головним з них це значення яке відповідає двом класам – 100 відсотків та 0 відсотків.

Лінійна функція - функція яка є більш складною у реалізації, але і більш розповсюджена у використанні, має таку формулу:

$$f(x) = c * x, \quad (1.2)$$

c - коефіцієнт пропорційності

Через свою лінійність не має популярності у використанні при створенні нейронної мережі.

Функція ReLU - нелінійна функція яка має вигляд:

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0, \end{cases} \quad (1.3)$$

Ця функція є найпопулярнішою серед інших функцій, оскільки може апроксимувати всі інші функції, а також найкраще підходить для глибинних нейронних мереж.

Сигмоїдна логістична функція – функція яка використовується у задачах зі

класифікацією, має вигляд:

$$f(x) = \frac{1}{1+e^{-x}}, \tag{1.4}$$

Ця функція також часто використовується, видає значення діапазоном від -1 до 1 Гіперболічний тангенс – інший вид сигмоїдної функції, має формулу:

$$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1, \tag{1.5}$$

Має більший коефіцієнт у градієнті ніж сигмоїдна функція.

Нейрона мережева модель складеться за трьох шарів

Перший шар має назву вхідний, він приймає дані, пропускає через функцію активації, вихідні значення помножуються на ваги та проходять через потайний шар, знову функція активація, помноження, потім вихідних шар, який видає кінцевий результат. На мові програмуванні нейронну модель інтерпретують як граф на рисунку 1.1 – приклад нейронної моделі[3].

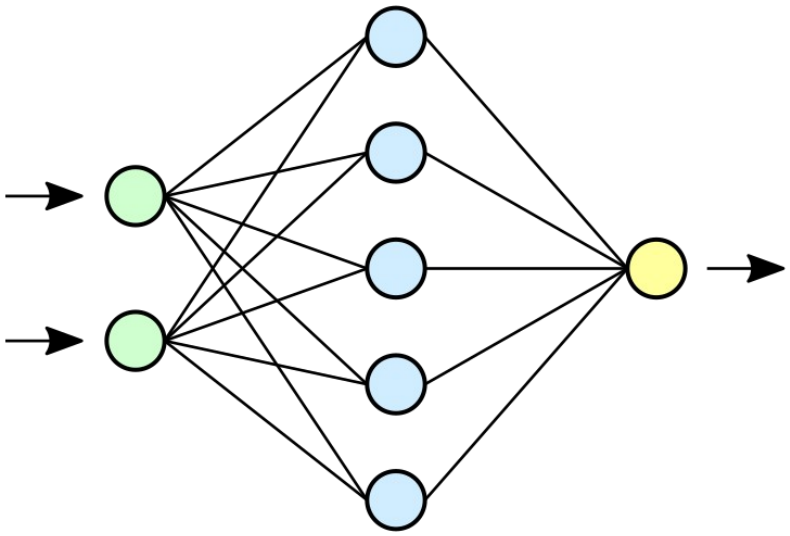

Рисунок 1.1 – Нейрона мережа

Штучні мережі застосовуються в таких областях, в яких потрібно виконувати роботу с великим обсягом даних, а також, де людський інтелект є малоефективним, попре це, штучний інтелект найкращим чином підходить для постійного аналізу даних, тобто для динамічного оновлення даних, таким чином є актуальне використання нейронної мережі у всіх сферах життя людини.

Нейронні мережі застосовуються у таких мережах: автоматизація прогнозування, автоматизація процесу класифікації, , автоматизація процесу розпізнавання, автоматизація процесу прийняття рішень; управління, кодування та декодування інформації; апроксимація залежностей та ін.

Компанія «Google» - перша компанія яка стала використовувати нейронні мережі у своїх додатків, так наприклад проводяться тести у галузі перекладання завдяки нейронної мережі[5].

Ще одна область - управління цінами та виробництвом (втрати від неоптимального планування виробництва часто недооцінюються). Оскільки попит та умови реалізації продукції залежать від часу, сезону, курсів валют та багатьох інших факторів, то і обсяг виробництва повинен гнучко варіюватися з метою оптимального використання ресурсів (нейрона мережа виявляє складні залежності між витратами на рекламу, обсягами продажів, ціною, цінами конкурентів, днем тижня, сезоном і т. д.). У результаті використання системи здійснюється вибір оптимальної стратегії виробництва з погляду максимізації обсягу продажу чи прибутку.

У медичній діагностиці нейронні мережі використовуються, наприклад, для діагностики слуху у немовлят. Система об'єктивної діагностики обробляє зареєстровані «викликані потенціали» (відгуки мозку), які у вигляді сплесків на електроенцефалограмі, у відповідь звуковий подразник, синтезований у процесі обстеження. Щоб почати діагностику слуху досвідченому аудіологу для цього необхідно провести аж до 1800 тестів, що займає близько двох годин. Система на основі нейронної мережі здатна з тією ж вірогідністю визначити рівень слуху вже по 200 спостереженнях протягом декількох хвилин, причому без участі кваліфікованого персоналу[3].

А також нейронні мережі широко використовуються для прогнозування довго-

строкових та/або короткострокових тенденцій у різних галузях (фінансовій, економічній, банківській та ін.).

Історія формування поняття «нейрона мережа» та способу її реалізації:

У 1943 - 1945 роках - У. Маккалок тп У. Піттс формалізують поняття нейронної мережі в фундаментальній статті про логічне обчислення ідей та нервової активності. На початку своєї співпраці з Піттсом Н. Вінер пропонує йому вакуумні лампи як засіб для реалізації еквівалентів нейронних мереж [5].

У 1948 році - опублікована наукова книга Н. Вінера про кібернетику. Основною ідеєю стало уявлення складних біологічних процесів математичними моделями та формулюваннями.

У 1949 році - канадський фізіолог та нейропсихолог. Д. Хебб пропонує перший алгоритм навчання, який припустив, що синоптичний зв'язок, що з'єднує два нейрони, буде посилюватися, якщо в процесі навчання обидва нейрони узгоджено відчувають збудження, або гальмування. Простий алгоритм, який реалізує такий механізм навчання, отримав назву правила Хебба. .

У 1958 році американський вчений у галузі психології, нейрофізіології та штучного інтелекту Ф. Розенблатт винаходить одношаровий перцептрон і демонструє його здатність вирішувати завдання класифікації.

У 1960 році Бернард Уідроу розробили «ADALINE» та «MADALINE», різниця між Адалін і стандартним нейроном Каллоха-Пітса полягає в тому, що у фазі навчання ваги ADALINE встановлюються відповідно до вагової суми входів формула 2.1, таким чином вихід спрощується до скалярного добутку. В стандартному перцептроні Розенблаттв на протязі етапу навчання - сигнали сітки надходять до передавальної функції, таким чином її вихід використовується для встановлення вагових коефіцієнтів.

$$y = \sum_{j=1}^{n} x_i * w_i, \tag{1.6}$$

У 1963 році в «Інституті проблем передачі інформації». Петровим проводи-

ться дослідження завдань «важких» для перцептрону. На цю роботу в галузі моделювання ІНС у СРСР спирався М. М. Бонгарда у своїй роботі як переробкою алгоритму для виправлення його недоліків».

У 1970 році Мінський анонсує доказ обмеженості алгоритму перцептрону Больцмана та показує його нездатність вирішувати деякі завдання, що пов'язані з інваріантністю уявлень.

У 1972 році Дж. Андерсон пропонує новий тип нейронних мереж, здатних працювати як пам'ять.

У 1973 році канадець Б. В. Хакімов запропонуваа нелінійну модель із синапсами, а також вводить її для вирішення задач у галузі медицині.

У 1974 році - Галушкін А. І. винаходить алгоритм зворотного поширення помилки на навчання багатошарових перцептронів.

У 1975 році - Фукусіма представляє когнітрон (своєю архітектурою когнітрон нагадує будову зорової кори, що має ієрархічну багатошарову організацію, у якій нейрони між шарами пов'язані лише локально.) - мережу, що має можливість до самоорганізуваня, саме таке призначення може використовується для інваріантного розпізнавання образів, але така ахітектура досягається тільки за допомогою запам'ятовування майже всіх станів образу.

У 1982 році - Дж. Хопфілд показав, що нейронна мережа зі зворотними зв'язками може являти собою систему, що мінімізує енергію (повнозв'язна нейронна мережа із симетричною матрицею зв'язків). Кохоненом представлені моделі мережі, що навчається без вчителя (нейронна мережа складається з адаптивних лінійних суматорів, вихідні сигнали шару обробляються так, щоб найбільший сигнал перетворюється на одиничний, інші звертаються в нуль), вирішальної задачі кластеризації, візуалізації даних (карта Кохонена) та інші завдання попереднього аналізу даних.

У 1986 році - Девідом І. Румельхартом, Дж. Є. Хінтоном і Рональдом Дж. Вільямсом - розвили метод зворотного поширення помилки.

У 2007 році - британський та канадський вчений математик, кібернетик та інформатик Хінтон в університеті Торонто – столиця Канади, створив спеціальни

алгоритми глибокого навчання багатошарових моделей нейронних мереж. Хінтон під час тренування нижніх шарів мережі споміг застосовувати малу «машину» Больцмана, що являє собою видом стохастичної архітектури рекурентної нейронної мережі. По Хінтону необхідно використовувати багато прикладів образів, що розпізнаються. Після навчання виходить готовий додаток, який швидко працює, здатне вирішувати конкретне завдання (наприклад, здійснювати пошук осіб на зображенні) [5].

1.2 Аналіз предметної області

1.2.1 Принцип побудови нейронної мережі

Створення штучного нейрона у більшому обумовлено вивченням біологічних нейронів. Штучний нейрон є основою будь-якої штучної нейронної мережі. У свою чергу нейрони являють собою простими, однотипними елементами, що імітують роботу нейронів мозку. Кожен нейрон характеризується своїм поточним станом за аналогією з нервовими клітинами головного мозку, які можуть бути збуджені і загальмовані, тому нейрони відносно просто описати в коді, оскільки працює за принципом транзистора, приймає або 1 або 0. Штучний нейрон, працює за прицнипои так само як і його природний прототип, оскільки має групу синапсів, тобто входів, які з'єднані зі виходами інших нейронів, а також спеціальний аксон – вихідний зв'язок даного нейрона, звідки сигнал збудження або гальмування надходить на синапс інших нейронів. Також використовується третій вид нейронів – «bias», нейрон, який використовується для змішення, щоб отримувати вихідний результат шляхом зсуву графіка функції активації[7].

Найпростіші нейронні мережі були створенні згідно концепції Френка Розенблатта, згідно його концепції – нейронна мережа являє собою систему, що складається з трьох шарів елементів: S (сенсорного), A (асоціативного) та R (реагуючого). При цьому ваги, що навчаються, є тільки між двома останніми шарами нейронів (A → R) піддаються корекції в процесі навчання. Шар синоптичних зв'язків S → A та-

кож містить ваги, але вони не змінюються в процесі навчання, їх можна конфігурувати вручну. Їх значення можуть дорівнювати або «1», або «-1», що відповідає збудливому синапсу і гальмує синапсів. Також могли існувати абсолютно гальмуючи синапси, тобто такі, ваги яких можна було прийняти рівними мінус нескінченності[4].

Отже, для створення нейронних мережі перше що потрібно - тренувальні дані, які поступають до вхідного шара (сенсорний), під час надходження до другого шару дані першого шару, перемножуються зі вагами та поступають до другого шару (асоціативного або прихований), після другого шару - дані надходять до остатнього шару (реагуючого або вихідний), де знов перемножуються зі вагами та поступають до активаційної функції, в залежності від обраної функції дані конвертуються та набувають інший вигляд, після чого дані зрівнюються зі фактичними, а у разі не співпадіння змінюються ваги спеціальним алгоритмом «back propagation» - метод зворотного розповсюдження помилки, який, у свою чергу, базується на алгоритмі градієнтного спуску.

Для змінювання вагових коефіцієнтів мережі так, щоб мінімізувати середню помилку на виході нейронної мережі при подачі на вхід послідовності навчальних вхідних даних. Формально, для того щоб зробити один крок за методом (алгоритмом) градієнтного спуску (зробити лише одну зміну параметрів мережі), для цього необхідно передати на вхід мережі послідовно абсолютно весь набір тренувальних даних, для кожного об'єкта тренувальних даних потрібно обчислити помилку та розрахувати необхідну зміну коефіцієнтів мережі, але після подачі всіх даних розраховувати суму в коригуванні кожного коефіцієнта нейронної мережі (сума градієнтів) і зробити корекцію коефіцієнтів «на один крок». Очевидно, що при великому наборі тренувальних даних алгоритм зворотного розповсюдження помилки працюватиме досить повільно, саме тому часто втілюють коригування коефіцієнтів мережі після кожного елемента навчання, де значення градієнта апроксимуються градієнтом функції вартості, обчисленому тільки на одному елементі навчання. Такий метод називають стохастичним градієнтним спуском чи оперативним градієнтним спуском. Стохастичний градієнтний спуск є являє собою формою стохастичного наближення.

Теорія стохастичних наближень дає умови збіжності способу стохастичного градієнтного спуску[5].

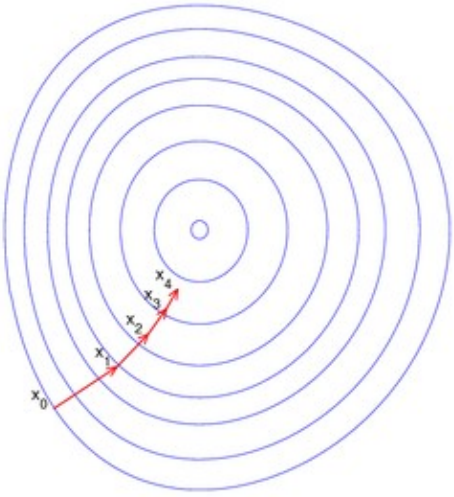

Рисунок 1.2 – Метод зворотного розповсюдження помилки

У випадку коли потрібно створити нейронну мережу, для роботи з великим обсягом даних, потрібно:

1. Оптимізувати алгоритм градієнтного спуску для прискорення навчання;
2. Ініціалізувати початкові значення вагових коефіцієнтів;
3. Виконати стандартизацію вхідних даних;
4. Підібрати найбільш підходящу функцію активації;
5. Обрати критерії якості навчання.

Алгоритм зворотного розповсюдження помилки має один великий недолік - попадання в локальний мінімум Рисунок 1.3

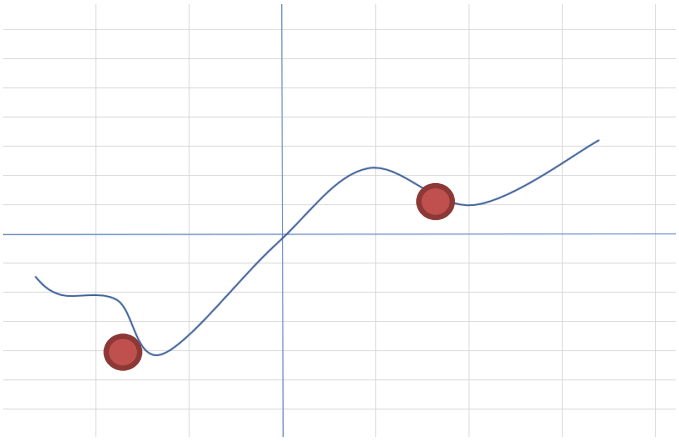

Рисунок 1.3 – Локальні мінімуми

Оскільки не існує універсального рішення цієї проблеми, то алгоритм навчання запускають зі різними навчальними даними вагових коефіцієнтів, таким чином обирає різні відправні точки на функції, для того щоб з більшої вірогідністю знайти локальні мінімуми.

Наступна проблема градієнтних спусків – це повільна збіжність на пологих ділянках функції, для її рішення використовують оптимізації, існують такі методи отпимізіції:

Оптимізація Нестора (Nesterov Momentum) - це розширення імпульсу, яке включає обчислення загасаючої ковзної середньої градієнтів прогнозованих позицій у просторі пошуку, а не самих фактичних позицій, тобто рух по точкам у визначеному направленні, при якому рух додатково переміщається у цьому ж напрямку, протягом деякого часу у майбутньому. Це дає ефект використання прискорювальних переваг імпульсу, дозволяючи уповільнювати пошук при наближенні до оптимуму і знижуючи ймовірність пропуску або перевищення його.

Оптимізація «Adagrad» - оптимізатор зі швидкостями навчання для конкретних параметрів, які адаптуються залежно від того, як часто оновлюється параметр під час навчання. Чим більше оновлень отримує параметр, тим менше оновлень.

Оптимізація «RMSProp» та «Adadelta» - суть оптимізації «RMSProp» полягає в зберіганні ковзного середнього квадрата градієнтів, а також розділення градієнта на корінь цього середнього. Суть оптимізації «Adadelta», полягає в розширення оптимізації «Adagrad», яке адаптує швидкість навчання на основі рухомого

параметра оновлень градієнтів, замість того, щоб накопичувати всі попередні градієнти. Таким чином, «Adadelta» продовжує навчатися, навіть якщо було зроблено багато оновлень. У порівнянні з «Adagrad», в оригінальній версії «Adadelta» не потрібно встановлювати початкову швидкість навчання. У цій версії встановлюється початкова швидкість навчання.

Оптимізатор «Adam» - це спосіб оптимізації, який можна використовувати замість традиційної процедури стохастичного градієнтного спуску для ітеративного оновлення ваги мережі на основі навчальних даних.

Наступна проблема – це значення вхідних даних, серед яких можуть виявитися великі значення і вони одразу перемістяться в область насичення функції активації, тобто буде не рівномірна функція. Для рішення цієї проблеми використовують стандартизацію вхідних даних по формулі:

$$\frac{x_i - min}{\max - min}, \tag{1.7}$$

У цій формулі «max» та «min» - максимальне і мінімальне значення вхідних даних по всій навчальній множині. В результаті вхідні дані під час навчання будуть перебувати в діапазоні від 0 до 1. Також потрібно провести нормалізацію даних на виході,через ті ж самі значення «max» та «min».

Для того щоб виконувати корекцію коефіцієнтів не відразу для кожного спостереження, а після прогін через мережу деякої їх кількості – використовують множину «batch» або «mini-batch», у свою чергу вибірка має назву «епоха», проганяючи mini-batch через мережу, підсумовують локальні градієнти на кожному нейроні, а потім, коригують ваги за сумою, що їх результує. У процесі такого підсумовування невеликі позитивні та негативні значення будуть скомпенсовані і залишиться корисне зміщення, яке призведе до зміни ваг у межах «mini-batch». Значення критерія якості отримується завдяки ітерацій по всій «епохи». У разі незадовільних результатів значення змішуються випадковим чином. Вкінці кожній «епохи» перераховується критерій якості, таким чином отримується графік який показано на рисунку

1.4[7].

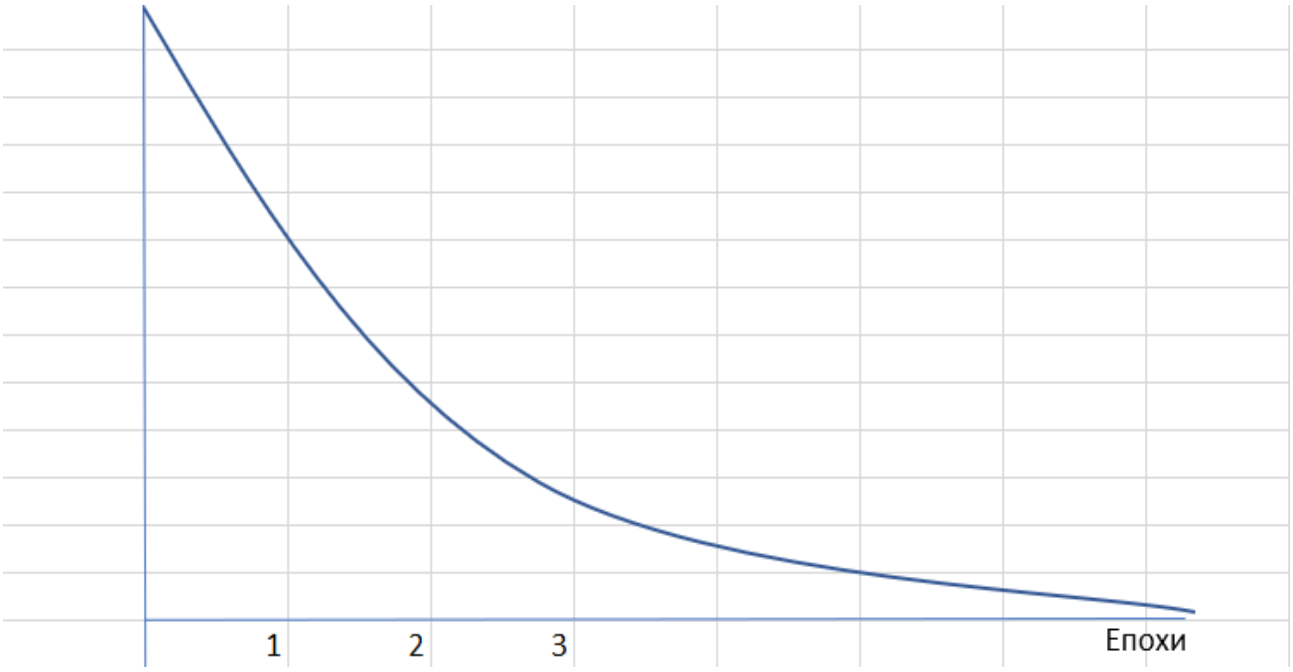

Рисунок 1.4 – Критерій якості по епохам

Ще одна проблема з якою можна зіткнутися – перенавчання, коли модель нейронної мережі пояснює тільки приклади з навчальної вибірки, адаптуючись до навчальних прикладів, замість того щоб вчитися класифікувати приклади, що не брали участь у навчанні. Рішенням цієї проблеми є метод «Dropout», який випадковим чином відключає нейрони, таким чином знижає спеціалізацію кожного окремого нейрона.

1.2.2 Типи нейронних мереж

Згорткові нейронні мережі – архітектура нейронних мереж, яка була придумана Яном Лекуном у 1988 році. Загальна ідея архітектури такої мережі була взята у біологічної зорової системи. Вчені з'ясували, що дендрити кожного нейрона поєднуються не з усіма рецепторами сітківки ока, а лише з деякою локальною областю. І вже дендрити всієї групи зорових нейронів покривають сітківку ока цілком. Таким чином вхідні дані (пікселі) зображення подається на вхід нейрона тільки в межах обмеженої області, як правило квадратної. Ця область даних зміщує-

ться вправо на заданий крок і входи подаються вже другого нейрон. Так відбувається сканування всього зображення. Вагові коефіцієнти для всіх нейронів цієї групи однакові. Сканування зображення по пікселям повторюється зі вже змінним набором коефіцієнтів. Таким чином отримуємо другу групу нейронів, третю, четверту та у загальному випадку маємо n різних груп. Так формується перший прихований шар нейронів загорткової нейронної мережі. Якщо розглянути більш детально то, мережа складається з великої кількості шарів. Після вхідного зображення (пікселі) сигнал проходить через загорткові шари, у яких чергується власне згортка. Саме завдяки такому чергуванні шарів з’являється можливість створювати ознаки, на кожному наступному шарі, саме така «карта» ознаків вбавляє у вигляді, і попре це збільшується кількість каналів. Це означає що нейронна мережа отримала здатність розпізнавання складних ієрархій ознак. На виході загорткових шарів мережі додатково встановлюють кілька шарів перцептрону, на вхід якого подаються кінцеві карти ознак. Тобто основна концепція загорткової нейронної мережі це утворення шар згортки, у свою чергу шар згортки включає в себе для кожного каналу свій фільтр, ядро згортки якого обробляє попередній шар за фрагментами (підсумовуючи результати поелементного сполучення для кожного фрагмента). Загальний вигляд загорткової нейронної мережі показано на рисунку 1.5. Для аналізу обчислених ознаків потрібно виконати ще одну операцію – «Pooling», тобто зміна масштабу картки ознаків, існує три вида pooling-у:

1. MaxPooling - відбір найбільших значень;
2. MinPooling - відбір найменших значень;
3. AveragePooling – вибір середніх значень.

Отже, отримуємо аналіз даних на більшому масштабі, завдяки цьому нейрони кожного наступного шару мають здатність находження більш загальних ознак на зображенні.

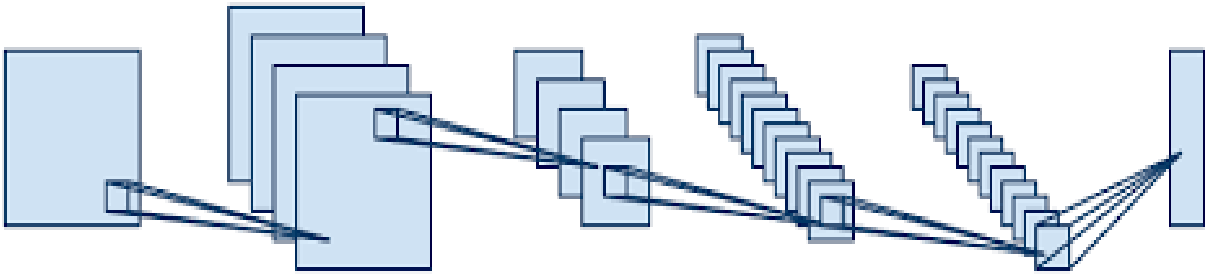

Рисунок 1.5 – Принцип роботи загорткової нейронної мережі

Рекурентні нейронні мережі («RNN») – архітектура нейронних мереж, яка була придумана Хопфилдом у 1982 році, а першою сучасною рекурентною мережею стала мережа Джеффа Елмана, представлена у 1990 р. Така архітектура нейронних мереж є ефективними для моделювання даних послідовності, таких як тимчасові ряди або природна мова. У мережі Елмана функції активації вихідних нейронів - лінійні і є один прихований шар з набором зворотних зв'язків (саме вони визначають рекурентність мережі). Вхідний та вихідний шари формуються як звичайні повнозв'язні. Спрощено цю архітектуру можна як трьох блоків: вхідні дані (x), «RNN», вихідні дані (h), схема цієї архітектури показано на рисунку 1.6.

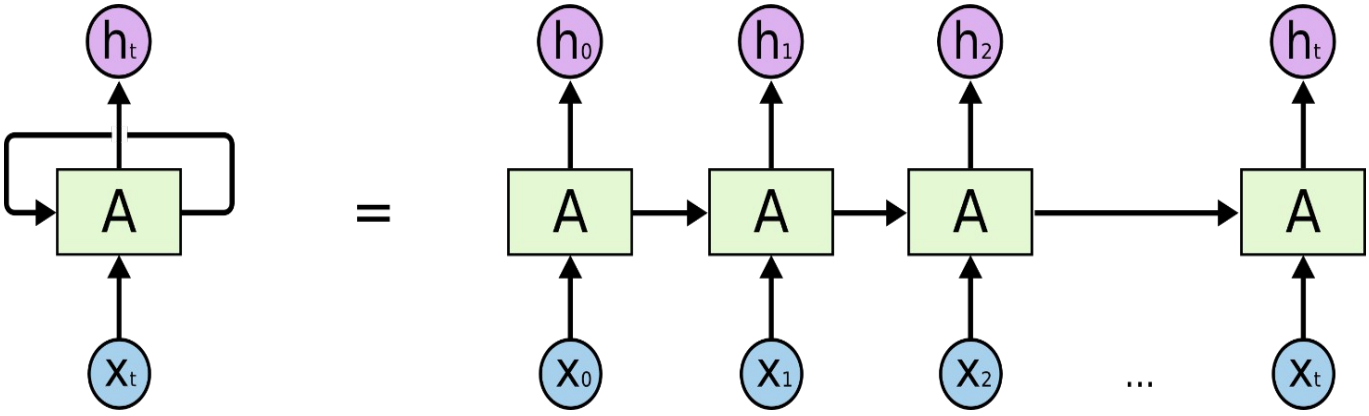

Рисунок 1.6 - Схема рекурентних нейронних мереж

На Рисунок 1.6 - мережа розгорнута у часі, таким чином можна наочно бачити кожен крок роботи цієї нейронної мережі. Така архітектура, коли безлічі вхідних векторів відповідає безліч вихідних, називається «many to many», також існують

архітектури: «many to one», «one to many» та «one to one».

Архітектура типу «many to one» має входи на кожному нейрону, але тільки один вихід, у свою чергу архітектура типу «one to many» має один вхід та виходи на кожному нейроні. Архітектура «one to one» має один вхід та один вихід.

Рекурентні мережі можна навчати алгоритмом зворотного розповсюдження помилки, оскільки має аналог зі багатошарової мережі при розгортанні процесу обробки в часі. Але є декілька проблем, наприклад: на вхід кожного нейрона надходить як вхідний сигнал, а й попередній стан мережі, також вагові коефіцієнти єдині всім шарів розгорнутої мережі, оскільки це, фактично, той самий шар, лише з різних ітераціях обчислення, тому використовують змінений алгоритм під назвою «Backpropagation Through Time» - алгоритм, при якому йде облік попередніх обчислювальних кроків, розгорнутих у часі[5].

1.2.3 Опис предметної області

Для того, щоб розпочати роботу з нейронною мережою, користувач повинен надати тренувальні дані типу Excel за форматом: дата, ціна – все у одній колонці. Нейронна мережа почне аналізувати дані та конвертувати їх для надходження у перший шар, у разі відсутності коефіцієнтів ваги – нейронна мережа почне тренування, після створення ваг – буде прогнозовано декілька значень по дням та створений відповідний графік, де користувач має можливість проаналізувати прогнозуючі дані та порівняти зі фактичними, з великої вірогідності дані будуть не точними, адже нейронна мережа потребую додаткового навчання, для більш точного прогнозування, користувач подає додаткові дані (фактичні), згідно яких – нейронна мережа, перенавчається, та знов видає прогнозуючі значення у виді графіку[4].

1.3 Огляд та аналіз існуючих аналогів

Існує лише декілька успішних аналогів використання нейронних мереж у прогнозуванні часових рядів.

Наприклад компанія «GMDH» - інноваційний глобальний постачальник рішень для планування ланцюжка поставок та прогнозної аналітики. Рішення «GMDH» побудовані на 100% запатентованій технології та керують кожною частиною процесу планування попиту та запасів, забезпечуючи повну прозорість всього ланцюжка поставок.

Компанія створює передові програмні рішення, які дозволяють використовувати можливості алгоритмів моделювання та прогнозування GMDH, забезпечуючи точне та гнучке прогнозування для бізнесу.

Продукт GMDH Streamline – використовується для прогнозування попиту та планування поповнення запасів, яке дозволяє підприємствам максимізувати віддачу від своїх капіталовкладень[9].

Рисунок 1.7 – Логотип компанії «GMDH»

Найкращим аналоговим прикладом є програма «NeuroShell Trader» - нейронна мережа, яка точно аналізує часові ряди для потреби трейдерів, тобто це програмне забезпечення для побудови торгових систем. Це не торгова система сама по собі, це набір інструментів як традиційних, так і методів штучного інтелекту, що можуть комбінувати для створення комп'ютеризованих торгових моделей. Моделі можуть складатися з індикаторів та правил, які трейдери використовували протягом багатьох років, методів штучного інтелекту або їх гібридів. Він буде будувати моделі для акцій, ф'ючерсів, товарів, опціонів, FOREX, індексів та багато іншого. Отже програма NeuroShell Trader прогнозує часові ряди завдяки моделі сгорткової нейронної мережі[10].

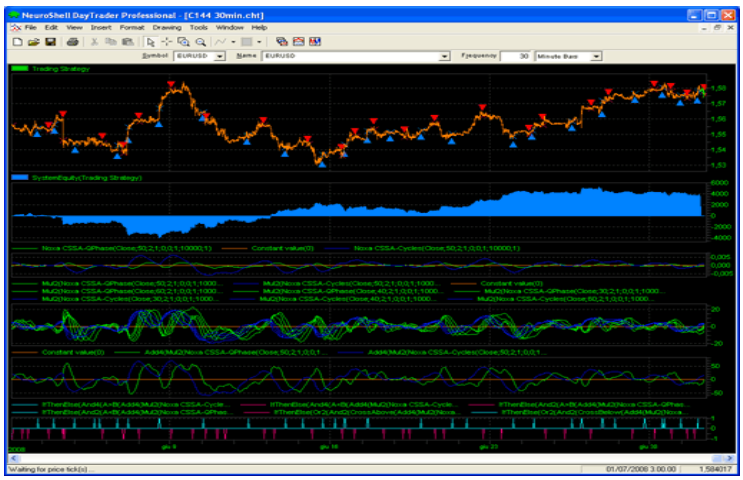

Рисунок 1.8 – Приклад використання «NeuroShell Trader»

Також існує аналог «BrainMaker» - програма яка призначена для побудови багатошарових нейронних мереж з алгоритмом зворотного розповсюдження помилки. Він включає програму підготовки та аналізу вихідних даних , навчання та запуску нейронних мереж, а також набір утиліт широкого призначення. який не тільки прогнозує значення для бірж, а також займається моделюванням кризових ситуацій. «BrainMaker» – одна із перших програм для біржових та фінансових прогнозувань часових рядів[11].

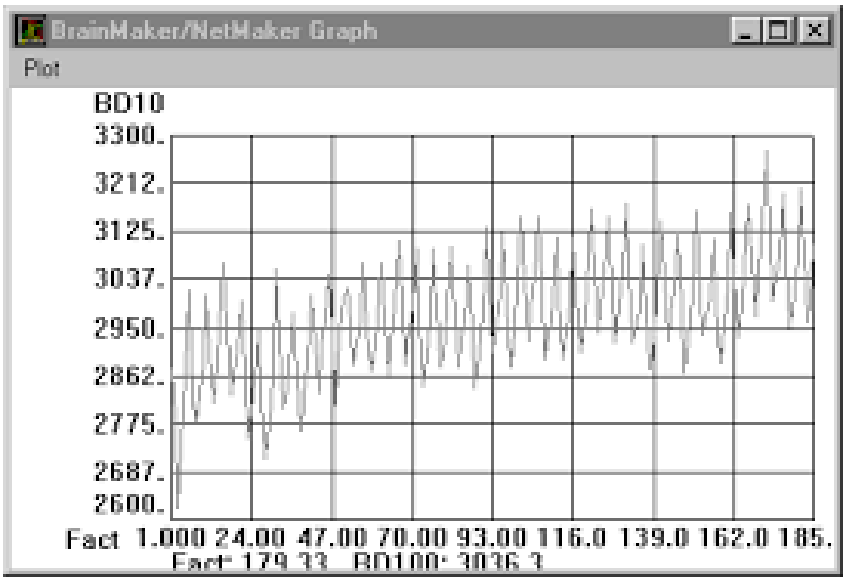

Рисунок 1.9 – Приклад використання «BrainMaker»

1.4 Висновок за розділом 1

1. Наведено характеристику об’єкту дослідження та визначено принцип робити;
2. Проведено огляд та аналіз кількох аналогів нейронних мереж для прогнозування часових рядів, що реалізують схожі функції предметної області;
3. Проведено етап аналізу вимог до розроблюваної нейронної мережі.

РОЗДІЛ 2
ІНСТРУМЕНТИ РОЗРОБКИ ПРОГРАМИ

2.1 Засоби розробки

2.1.1 Мова програмування Python

Python - це високорівнева, універсальна і дуже популярна мова програмування, згідно індексу «PYPL», рис 2.1 . Мова програмування Python використовується у веб-розробці, особливо у програмах машинного навчання, а також у всіх передових технологіях в індустрії програмного забезпечення, також Python є мультипарадигмальною мовою програмування, що підтримує імперативне, процедурне, структурне, об'єктно-орієнтоване програмування. Розробка мови програмування Python, а також реалізація почалася у 1989 року співробітником голландського інституту «CWI» Гвідо ван Россумом.

Історія мови програмування Python почалася ще у в 1980-х голландським програмістом Гвидо ван Россумом, а розпочав його створення в червні 1989 в центрі математичного та інформаційного університету в Нідерландах. Мова програмування Python була задумана як нащадок мови програмування ABC, у свою чергу ABC замислювався для використання в цілях конкуретності з такими мовами програмування як Бейсік, Паскаля і AWK. Мова програмування Python не призначалась для системного програмування, але замислювалась як хороша основа вивчення програмування та використання без високих навиків у програмуванні у повсякденній роботі. Python здатний до обробки спеціальних винятків та взаємодії з операційною системою «Amoeba». Ван Россум є визначним автором мови програмування Python і продовжував виконувати головну роль у прийнятті рішень щодо розвитку мови до 5 липня 2018.

Початок розвитку був зі Python 1.0, якмй з'явився у липні 1994 року. Генеральними можливостями було включення до цього релізу засобів функціонального програмування: лямбда-обчислення, map, filter та згортка списку. Рос-

сум запровадив, що "Python придбав функції lambda, reduce(), filter() і map() завдяки аматору Lisp, якому їх не вистачало, і він надав патчі, що реалізують ці функції".

Кінцевою версією, випущеною Ван Россумом під час роботи в університеті математики та інформатики був Python 1.2, який з грудня 1995 року Россум повернувся до роботи над Python-ом у корпорації національних дослідницьких ініціатив в штаті Вірджинія, де було випущено декілька версій мови.

До версії 1.4 Python вводив безліч нових функцій, найбільш вагомими були запозичені в Modula-3 іменовані параметри і вбудована спеціальна підтримка комплексних чисел, так у Python версії 1.4 з'явилася проста форма інкапсуляції даних за допомогою фунції «name_mangling». Name_mangling змінює побудову компілятора, тобто зміна імен (декоруванням імен) — це техніка, яка використовується для вирішення багатоманітних проблем, викликаних необхідністю вирішування унікальних імен для об’єктів програмування. Він забезпечує засіб кодування та декодування інформації в імені функції, структури, абр класу та іншого типу даних, щоб передати більше семантичної інформації від компілятора до компоновки.

Під час перебування в університеті математики та інформатики Россум запатентував проект «Програмування для всіх», цей проект призначений зробити програмування доступним для більшої кількості людей різних навиків програмування, на основі отримання базового знання мови та математики, необхідних для більшості професій. Python відігравав генеральну роль у цій ініціативі завдяки відносно простому синтаксису. Проект Россума фінансувався «управлінням перспективних дослідних проектів міністерства оборони США», в даний час проект закритий[12].

У 2000 році кістяк команди розробників мови програмування Python, сформували команду «BeOpen PythonLab». Python 2.0 був єдиним релізом BeOpen PythonLab. Після нього Ван Россум та команда розробників PythonLab приєдналися до компанії Digital Creations.

Реліз версії 1.6 включав спеціальну ліцензійний договір від корпорації національних дослідницьких ініціатив, яка була значно довшою за ліцензію організації «CWI», яка використовувалася насамперед. Новий ліцензійний договір включав

статтю, що угода регулюється законами штату Вірджинія. Фонд вільного програмного забезпечення (коротко FSF) заявив про те, що ця стаття порушує вибір правової норми, тобто суперечить ліцензії на вільне програмне забезпечення. Організація BeOpen та FSF підписали договір по вільну ліцензію Python. Python 1.6.1 відмінився від Python 1.6 лише з виправленням дрібних помилок та з новою ліцензією.

Python з версією 2.0 була випущена 15 жовтня 2000 року та вводила багато нових обширних функцій, наприклад збирач «сміття» і підтримка спеціальній кадровикі Unicode. У версії Python 2.0 з'явилась можливість до «спискового» включення – функція, запозичена із функціональних мов програмування SETL та Haskell, які призначенні для роботи зі множинами. Синтаксис у Python для цієї версії аналогічний на синтаксису мови програмування Haskell, за винятком того, що в Haskell віддали перевагу пунктуації написання символів, у свою чергу в Python - ключові слова. Також у Python версії 2.0 було доповнено систему складання сміття з підтримкою циклічних посилань.

Python 2.1 доволі схожий на Python версії 1.6.1 та Python 2.0. Ліцензія для клієнтів, була зареєстрована під назвою «Python Software Foundation License». Починаючи з бета релізу Python 2.1, технічна документація та специфікації належать до організації Python Software Foundation, створеній у березні 2001 року. Цей реліз вписував зміну специфікацію мови, що підтримує включені області видимості, як і мовами зі лексичною областю видимості. Ця можливість була включена за замовчуванням.

Визначним нововведенням у Python версії 2.2 - об'єднання базових типів Python та класів, які вироджуються користувачем, в одній ієрархії. Це перетворило мову програмування Python у повністю об'єктно-орієнтовану мову. Тоді були додані генератори, ідея яких запозичена з Icon, генератор - це фукція для керування ітераційною поведінкою циклу .

У грудні 2014 року було проголошено про те, що Python 2.7 буде підтримуватися до 2020 року, але визначною зміною була зміна прогресу розвитку мови, переходом до більш прозорого процесу творення, з цього и почався розвиток мови програмування Python під сучасний лад[14].

Python 3.0 розроблявся з метою усунення фундаментальних проблем в мові.

Такі зміни не могли бути розроблені за умови збереження повної зворотної сумісності з версією 2. Провідним принципом розробки Python 3 було: «скорочення функціональності, що дублюється, а також усуненням застарілих способів зробити це».

Основні можливості Python 3.0:

1. Синтаксична здібність до анотації параметрів і результату функцій;
2. Повний перехід на Unicode для рядків;
3. Введення нового типу так званих «незмінних байтів» і типу «мінливого буфера». Обидва вони зобов'язані представляти двійкові дані;
4. Нова підсистема вводу-виводу, яка має окремі подання для двійкових і текстових даних;
5. Абстрактні класи, абстрактні методи (доступні в 2.6).
6. Ієрархія типів чисел;
7. Вирази для словників і множин;
8. Зміни для друку на вбудовану функцію, що дозволить модулю «print_function _future» вносити зміни для відмінних вживань функції, а також спростить код;
9. Переміщення скорочення від вбудованого простору до модуля «functools;
10. Новий синтаксис для мета класів;
11. Синтаксис призначення змінено[13].

Тобто, мова програмування Python дуже добре підходить для початківців, а також для досвідчених програмістів, які використовують інші мови програмування, такі як C++ і Java. Головною особливістю цієї мови програмування – це простота використання конструкцій для створення програми, завдяки чому використовується для розробки нейронних мереж, а також має можливість використання спеціальних модулей «TensorFlow» та його доповнення модуль «Keras». Мова програмування Python має у своєму розпорядженні модуль «matplotlib», у свою чергу модуль matplotlib - це велика бібліотека для створення статичних, анімованих та інтерактивних візуалізацій на мові програмування Python, завдяки яким можливо графі-

чно описати прогнозуючи дані та співвідносити їх зі фактичними[21].

Для форматування даних було використано модуль «csv». CSV (comma-separated value) - це спеціальний формат представлення табличних даних, у цьому форматі надаються вхідні дані для тренування нейронної мережі, тому для робити з цим форматом використовується модуль csv[23].

Для роботи зі багатовимірними масивами (матрицями) було використано плагін «NumPy». Бібліотека NumPy надає можливість реалізації обчислювальних алгоритмів, які мають вигляд функцій та операторів, оптимізованих для роботи з багатовимірними масивами. В наслідку будь-який алгоритм математичних обчислювань, який може бути виражений у вигляді послідовності операцій над масивами або матрицями і реалізований з використанням бібліотеки NumPy, працює так само швидко, як еквівалентний код, що виконується «MATLAB»[22].

Отже, було обрано мову програмування Python, оскільки ця мова має можливість використання конструкцій з модуля TensorFlow та має можливість їх описати завдяки модулю matplotlib

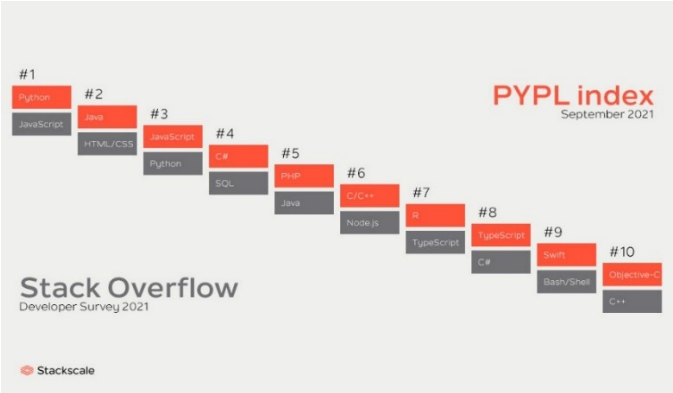

Рисунок 2.1 – Індекс популярності мов програмування по «PYPL»

Рисунок 2.2 – Логотип мови програмування Python

2.1.2 Середа розробки PyCharm

PyCharm – це інтегроване середовище розробки мови програмування Python, яка надає засоби для аналізу коду, графічний налагоджувач, а також інструмент для запуску спеціальних юніт-тестів та підтримує веб-розробку на Django. Це середовище розробки має добре налагоджену файлову систему, а також відлачик, завдяки якому можливу прослідкувати за кожним кроком роботи програми, що є невід'ємною частотною у розробці нейронних мереж на мові програмування Python.

Можливості середи розробки PyCharm:

1. Рефакторинг коду, тобто процес змінення внутрішньої структури програми, який надає широкі можливості щодо виконання швидких глобальних змін у проекті;
2. Підтримка Git, SVN, Mercurial, тобто багатогранність контролю за версіями;
3. Можливість автодоповнення коду;
4. Надає можливість користувачам писати свої плагіни, цим розширювати можливості PyCharm;
5. Крос-платформність;
6. Гнучкий перегляд документації для будь-якого елемента у вікні редактора, перегляд документації через браузер, підтримка «docstring», тобто генерацію, підсвічування, автодоповнення та[17].

Середа розробки PyCharm надає можливість працювати з ноутбуками «Jupyter», запускати команди в інтерактивній консолі Python, підключати бібліотеки «Anaconda», а також працювати з іншими бібліотеками для наукових обчислень та аналізу даних, включаючи Matplotlib та NumPy.

PyCharm був випущений на ринок інтегрованих середовищ розробки для мови програмування Python для створення конкуренції з PyDev та більш поширеного середовища розробки Komodo IDE. Бета-версія була випущена у червеі 2010 року, версія PyCharm версії 1.0 була випущена місяцями пізніше. Версія 2.0 вийшла 13 грудня 2011 року, версія 3.0 була випущена 24 вересня 2013 року. У березні 2016

року JetBrains перейшла на підписну модель ліцензування, а разом із цим змінилася і нумерація версій[18].

Рисунок 2.3 – Логотип середовища розробки PyCharm

2.1.3 Фреймворк PyQt

Фреймворк PyQt – це графічний фреймворк для мови програмування Python, виконаний як розширення Python, що має відкритим вихідним код для віджет-інструментарію Qt. У свою чергу Qt - це набір бібліотек C++ та інструментів розробки, який включає незалежні від платформи абстракції для графічних інтерфейсів. PyQt був розроблений британською компанією «RiverBank Computing»[19].

PyQt працює зі кроссплатформенним середовищем для розробки графічних інтерфейсів «Qt Designer». Ця середа розробки дозволяє створювати графічні інтерфейси користувача за допомогою ряду інструментів, що спрощує створення інтерфейсу програми[20].

Рисунок 2.4 – Логотип PyQt

2.1.4 СУБД MongoDB

MongoDB – нереляційна, крос-платформна документно орієнтована система

управління базами даних, яка не описує схеми таблиць. Вважається одним із класичних прикладів NoSQL-систем. NoSQL-система, яка зберігає документи - документи можуть зберігати складну структурою інформацію, цей документ можна подати як сховище ключів та прив’язні до них значень. NoSQL-система система має три головні характеристики:[24]

1. Швидка технологія, тобто NoSQL бази дані не потребують того ж обсягу підготовчих дій, які зазвичай потрібні для реляційних баз;

2. NoSQL - системи мають можливість зберігання великих обсягів неструктурованої інформації. Завдяки нереляційним баз даних, є можливість зберігання різних типів даних, також, при необхідності, в процесі створення запитів можна додавати нові типи даних;

3. NoSQL – системи зберігають дані поетапно, тобто спочатку на локальному носії, потім перенесення системи в хмару, де і працює[26].

Система управління MongoDB написана на мові C++, тому має можливість швидкого відклику до запитів, що покращить продуктивність програми, де треба зберігати складні структури дані, також цю систему можна портувати на різні платформи. MongoDB надає можливість розгорнути на платформах Linux, MacOS, Windows, Solaris.. Функціональність NoSQL-системи MongoDB дозволяє розташувати декілька баз даних на кількох фізичних серверах, у свою чергу ці бази даних можуть обмінюватися своїми даними, завдяки запитів та зберігати цілісність. MongoDB може працювати з механізмом синхронізації вмісту кількох копій об'єкта, тобто набором реплік, таким чином містити дві або більше копії даних різних вузлах. Кожен екземпляр такої репліки може виступати у ролі основної чи допоміжної репліки, таким чином мати різні зв’язки між собою. Всі дії запису та читання за замовчуванням здійснюються з основною реплікою. А вже допоміжні репліки підтримують копію даних у актуальному стані. У випадку коли основна репліка дає збій, тоді набір реплік проводить автовибір, де вибирається яка з реплік повинна стати основною. Інші репліки можуть додатково стати джерелом для операцій читання. Ще одна можливість, яку надає СУБД MongoDB - це можливість працювати відповідно до парадигми «MapReduce», тобто розподіляти обчислення , що викори-

стовується для паралельних обчислень над великим наборами даних. Для агрегації даних передбачено аналог SQL-виразу GROUP BY; оператори агрегації можуть бути пов'язані в ланцюзі подібно до Unix-конвеєрів, тобто мати структуру перенаправлення введення-виведення: те, що виводить на потік стандартного виводу попередній процес, потрапляє в потік стандартного введення наступного процесу.

Технологія «GridFS» - це технологія для зберігання та вилучення файлів, розмір яких перевищує 16 МБ. Замість зберігання файлу в одному документі (що перевищує розмір, технологія GridFS роздроблює файл на частини або фрагменти і зберігає кожен такий фрагмент в окремому документа. За замовчуванням технологія GridFS використовує розмір фрагмента у 255 КБ, тобто GridFS ділить файл на фрагменти по 255 КБ, крім останнього фрагмента. Останній шматок має розмір згідно граничного розмір документа BSON. Так само файли, розмір яких не перевищує розмір фрагмента, мають тільки останній фрагмент. Коли створюється запит технологія GridFS повторно збирає фрагменти. Таким чином можена виконувати запити діапазону до файлів, які зберігаються в GridFS. Також можна одержати доступ до інформації з довільних розділів файлів, наприклад перекинутись до середини файлу.

MongoDB замість таблиць використовує колекції, тому якщо в реляційних баз даних таблиці зберігають однотипні жорстко структуровані об'єкти, то завдяки системи колекції можна містити різні об'єкти, які мають різну структуру і різний набір властивостей[25].

Особливості NoSQL-системи MongoDB:

1. Хмарне зберігання бази даних для клієнта;
2. Використання технології GridFS, яка має дві колекції;
3. СУБД MongoDB здійснює пошук за запитам, тобто користувач може створити діапазонний запит та миттєво отримати відповідь, швидкість обумовлена тим, що MongoDB написана на мові C++;
4. MongoDB використовує формат зберігання інформації BSON;
5. СУБД MongoDB налагоджує баланс використання потужності процесора, для розподілу навантаження між різними базами даних. При цьому бази

даних, розташовані на різних вузлах, синхронізовані між собою та забезпечують цілісність інформації для клієнта[26].

Рисунок 2.5 – Логотип системи управління базами даних MongoDB

2.1.5 Технологія TensorFlow

Технологія TensorFlow – це програмний модуль мови програмування Python для машинного навчання, тобто технологія TensorFlow використовується для вирішення завдань побудови та тренування нейронної мережі , розроблена компанією «Google», що дозволяє проводити обчислення за допомогою спеціального графа - граф обчислень, вузли якого відповідають операціям чи змінним. Змінні можуть передавати своє значення операції, а операції можуть передавати свої результати інші операції. Таким чином, кожен вузол у графі визначає функцію змінних. Такий граф також дозволяє ефективно обчислювати і похідні функції, що дуже корисно в розробки алгоритмів машинного навчання, а також під час вирішення різних оптимізаційних завдань.

Tensorflow виконує чисельне диференціювання функцій, використовуючи для цього метод зворотного обчислення похідних, що є набором методів для оцінки похідної функції, заданої програмою. Диференціювання використовує той факт, що кожна комп'ютерна програма, якою б складною вона не була, виконує послідовність елементарних арифметичних операцій (складання, віднімання, множення, поділ і т. д.) та елементарних функцій (log, sin, cos і ін). При багаторазовому застосуванні ланцюгового правила до цих операцій похідні довільного порядку можуть бути обчислені автоматично з точністю до робочої точності та з використанням не більше ніж на невеликий постійний коефіцієнт більше арифметичних операцій, ніж вихідна програма.

Назву технології Tensorflow можна перекласти як потік тензорів, тут це факти-

чно означає потік обчислень з багатовимірними матрицями, тобто тензори виступають як багатовимірні масиви з єдиним типом. Таки тензори працюють по аналогії з масивами NumPy, де основним об'єктом є однорідний багатовимірний масив, що представляє собою таблицю елементів одного типу, індексованих кортежом невід'ємних цілих чисел. У таких масивів вимірювання називаються осями.

Тензори мають особливість «Broadcasting», Broadcasting - це концепція, запозичена з еквівалентної функції NumPy. Концепція Broadcasting - обробляє масиви різної форми під час арифметичних операцій. З урахуванням певних обмежень менший масив «транслюється» більшим масивом, щоб вони мали сумісні форми. Мовлення надає засоби векторизації операцій з масивами, так що зациклювання відбувається в мові програмування C, тобто мають високу швидкість, а не в мові програмування Python. Така концепція робить це без створення непотрібних копій даних та зазвичай призводить до ефективної реалізації алгоритму[13].

У свою чергу зміна у TensorFlow - це рекомендований тип об'єкта, що представляє загальний та постійний стан, яким можна керувати за допомогою будь-якої операції, включаючи моделі TensorFlow. Маніпуляція відноситься до будь-якої зміни значення чи параметра. Ця характеристика є найбільш характерною рисою змінних.

Історія створення технології TensorFlow починається зі закритої системи машинного навчання «DistBelief», яка розроблялася компанією «Google Brain» для внутрішніх проектів з 2010 року для роботи з нейронними мережами глибокого навчання. Ця машинна навчання стала використовуватися у багатьох дослідницьких та комерційних проектах групи фірм холдингу «Alphabet». Після успіху у використання технології DistBelief, фірма Google вирішила вивести проект на нову ступінь розвитку, і для рефакторингу виділила групу з кількох розробників, до якої увійшов американський учений у галузі інформатики та програміст Джефф Дін, метою групи було спрощення та оптимізація кодів бібліотеки, збільшення надійності та зручності користування. Ця бібліотека (модуль) отримала назву TensorFlow. У 2013 році до цього проекту приєднався британський та канадський вчений математик, кібернетик та інформатики Джеффрі Хінтон - вчений, який у 2009 - 2010 роках, перетворив

метод узагальненого зворотного розповсюдження помилки та ряд інших поліпшень, що дозволили суттєво покращити точність нейронних мереж.

Технологія TensorFlow з 8 листопада 2015 році стала програмним забезпеченням з відкритим вихідним кодом. Технологія TensorFlow - система машинного навчання компанії Google другого покоління. У той час як перше покоління працює на одиничних пристроях, система другого покоління може працювати на багатьох паралельних процесорах як CPU, так і GPU, спираючись на архітектуру CUDA (програмно-апаратна архітектура паралельних обчислень, що дозволяє суттєво збільшити обчислювальну продуктивність завдяки використанню графічних процесорів фірми Nvidia.) для підтримки обчислень загального призначення на графічних процесорах, таким чином значно покращити обчислювальну продуктивність технології другого покоління над першим. Технологія TensorFlow підтримує кросплатформність, тобто доступна для 64-розрядних Linux, macOS, Windows, та для мобільних обчислювальних платформ, включаючи Android та iOS.

У травні 2016 року Google повідомила про застосування для завдань глибинного навчання апаратного прискорювача власної розробки – тензорного процесора (TPU). Тензорний процесор (TPU) - процесор, що відноситься до класу нейронних процесорів, які використовуються для апаратного прискорення роботи алгоритмів штучних нейронних мереж, що є спеціалізованою інтегральною схемою, розробленою компанією Google і призначеною для використання з бібліотекою машинного навчання TensorFlow. Цей процессор представлений у липні 2016 році на річній конференції, яка орієнтована на розробників, що проводиться компанією Google для обговорення розвитку відкритих технологій та сервісів Google. Порівняно з графічними процесорами (GPU), наприклад компанії Intel , розрахований більш високий обсяг обчислень зі зниженою точністю більш високої продуктивності на ват і відсутності модуля для растризації і текстурних блоків. Також корпорація застосувала тензорні процесори для обробки фотографій «Google Street View» (функція Google Maps та Google Earth, що позволяє дивитися види вулиць багатьох міст світу з висоти близько 2 - 2,5 метрів.) на предмет отримання тексту, повідомлялося, що весь обсяг оброблений менш ніж за чотири дні. У Google Фото один тензорний процесор

може обробляти понад ста мільйонів фотографій за день. Також пристрій застосовується для системи самонавчання RankBrain, яка підтримує обробку результатів пошуку Google та забезпечує більш релевантні результати для користувачів. Після використання тензорного процесору у власних завдань, компанії Google з обробки даних вдалося досягти на порядок кращих показників продуктивності на ват витраченої енергії.

Рисунок 2.6 – Логотип TensorFlow

Отже, технологія Tensorflow – це використання тензорів (матриці), завдяки яким можна описати графи, у свою чергу графи описують нейронний зв'язок. [16]

Фреймворк «Keras»

Фреймворк – це спеціальна програмна платформа, яка має можливість визначати структуру програмної системи, тобто програмне забезпечення, яке спрощує розробку та поєднання різних компонентів великого за розміром програмний проект. Фреймворк різниться від поняття бібліотеки (модуль) тим, що бібліотека використовується в програмному продукті як набір близької функціональності підпрограм, що не впливає на архітектуру програмного продукту і не накладаючи на неї певні обмеження. У той час як фреймворк диктує правила побудови архітектури додатка, задаючи на початковому етапі розробки поведінку за умовчанням, яку потрібно буде розширювати та змінювати відповідно до зазначених вимог.

Фреймворк Keras – це фреймворк, який написаний мовою програмування Python і забезпечує взаємодію зі штучними нейронними мережами, а саме забезпе-

чує роботу зі Tensorflow, оскільки цей фреймворк є надбудовою над технологією TensorFlow. Keras є найпопулярнішим фреймворком для використання нейронних мереж (після Tensorflow), згідно рисунку 2.6. Фреймворк Keras націлений на оперативну роботу з мережами глибинного навчання. Keras був створений як частина дослідницьких зусиль проекту ONEIROS, засновником є Франсуа Шолл, інженер Google. На початку створення фреймворка Keras, планувалося, що компанія Google підтримуватиме Keras в основній бібліотеці TensorFlow, проте Шолле виділив Keras в окрему надбудову, оскільки згідно з концепцією Keras є скоріше інтерфейсом, ніж наскрізною системою машинного навчання. Keras - це високорівневий API TensorFlow : доступний, високопродуктивний інтерфейс для вирішення задач машинного навчання з сучасним глибоким навчанням. Він надає необхідні абстракції та будівельні блоки для розробки та доставки рішень машинного навчання із високою швидкістю ітерації. Keras дозволяє інженерам та дослідникам повною мірою скористатися перевагами масштабованості та кросплатформових можливостей TensorFlow. Keras дає можливості запускати на TPU або на великих кластерах графічних процесорів, а також експортувати свої моделі Keras для запуску у браузері або на мобільному пристрої.

Використовуючи API Keras - створюємо нейронну мережу на TensorFlow, але робимо це набагато швидше. Фреймворк Keras надає можливості:

1. Зниження когнітивного навантаження на розробників, дозволяючи зосередитись на справді важливих частинах проблеми;
2. Використовує принцип поступового розкриття складності, тобто спрощує початок роботи, але дозволяє обробляти довільно розширені варіанти використання, вимагаючи лише поступового навчання кожному етапі;
3. Забезпечення найкращу в галузі продуктивність і масштабованість, він використовується організаціями та компаніями, включаючи NASA, YouTube або Waymo[15].

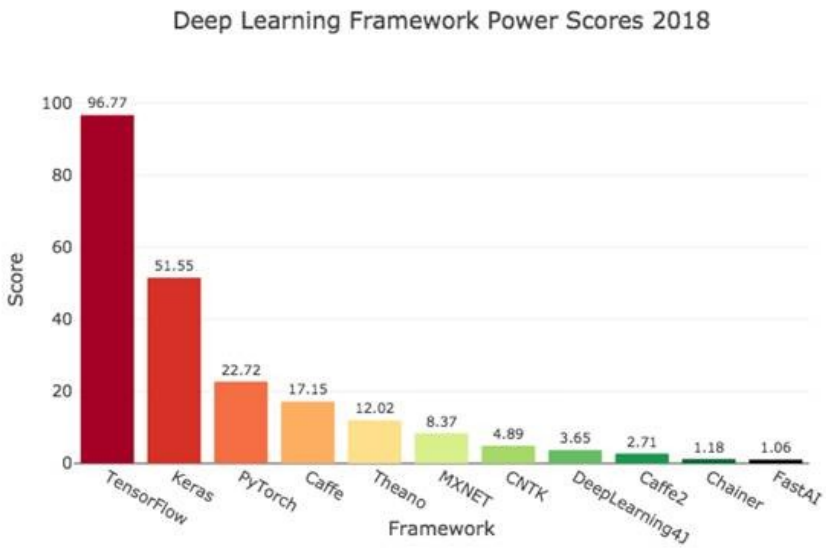

Рисунок 2.7 – Популярність використання фреймворків для нейронних мереж

Отже, для оптимального використання технології TensorFlow, було вирішено використовувати фреймворк Keras.

2.2 Розробка архітектури програми

Архітектура програми – це абстракція елементів системи на певній фазі її роботи. Оскілки стан програми залежить від дій користувача, то архітектуру програми можна представити діаграмою станів. Розроблювальна програма використовує підключення до бази даних зареєстрованих користувачів, яка використовується програмою при авторизації користувача, тобто повертає звіт чи зареєстрований даний користувач в системі, рисунок 2.7.

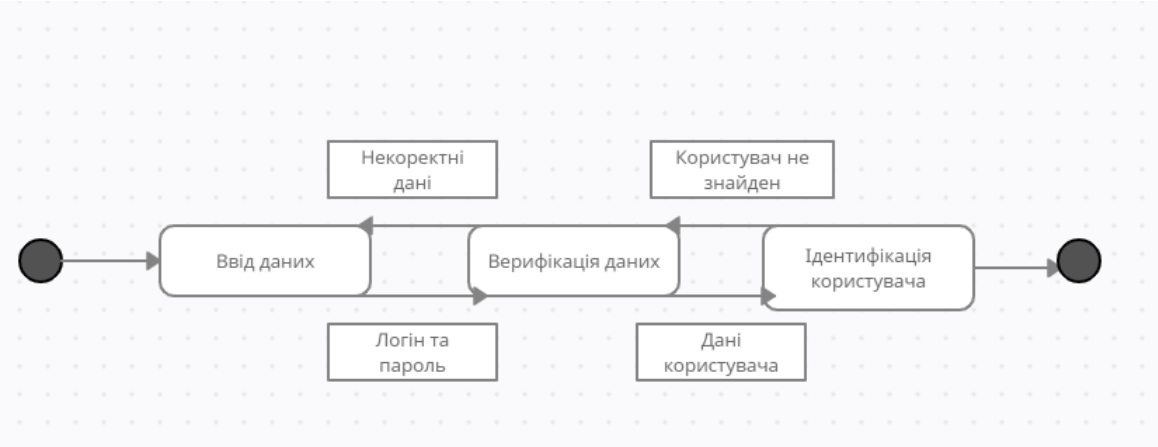

Рисунок 2.8 – Діаграма станів (Авторизація)

Після проходження авторизації, користувач «переноситься» на наступне вікно, де вже може користуватися послугами програми, рисунок 2.8.

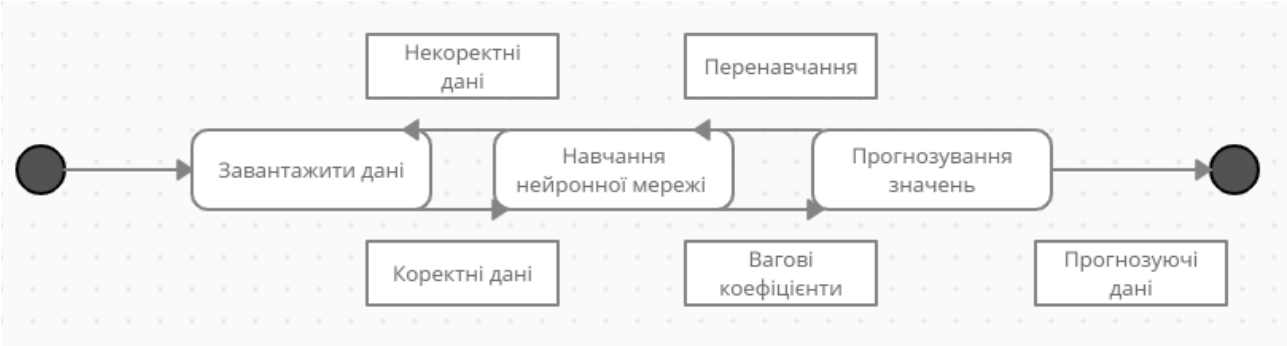

Рисунок 2.9 – Діаграма станів (Після проходження авторизації)

Користувач завантажує дані для тренування нейронної мережі, після чого, дані потрапляють у нейронну мережу, де, за необхідністю, нейронна мережа перенавчається для кращих прогнозуючих значень, після чого вагові коефіцієнти зберігаються. На основі вагових коефіцієнтів нейронна мережа прогнозує значення, та передає ці значення на огляд користувача.

Створення архітектури нейронної мережі

Для створення нейронної мережі для прогнозування часових рядів було зіставлено дві архітектури: архітектура рекурентних нейронних мереж та архітектура згорткових нейронних мереж. Важливою особливістю при створення нейронної мережі є пам'ять про часові залежності в даних. Адже у разі коли кожен батч вхідних

даних обробляється незалежно, без збереження стану між ними, не основуючись на попередній інформації – нейронна мережа тренується не коректно. Для врахування цієї особливості, було взято архітектури рекурентних нейронних мереж та загорткових нейронних мереж, що мають можливість обробляти елементи часового ряду послідовно з використанням інформації, отриманої при обробці попередніх його елементів. Щоб переконатися у важливості володіння пам'яттю про попередні значення часового ряду (а також щоб краще розпізнати закономірності, що містяться в ньому) нейронною мережею, було проаналізовано статтю «Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction»[6], автор якої Milos Miljanovic на прикладі часових рядів: «Mackay-Glass», «Sunspots», «S&P 500» показав - для того, щоб робити прогнозування, мережі повинні містити пам’ять про попередні значення часового ряду, щоб знайти ознаки, які містяться в цих даних.

Якщо подивитись на тренувальні данні, то ці дані представляють собою послідовність, у якій важливі причина і наслідок. Для рішення задач пошуку закономірності причини і наслідку, теоретично, ідеальною архітектурою є рекурентні нейроні мережі, але у разі великої кількості даних робота архітектура потребує велику кількість обчислювальних ресурсів, тобто така архітектура працює повільно. У свою чергу властивості згорткових нейронних мереж, які роблять їх ідеальним вибором для розпізнавання образів також підходять для обробки послідовностей. Час можна розглядати як просторовий вимір, подібно до висоті або ширині двовимірного зображення. Така згорткова нейронна мережа вимагає менше обчислювальних ресурсів. Таким чином згорткові нейронні мережі можуть бути альтернативою рекурентним мережам.

Оскільки згорткова мережа обробляє вхідні дані незалежно, вона нечутлива до порядку проходження часових даних, на відміну від рекурентної мережі. Але якщо об'єднати швидкість і легкість згорткових мереж з чутливістю до порядку рекурентних мереж, де згорткових мережі використовуються для попередньої обробки даних перед передачею їх у рекурентну мережу, то така модель нейронної мережі отримує все найкраще від рекурентних мереж та сгорткових мереж.

Таким чином, згорткова частина обробки даних перетворить вхідну послідовність на більш коротку послідовність високорівневих ознак. Після чого ця послідовність високорівневих ознак подається на вхід рекурентної частини нейронної мережі. Така модель нейронної мережі показана на рисунку 2.7.

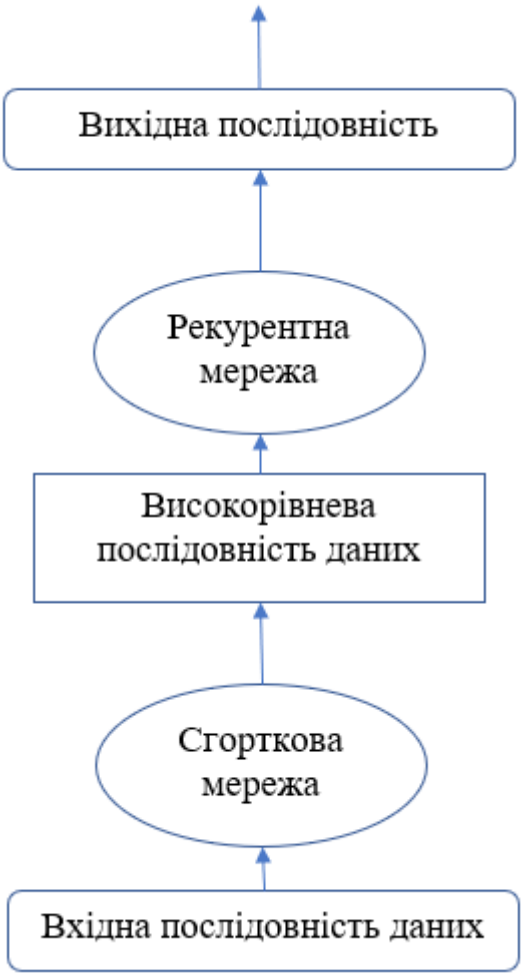

Рисунок 2.10 – Загальна архітектура нейронної мережі

Модель нейронної мережі, згідно рисунку 2.7, дозволяє обробляти великі обсяги даних, відносно великої швидкості, але у разі потреби обробки невеликих за обсягом даних, така модель не є підходящою. Для вирішення цієї проблеми було вирішено створити окрему архітектуру нейронної мережі, основаною на архітектури рекурентних нейронних мереж. Було вирішено використовувати декілька слоїв рекурентних нейронних мереж, це дозволить побороти проблему перенавчання та підвищити якість нейронної мережі. Така модель нейронної мережі показана на рисунку

2.8[7][8].

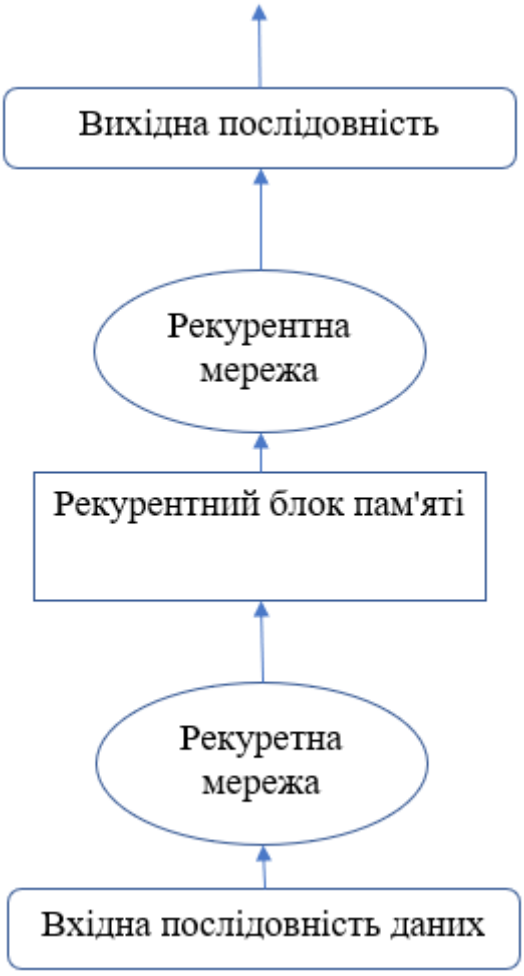

Рисунок 2.11 – Рекурентна архітектура нейронної мережі

2.3 Проектування структури бази даних

Проектування бази даних – це процес створення моделі бази даних та визначення необхідних обмежень цілісності. Існують чотири основні задачі проектування:

1. Забезпечення зберігання у БД всієї необхідної інформації;
2. Забезпечує можливість отримання даних за всіма необхідними запитами;
3. Скорочення надмірності та дублювання даних;
4. Забезпечення цілісності бази даних.

Тому під час розробки програми необхідним етапом розробки є проектування бази даних, яка забезпечить зберігання тренувальних даних, даних користувача та прогнозуючі дані, що наддасть змогу проводити операції над цими даними.

Для зберігання даних у базі потрібно створити «колекції», оскільки було обрано нереляційну СУБД MongoDB. Для зберігання всієї необхідної інформації вирішено створити три колекції: «Users», «previous_data», «prognoz_data». У свою чергу зв'язок між колекціями прописаний у коді.

При авторизації система використовує підключення до бази даних, а саме до колекції «Users», де шукає користувача у базі даних зареєстрованих користувачів, після чого повертає звіт чи зареєстрований даний користувач в системі. У разі неуспішній авторизації користувача повертає звіт зі значення «false», а у разі успішній авторизації повертає звіт зі значення «true», та користувач переходить до головного вікна. У головному вікні користувач має можливість надати дані для тренування для подальшого прогнозування, ці дані додаються та зберігаються у колекції «previous_data». Після чого на основі цих даних прогнозуються значення, які зберігаються у колекції «prognoz_data»[27].

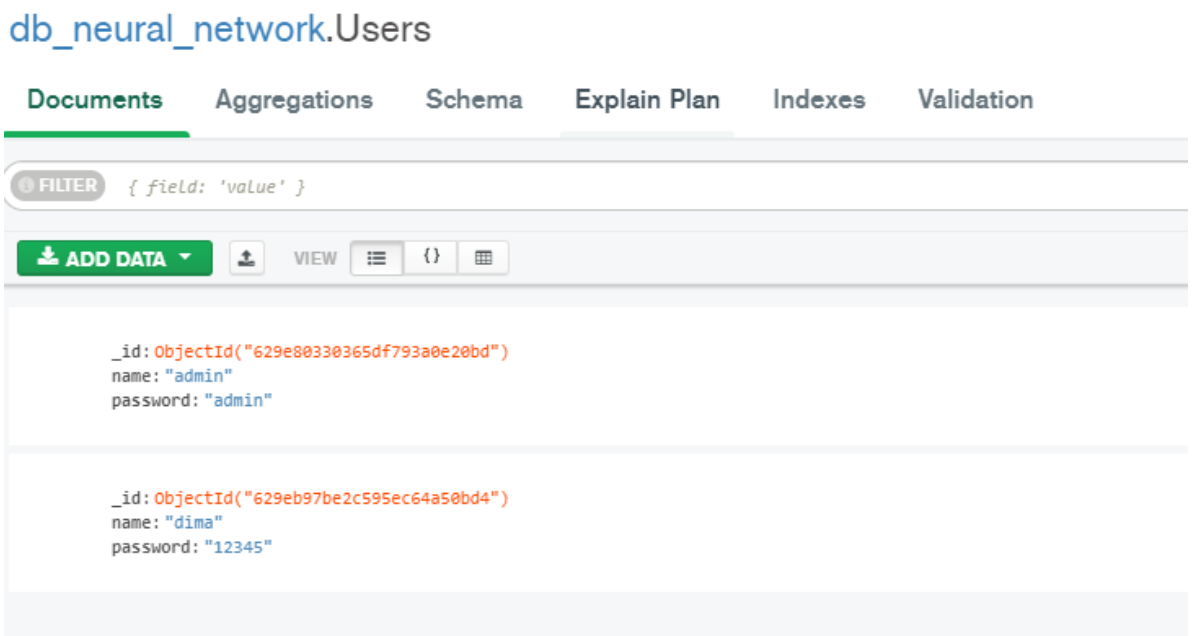

Рисунок 2.12 – Колекція «Users»

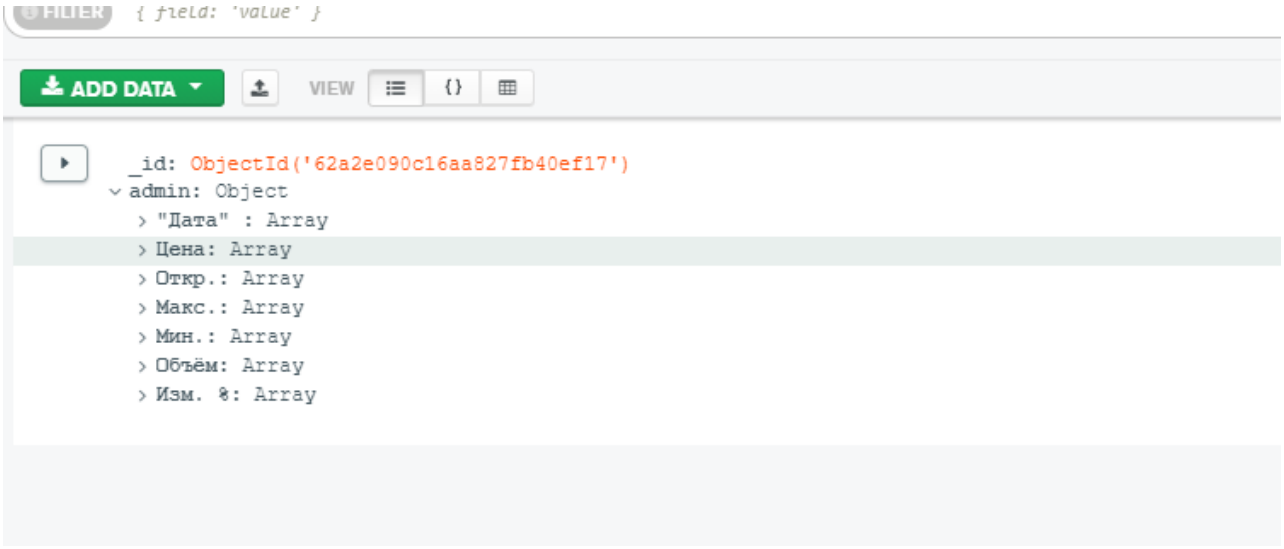

Рисунок 2.12 – Колекція «previous_data»

2.4 UML-діаграми

Unified Modeling Language (UML) – уніфікована мова моделювання. Розшифруємо: modeling має на увазі створення моделі, що описує об'єкт. Unified (універсальний, єдиний) - підходить для широкого класу проектованих програмних систем, різних галузей додатків, типів організацій, рівнів компетентності, розмірів проектів. Unified Modeling Language був створений для визначення, візуалізації, проектування та документування, програмних систем. На основі UML-діаграм створюють програми, головна причина використання UML-діаграм – це можливість розробникам програмного забезпечення добитися угоди у графічних позначеннях для представлення загальних понять[28].

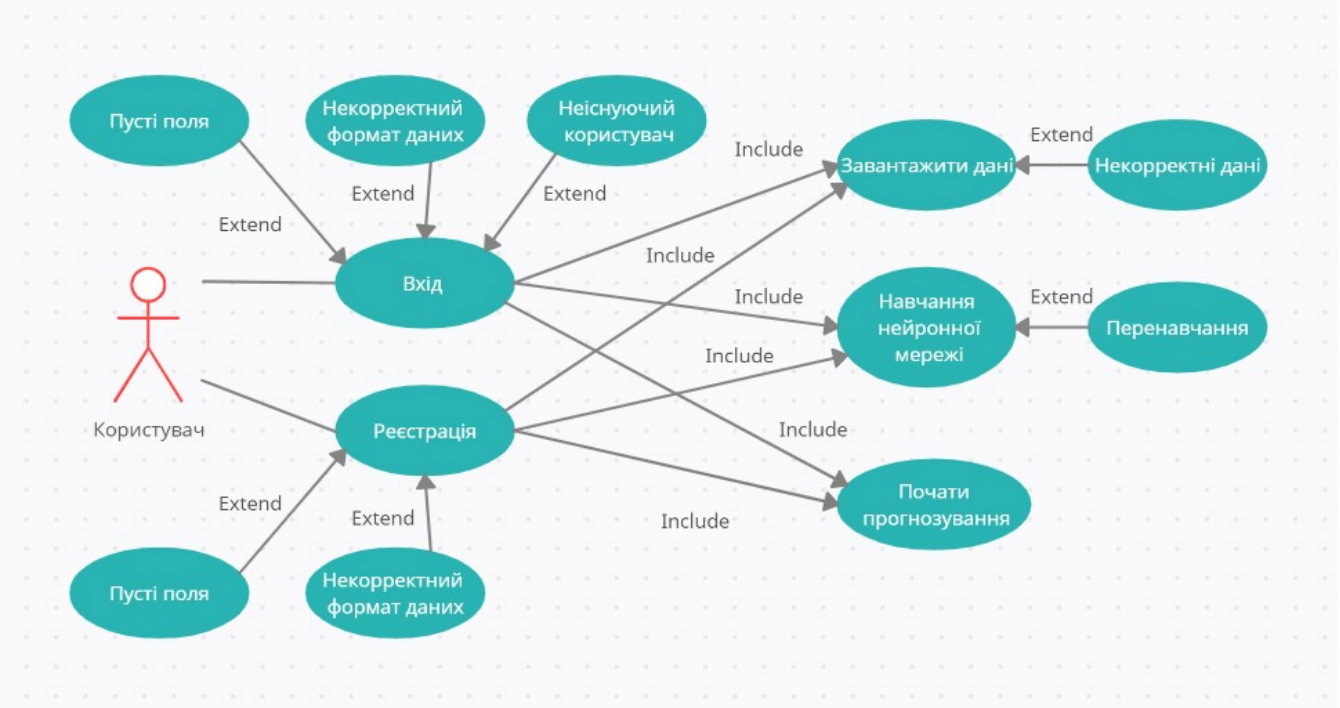

Рисунок 2.12 – Діаграма прецедентів

На рисунку 2.12 зображено як виконується взаємодія користувача та програмою. Користувач авторизується та починає працювати зі основною програмою.

2.5 Моделювання варіантів використання нейронної мережі

В сучасному світі використання нейронних мереж тільки набирає обертів, особливо стосовно розділу «Data science». Data science – це галузь, яка використовує наукові методи, процеси, алгоритми та системи для отримання знань та ідей із шумних, структурованих і неструктурованих даних, та застосування знань із даних у широкому діапазоні областей застосування. Зокрема, використання нейронних мереж для прогнозування часових рядів – має широкий спектр використання. Згідно теми роботи, у якості прикладу, програму було використано для прогнозування фінансових рядів для подальшого аналізу та створення обліку згідно показників. Але цю програму можна використовувати для прогнозування не тільки фінансових рядів, а й для прогнозування інших типів часових рядів, наприклад для прогнозування природних явищ, таких як температура, оскільки нейронна мережа орієнтуються на попередніх рядів незалежно від типу цього ряду, то вона зможе спрогнозувати дані на майбутнє. Також цю програму можливо використовувати у таких сферах:

1. Енергетика – ціни, попит, виробничі графіки;
2. Роздрібна торгівля – продажі, споживчий попит на певні товари;
3. Транспорт - попит на майбутні поїздки.

Для аналізу великого обсягу даних потрібно багато часу, але за допомогою штучних мереж можливо в рази прискорити аналіз даних, згідно використання програми, для послідуючого прогнозування майбутніх значень.

2.6 Висновок до розділу 2

1. Проаналізувавши базові технології розробки програм було визначено, що для даного проекту буде використано середовище розробки PyCharm, мова програмування Python, СУБД MongoDB, інтерфейс PyQt, технологія TensorFlow;
2. Побудовано архітектуру програми, зокрема, архітектуру нейронної мережі;
3. Спроектовано структуру бази даних;
4. Побудовано UML- діаграми для створення програми.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМИ

3.1 Процес розробки програми

Перше що потрібно розробити – це інтерфейс програми для простої взаємодії користувача зі можливостями програми. Для розробки інтерфейсу було обрано фреймворк «PyQt» та мову програмування «Python», у свою чергу середовище розробки інтерфейсу - «Qt Designer».

Для того щоб користувач почав працювати зі програмою він має пройти авторизацію, для цього було створено вікно «authorization_interface» у середовищі розробки «Qt Designer» - рисунок 3.1.

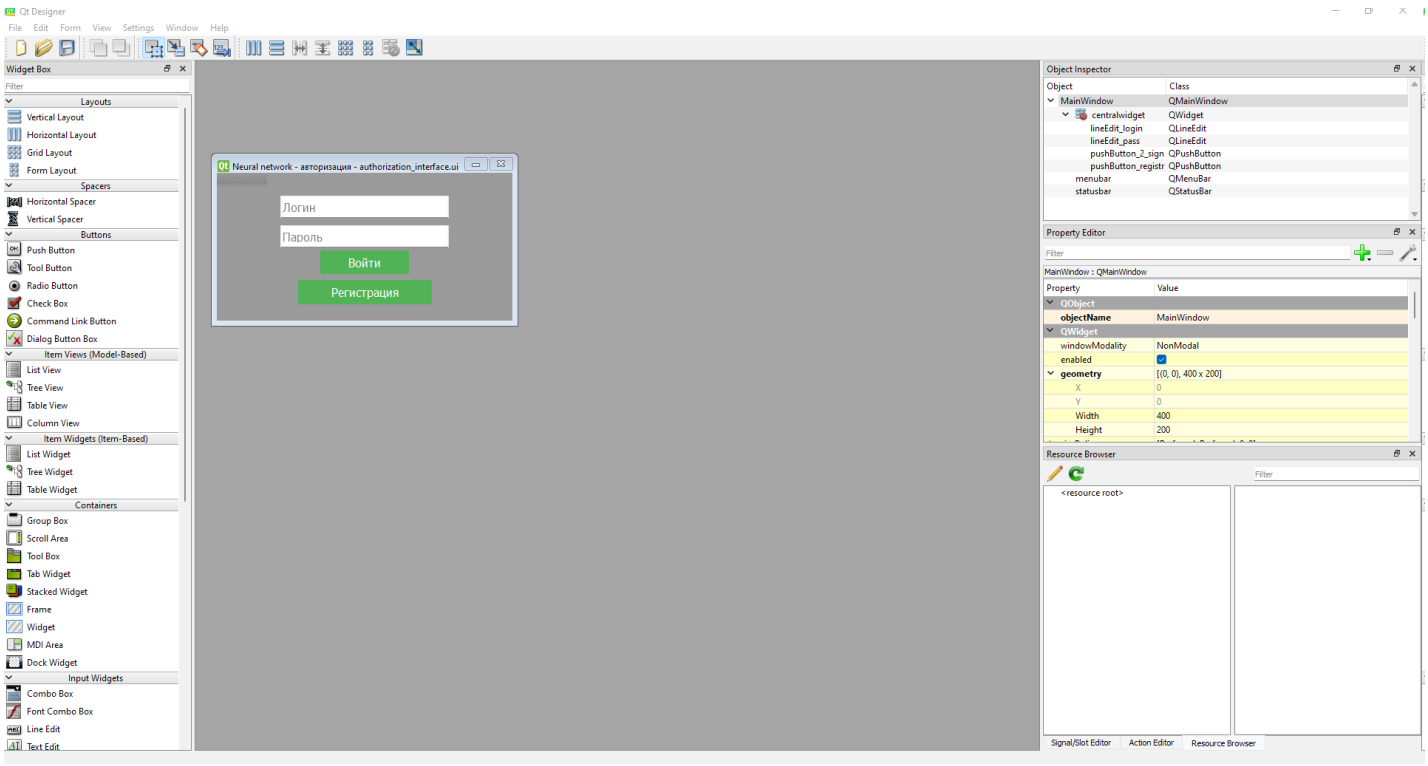

Рисунок 3.1 – Вхідне вікно «authorization_interface»

Після того як користувач натисне на кнопку «Войти», або «Регистрация» - дані які були надані у полях «LineEdit» почнуть зіставлятися зі даним бази даних, у разі каретних даних користувач потрапить в наступне вікно – головне вікно, у разі

некоретних даних з’явиться модальне вікно відповідно помилки рисунок 3.2.

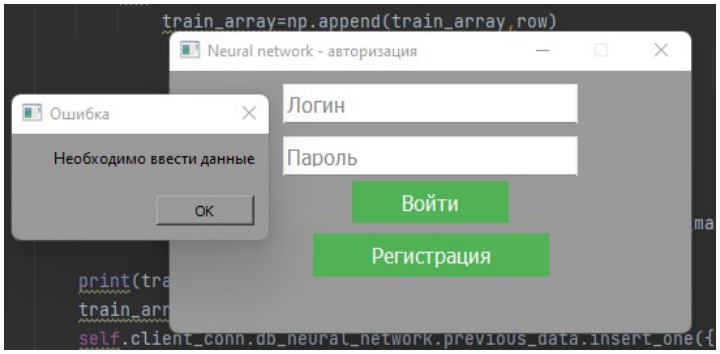

Рисунок 3.2 – Модальне вікно помилки

У разі проходження верифікації користувач потрапляє на головне вікно рисунок 3.3, де вже має можливість працювати зі програмою.

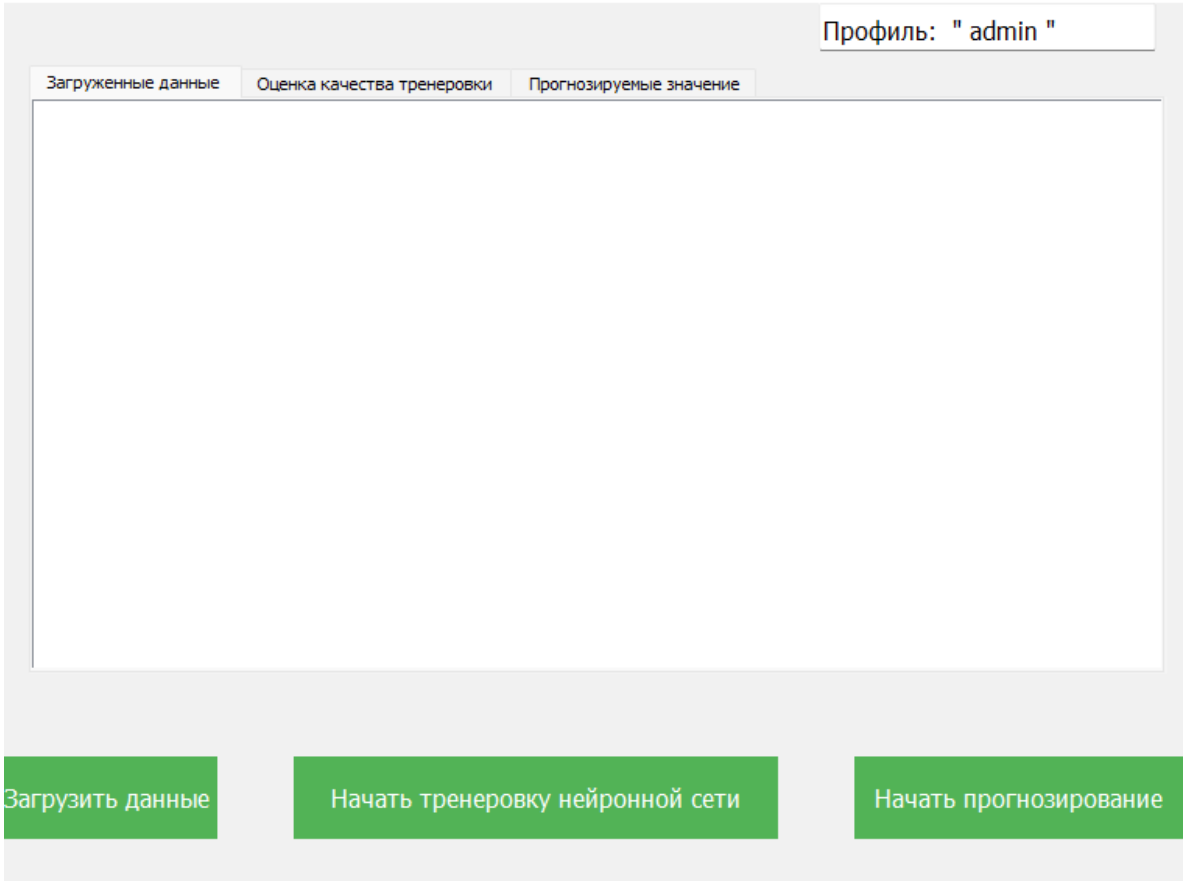

Рисунок 3.3 – Головне вікно програми

У головному вікні є три функціональні кнопки: «Загрузить данные», «Начать тренировку нейронной сети», «Начать прогнозирование». При натисканні

кнопці «Загрузить данные» - користувач має можливість додати дані для тренування нейронної мережі. Ці дані будуть відформовуванні та відсортовуванні перед наданням до нейронної мережі, на рисунку 3.4 надано приклад форматування даних.

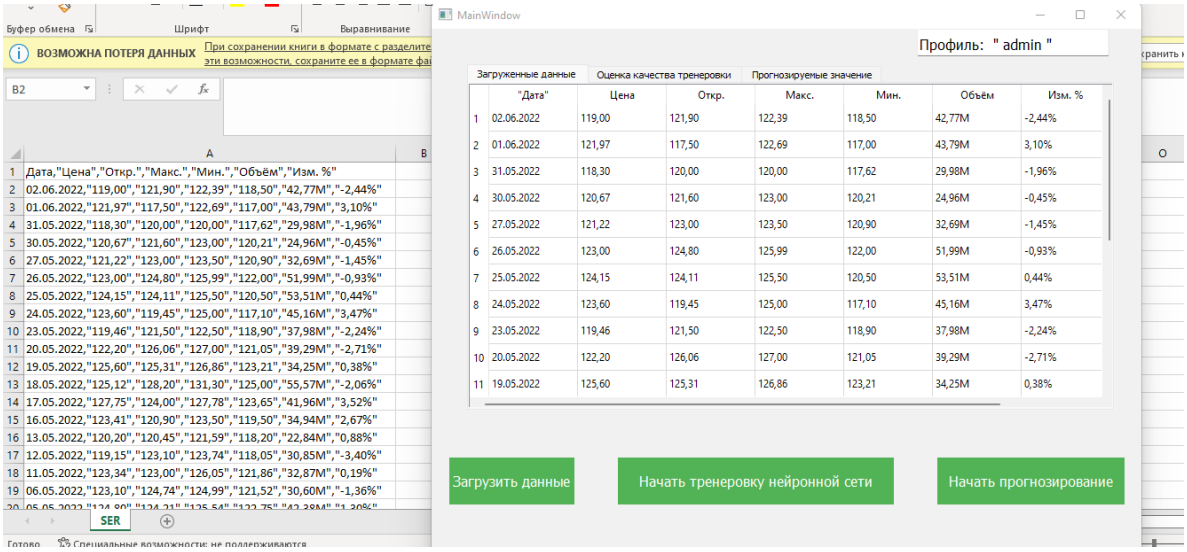

Рисунок 3.4 – Форматування даних

Після чого користувач може натиснути на кнопку «Начать тренировку нейронной сети», де ці дані вже підуть на вхід нейронної мережі. Також щоб користувач мав розуміння готовності даних, на екрані з’явиться «ProgressBar» - повзунок готовності тренування перед прогнозуванням. І вже після тренування нейронної мережі користувач має можливість побачити якість тренування нейронної мережі у вкладці «Оценка качества тренировки» - рисунок 3.6.

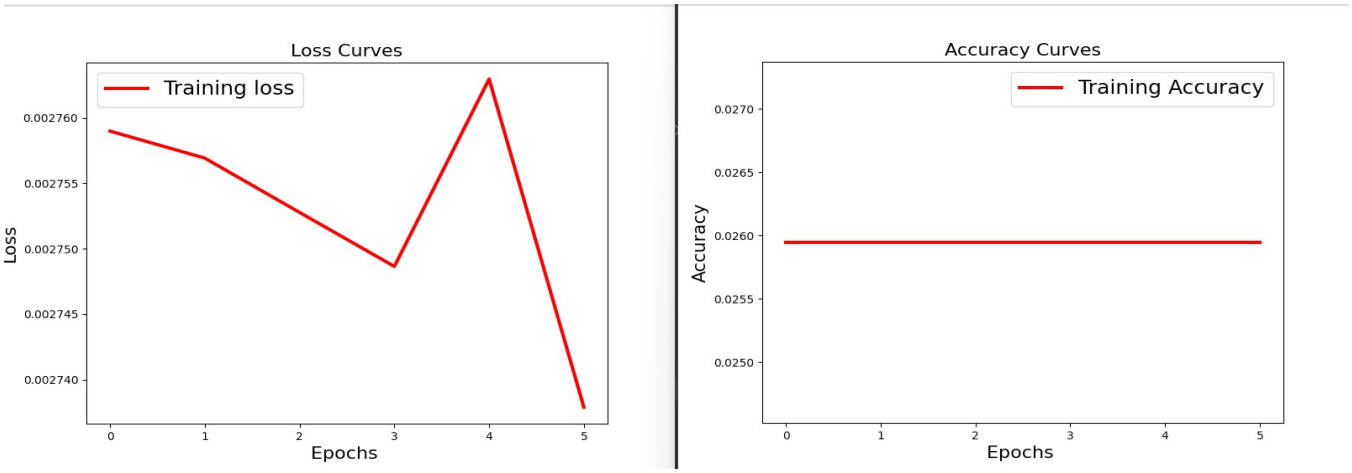

Рисунок 3.5 – Оцінка якості тренування

На цій вкладці користувачу надано можливість побачити «Mean squared error» - середньоквадратичну помилку, тобто середньоквадратичну різницю між оцінними значеннями та фактичним значенням. Для цього було вирішено використовувати функцію втрат.

Після виконання всіх цих процедур (надання даних та їх тренування) – користувач отримує змогу натиснути на кнопку «Начать прогнозирование», де на основі тренувальних даних, тобто вагових коефіцієнтів, нейронна мережа починає етап прогнозування. Після прогнозування для користувача створюється графік прогнозуючих значень у вкладці «Прогнозируемые значение» (рисунок 3.6), що надає змогу користувачу проаналізувати ці дані та створити облік згідно свого аналізу.

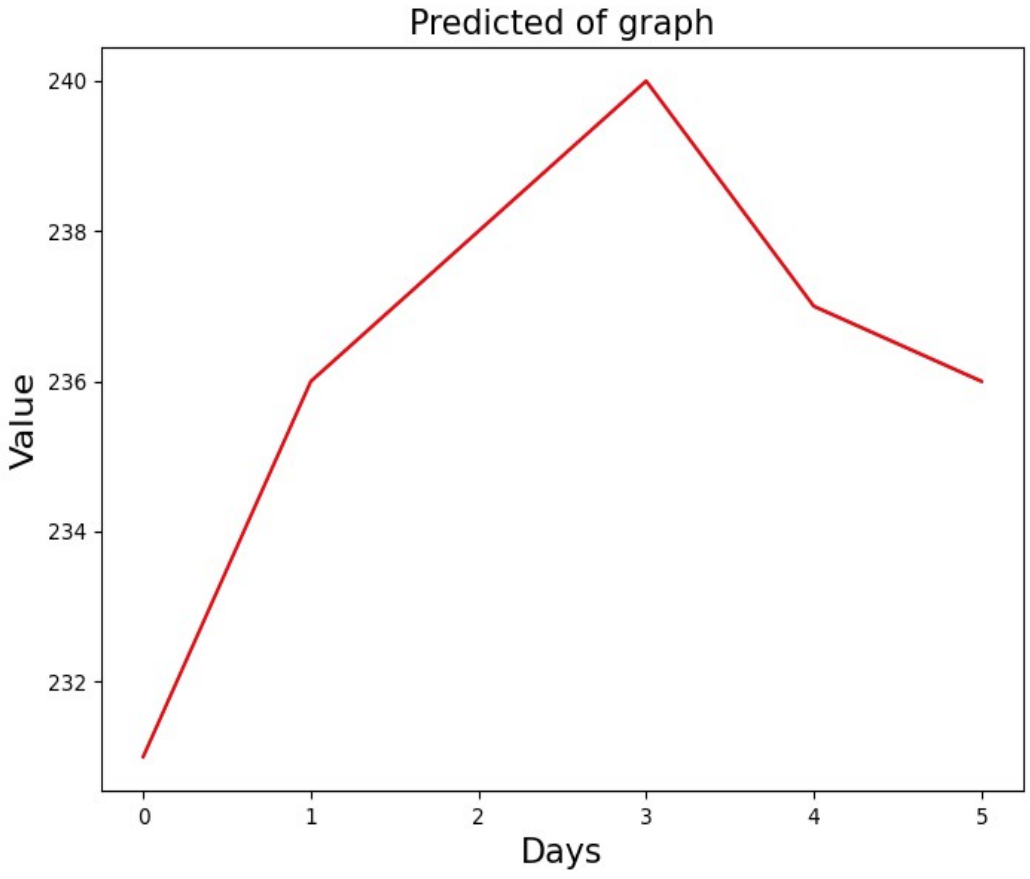

Рисунок 3.6 – Прогнозуючі значення

3.1.1 Розробка нейронної мережі

При розробці нейронної мережі для прогнозування потрібно дотримуватися таких парадигм форматування даних:

1. Отримання тренувальних даних (форматування);
2. Нормалізація цих даних (від -1 до 1);
3. Крос-валідація даних, тобто розподіл даних для тренування та дані для тесту, що не беруть участі в навчанні.

Дані для тренування надає користувач, при форматування даних зберігаються два стовпця: остаточна ціна за день та об`єм капіталізації за день. Саме ці дані і будуть основою для формування вагових коефіцієнтів нейронної мережі.

Нормалізація даних відбувається за зміненою формулою 1.7. А саме бере саму високу ціну та ділить всі інші значення на цю ціну, та відраховується один, таким чино виконується стандартизація, код з виконанням нормалізації наведено на лістингу 3.1

Лістинг 3.1- Нормалізація даних в коді

```
def normalisation(self, window_data, single_window=False):
        normalised_data = []
        window_data=[window_data] if single_window else window_data
        for window1 in window_data:
            normalised_window=[]
            for col_i in range(window1.shape[1]):
                    normalised_col = [((float(p)/float(window1[0, col_i]))-1) for p in
window1[:,col_i]]
                normalised_window.append(normalised_col)
            normalised_window=np.array(normalised_window).T
            normalised_data.append(normalised_window)
        return np.array(normalised_data)
```

У свою чергу для оптимізації нейронної мережі було обрано алгоритм «Adam», саме цей оптимізатор буде найпідходящим для даної моделі нейронної мережі.

Після цього потрібно розділити дані на тренувальні та на дані для тесту, тоб-

то крос-валідація даних. Для цього було поділено дані: 85% даних це дані для тренування, а решта даних це дані для тесту, які не будуть брати участі у навчанні. Крос-валідацію даних у коді наведено на лістингу 3.2.

Лістинг 3.2 - Крос-валідація даних

```
split = 0.85
        i_split = int(len(df)*split)
        cols=["Close", "Volume"]
        data_train = df.get(cols).values[:i_split]
        data_test = df.get(cols).values[i_split:]
        len_train = len(data_train)
        len_test = len(data_test)
```

Коли дані будуть відформатовані згідно трьох парадигм, можна почати процедуру створення моделі нейронної мережі рисунок 3.9. Для прикладу буде описано модель нейронної мережі згідно рекурентної архітектури (рисунок 2.11). Для створення такої моделі було використано фреймворк «Keras» зі його рекурентними шарами.

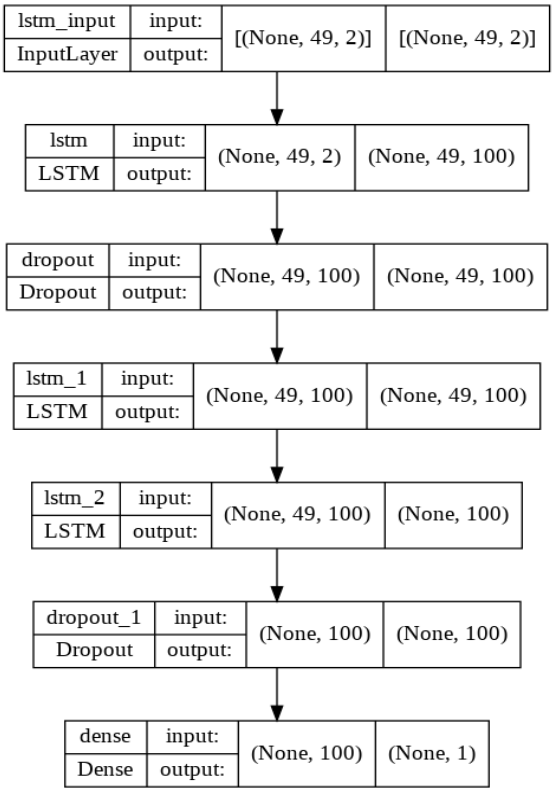

Рисунок 3.9 – Розгорнута рекурентна архітектура нейронної мережі

На рисунку 3.9 представлена розгорнута рекурентна архітектура нейронної мережі. Згідно цій архітектурі є вхідній шар, який приймає дані та має 49 нейронів, цей вхідний шар працює у парі зі «LSTM» шаром, який також має 49 нейронів, таким чином вхідний шар має 100 нейронів. Далі йде наступний шар – шар «Dropout», цей шар відповідає за перенавчання, а саме для його запобігання. Після шару шар «Dropout», йде знов шар «LSTM», який має 100 нейронів, потім знов шар «LSTM», після чого нейрони підвергаються регуляризації шаром «Dropout». Далі дані поступають до лінійної функції активації, після чого прогнозуються значення, ці значення зіставляються зі даними для тесту, у разі не співпадіння – змінюються вагові коефіцієнти, і знов зіставляються зі даними для тесту, тобто працює алгоритм «зворотного розповсюдження помилки».

3.2 Тестування програми

Для того щоб протестувати коректність функціоналу програми – було прийнято рішення користуватися програмою від імені користувача, а саме, виконання всіх можливих дій у програмі для виявлення помилок які не оброблені у коді.

Під час тестування було виявлено декілька помилок: помилка при введені некоректних даних тренування для нейронної мережі, для вирішення цієї помилки потрібно проводити інвентаризацію даних алгоритмом, таким чином виявляти коректність даних, ще одна помилка пов’язана зі роботою нейронної мережі, а саме не цілісна нормалізація даних, що приводить до неточності при прогнозування, для вирішення цієї помилки потрібно змінити функцію нормалізації на відсоткове співвідношення.

Всі ці зміни будуть інтегровані у майбутніх оновленнях програми

3.3 Висновок до розділу 3

В даному розділі було створено та проаналізовано функціонал програми, а

саме створено інтерфейс програми та описано його функціонал, спроектовано повну архітектуру нейронної мережі, а також саму нейронну мережу за цією архітектурою. Також в цьому розділі було проведено тестування програми, де були виявленні помилки, а також можливість їх усунення.

ВИСНОВКИ

В ході виконання випускної роботи молодшого спеціаліста, мною було досягнуто поставлених цілей та задач, а саме було проведено аналіз літератури, що направлена на розробку програми, а саме дослідження книжок та наукових статей щодо розробки та використання нейронних мереж, зокрема, використання нейронних мереж для прогнозування часових рядів. Було проведено огляд існуючих аналогів, а також попит використання нейронних мереж.

Головною метою виконання випускної роботи молодшого спеціаліста - розробити модель факторного аналізу фінансових результатів, проаналізувати існуючі нейронні мережі для подальшого аналізу цих рядів для створення обліку, а також використання можливостей технології «TensorFlow». При розробці програми ця мета була досягнена.

При написання випускної роботи молодшого спеціаліста були виконані наступні завдання:

1. Досліджено літературу стосовно створення програми;
2. Виконано огляд мов програмування, середовищ розробки, технологій для створення нейронної мережі, видів систем управляння базами даних, а також середовищ розробок інтерфейсу;
3. Протестовано різні моделі нейронних мереж у програмі, завдяки чому було виявлено найпідходящу модель нейронної мережі;
4. Протестована програма для аналізу часових рядів на виявлення помилок;
5. Розроблено програму на тему випускної роботи молодшого спеціаліста та зроблено висновок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Часові ряди [Електронний ресурс]. Метод доступу: www. URL: https:// ru.wikipedia.org/wiki/Временной ряд – 02.06.2022
2. Часові ряди [Електронний ресурс]. Метод доступу: www. URL: https:// habr.com/ru/post/553658/ – 02.06.2022
3. Книга про нейронні мережі та часові ряди [Електронний ресурс]. Метод доступу: www. URL: https://scask.ru/p_book_ins.php?id=40 – 01.06.2022
4. Актуальність нейронних мереж [Електронний ресурс]. Метод доступу: www. URL: https://bigenc.ru/technology_and_technique/text/4114009 - 30.05.2022
5. Стаття про нейронну мережу [Електронний ресурс]. Метод доступу: www. URL: https://ru.wikipedia.org/wiki/Нейронная_сеть - 27.05.2022
6. Стаття про важливість залежності значення від часу [Електронний ресурс]. Метод доступу: www. URL: https://www.researchgate.net/publication/ 267555543_Comparative_analysis_of_Recurrent_and_Finite_Impulse_Response_ Neural_Networks_in_Time_Series_Prediction - 31.05.2022
7. Книга про нейронні мережі [Електронний ресурс]. Метод доступу: www. URL:https://codernet.ru/books/python/ glubokoe_obuchenie_na_python_sholle_fransua/ - 20.05.2022
8. Застосування глибоких нейронних мереж для короткострокового прогнозу [Електронний ресурс]. Метод доступу: www. URL: https://zenodo.org/record/ 3253019#.YqM-PmhBxD8 – 23.05.2022
9. Програма аналог GMDH [Електронний ресурс]. Метод доступу: www. URL: https://gmdhsoftware.com/ – 02.06.2022
10. Програма аналог NeuroShell Trader [Електронний ресурс]. Метод доступу: www. URL: https://try.neuroshell.com/index/ – 02.06.2022
11. Програма аналог BrainMaker [Електронний ресурс]. Метод доступу: www. URL: https://quantpro.ru/archives/4865 – 02.06.2022
12. Про мову програмування Python [Електронний ресурс]. Метод доступу: www. URL: https://www.python.org/ - 03.06.2022

13. Про мову програмування Python та нейроні мережі [Електронний ресурс]. Метод доступу: www. URL: https://proproprogs.ru/data - 29.05.2022
14. Про історію створення мови програмування Python [Електронний ресурс]. Метод доступу: www. URL: https://ru.wikipedia.org/wiki/Python - 28.05.2022
15. Про фреймворк Keras [Електронний ресурс]. Метод доступу: www. URL: https://keras.io/ - 01.06.2022
16. Про фреймворк Keras та TensorFlow [Електронний ресурс]. Метод доступу www. URL: https://www.tensorflow.org/ - 20.05.2022
17. Про середовище розробки PyCharm [Електронний ресурс]. Метод доступу www. URL: https://www.jetbrains.com/ru-ru/pycharm/ - 29.05.2022
18. Про середовище розробки PyCharm [Електронний ресурс]. Метод доступу www. URL: https://ru.wikipedia.org/wiki/PyCharm - 29.05.2022
19. Про середовище розробки інтерфейсів PyQt [Електронний ресурс]. Метод доступу www. URL: https://riverbankcomputing.com/software/pyqt/ - 30.05.2022
20. Про середовище розробки інтерфейсів PyQt [Електронний ресурс]. Метод доступу www. URL: https://ru.wikipedia.org/wiki/PyQt - 30.05.2022
21. Про модуль Matplotlib [Електронний ресурс]. Метод доступу www. URL: https://matplotlib.org/ - 23.05.2022
22. Про модуль NumPy [Електронний ресурс]. Метод доступу www. URL: https://numpy.org/ - 24.05.2022
23. Про модуль csv [Електронний ресурс]. Метод доступу www. URL: https:// pythonworld.ru/moduli/modul-csv.html - 26.05.2022
24. Про СУБД MongoDB [Електронний ресурс]. Метод доступу www. URL: https://www.mongodb.com/ - 26.05.2022
25. Про СУБД MongoDB [Електронний ресурс]. Метод доступу www. URL: https://ru.wikipedia.org/wiki/MongoDB - 26.05.2022
26. Про NoSQL СУБД [Електронний ресурс]. Метод доступу www. URL: https://ru.wikipedia.org/wiki/NoSQL - 26.05.2022
27. Проектування бази даних [Електронний ресурс]. Метод доступу www.

URL: https://habr.com/ru/post/514364/ - 27.05.2022

28. Про UML-діаграми [Електронний ресурс]. Метод доступу www. URL: https://ru.wikipedia.org/wiki/UML - 29.05.2022

ДОДАТОК А

ЛІСТИНГ КОДУ