Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет Комп'ютерні технології та системи
Кафедра Комп'ютерні інформаційні технології

Пояснювальна записка
До кваліфікаційної роботи
магістра

на тему: «Прогнозування динаміки цін на українській міжбанківській біржі за допомогою нейромережевих технологій»
за освітньою програмою інформаційні технології
зі спеціальності: 121 Інженерія програмного забезпечення.

Виконав: студент групи ПЗ2121М: /Денис ГІЛЯВСЬКИЙ/

Керівник: /Світлана ВОЛКОВА/

Нормоконтролер: /Світлана ВОЛКОВА/

Консультанти:
Економічний розділ /Микола ГНЕНИЙ/

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент

Дніпро – 2022 рік

Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет Комп'ютерні технології та системи
Кафедра Комп'ютерні інформаційні технології

Пояснювальна записка
До кваліфікаційної роботи
магістра

на тему: «Прогнозування динаміки цін на українській міжбанківській біржі за допомогою нейромережевих технологій»
за освітньою програмою інформаційні технології
зі спеціальності: 121 Інженерія програмного забезпечення.

Виконав: студент групи ПЗ2121М: /Денис ГІЛЯВСЬКИЙ/

Керівник: /Світлана ВОЛКОВА/

Нормоконтролер: /Світлана ВОЛКОВА/

Консультанти:
Економічний розділ /Микола ГНЕНИЙ/

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент

Дніпро – 2022 рік

Ministry of Education and Science of Ukraine

Ukrainian State University of Science and Technologies

Faculty Computer technologies and systems
Department Computer information technology

Explanatory Note
To Master's Thesis
master

on the topic: «Forecasting price dynamics on the Ukrainian interbank exchange using neural network technologies»
according to educational curriculum information technologies
in the Speciality: 121 software engineering.

Done by the student of the group ПЗ2121М: /Denys HILIAVSKYI/

Scientific Supervisor: /Svitlana VOLKOVA/

Normative controller: /Svitlana VOLKOVA/

Supervisors:
Economic part /Mykola GNENNIYN/

Dnipro – 2022

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Комп'ютерних технологій і систем
Кафедра: Комп’ютерні інформаційні технології
Рівень вищої освіти: магістр
Освітня програма: Інформаційні технології
Спеціальність: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
_____ грудня 2022 р.

ЗАВДАННЯ

На кваліфікаційну роботу ОС Магістр
студенту Гілявському Денису Олександровичу

1. Тема дипломної роботи: Прогнозування динаміки цін на український міжбанківській біржі за допомогою нейромережевих технологій.
   Керівник роботи: Волкова Світлана Анатолівна
   затверджені наказом від 173ст від 11.02. 2022 року.

2. Строк подання студентом роботи 21.12.2022 року

3. Вихідні дані до дипломної роботи: результат навчання нейронної мережі, результат роботи нейронної мережі, функції аналізу даних, графікі зміни цін на акції.

4. Зміст пояснювальної записки (перелік питань до розробки):
   4.1. Аналітична частина: огляд предметної галузі;
   4.2. Основна частина : огляд існуючих рішень, проектування та розробка програмного забезпечення;
   4.3. Дослідження ефективності використання нейромереж;
   4.4. Економічна частина: техніко-економічні розрахунки;
   4.5. Загальні висновки.

5. Перелік демонстраційного матеріалу:
   5.1. доповідь;
   5.2. презентація;
   5.3. демонстраційне відео.

6. Консультанти:

Розділ	Консультант	Завдання видав	Завдання прийняв
Техніко-економічні розрахунки	доц. Гнений М.В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Срок виконання етапів	Примітка
1	Вступ	03.09.22 – 14.09.22	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	14.09.22 – 12.10.22	
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	12.10.22 – 18.10.22	
4	Постановка задачі, технічне завдання	18.10.22 – 20.10.22	
5	Техніко-економічні показники	20.10.22 – 07.11.22	
6	Розробка інструментальних засобів дослідження	08.11.22 – 13.11.22	
7	Виконання досліджень	14.11.22 – 18.11.22	
8	Оформлення тез доповідей	19.11.22 – 23.11.22	
9	Оформлення статті у фаховий журнал	24.11.22 – 28.11.22	
10	Оформлення пояснювальної записки	29.11.22 – 05.12.22	
11	Розробка демонстраційних матеріалів	02.09.22 – 15.09.22	
12	Подання кваліфікаційної роботи до кафедри	23.12.22	
13	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	27.12.22	

Студент Денис ГІЛЯВСЬКИЙ

Керівник роботи Світлана ВОЛКОВА

РЕФЕРАТ

Пояснювальна записка до кваліфікаційной роботи магістра 176 с., 70 рис. , 8 табл., ___ додатки, 58 джерел.

Об’єктом дослідження є нейронні мережі за визначеними критеріями. Предметом дослідження є прогнозування значень часових рядів за допомогою методів машинного навчання.

Метою дослідження в контексті даної роботи є виявлення підходящої для прогнозування архітектури нейронних мереж, її реалізація за допомогою бібліотек Numpy та TensorFlow для Python, як найбільш популярних фреймворків для побудови передбачуваних моделей, оцінка точності прогнозів моделі та підбір оптимальних параметрів.

Методи дослідження: навчання нейромережі за методом регресії, який передбачає те, що минулі значення тимчасового ряду будуть використовуватися для передбачення наступного значення. При розробці програмної реалізації побудованого методу була застосована технологія об’єктно-орієнтованого проектування, та моделювання на основі UML.

Пояснювальна записка складається зі вступу, чотирьох розділів, висновків, бібліографічного списку та восьми додатків.

Вступ описує суть, мету та актуальність роботи (2 сторінки).

Перший розділ: огляд предметної галузі (18 сторінок).

Другий розділ: аналіз та проблеми сучасних нейронних мереж (15 сторінок).

Третій розділ проектний розділ (30 сторінок).

Четвертий розділ: дослідження нейромереж (14 сторінки).

Додатки: діаграма варіантів використання, технічне завдання, робочий проект, опис програми, керівництво користувача,текст програми, тези двох доповідей.

Ключові слова: нейронная мережа; нейрон; графічне представлення роботи нейронної мережі; алгоритм навчання; датасет.

ЗМІСТ

ВСТУП

Актуальність роботи. Використання нейронних мереж у задачах прогнозування, на сьогодні є найефективнішим методом. Нейромережі мають багато переваг над іншими алгоритмами. За допомогою нейронних мереж можна легко досліджувати залежність прогнозованих даних від незалежних змінних. Як приклад можна навести таке: є припущення, що стан підприємства через місяць якимось чином залежить від економічних, політичних, ринкових, виробничо-технологічних, конкурентних та міжнародних. Отже, необхідно побудувати систему, яка все це враховуватиме, і на основі цих даних даватиме короткострокові прогнози. Використання більшої частини стандартних методів прогнозування у цій задачі просто неможливе.

Досить легко отримати працюючу систему прогнозування, навіть використовуючи найпростішу нейромережну структуру (перцептрон з одним прихованим шаром) та базу даних. Облік або облік системи зовнішніх параметрів визначатиметься включенням, або виключенням відповідного входу в нейронну мережу. Прогнозист може використовувати будь-який алгоритм для визначення вагомості вхідних змінних, щоб потім можна було виключити з обліку параметри, які незначно впливають.

Нейромережа, що розроблюється може бути додатковим елементом при аналізі роботи навчання та прогнозування нейронних мереж

Тема роботи: «Прогнозування динаміки цін на українській міжбанківській біржі за допомогою нейромережевих технологій».

Об’єктом дослідження є нейронні мережі за визначеними критеріями.

Предметом дослідження є прогнозування значень часових рядів за допомогою методів машинного навчання.

Мета та задачі дослідження. Виявлення підходящої для прогнозування архітектури нейронних мереж, її реалізація за допомогою бібліотек Numpy та

TensorFlow для Python, як найбільш популярних фреймворків для побудови передбачуваних моделей, оцінка точності прогнозів моделі та підбір оптимальних параметрів.

Для досягнення мети буде поставлено та вирішено такі задачі:

– дослідження відомих методів розробки нейронних мереж;

– дослідження відомих методів навчання нейронних мереж;

– дослідження відомих методів усунення перенавчання нейронних мереж;

– розробка нейронної мережі, яка навчається та проводить прогнози.

Методи дослідження. навчання нейромережі за методом регресії, який передбачає те, що минулі значення тимчасового ряду будуть використовуватися для передбачення наступного значення. При розробці програмної реалізації побудованого методу була застосована технологія об’єктно-орієнтованого проектування, та моделювання на основі UML.

Наукова новизна. Розроблено реккуретну мережу та метод усунення її перенавчання.

Практичне значення. Впровадження розробленної нейромережі дозволяє вирішити завдання, спираючись на викривлену, неповну інформацію. Головною перевагою від використання нейронної мережі для обробки даних є набагато збільшена швидкодія процесу в порівнянні зі звичайними математичними методами, навчання нейронних мереж за зразками та зміна топології мережі, виходячи з вимог розв'язуваного завдання.

Апробація результатів дослідження та публікації. Результати магістерської роботи доповідались на семінарі кафедри КІТ 05.11.2022 р., представлено конференціях: VI International Scientific and Practical Conference “MODERN RESEARCH IN WORLD SCIENCE”, (4-6 September 2022 p..), VII International Scientific and Practical Conference//“MODERN RESEARCH IN WORLD SCIENCE” 24 Hours of Participation (2-4 October 2022 p.)

1 ОГЛЯД ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Штучний нейрон та штучні нейронні мережі

Штучні нейрони – спрощена модель біологічних нейронів, їх прототипи. Такі нейрони це вузли штучної нейронної мережі (ШНМ). Нейрони можуть приймати певну кількість сигналів і на основі цих сигналів вони генерують вихідний сигнал, який далі подається на вхід або вихід наступним нейронам. Коли нейрони з'єднані між собою, вони утворюють ШНМ. На рисунку 1.1 показано модель штучного нейрона.

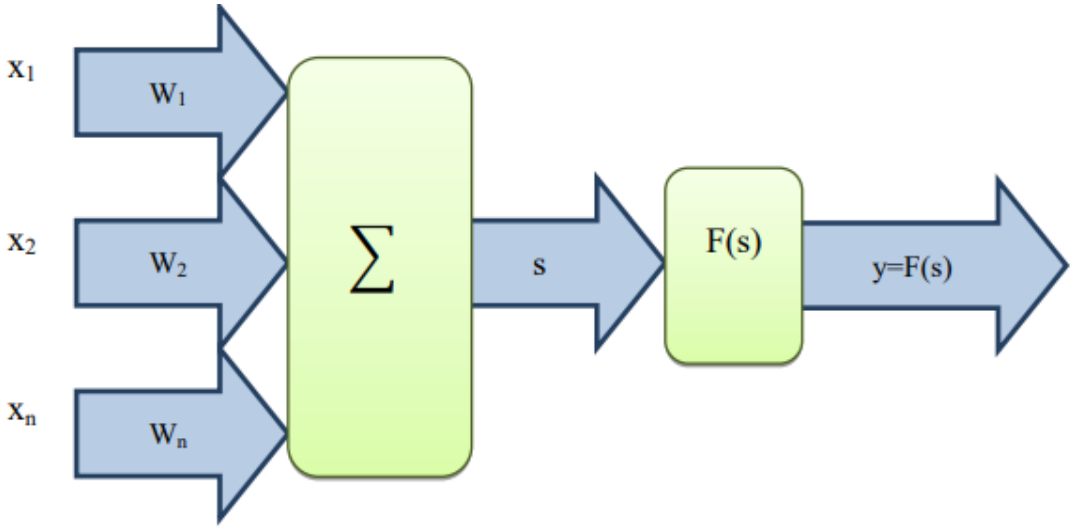


Рис. 1.1 – Модель штучного нейрона

У цій моделі всі вхідні нейрони множаться на відповідний ваговий коефіцієнт. Після цього всі сигнали потрапляють до суматора, в якому відбувається визначення рівня активації нейрона.

Функція активації f(x) відображає залежність вихідного сигналу нейрона від суми зважених вхідних. Для нелінійності роботи нейрона застосовують певні передавальні функції.

Порогова передавальна функція є перепадом, що далі показано на рисунку 1.2. Якщо вхідне значення буде більше порогового, то значення функції активації дорівнює максимально допустимому, в іншому випадку значення буде

мінімально допустимим.

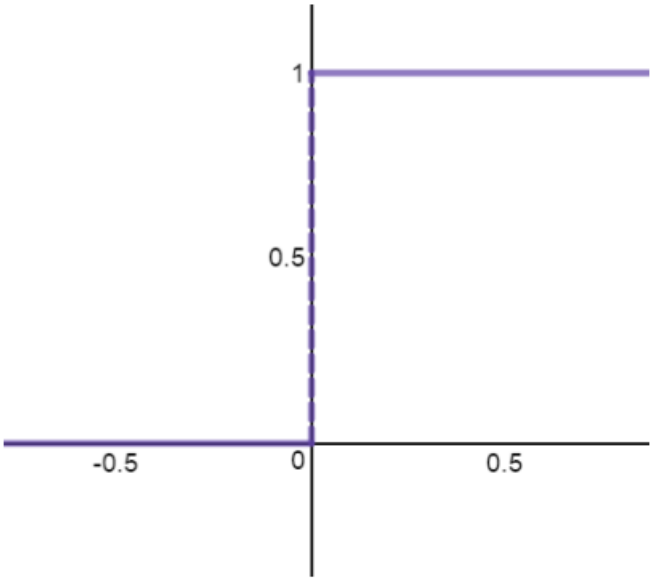


Рис. 1.2 – Порогова функція активації

Лінійна передатна функція представлена далі на рисунку 1.3. На ній присутні дві лінійні ділянки, в яких функція активації тотожно дорівнює максимально і мінімально допустимому значенню, а також є ділянка, де функція строго монотонно зростає.

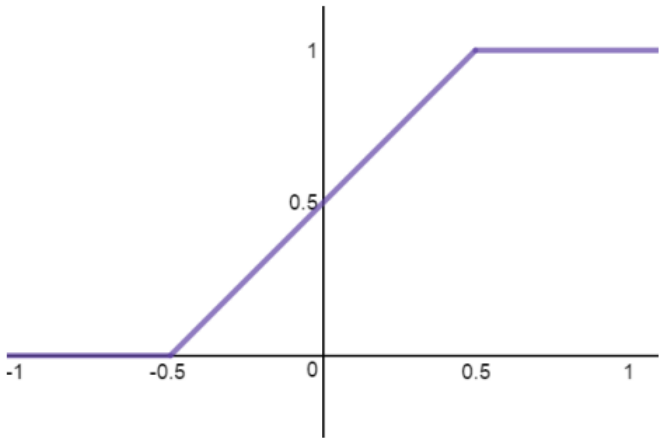


Рис. 1.3 – Лінійна функція активації з начисленням

Обмеженість нейромереж з граничною функцією активації обумовлено

введенням функції сигмоїдального типу. Далі на рисунку 1.4 представлено, що функція активації відбувається плавно, завдяки чому можна перейти від бінарних виходів до аналогових. Особливістю сигмоїдальної функції активації є те, що вона не насичується від сильних сигналів, а слабкі сигнали вона посилює.

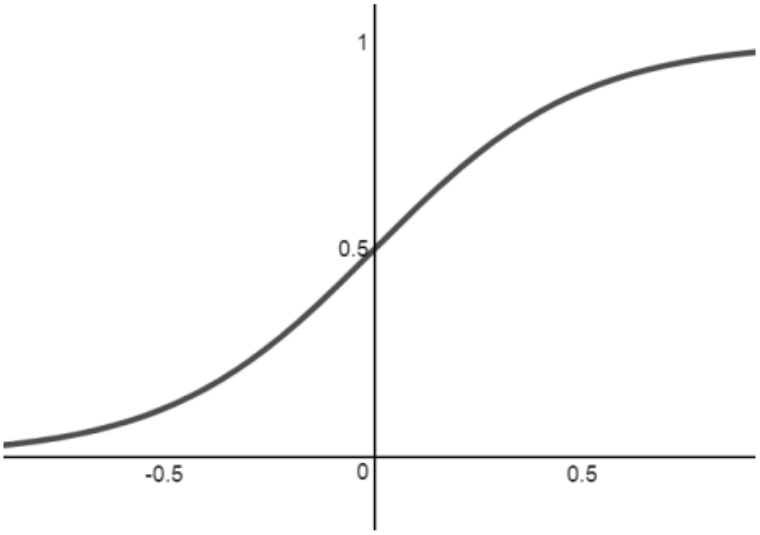


Рис. 1.4 – Сигмоїдальна функція активації

Сигмоїдальний сигнал можна задати різними способами, наприклад, логістична функція і гіперболічний тангенс, у яких відмінність тільки в масштабі осей.

Дуже рідко зустрічаються функції активації типу одиничного стрибка та лінійного порога. Найчастіше використовують саме сигмоїдальну функцію активації.

З цього слідує, що пов'язані між собою нейрони утворюють нейронну мережу. Самі нейромережі, у свою чергу, відрізняються архітектурою: структурою зв'язків серед нейронів, кількістю шарів, функцією активації, алгоритмом навчання.

Нейронні мережі в біології утворюються завдяки нейронам та всіляким сполукам між ними. Однак для реалізації сучасних нейромереж є деякі фізичні

обмеження.

Нейрони, що об'єднуються в мережі, утворюють системи обробки інформації. Такі моделі адаптуються до постійних змін довкілля. Під час роботи відбувається процес перетворення вхідного вектора сигналів у вихідний. Те, який буде вид перетворення вже залежить від архітектури нейромережі, засобами управління та синхронізації інформаційних потоків між нейронами та характеристиками самих нейронів.

Одним з важливих факторів ефективності ШНМ є встановлення кількості нейронів і типу зв'язку між ними.

При описі нейромереж часто використовуються дані терміни:

- структура - спосіб зв'язків нейронів у ШНМ;

- архітектура – структура ШНМ та типи нейронів;

- парадигма - спосіб навчання та використання.

Допускається, що можуть бути реалізовані нейронні мережі різної структури з однаковою парадигмою і навпаки.

На наступному рисунку 1.5 зображені групи нейронних мереж, у яких кожен нейрон пов'язаний лише з сусідніми нейронами, таке називають слабозв'язаною нейромережею (персептрон, нейромережі прямого поширення та ін.)

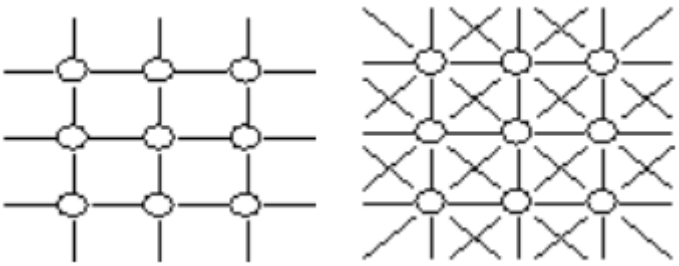


Рис. 1.5 – Слабозв'язані нейромережі

На рисунку 1.6 представлена повнозв'язкова нейронна мережа. У ній вихід кожного нейрона пов'язані з входами решти (мережі Кохокена, мережі Хопфілда).

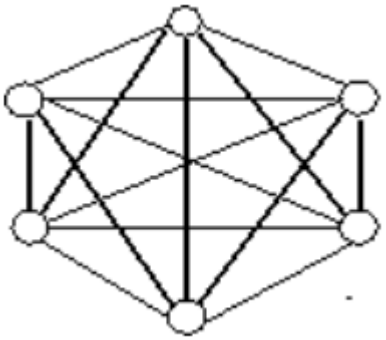


Рис. 1.6 – Повнозв'язана нейромережа

На наступному рисунку 1.7 представлено найпоширеніший варіант архітектури - багатошарова нейронна мережа. У цій архітектурі нейрони об'єднуються у шари, які мають єдиний вектор сигналів входів. Зовнішній вхідний вектор подається на рецептори, яким є шар нейронної мережі. Ефектори – виходи нейромережі, а саме вихідні сигнали останнього шару. Крім вхідних і вихідних шарів ШНМ може мати один або кілька прихованих шарів нейронів. Такі нейрони позбавлені контакту із зовнішнім середовищем.

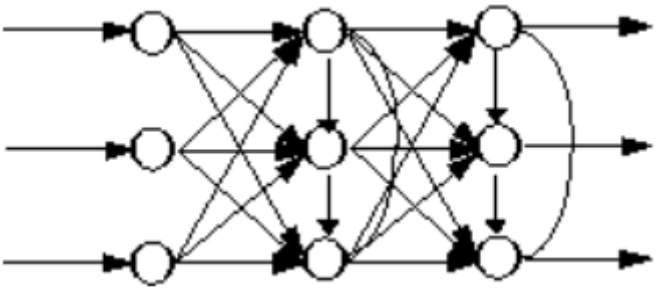


Рис. 1.7 – Багатошарова нейронна мережа

Важливим аспектом нейронних мережах є напрям зв'язків від одного нейрона до іншого. У більшості нейронних мереж кожен нейрон прихованого шару отримує сигнали від усіх нейронів з попереднього шару. Нейрон після виконання операцій над вхідними сигналами передає свій вихід до всіх наступних нейронів, забезпечуючи передачу вперед на вихід. Такі нейронні мережі називають мережами прямого поширення.

Існують також мережі із зворотним зв'язком, у яких сигнал виходів з

нейронів прямує до нейронів з попереднього шару (рисунок 1.8).

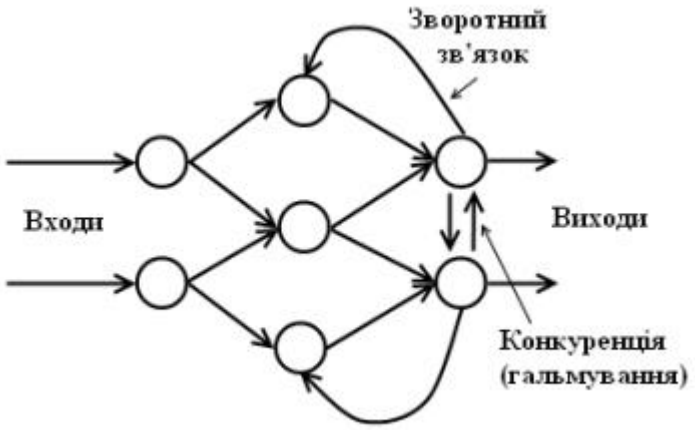


Рис. 1.8 – Рекурентна мережа

Шлях, яким з’єднуються нейрони між собою має значний вплив на роботу мережі.

1.2 Рекурентні нейронні мережі

Рекурентні нейронні мережі — це нейронні мережі, які запрограмовані на обробку послідовних значень. Дуже часто їх застосовують для:

- обробки природної людської мови;
- аналізу написаного тексту;
- машинного перекладу тексту;
- генерації тексту;
- генерації чисел;
- та ін.

Працюючи з текстами вони здатні оцінювати його граматичну і семантичну коректність. Рекурентні нейронні мережі здатні проаналізувати будь-який текст, а потім на основі цього аналізу скласти читальний та схожий на проаналізований текст. Наприклад, можна взяти текст Т. Г. Шевченко, проаналізувати його, а потім спробувати згенерувати схожий текст. Рекурентні

нейронні мережі здатні зробити текст зі схожим стилем.

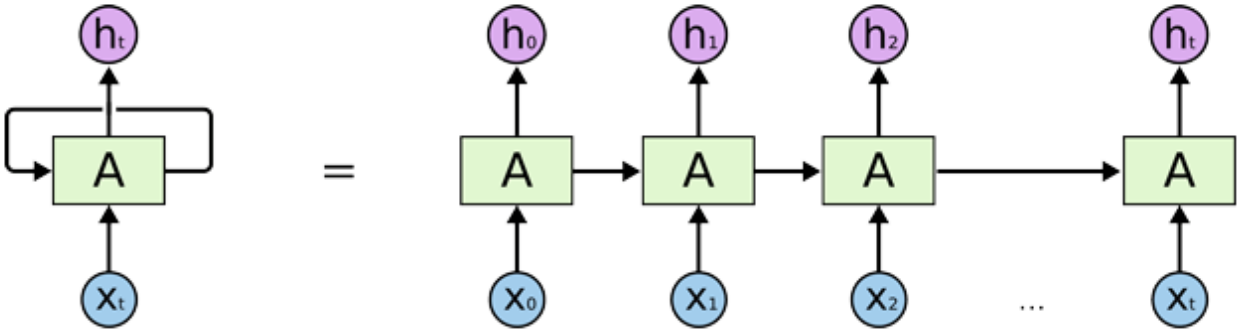


Рис. 1.9 – Схема одношарової рекурентної нейронної мережі

Ідея рекурентності полягає в тому, щоб послідовно використати інформацію. У традиційних нейромереж обробка інформації йде за іншим принципом: там немає послідовності, лише «хаотичність» і незалежність входів і виходів при обробці інформації. Такий підхід не може ефективно працювати у всіх випадках. Наприклад, потрібно визначити якесь значення, але за основу необхідно взяти попереднє значення. У цьому випадку традиційна нейронна мережа не впорається із завданням, а рекурентна впорається. Чому це стає можливим? Тому що в рекурентній нейромережі є таке поняття, як пам'ять. У пам'яті зберігається інформація про попереднє значення і на основі попереднього визначається наступне.

Рекурентні нейронні мережі працюють із заданими значеннями. Вони приймають певні фіксовані вхідні значення та у відповідь дають такий же фіксований результат. Рекурентні мережі досліджують значення за чітко заданими принципами. Відштовхуючись від принципів дослідження значень, рекурентні мережі бувають наступних типів:

1. "Один до одного". Такий тип нейромережі застосовний, коли на вхід надходить поодинока інформація і на виході також виходить поодинока інформація. Наприклад, такий підхід актуальний при кодуванні та розкодуванні інформації.

2. "Один до багатьох". Такий тип нейромережі застосовується, коли на вході надходить поодинока інформація, а на виході результатом є послідовність. Наприклад, на вхід надходить одне зображення, а на вихід текстовий опис того, що зображено на зображенні.

3. «Багато до одного». Такий тип нейромережі застосовується, коли на вхід надходить безліч інформації, а на виході результатом є єдиний результат. Наприклад, на вхід йде безліч зображень, а на вихід на всіх зображеннях знаходиться "автобус".

4. «Багато до багатьох». Такий тип нейромережі застосовується, коли вхід надходить послідовність інформації, але в виході результатом є змінена послідовність інформації. Наприклад, при машинному перекладі тексту на вхід надходить текст однією мовою, а на виході виходить текст іншою мовою.

Рекурентні нейронні мережі застосовуються у таких областях:

1. Мовне моделювання та генерування тексту. Досліджується послідовність слів. На основі її дослідження нейромережа здатна "передбачити" ймовірність кожного наступного слова. У такий спосіб формуються тексти. Для того, щоб нейронна мережа згенерувала якийсь текст, їй на вхід потрібно надати шаблон тексту для генерування.

2. Машинний переказ. Ця сфера трохи схожа на попередню, тому що як вхідні дані використовується послідовний текст. Вводиться послідовний текст вихідної мови, наприклад текст російською. Потім вводиться послідовний текст мови, якою потрібно перекладати, наприклад, текст англійською. Нейронна мережа вивчає обидва тексти та зіставляє послідовність перекладу слів.

3. Розпізнавання людської мови. Для такої дії як вхідні дані для навчання нейромережі застосовують послідовність акустичних сигналів у вигляді аудіозаписів. Аналізуючи аудіозаписи, нейромережа буде здатна передбачати можливу послідовність людської мови.

4. → Генерація· зображень.· У· «чистому»· вигляді· рекурентні· нейронні· мережі·не·в·змозі·генерувати·зображення,·але·в·тандемі·з·нейронними·мережами· згортками·роблять·це·досить·легко.¶

Нескладно·вигадати,·де·використовувати·рекурентні·мережі,·якщо·вони· якісно·натреновані.·Складніше·правильно·натренувати·мережу,·адже·рекурентна· мережа·має·одну·проблему·—·маленьку·пам'ять.·Пам'ять·не·здатна·"утримати"· великі· послідовності.· Через· це,· коли· пам'ять· перевантажується· інформацією,· нейромережа·видає·помилки·і·навчається·негаразд·ефективно,·як·хотілося·б.·Щоб· частково·вирішити·цю·проблему,·було·трохи·«розширено»·блоки·пам'яті.·Таким· чином,·у·зв'язку·з·різним·обсягом·осередку·пам'яті·розрізняють·декілька·видів· рекурентних·мереж.¶

Залежно від осередку пам'яті, розрізняють 3 категорії рекурентних мереж:

1. → Звичайні· рекурентні· мережі· RNN.· Проблема· з· переповненням· пам'яті·пов'язана·з·ними,·тому·їх·використовують·у·тих·завданнях,·коли·потрібно· не·генерувати·багато·значень,·а·автоматично·доповнювати·якусь·інформацію.¶

2. → Рекурентна·мережа·з·довгою·пам'яттю,·вона·LSTM.·Подібні·мережі· вирішили· проблему· з· пам'яттю· за· рахунок· розширення· осередку· пам'яті.· Такі· мережі·годяться·до·виконання·складних·завдань,·наприклад,·з·допомогою·можна:· складати·музику,·генерувати·текст,·схожий·за·стилем·текст·Т.·Г.·Шевченко·та· інших.· Але· на· відміну· першого· виду,· вони· споживають· багато· системних· ресурсів.¶

3. → Керовані·рекурентні·мережі,·вони·ж·є·GRU.·По·суті,·є·лайт-версією· попереднього·виду·мережі.·Осередки·з·таким·самим·обсягом·пам'яті,·але·обробка· інформації·йде·дещо·іншим·шляхом,·злегка·«обрізаним»·за·функціональністю.¶

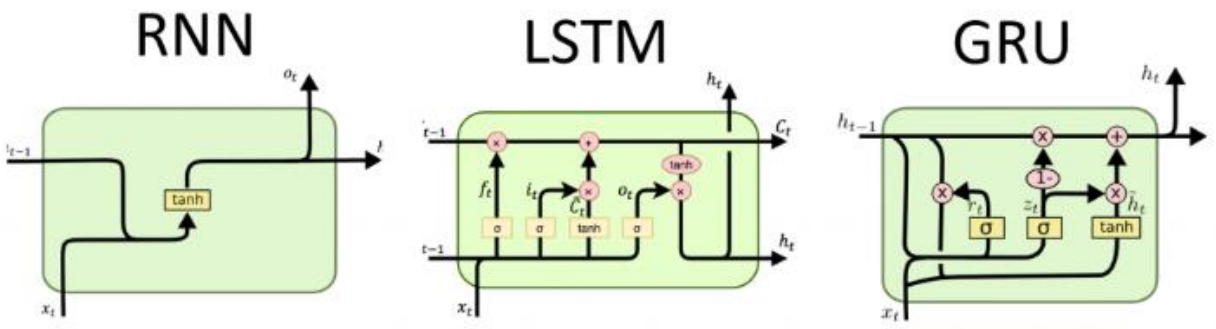


Рис. 1.10 – Рекурентні нейронні мережі

1.3 Основні поняття

Моделювати· економічні· процеси· можна· з· допомогою· традиційних· математичних·методологій,·і·навіть·з·допомогою·сучасних·способів.·До·таких· методів·належать·нейронні·мережі.·При·використанні·нейронних·мереж·можна· вирішувати·завдання,·які·не·можуть·бути·вирішені·за·допомогою·традиційних· методів.· Нейронна· мережа· дозволяє· вирішити· завдання,· спираючись· на· викривлену,· неповну· інформацію.· Головною· перевагою· від· використання· нейронної·мережі·для·обробки·даних·є·набагато·збільшена·швидкодія·процесу·в· порівнянні·зі·звичайними·математичними·методами,·навчання·нейронних·мереж· за·зразками·та·зміна·топології·мережі,·виходячи·з·вимог·розв'язуваного·завдання.¶

Часовий· ряд· -· це· масив· точок· будь-яких· даних,· проіндексованих· (· перелічених,·або·відкладених·на·графіку)·в·хронологічному·порядку.·Найчастіше· часовий·ряд·є·послідовністю,·взятою·на·рівновіддалених·точках·в·часі,·які·йдуть· одна·за·одною.·Отже,·часовий·ряд·є·послідовністю·даних·у·дискретному·часі.· Прикладами·часових·рядів·є·температура·повітря,·швидкість·вітру,·економічний· індекс·фондової·біржі,·тощо.¶

Найкращим способом представлення часових рядів є лінійні діаграми. Часові ряди використовують певною мірою в усіх галузях науки та інженерії, які включають до себе часові вимірювання статистика, обробка сигналів, розпізнавання образів, економетрика, фінансова математика, прогнозування

погоди, розумний транспорт та передбачення траєкторій, передбачення землетрусів, електроенцефалографія, автоматичне керування, астрономія, технології зв'язку, тощо.

Аналіз часових рядів застосовується з метою отримання тих чи інших значимих статистик та важливих характеристик даних цих рядів.

Прогнозування часових рядів – використання моделі для прогнозування майбутніх значень на основі попередньо спостережених.

Аналіз часових рядів може застосовуватися до дійснозначних неперервних, дискретних числових, та дискретних символьних даних (наприклад, послідовностей символів, таких як літери, слоги та слова латинського алфавіту).

Штучні нейронні мережі (ШНМ) – це обчислювальні системи, натхнені біологічними нейронними мережами, що складають мозок тварин. Такі системи навчаються задач (поступально покращують свою продуктивність на них), розглядаючи приклади, загалом без спеціального програмування під задачу.

ШНМ ґрунтується на сукупності з'єднаних вузлів, що називають штучними нейронами (аналогічно до біологічних нейронів у головному мозку тварин). Кожне з'єднання (аналогічне синапсові) між штучними нейронами може передавати сигнал від одного до іншого. Штучний нейрон, що отримує сигнал, може обробляти його, й потім сигналізувати штучним нейронам, приєднаним до нього.

В поширених реалізаціях ШНМ сигнал на з'єднанні між штучними нейронами є дійсним числом, а вихід кожного штучного нейрону обчислюється нелінійною функцією суми його входів. Штучні нейрони та з'єднання зазвичай мають вагу, яка підлаштовується в перебігу навчання. Вага збільшує або зменшує силу сигналу на з'єднанні. Штучні нейрони можуть мати такий поріг, що сигнал надсилається лише якщо сукупний сигнал перетинає цей поріг.

Штучні нейрони зазвичай організовано в шари. Різні шари можуть

виконувати різні види перетворень своїх входів. Сигнали проходять від першого (входового) до останнього (вихідного) шару, можливо, після проходження шарами декілька разів [8].

Нейронна мережа, як поняття, спочатку було з біології. У біології це поняття визначається як мережа, що складається з біологічних нейронів, вони пов'язані і об'єднані в нервовій системі. Нейросеть є спрощеною моделлю мозку людини, вона дуже успішно застосовується при вирішенні всяких різних завдань. В економіці використання штучних нейронних мереж збільшується з часом, адже вони добре проявили себе у вирішенні безлічі прикладних економічних та фінансових завдань. Нейросети вже стали незамінними при якості обробки величезної кількості даних, без них дуже складно, а часом і зовсім неможливо адекватно оцінити поточну ситуацію на ринку і прийняти правильне рішення. Це все вказує на необхідність подальшого вивчення, удосконалення та використання штучних нейронних мереж на практиці.¶

Раціонально використовувати нейронні мережі для вирішення завдань, які потребують трудомістких обчислень. Ці завдання поділяються на:¶

1. Прогнозування. Це перший клас економічних завдань і їх можна вирішити при використанні штучних нейромереж. Їхня здатність до виявлення прихованих залежностей в елементах мережі та їх здатність до узагальнення дозволяє впоратися з такими завданнями. Приклади:¶

- прогнозування рівня попиту на новий товар чи послугу;
- прогнозування обсягів продажу;
- прогнозування поведінки клієнтів;
- аналіз надійності компанії та визначення ймовірності її банкрутства;
- передбачення зміни вартості акцій у певний період;
- прогнозування доцільності впровадження інноваційних проектів та їх економічної ефективності;
- оцінка платоспроможності клієнта та ризику надання йому кредиту.

2. Класифікація об'єктів економічного аналізу. Наступний тип завдань, які можна вирішити з допомогою нейронних мереж. Наприклад, можна класифікувати клієнтів за рівнем ризику надання їм кредиту.

У міжнародній практиці деякі компанії застосовують нейромережі у своїй роботі на вирішення завдань.

Фінансова·компанія·“Citicorp”·є·гарним·прикладом·щодо·використання·штучних·нейронних·мереж.·У·цій·компанії·застосовується·великий·спеціальний·нейрокомп'ютер,·який·дозволяє·аналізувати·та·давати·короткострокове·передбачення·коливання·курсу·валют.·Результати,·отримані·від·нейромережі·цієї·компанії,·перевершили·результати,·отримані·від·брокерів,·які·працюють·у·цій·компанії·довгі·роки.¶

Також,·як·приклад,·можна·розглянути·іншу·фірму,·яка·теж·впровадила·у·свою·діяльність·штучну·нейромережу·для·аналізу·ринку,·ця·фірма·–·“Richard·Borst”.·Ця·компанія·займається·торгівлею·нерухомістю.·Після·того,·як·вони·стали·використовувати·нейромережу,·оборот·у·корпорації·в·Пенсільванії·та·Нью-Йорку·зріс·на·6%,·що·є·великим·успіхом.¶

Нейронні·мережі·-·це·системи,·що·складаються·з·нейронів·(безліч·простих·обчислювальних·елементів),·які·пов'язані·між·собою·певним·чином.·Найпоширеніші·це·багатошарові·мережі,·у·яких·нейрони·з'єднані·у·шари.·Шар·–·це·сукупність·нейронів,·куди·кожен·такт·часу,·паралельно·надходить·інформація·з·інших·нейронів.·Нейрони·з'єднуються·з·входами·інших·нейронів,·саме·так·сигнали·одного·елемента·передається·іншому.¶

.

Коли визначено число елементів у кожному з шарів і кількість шарів, мережу необхідно навчити, тобто потрібно визначити значення для порогів мережі і ваг. Завдяки цьому буде мінімізовано помилку прогнозу. Помилка для конкретного завдання можна визначити, якщо прогнати через нейромережу всі наявні спостереження і провести порівняння результатів, що реально видаються з бажаними. Можна сказати, що навчання нейромережі – це процес припасування

моделі, що реалізується мережею, до вже наявних навчальних даних.

Приклад роботи нейромережі показаний на рисунку 1.11. Вхідний шар служить введення значень вхідних змінних. Кожен із вихідних та прихованих нейронів з'єднані з усіма елементами попереднього шару. В основному віддаються переваги нейромереж з повною системою зв'язків, тому мережі, в яких нейрони пов'язані тільки з деякими з нейронів попереднього шару, розглядати не раціонально.

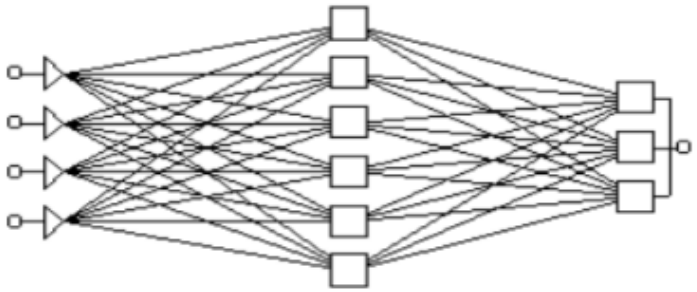


Рис. 1.11 – Схема нейронної мережі

На сьогоднішній день є кілька структур нейромереж. Оскільки всі штучні нейронні мережі містять нейрони, з'єднання і передавальні функції, існує схожість між різними структурами нейромереж. Багато змін проходять виходячи з різних правил навчання. Мати модель зовнішнього середовища, в якій працює нейромережа, необхідно для навчання. Ще потрібно визначити, як саме модифікувати вагові параметри для нейромережі. Правила навчання для налаштування ваги – це алгоритм навчання. Є три загальні типи навчання:

- з учителем;
- без вчителя (самонавчання);
- змішане.

Навчання нейронної мережі означає, що їй повідомляється, чого від неї хочуть досягти. Як вже було сказано вище, нейромережа може навчатися з учителем або без нього. Стабілізація ваг нейронної мережі відбувається після

багаторазового· пред'явлення· прикладів,· у· своїй· нейромережа· дає· правильні· відповіді· на· всі· приклади· з· бази· даних.· Прийнято· говорити· в· такому· разі,· що· "нейронна· мережа· вивчила· всі· приклади"· або· "нейронна· мережа· навчена".· У· програмній·реалізації·можна·побачити,·що·під·час·навчання·величина·помилки· (сума· квадратів· помилок· на· всіх· виходах)· поступово· зменшується.· Нейронна· мережа·вважається·натренованою·та·готовою·до·застосування·нових·даних,·коли· величина·помилки·досягне·нуля·або·прийнятного·рівня.¶

Дуже·важливою·фінансовою·областю,·в·якій·використовуються·нейронні· мережі·є·прогнозування·на·фондовому·ринку.·Існують·стандартні·методи,·що· ґрунтуються· на· фіксованому· наборі· правил,· які· з· часом· втрачають· свою· актуальність·через·зміну·умов·торгів·на·біржі.·Ці·методи·не·використовують· нейромережі.·Системи·такого·типу·є·надто·повільними·у·ситуаціях,·коли·від· учасника· торгів· потрібне· миттєве· ухвалення· рішення.· Тому· використання· нейронних· мереж· є· потужним· методом· прогнозування,· які· дозволяють· відтворювати· дуже· складні· прорахунки.· Переваги· нейромереж· для· прогнозування·фондового·ринку:¶

– → простота·у·використанні,·через·те,·що·нейромережі·навчаються·на· прикладах.·Алгоритм·використання·дуже·простий,·спочатку·користувач·підбирає· представницькі·дані,·після·чого·запускає·навчання·нейромережі;¶

– → нейромережі·залучають·з·інтуїтивної·точки·зору·через·те,·що·вони· засновані·на·біологічній·моделі·нервових·систем;¶

– → необхідний· елемент· інвестиційної· діяльності· –· передбачення· фінансової·складової.¶

Ідея·інвестицій·для·отримання·доходу·в·майбутньому·ґрунтується·на·ідеї· прогнозування.·Фінансове·прогнозування·є·основою·діяльності·індустрії· інвестицій·(бірж·і·позабіржових·систем·торгівлі·цінними·паперами).·Для· якісного·прогнозу·потрібно·якісно·підготувати·дані·та·нейропакет·з·великою· функціональністю.·Існує·безліч·програм·для·роботи·з·нейронними·мережами,· одні·універсальні,·інші·ж·вузькоспеціалізовані.¶

1.4 Постановка проблеми

Актуальною проблемою на сьогодні у плані функціонування та розвитку підприємства є його фінансовий стан. Потрібно розуміти, що 100% прогноз не може дати ні людина, ні машина, тому прогнозування здійснюється певною похибкою. Відбувається це тому, що при прогнозуванні неможливо врахувати всі фактори, які можуть вплинути на діяльність підприємства. Але все ж таки, ймовірність помилки не є критичною проблемою, щоб не проводити прогнозування. За допомогою детальних економічних та фінансових прогнозів можна представляти потенційні можливості підприємства, що дозволить конструктивніше будувати шлях розвитку підприємства. Для максимізації точності та надійності прогнозування необхідно мати повну за обсягом, структурою та змістом інформацію. Дуже важливою частиною діяльності будь-якого підприємства є прогнозування, тому що саме на основі прогнозу очікуваних значень тих чи інших показників приймаються управлінські рішення.

1.5 Постановка завдання

Актуальність та доцільність застосування методів машинного навчання на фінансових ринках продиктована їхньою ефективністю у вирішенні завдань прогнозування часових рядів. Нейронні мережі вирішують такого роду завдання набагато ефективніше за класичні алгоритми статистичного аналізу, побудованих на ковзних середніх, авторегресії та умовній гетероскедастичності.

Ця тема отримала відображення в науковій літературі в останнє десятиліття завдяки таким дослідникам, як Siami-Namini S., Siami-Namin A. (Режим зміни та тенденції Prediction for Bitcoin Time Series Data), Fokkema D. (Machine Learning in Finance), Shah D., Zhang K. (Bayesian Regression and Bitcoin), Lahmiri S. (A Comparison of PNN and SVM for Stock Market Trend Prediction Using Economic and Technical Information), роботи яких докладно описано у п. 1.5. Значний внесок у розвиток прогнозування тимчасових рядів та машинного

навчання зробили Cortes, C. Vapnik, V. (Support-vector networks, 1995), Elman, J. L. (Finding Structure in Time, 1990), Gianluca Bontempi (Machine Learning Strategies for Time Series , 2013) та ін.

Для ефективнішого функціонування підприємства, постійно створюються та вдосконалюються безліч моделей та методів, а також спеціальні програмні засоби. Тим не менш, безліч методів містять істотний недолік, який дає можливість описати множину процесів лінійної залежності, та можливість надати однозначність стаціонарного рішення в системі лінійних рівнянь. Цим недоліком є лінійність, що робить її недостатньо коректною. Саме в таких випадках стає розумно використовувати нейронні мережі як спосіб моделювання цих моделей і методів. Мета дослідження розібратися у поняттях нейронних мереж та в основах її роботи. Дослідити застосування штучних нейронних мереж для задач прогнозування та моделювання фінансових та економічних процесів, а також дослідити різні види програмного забезпечення для роботи з нейронною мережею.¶

Об'єкт дослідження – нейронні мережі за визначеними критеріями.¶

Предмет дослідження – прогнозування значень часових рядів за допомогою методів машинного навчання.¶

Мета роботи – виявлення підходящої для прогнозування архітектури нейронних мереж, її реалізація за допомогою бібліотек Numpy та TensorFlow для Python, як найбільш популярних фреймворків для побудови передбачуваних моделей, оцінка точності прогнозів моделі та підбір оптимальних параметрів.¶

Висновки до першого розділу

В даному розділі були підняті питання стосовно поставленої мети, її доцільності та наукової новизни. Також була проаналізована актуальна проблема на сьогоднішній час щодо застосування нейромереж у фінансовій сфері. Було поставлено завдання щодо актуальності та доцільності застосування методів машинного навчання на фінансових ринках.

Було проведено первинний аналіз щодо нейронних мереж та їх основних понять.

На основі первинного огляду предметної галузі було визначено об’єкт дослідження та предмет дослідження, також поставлена мета роботи.

2 АНАЛІЗ ТА ПРОБЛЕМИ СУЧАСНИХ НЕЙРОННИХ МЕРЕЖ

2.1 Аналіз останніх досліджень по темі

Сьогодні існує багато методик для прогнозування фінансового та економічного стану підприємства. Для економічного прогнозування використовують три допоміжні класи методів прогнозування:

- екстраполяція;
- експертні оцінки;
- моделювання.

Саме прогнозування здійснює фундаментальну основу управлінської та підприємницької діяльності у будь-якій сфері, при виконанні різних властивих їм функцій. В інтересах розвитку організаційно-виробничої системи в умовах випадковості, визначеності або невизначеності відбувається прогнозування. В результаті передпрогнозних досліджень, прогнозисту необхідно структурувати інформацію, провести аналіз та прийняти рішення про те, який із способів відповідатиме конкретним умовам прогнозу. Якості управлінських рішень вирішальним чином сприяє правильність вихідних передумов та методологічних засад прогнозу. Результати прогнозу включаються до мети організації, що визначає керівництво.

Використання нейронних мереж у задачах прогнозування, на сьогодні є найефективнішим методом. Нейромережі мають багато переваг над іншими алгоритмами. За допомогою нейронних мереж можна легко досліджувати залежність прогнозованих даних від незалежних змінних. Як приклад можна навести таке: є припущення, що стан підприємства через місяць якимось чином залежить від економічних, політичних, ринкових, виробничо-технологічних, конкурентних та міжнародних. Отже, необхідно побудувати систему, яка все це враховуватиме, і на основі цих даних даватиме короткострокові прогнози. Використання більшої частини стандартних методів прогнозування у цій задачі

просто неможливе.

Досить· легко· отримати· працюючу· систему· прогнозування,· навіть· використовуючи· найпростішу· нейромережну· структуру· (перцептрон· з· одним· прихованим·шаром)·та·базу·даних.·Облік·або·облік·системи·зовнішніх·параметрів· визначатиметься·включенням,·або·виключенням·відповідного·входу·в·нейронну· мережу.·Прогнозист·може·використовувати·будь-який·алгоритм·для·визначення· вагомості· вхідних· змінних,· щоб· потім· можна· було· виключити· з· обліку· параметри,·які·незначно·впливають.¶

Вагомою·перевагою·нейромереж·є·те,·що·користувач·не·є·заручником· вибору·математичної·моделі·поведінки·часового·ряду.·Модель·нейронної·мережі· будується· під· час· її· навчання· без· участі· користувача,· а· також· нейромережа· використовує·приклади·з·бази·даних,·і·сама·підлаштовується·під·ці·дані.¶

Недетермінованість·є·недоліком·нейронних·мереж.·Часто·нейромережа· порівнюють·із·«чорною·скринькою»,·тому·що·після·навчання·нейромережі·логіка· прийняття· нею· рішень· цілком· прихована· від· користувача.· Хоча· вже· давно· існують·алгоритми,·за·допомогою·яких·можна·отримувати·знання·з·нейронної· мережі.· Вони· формалізують· нейронну· мережу· до· списку· логічних· правил· і· створюють на основі нейромережі експертну систему. Ці алгоритми не вбудовані в нейромережевые пакети, адже набір правил, які генерують ці алгоритми, дуже об'ємні. Експерти, які знають нюанси у роботі нейронних мереж, у її налаштуванні, навчанні та застосуванні, непрозорість нейромереж не вважають серйозним недоліком.

Використання нейронних мереж для прогнозування фінансового стану підприємства, отже, надає такі переваги:

– незалежність нейромережевої моделі від математичної моделі поведінки часового ряду;

– легкість дослідження залежності прогнозованої величини від незалежних змінних;

– можливість визначення значущості вхідних змінних.

Виходячи з цього, ці переваги значно зменшують важливість головного недоліку нейронних мереж - недетермінованість.

2.2 Аналіз наукових статей по темі

У статті [12] розглянуто основні принципи використання нейронних мереж для передбачення фінансових рядів, на прикладі акцій. Їхня ефективність пояснюється здатністю нейронних мереж вловлювати нелінійні залежності. Це якраз потрібно у фінансовій сфері, оскільки модель, яка суворо описує майбутню поведінку фінансових активів за будь-якими параметрами, не існує. Відзначено нелінійну залежність між цінами на акції, їх дивідендами та обсягами торгів, а також нелінійну залежність між індексом акцій та макроекономічними індикаторами.

Автори роботи [7] застосували рекурентні нейронні мережі для прогнозування ціни на криптовалюту Ethereum і провели порівняльний аналіз результатів. Як вхідні дані були використані котирування закриття торгової сесії та середньозважені котирування протягом дня. Порівнювалися моделі: CNN-2l, CNN-3l, LSTM, sLSTM, BiLSTM, GRU, найкращі результати на тестуючій частині датасету показала звичайна мережа LSTM. Архітектурно вона собою представляла вхідний LSTM шар з 50 нейронів з функцією активації - гіперболічний тангенс і ймовірністю ”вимикання нейронів” - 0.2. Також була застосована L2 регуляризація зі значенням 0.0001 для вирішення проблеми перенавчання.

У роботі [11] автори використовували нейронні мережі для прогнозування подальшого тренду руху криптовалют. З особливостей варто відзначити високочастотність пророцтв (використовувалися хвилинні котирування), також була використана не одна валюта, а шістка найпопулярніших. Як вхідні дані були обрані різноманітні технічні індикатори,

всього 18 штук. Були використані лише 4 архітектури нейронних мереж: CNN, CLSTM, MLP, RFBNN. Найкращий результат показала CLSTM мережа, причому абсолютно на всіх валютах, що підтверджує припущення щодо ефективності мереж із пам'яттю. Також дано конкретні установки мережі з усіма параметрами.¶

Цікаво вивчення передбачуваної можливості технічних індикаторів [13]. Хоча у наукових працях, донедавна, вони зустрічалися досить рідко і переважно використовувалися практиками біржової торгівлі. Незважаючи на це, результати роботи полягають у тому, що технічні індикатори можна використовувати для передбачення доходності активів самостійно, так і разом з макроекономічними величинами. Причому спільне використання цих двох способів дає найкращий результат, тому вони вловлюють різні види залежностей і не взаємозамінні один одному.¶

У роботі «On stock return prediction with LSTM networks» [14] розглядалася мережа LSTM. Використовували два підходи: регресійний та класифікаційний (можна передбачати не точно значення прибутковості на наступний часовий період, а напрямок зміни). При цьому як дані були використані індекси акцій США, Бразилії та Швеції. Ця робота представляє інтерес, як стаття, де жодна архітектура LSTM (тривіальна, глибока, з різними функціями активації) не змогла передбачити індекс S&P500 та індекс BOVESPA на рівні, що перевищує статистичну похибку, хоча шведський індекс OMX все ж таки вдалося передбачити з більш високою точністю. Автор пояснює це ефективністю ринку перших двох країн.¶

У статті [8] автори перевіряють передбачувальну здатність нейронної мережі GRU і порівнюють її з LSTM (GRU - більш сучасна модель, в якій менше вхідних фільтрів і як наслідок менше вироблених операцій, але, природно, це впливає на здатність вловлювати приховані залежності). Як вхідні дані було обрано 20 різних ознак, більшість з яких носять більш фундаментальний характер. При цьому, це одна з небагатьох робіт, де було порушено питання мультиколінеарності ("схожості" значення ознак). Для кожної з ознак був

підрахований·VIF·-·variance·inflation·factor,·і·якщо·значення·VIF·було·вище·10,·то· ознака·відбраковувався,·що·допомогло·скоротити·датасет·на·25%,·і,·як·наслідок,· підвищити· швидкість· навчання· нейронної· мережі.· Також· у· цій· роботі· є· обґрунтування·вибору·різних·архітектурних·рішень,·чого·немає·у·інших·авторів.· В· результаті· виконаної· роботи· вийшло,· що· найкращі· результати· показала· нейронна· мережа· з· архітектурою· GRU.· Причому· ”тривіальна”· мережа· з· 1-го· шару· GRU· та· 1-го· Dense· шару· має· помилку· менше,· ніж· архітектура· із· прихованим·шаром·GRU¶

У·статті·Forecasting·directional·movement·of·Forex·data·using·LSTM·with· technical·and·macroeconomic·indicators·[15]·автори·будують·нейронні·мережі,·які· виконують· завдання· класифікації:· передбачається· напрям,· куди· піде· ціна.· У· цьому·використовується·дуже·цікава·архітектура.·Початковий·набір·ознак·був· розбитий·на·дві·групи:·технічні·індикатори·та·макроекономічні·параметри.·По· кожній· групі· навчалася· окрема· нейронна· LSTM· мережа,· а· вже· потім· при· узгодженій· роботі· цих· мереж,· робиться· прогноз· на· напрямок· руху· ціни.· Ця· гібридна·мережа·LSTM·показала·результати·краще,·ніж·проста·мережа·LSTM· побудована·по·повному·набору·ознак.¶

2.3 Аналіз існуючих аналогів

1. Matlab – середовище робочого столу, налаштоване для ітеративного аналізу та процесів проектування, з мовою програмування, яка прямо виражає математику матриць та масивів. Він включає Live Editor для створення сценаріїв, які об'єднують код, виведення і форматований текст у виконуваний блокнот. Matlab містить інструментарій для нейромереж – Anfis Editor (навчання, створення, тренування та графічний інтерфейс), командний інтерфейс для програмного завдання мереж, також містить nnTool, який служить для більш тонкої конфігурації мережі. Підходить для первинної роботи прогнозування ринку Forex. На рисунку 2.1 надано результати моделювання нейромережею індексу ПФТС за даними за перші 20 днів 2015 року. Чорним кольором позначені

фактичні дані, сірим показані прогнозовані дані.

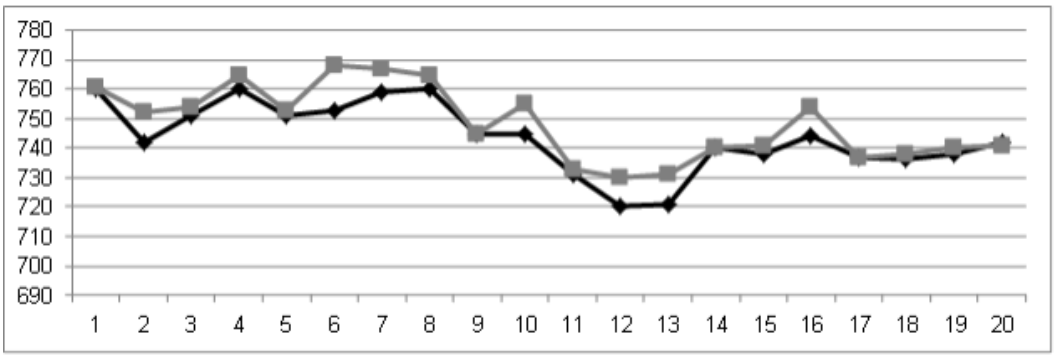


Рис. 2.1 - Приклад моделювання нейромережею індексу ПФТС

Нейронна·мережа·Елмана·використана·для·моделювання·в·пакеті·Matlab.· Вона· запам'ятовує· свої· попередні· дії· та· реалізує· завдання· навчання,· яке· розгортається·у·часі,·і·це·є·актуальним·для·прогнозування·тимчасових·рядів·із· пам'яттю.¶

2.· Statistica· –· є· дуже· потужним· програмним· забезпечення,· яке· застосовують· для· аналізу,· пошуку· даних· та· виявлення· статичних· закономірностей.·Робота·з·нейронними·мережами·в·даному·пакеті·відбувається· в· модулі· STATISTICA· Neural· Networks,· що· є· реалізацією· повного· набору· нейромережевих·методів·обробки·та·аналізу·даних.·¶

.

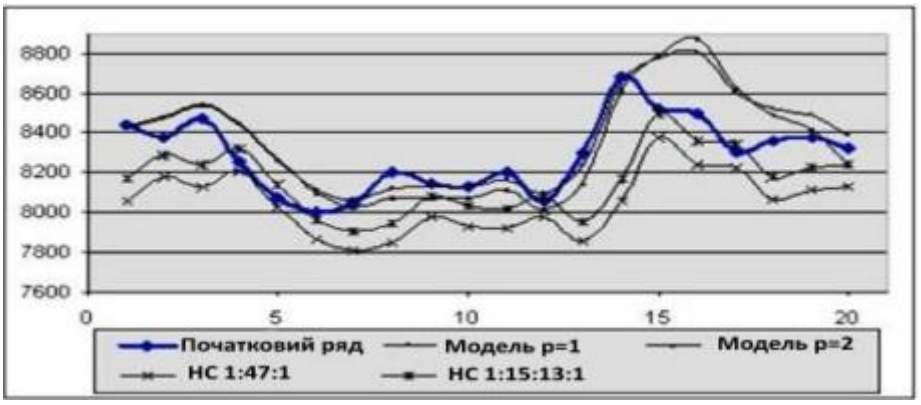


Рис. 2.2 – Результати прогнозів, отриманих нейромережею для динаміки курсу акцій з використанням ST Neural Networks

3. NeuroShell Day Trader – система до роботи з нейронними мережами, яка враховує специфічні потреби учасників торгів. Дана система легка у використанні, проте вона є дуже вузькоспеціалізованою. Для торгівлі вона підходить, але за своєю суттю ця система дуже близька до чорної скриньки.

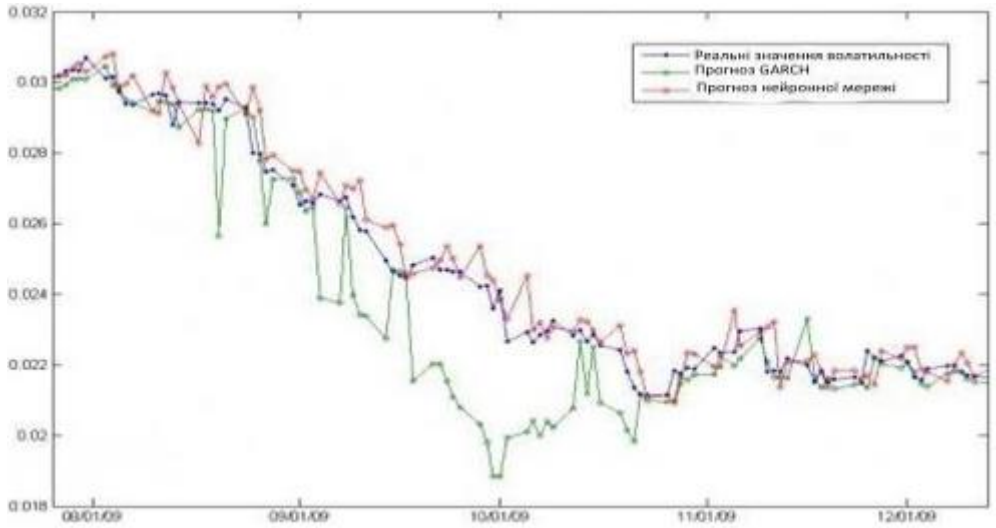


Рис. 2.3 – Результати розрахунку індексу з використанням NeuroShell Day Trader

При створенні торгової стратегії навіть без використання нейромереж виникає низка питань, а під час створення нейронної мережі до роботи з біржовим ринком тим паче. Ринок постійно змінюється, тому потрібно визначити період, з якого будуть братися дані навчання нейромережі. Через це можуть виникати проблеми, наприклад, якщо взяти невдалий період (період з високою волатильністю) викликаний непрогнозованими і в майбутньому неповторними подіями, це призведе до помилки або низької точності прогнозу (що є критично неприпустимим при роботі з нейромережами).

4. BrainMaker – призначений для вирішення завдань, у яких алгоритми та формальні методи не знайдені, початкові дані є неповними чи суперечливими. Такими завданнями є фінансове та біржове прогнозування, моделювання кризових ситуацій, розпізнавання образів та інші.

BrainMaker – один з перших пакетів, а також є одним з лідерів ринку.

Спочатку створювався на замовлення військових. Для роботи з фінансами він був адаптований у 1990 році і був нагороджений престижною премією журналу PC Magazine «Найкраще програмне забезпечення року». До цього дня це ПЗ щорічно перемагає на різних конкурсах, було зроблено 20тис. установок (для спеціалізованого програмного забезпечення це досить багато). У США даний нейропакет є найпопулярнішим..

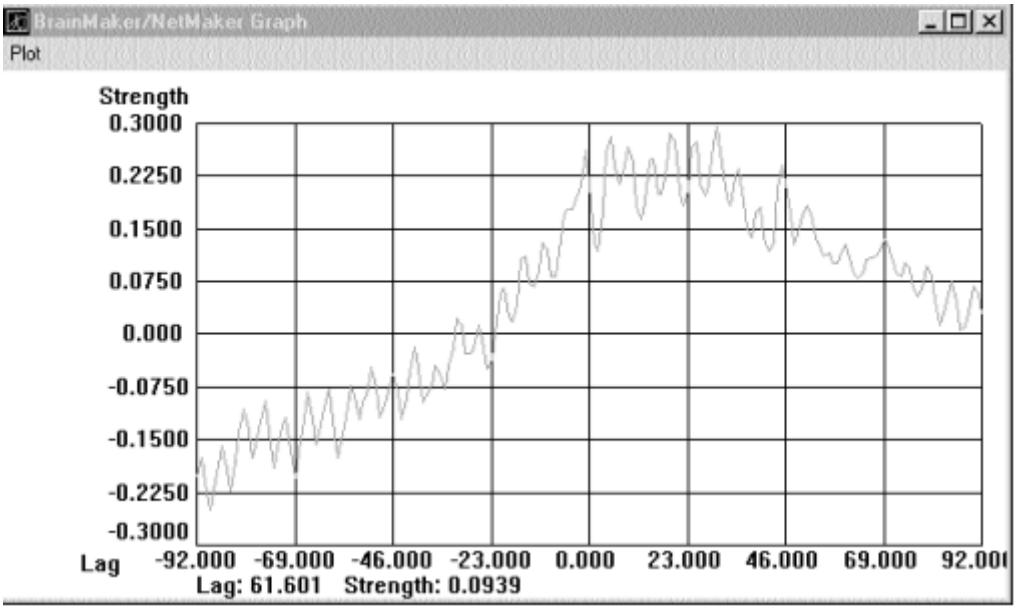


Рис. 2.4 – Приклад прогнозування зміни цін на акції компанії Bart-Davis-100 (BD100) з використанням BrainMaker

2.4 Аналіз інструментальних серед розробки

Для·програмування·нейронних·мереж·нині·найчастіше·використовується· мова· Python· завдяки· безлічі· бібліотек· з· набором· вбудованих· математичних· функцій,· таких· як· витвір· векторів,· транспонування· тощо.· Наприклад,· використовуючи·бібліотеку·Numpy,·можна·розробити·просту·нейронну·мережу,· вирішальну· задачу· прогнозування.· Бібліотека· Keras· застосовується· при· програмуванні·мереж·прямого·поширення·та·вирішення·завдань·розпізнавання· мови.· Для· нейронних· мереж,· що· працюють· із· зображеннями,· необхідне· підключення·іншого·модуля,·наприклад·TensorFlow.·¶

Крім· Python,· для· написання· програмного· коду,· що· реалізує· нейронну·

мережу, використовуються мови R, C Sharp, C++, Haskell, Java, Go та Swift. Як і раніше, застосовуються такі пакети прикладних програм, як MatLab та Deductor. Однак їх використання обмежене відсутністю вибору видів та архітектур нейронних мереж.

TensorFlow – бібліотека для машинного навчання від Google з відкритим вихідним·кодом.·Застосовується·для·побудови·та·тренування·нейронної·мережі,· вирішальної·задачі·знаходження·та·класифікації·образів.·Бібліотека·побудована· на· парадигмі· програмування· потоків· даних,· що· дає· змогу· оптимізувати· математичні·обчислення.·Обчислення·в·TensorFlow·виконуються·за·допомогою· графа·потоків·даних,·вузли·якого·відображають·операції,·а·ребра·–·потоки·даних· між·вузлами.·Як·і·більшість·фреймворків·глибокого·навчання,·TensorFlow·має· API· на· Python· поверх· механізму· C· та· C++,· що· прискорює· його· роботу.· TensorFlow·має·гнучку·екосистему·інструментів,·бібліотек·та·ресурсів·спільноти.· Це· дозволяє· дослідникам· використовувати· найсучасніші· МО-технології,· а· розробникам· —· створювати· та· розгортати· програми· на· базі· машинного· навчання.·Платформа·надає·інтуїтивно·зрозумілі·високорівневі·API-інтерфейси,· наприклад· Keras,· із· швидким· виконанням,· що· забезпечує· негайну· ітерацію· моделі· та· просте· налагодження.· За· рахунок· мультиплатформенності· рішення· дозволяє·навчати·і·розгортати·моделі·в·хмарі·і·локально,·незалежно·від·мови,·що· використовується·користувачем.·Якщо·необхідно·запустити·модель·машинного· навчання·на·смартфоні·або·IoT-пристрої,·то·знадобиться·середовище·глибокого· навчання·з·відкритим·кодом·TensorFlow·Lite.·Коли·потрібно·створити·та·навчити· МО-модель· на· JavaScript,· а· потім· розгорнути· її· у· браузері· або· на· Node.js,· то· можна·скористатися·бібліотекою·TensorFlow.js.¶

Варто зазначити, що фреймворк постійно розвивається рахунок відкритого вихідного коду і великої спільноти ентузіастів. Також за рахунок його популярності є безліч вже вирішених завдань, що значно спрощує життя новоспеченим розробникам.

Однак фреймворк не позбавлений недоліків. Компанія Google відома

своєю любов'ю до створення власних стандартів, що торкнулося і фреймворку. Наприклад, якщо під час роботи з TensorFlow у коді вилітає помилка, то фреймворк не покаже користувачеві конкретну сходинку, що її спровокувала. Чому так? Це з особливістю API TensorFlow, де всі операції виконуються через клас tf.Session. У програмуванні є дві основні парадигми — імперативна та декларативна. У TensorFlow використовується другий підхід.

Ще одна проблема TensorFlow, яку слід виділити - фреймворк завжди забирає всю відеопам'ять. Якщо хочеться його обмежити, необхідно створювати файл конфігурації і явно вказувати, що можна брати. Також фреймворк можна обмежити безпосередньо, наприклад, дозволити взяти не більше 50% відеопам'яті. Через подібну поведінку з пам'яттю можуть виникнути проблеми у роботі. Припустимо, що в одному проекті багато різних моделей, одні з яких написані на PyTorch, а інші - на TensorFlow. Якщо спочатку ми створимо TensorFlow-модель і не обмежимо її, вона використовує всю відеопам'ять при тому, що їй необхідно всього 0,5 ГБ, і в результаті на PyTorch-моделі просто не вистачить місця.

Плюси:

– відмінний фреймворк для створення нейронних мереж, які працюватимуть у продакшені;

– бере на себе оптимізацію ресурсів для обчислень;

– величезне ком'юніті;

– за рахунок популярності вища ймовірність, що проблему, подібну до вашої, вже вирішили.

Мінуси:

– складний у використанні та освоєнні;

– недружелюбний;

– необхідно постійно контролювати відеопам'ять, що використовується;

– має свої стандарти;

– погана документація;

– завжди є п'ять способів вирішити завдання, але три з них застаріли, один не працює, а той, що працює, не задокументований.

PyTorch – це середовище машинного навчання мовою Python з відкритим вихідним кодом, що забезпечує тензорні обчислення з GPU-прискоренням. Вона була розроблена компанією Facebook та представлена у жовтні 2016 року, а відкрита для сторонніх розробників – у січні 2017 року. Фреймворк підходить для швидкого прототипування у дослідженнях, а також для аматорів та невеликих проектів. Фреймворк пропонує динамічні графи обчислень, які дозволяють обробляти введення та виведення змінної довжини, що корисно, наприклад, при роботі з нейронними рекурентними мережами. Якщо коротко, то зарахунок цього інженери і дослідники можуть змінювати поведінку мережі «нальоту».¶

За рахунок глибокої інтеграції фреймворку з кодом C++ розробники можуть програмувати C і C++ за допомогою API-розширення на основі FFI для Python. На відміну від TensorFlow, PyTorch менш гнучкий у підтримці різних платформ. Також у ньому немає рідних інструментів для візуалізації даних, але є сторонній аналог, званий tensorboardX. Однак, знову ж таки, на відміну від TensorFlow, якщо при роботі з PyTorch вилітає помилка, то це конкретна недоробка в коді і система виділить вам саме рядок, який її спровокував. Також під час розгортання мереж на GPU PyTorch самостійно займе лише необхідну

Плюси:

– має багато модульних елементів, які легко комбінувати;

– легко писати власні типи шарів та працювати на GPU;

– має широкий вибір попередньо навчених моделей.

Мінуси:

– доведеться самостійно писати тренувальний код;

– погана документація, щоразу траплятимуться функції та методи, документація яких існує виключно на форумах спільноти та отримана емпіричним шляхом.

Keras – відкрите середовище глибокого навчання, написане на Python. Вона була розроблена інженером з Google Франсуа Шолле та представлена у березні 2015 року. Фреймворк націлений на оперативну роботу з нейромережами і є компактним, модульним і таким, що розширюється. Підходить для невеликих проектів, оскільки створити щось масштабне на ньому складно і він явно програватиме у продуктивності нейромереж того ж TensorFlow.

Keras· працює· поверх· TensorFlow,· CNTK· і· Theano· і· надає· інтуїтивно· зрозумілий·API,·який,·на·думку·наших·інженерів,·поки·що·є·найкращим·у·своєму· роді.· Фреймворк· містить· численні· реалізації· будівельних· блоків· нейронних· мереж,· що· широко· застосовуються,· таких· як· шари,· цільові· та· передавальні· функції,· оптимізатори,· а· також· безліч· інструментів· для· спрощення· роботи· із· зображеннями·і·текстом.·Deeplearning4j·використовує·Keras·як·свій·Python·API·і· дозволяє·імпортувати·моделі·з·Keras,·а·також·через·Keras·з·Theano·і·TensorFlow.¶

Плюси:¶

–→зручний·у·використанні;¶

–→легкий·в·освоєнні;¶

–→фреймворк,·що·швидко·розвивається;¶

–→гарна·документація;¶

–→вбудований·у·TF.¶

Мінуси:¶

–→не·підходить·для·великих·проектів.¶

_

.

Darknet - це фреймворк з відкритим вихідним кодом, написаний мовою C з використанням програмно-апаратної архітектури паралельних обчислень CUDA. Він швидкий, легкий та зручний у використанні. Також Darknet підтримує обчислення на базі CPU та GPU. Навчені ваги Darknet зберігає у

форматі, який може бути розпізнаний за допомогою різних методів на різних платформах. Однак це може стати проблемою, якщо ви вирішите натренувати модель на одному надпотужному обладнанні, а потім використовувати її на іншому.

Так як фреймворк написаний на C і не має іншого API, то у випадку, коли вимоги платформи або власні переваги змусять звернутися до іншої мови програмування, доведеться заморочитися над його інтеграцією. До того ж він поширюється лише у форматі вихідного коду, і процес компіляції на деяких платформах може бути проблематичним. Фреймворк не рекомендується використовувати для складних проектів, хіба що необхідно створити надшвидкий детектор об'єктів.

Плюси:

– простий;

–→швидкий;¶

–→зручний.¶

Мінуси:¶

–→крім·завдань·із·виявленням,·більше·ніде·не·використовується;¶

–→не·рекомендується·для·великих·проектів;¶

–→погана·документація.¶

XGBoost·-·це·фреймворк·з·відкритим·вихідним·кодом,·який·пропонує·систему·градієнтного·бустингу·C++,·Java,·Python,·R,·Julia.·Він·розроблений·для·забезпечення·високої·ефективності,·гнучкості·та·портативності.·Цей·фреймворк·відноситься·не·до·глибокого·навчання,·як·усі·вищепредставлені,·а·до·класичного.¶

Спочатку·це·був·дослідницький·проект·Тяньцзі·Чена·та·Карлоса·Гестріна·у·складі·Distributed·[Deep]·Machine·Learning·Community,·але·пізніше·він·був·розширений·та·представлений·публіці·на·конференції·SIGKDD·у·2016·році,·де·спричинив·фурор.·Після·своєї·презентації·фреймворк·лідирував·у·змаганнях·Kaggle·і·досі·залишається·фаворитом·для·вирішення·більшості·завдань·на·

платформі. XGBoost фокусується на швидкості обчислень та продуктивності моделі та підходить для вирішення завдань регресії, класифікації та впорядкування. Якщо дані можна подати у вигляді таблиці, то точність і продуктивність будуть значно вищими, ніж у DeepLearning-рішень. Улюблений інструмент Data Scientist-ів. Фреймворк сумісний з операційними системами Windows, Linux та OS X, а також підтримує кластери AWS, Azure та Yarn, добре працює з Flink, Spark.

Плюси:

– дуже швидкий та зручний інструмент для тренування моделей типу «дерево рішень»;

– точний;

– відмінно підходить для перевірки гіпотез.

Мінуси:

– вузькоспеціалізований.

З· усього· вище· перерахованого· випливає· висновок,· **TensorFlow**· добре· підходе·для·розвинених·проектів,·таких·як·створення·багатошарових·нейронних· мереж.· Може· використовуватися· для· розпізнавання· мовлення,· об'єктів· та· зображень,· а· також· для· роботи· з· текстом.· **PyTorch**· підійде· у· випадку,· коли· необхідно· навчити· моделі· швидко· та· ефективно.· Зручний· для· швидкого· прототипування·у·дослідженнях,·а·також·для·аматорів·та·невеликих·проектів.· **Keras**·підходить·для·швидкого·прототипування.·Добре·проявляє·себе·у·задачах,· пов'язаних·із·перекладом,·розпізнаванням·зображень·та·мови.·**Darknet**·підходить· для·невеликих·проектів.·Добре·працює·у·завданнях·виявлення.·**XGBoost**·може· використовуватися·для·вирішення·задач·регресії,·класифікації,·впорядкування·та· користувальницьких·завдань·на·передбачення.¶

.

Висновки до другого розділу

Для економічних процесів використання нейромереж дуже актуально, особливо коли аналіз та обробка великої кількості даних людським інтел не ефективно, а використовувати традиційні обчислювальні процеси не доц і трудомістко.¶

Використання нейромереж є досить потужним методом прогнозув що дозволяє відтворювати дуже важкі залежності. Нейронні мережі на сьо є актуальним методом для роботи з фінансовою та економічною складовою

Прогнозування на фондовому ринку є однією з важливих га застосування нейромереж. Економісти-аналітики не обмежені у в програмного забезпечення для створення нейронних мереж, адже в сучасно існує чимала кількість. Вибір складатиметься лише з поставленого завда якому нейронна мережа буде використовуватися, а також від дослідження.

.

3 ПРОЕКТНИЙ РОЗДІЛ

3.1 Опис вхідної та вихідної інформації

Вхідною інформацією є дані, взяті із сайту Kaggle. Kaggle - це співтовариство фахівців з Data Science. Тут можна вивчати машинне навчання, писати свої та розбирати чужі прогнозні моделі, брати участь у змаганнях та спілкуватися з дата-саентистами. Сервіс повністю безкоштовний. Середовище організоване як публічна веб-платформа, на якій користувачі та організації можуть публікувати набори даних, досліджувати та створювати моделі, взаємодіяти з іншими фахівцями з даних та інженерами з машинного навчання, організовувати конкурси з дослідження даних та брати участь у них. У системі розміщено набори відкритих даних, надаються хмарні інструменти для обробки даних та машинного навчання. Також реалізовані навчальні ресурси, є розділ для розміщення вакансій роботодавцями, де також можлива організація конкурсів для відбору найкращих кандидатів.¶

Дані представлені у вигляді датасету біржових цін на акції компанії Apple на період 1984 – 2017р. Kaggle надає дані у вигляді файлу у розширенні csv, який є текстовим файлом. Датасет цін поділено на кілька категорій: ціна відкриття та закриття торгів, мінімальна та максимальна ціна в день торгів на акції, а також обсяг торгів у цей день.¶

Щоб не перевантажувати систему, дані було взято за період 6 років, з 2011 до 2017 року. Також вхідною інформацією є дані, які в процесі розробки програми були поділені на тренувальну та валідаційну частину у співвідношенні 80% та 20%.¶

Вихідною інформацією в додатку є дані які доступні для перегляду звичайному користувачеві. Наприклад, такими даними є графіки зміни цін на акції з часом, функції аналізу даних.¶

.

Основною частиною вихідної інформації є результат навчання нейронної

мережі, а саме порівняння передбачуваного результату з наявним. Таким порівнянням є графіки, які демонструють здатність моделі до передбачення, а також функції точного розрахунку метрик, які демонструють похибку моделі.

3.2 Обґрунтування вибору середи розробки

За всіма параметрами, для проектування і створення нейронної мережі підійшов хмарний сервіс від Google – Google Collaboratory.

Також необхідне середовище для розробки, для експериментів із нейронними мережами прийнято використовувати Jupyter Notebook.

Jupyter Notebook – інструмент для інтерактивної розробки та представлення програмних проектів, свого роду блокнот, що поєднує програмний код у єдиному документі. Якщо немає можливості встановити цей інструмент через різні версії розрядності операційних систем та інше, Google пропонує аналог – Colaboratory, що не потребує встановлення на комп'ютер.

Google Colaboratory – сучасний хмарний сервіс, спрямований на спрощення досліджень у галузі машинного та глибокого навчання, оскільки всі обчислення в Colaboratory виконуються на віддалених серверах. У Colaboratory встановлено TensorFlow і практично всі необхідні для роботи бібліотеки Python. Якийсь відсутній пакет можна встановити командою pip або утиліта apt-get.

Крім того, що в Colaboratory можна писати та виконувати код Python у браузері, не потрібне налаштування сервісу, є безкоштовний доступ до графічних процесорів та документів інших користувачів. Все це робить хмарний сервіс Colaboratory доступним рішенням для навчання студентів основ нейронних мереж.

В основі «Колабораторії» є блокнот Jupyter для роботи на Python, тільки з базою на Google Диску, а не на комп'ютері. Тут ті ж осередки (cells), що підтримують текст, формули, зображення, розмітку HTML і не тільки. Тобто можна програмувати на Python і не качати купу бібліотек, не перевантажувати

машину і не переживати, що місце на харді закінчиться. Єдина умова — потрібно мати обліковий запис Google. Головна особливість «Колабораторії» — безкоштовні потужні графічні процесори GPU та TPU, завдяки яким можна займатися не лише базовою аналітикою даних, а й складнішими дослідженнями в галузі машинного навчання. З тим, що CPU обчислює годинами, GPU або TPU справляються за хвилини або секунди.

CPU – центральний процесор – мозок комп'ютера, який виконує операції з даними. Настільки універсальний, що може використовуватися майже всім завданням: від запису фотографій на флешку до моделювання фізичних процесів.

GPU – графічний процесор. Обробляє дані швидше, оскільки завдання виконує паралельно, а чи не послідовно, як CPU. Він заточений виключно під графіку, тому на ньому зручніше працювати із зображенням та відео, наприклад займатися 3D-моделюванням або монтажем.

TPU – тензорний процесор, розробка Google. Він призначений для тренування нейромереж. У цього процесора у рази вища продуктивність при великих обсягах обчислювальних завдань.

Самі процесори дорогі, і не кожен може їх собі дозволити. Google Colaboratory дає можливість безкоштовно та безперервно користуватися ними протягом 12 годин. Будьте уважні: як тільки цей час мине, Colab зітре всі дані і доведеться починати спочатку.

Крім того, Google відключає блокноти після приблизно 30 хвилин бездіяльності, щоб не перевантажувати процесори. Система Colab так влаштована спеціально: багато чинників, зокрема час простою, максимальна активність, загальні обмеження обсяг пам'яті іноді динамічно змінюються. Активним учасникам ненадовго можуть обмежити доступ до GPU, щоб надати можливість використовувати процесор іншим.

Особливості Google Colaboratory:

1. Як і Документи Google, він дає можливість працювати з Python-бібліотеками для аналізу даних онлайн.

2. Colab надає потужні процесори для хмарних обчислень. Він має інтуїтивно зрозумілий інтерфейс, який дозволяє не перевантажувати комп'ютер і робити всі обчислення швидко.

3. Усі блокноти під рукою. Google Colab зберігає доступ до облікового запису з будь-яких пристроїв. Правда, якщо ви з обережністю ставитеся до своєї конфіденційності, Jupyter Notebook залишиться кращим варіантом.

4. Серце Colab – це спільне використання. При роботі над проектом у команді Colab дає можливість вільно правити, коментувати та редагувати код з різних облікових записів, навіть якщо ви сидите на жорсткому локдауні десь у Лондоні.

Ще одна перевага Colab - інтеграція з GitHub. Він відкриває доступ до будь-якого сховища, якщо йому надати профіль на сервісі.

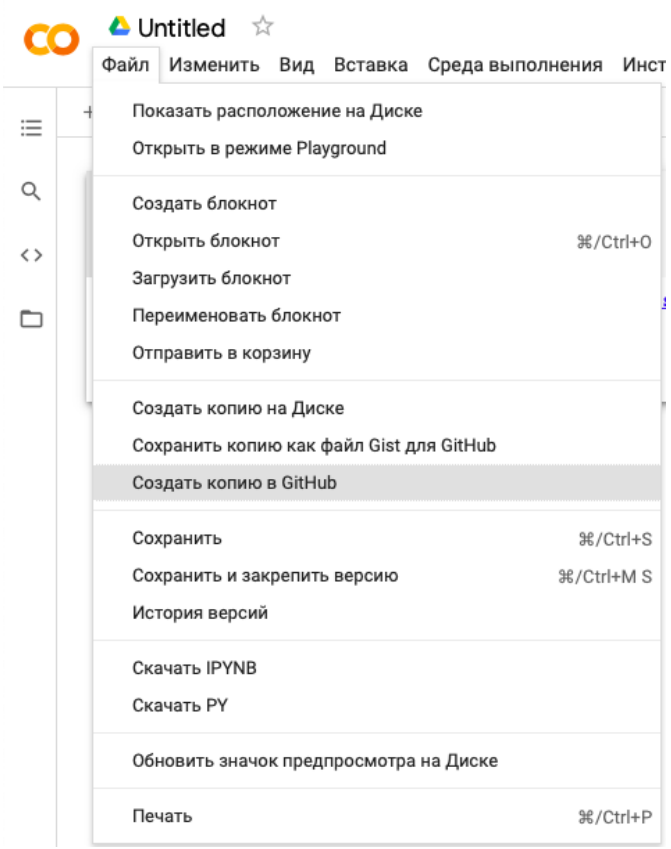


Рис. 3.1 – Взаємодія Google Colab з GitHub

Крім того, для певних завдань можна вибрати підходящий за потужністю процесор. Необхідно просто змінити середовище виконання у потрібній вкладці і вже в налаштуваннях блокноту вибрати між GPU та TPU.

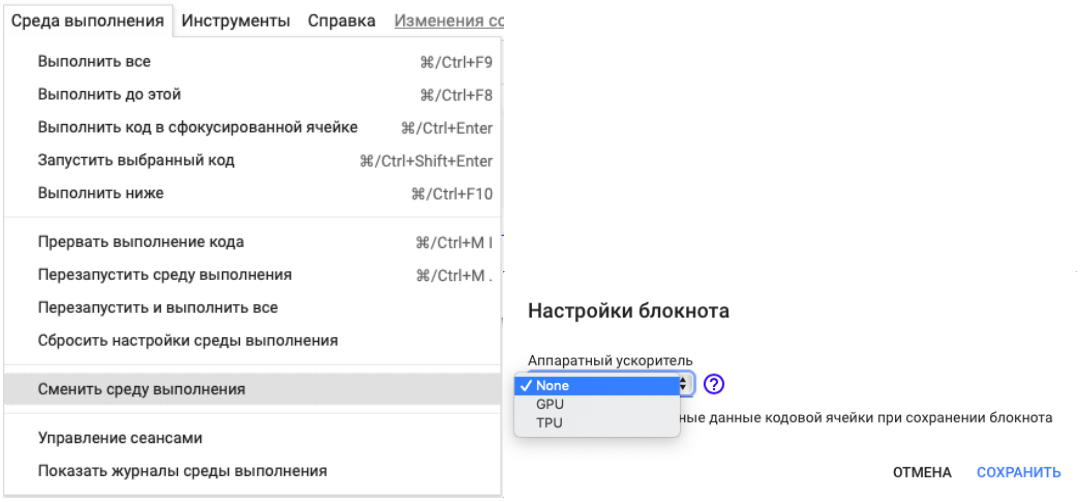


Рис. 3.2 – Вибір процесора в Colab

3.3 Моделювання об’єкту проектування

UML (Unified Modeling Language) – уніфікована мова моделювання. Дана мова є системою позначень, яка базується на діаграмах і призначається для моделювання систем на основі об'єктно–орієнтованого підходу.

UML – "мова для визначення, візуалізації і конструювання артефактів програмних систем". Це система позначень (включаючи семантику), призначена для моделювання систем на основі об'єктно–орієнтованого підходу.

Щоб створити програмне забезпечення, необхідно описати проблему і вимоги до системи. Етап аналізу полягає в дослідженні проблеми, а не в пошуках шляхів її вирішення.

При розробці програми необхідно також забезпечити високий рівень і докладний опис логіки рішення, що задовольняє вимогам до системи і накладаються обмеженням. В процесі проектування основна увага приділяється логічному рішенню, що забезпечує виконання основних вимог.

Діаграма – це графічне представлення безлічі елементів, найбільш часто зображується як зв'язний граф з вершин (предметів) і дуг (відносин).

В UML присутні наступні типи діаграм:

– діаграми варіантів використання (usecase diagrams);

– діаграми класів (class diagrams);

– діаграми поведінки системи (behavior diagrams);

– діаграми взаємодії (interaction diagrams);

– діаграми станів (statechart diagrams);

– діаграми діяльностей (activity diagrams);

– діаграми реалізації (implementation diagrams): діаграми компонентів (component diagrams);

– діаграми розміщення (deployment diagrams).

Для розробки інтерфейсу програми були використані діаграми: варіантів використання та класів.

Діаграми варіантів використання описують функціональне призначення системи або те, що система повинна робити. Розробка діаграми переслідує такі цілі:

– визначити спільні кордони і контекст модельованої предметної області;

– сформулювати загальні вимоги до функціональної поведінки проектованої системи;

– розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей;

– підготувати вихідну документацію для взаємодії розробників системи з її замовниками і користувачами.

Суть діаграми варіантів використання: проектована система представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою варіантів використання. При цьому актором або дійовою особою називається будь–яка сутність, що взаємодіє з системою ззовні. Це може бути людина, технічний пристрій, програма або будь–яка інша система, яка може

служити джерелом впливу на модельовану систему так, як визначить сам розробник. Варіант використання служить для опису сервісів, які система надає актору. Діаграма варіантів використання може доповнюватися пояснювальним текстом у вигляді коментарів, які розкривають зміст або семантику складових її компонентів.

Окремий варіант використання позначається на діаграмі еліпсом, усередині якого міститься його коротка назва або ім'я у формі дієслова з пояснювальними словами. Мета варіанту використання полягає в тому, щоб визначити закінчений аспект або фрагмент поведінки деякої сутності без розкриття її внутрішньої структури.

Актор є будь–якою зовнішньою по відношенню до модельованої системі сутністю, яка взаємодіє з системою і використовує її функціональні можливості для досягнення певних цілей. При цьому актори служать для позначення узгодженої безлічі ролей.

Будь–який актор може розглядатися як якась окрема роль щодо конкретного варіанту використання. Стандартним графічним позначенням актора на діаграмах є фігурка чоловічка, під якою записується ім'я актора. У деяких випадках актор може позначатися у вигляді прямокутника класу з ключовим словом «актор» і звичайними складовими елементами класу. Імена акторів повинні записуватися великими літерами і слідувати рекомендаціям використання імен для типів і класів моделі.

Актори взаємодіють з системою за допомогою обміну повідомленнями з варіантами використання. Повідомлення являє собою запит актором певного сервісу системи і отримання цього сервісу.

Інтерфейс служить для специфікації параметрів моделі, які видимі ззовні, без вказівки їх внутрішньої структури. У мові UML інтерфейс є класифікатором і характеризує тільки обмежену частину поведінки модельованої сутності. На діаграмі варіантів використання інтерфейс зображується у вигляді маленького

кола, поруч з яким записується його ім'я. Як ім'я може бути іменник або рядок тексту. Якщо ім'я записується англійською мовою, то воно повинно починатися з великої літери I.

Діаграма варіантів використання представлена в Додатку А.

Між елементами діаграми варіантів використання можуть існувати різні відносини, які описують взаємодію екземплярів акторів і варіантів використання. У мові UML існує кілька стандартних видів відносин між акторами і варіантами використання:

– асоціації (association relationship);
– розширення (extend relationship);
– спілкування (generalization relationship);
– включення (include relationship).

3.4 Технічне забезпечення

Дані про обчислювальну техніку, на якій була створена і протестована програма, представлені нижче:

– операційна система Windows 10 x64;
– відеоадаптер Nvidia GeForce GTX 1080;
– процесор AMD Ryzen 5 2600x Six-Core Processor;
– оперативна пам'ять 16309Мб.

Оскільки програма розроблена в Google Collaboratory, який є хмарним сервісом, вимоги до обчислювальної техніки будуть мінімальні, всі обчислення проводяться на сервері. Рекомендовані системні вимоги:

– операційна система Windows 10, і більш пізні версії;
– процесор двоядерний 2 ГГц;
– оперативна пам'ять 4096Мб;
– браузер Google Chrome;
– широкосмугове підключення до інтернету;

– пристрої взаємодії з користувачем – клавіатура і миша.

3.5 Аналіз та результати роботи програмного забезпечення

Прогнозування – одне з найбільш популярних, але при цьому одне з найскладніших завдань інтелектуального аналізу даних. Проблеми прогнозування пов'язані з недостатньою якістю та кількістю вихідних даних, змінами середовища, в якому протікає процес, впливом суб'єктивних факторів. Загалом завдання короткострокового прогнозу цін на ринку, навіть і з використанням нейронних мереж, є досить складним, особливо на українському фондовому ринку, що стрімко змінюється. Зовнішні дані, що постійно змінюються, впливають на результат, а саме на точність прогнозування. Від цього залежить міра впливу на фінансовий ринок.¶

Нейронні мережі – затребуваний та потужний інструмент розробника, але застосовувати їх необхідно лише у певних ситуаціях, пов'язаних із розпізнаванням та прогнозуванням, тобто там, де потрібен приблизний аналіз даних. При цьому нейронні мережі мають вкрай високу швидкодію, що так само часто є одним з важливих критеріїв при виборі способу обробки даних.¶

Початковим етапом у вирішенні цієї проблеми є вивчення предметної галузі, що дозволяє виділити значні інформаційні ресурси.¶

Наступним етапом є вибір найбільш придатного методу прогнозування курсу акцій і зрештою його програмна реалізація.¶

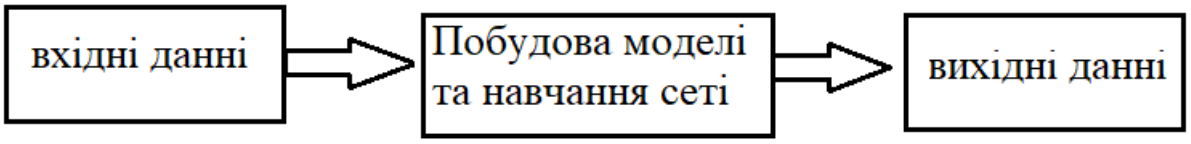
У роботі проведено три етапи створення моделі нейромережі для вирішення задачі передбачення:¶

:

– перший етап – підготовка даних;

– другий етап – побудова моделі та навчання нейромережі;

– третій етап – вибір функції помилки.



Підготовка даних та навчання мережі

Нейромережа, являє собою багатошарову мережеву структуру однотипних елементів - нейронів, з'єднаних між собою та згрупованих у шари. У нашій нейромережі визначено 3 шари нейронів:

– вхідний шар;
– шар регулювання (обчислювальний та аналізуючий блок);
– шар виводу.

На нейрони вхідного шару подається інформація. У нашому випадку вхідний шар складається з 32 нейронів, на які послідовно подаються ціни за 5 років.

Вхідні дані є вибіркою, елементи якої містять наступну інформацію:

– дані про ціни на акції;
– період для аналізу;
– різноманітних кореляційні показники взаємодії цінних паперів;
– неекономічні складові, що впливають зміну ціни;
– інша інформація, необхідна побудови моделі.

У обчислювальному та аналізуючому блоці відбуваються обчислювальні, корегують та аналізують процеси обробки вхідної вибірки. Розраховані нейронною мережею показники подаються у вихідні дані, які необхідні прийняття будь-яких рішень.

Таким чином, з вихідного шару знімається результат. На цьому шарі визначено один нейрон, який видаватиме прогнозоване значення ціни на один день. Ця ціна є предметом дослідження. При проходженні мережею вхідні сигнали посилюються чи послаблюються, що визначається вагами

міжнейронних зв'язків. Перед застосуванням нейромережу необхідно навчити на прикладах, тобто за відомими вхідними параметрами та результатом. У цьому випадку мережа дає результат, максимально близький до правильного.

Спочатку була обрана функція активації – сигмоїд. Це найпоширеніша функція активації, її діапазон значень [0,1]. Мінусом цієї функції активації є те, що ми не можемо вийти за діапазон значень, але це обмеження легко обійти в програмній реалізації.

Після цього обрали спосіб навчання нейромережі. Для навчання мережі використано метод зворотного розповсюдження помилки.

Для навчання нейронної мережі дані розділили всі дані на 2 частини: тренувальну та валідаційну. Для цін на акції у тренувальному наборі майбутня ціна акцій використовується для навчання нейронної мережі. Розділ даних проведено у співвідношенні 4/5 до 1/5. Тобто для тренування взяли вибірку обсягу 4/5, для валідації – вибірку обсягу 1/5. Усі дані групуємо у блоки. Наші дані згруповані в блок п'ять років з інтервалом в умовний один день.

Приклад застосування моделі

Для моделювання прогнозу нейронна мережа використовує знання про значення котирувань за період 5 років. Аналізуючи ціни акцій компанії Apple за 2011, 2012, 2013, 2014, 2015 роки. Нейронна мережа прогнозує ціну на 2016/2017 рік із певною часткою ймовірності. Таким чином, для побудови прогнозу потрібно знати значення котирувань минулих часових періодів. На рис.3.5. представлений набір дат та цін котирувань на кожну дату, які є тренувальною вибіркою нейронної мережі. Інформація «знімалася» щодня за винятком суботи та неділі. Тобто інтервал між зняттям даних не є рівномірним.

Вхідні значення котирувань – це значення, що подаються на вхід нейронної мережі. Цільові значення - це значення, яких повинна прагнути нейронна мережа при валідації. Виходячи з наявних даних, мережа має отримати набір вагових коефіцієнтів, що відображають залежність змін котирувань цін за

вказаний період. Помилка між реальним та отриманим значеннями нейронної мережі становить приблизно 3,7%.

При навчанні мережі визначається нейрон-переможець шляхом обчислення евклідової відстані, як показано у формулі

$$\left||x - w_c|\right|^2 = \min_j \left(\left||x - w_j|\right|^2\right) \quad (1)$$

де

x - вектор входів;

wc - вектор ваг;

wj - Вектор ваг j нейрона Кохонена.

Ваги нейрона, що виграв, змінюються за наступним правилом у поданій нижче формулі:

$$w_c(k+1) = \begin{cases} w_c(k) + \alpha_c(k) * [x(k) - w_c(k)] \\ w_c(k) - \alpha_c(k) * [x(k) - w_c(k)] \end{cases} \quad (2)$$

де

wc (k+1) - наступне значення ваги;

wc(k) - попереднє значення ваги;

x(k) - значення входу,

αc(k) - коефіцієнт навчання.

Перший варіант використовується, якщо x і wc належать одному класу, інакше використовується другий варіант. Параметр навчання αc(k) обчислюється за такою формулою:

$$\alpha_c(k) = \frac{\alpha_c(k-1)}{1+\alpha_c(k-1)*S(k)} \quad (3)$$

де

αc(k) - наступне значення параметра навчання;

αc(k-1) - Попереднє значення параметра;

S(k) обчислюється за такою формулою:

$$S(k) = \begin{cases} +1, \text{якщо } w \text{ та } x \text{ належать одному класу} \\ -1, \text{у протилежному випадку} \end{cases} \quad \text{..... (4)}$$

Вибір функції помилки

На цьому кроці визначили спосіб підрахунку середньої квадратичної помилки. У програмній реалізації використовується Mean Squared Error, який є середнім відхиленням на всіх ітераціях навчання.

Для перевірки точності тестування подамо на вхід значення тестуючої вибірки. Проаналізувавши їх, нейронна мережа має видати на виході значення, близькі до реальних. Про успішність навчання можна судити за ступенем відмінності тестованого та реального значень котирувань (цін). Значення, наведені у табл. 3.1, показують, що ступінь відмінності невелика, це свідчить про малу похибку та успішне тестування мережі.

Таблиця 3.1 – Значення та похибка

Дата	Значення ФАКТ	Вихід нейронної мережі
…….	…….	…….
2017-06-01	152.02	157.64
2017-06-02	152.43	158.1
2017-06-05	153.19	158.89
2017-06-06	152.75	158.71
2017-06-07	153.86	159.4
…….	…….	…….

Тепер оцінимо точність прогнозу. Для цього виберемо наступні за періодами навчальної вибірки періоди значення котирувань цінних паперів (табл. 3.1), які подаємо на вхід нейронної мережі. Отримавши виході певне значення, можемо чисельно оцінити точність прогнозування. Обробивши дані

тестованої вибірки, нейронна мережа отримала на виході значення котирувань цінних паперів, представлені в таблиці. 3.1.

Точність результату вимірів є характеристикою якості виміру, що відбиває близькість до нуля похибки її результату.

Крім того, розглянувши напрям руху цін у бік зростання та спадання, а також показань на виході нейронної мережі, можна зробити висновок про те, що моделлю показані ймовірні напрями змін цін.

Програмна реалізація

Програмна реалізація проведена мовою Python. Основною функцією програми є прогнозування цін на акції за допомогою нейромережі. Ця програма демонструє можливості нейромереж та мови Python.

Програма створена за допомогою хмарного сервісу Google Collaboratory від Google. Вона дозволяє провести упорядкування і систематизування інформації пов'язаной з цінами на акції, побудувати графіки динаміки цін на бирже, при заданих метриках. Розроблений додаток дозволяє гнучко моделювати нейронні мережі, що допоможе надалі отримати якіснішу модель для вирішення задачі.

При відкритті файлу з програмою, з’являється поле Jupyter notebook з кодом програми, яке розбите на кілька блоків (Рис. 3.3). Це блоки:

- бібліотеки;
- аналіз даних використовуючи бібліотеку Pandas;
- розбиття датафрейму на тренувальну та валідаційну частину;
- створення датасету для часового ряду;
- навчання RNN моделі використовуючи Tensorflow;
- усунення перенавчання. Метод регуляризації Dropout.

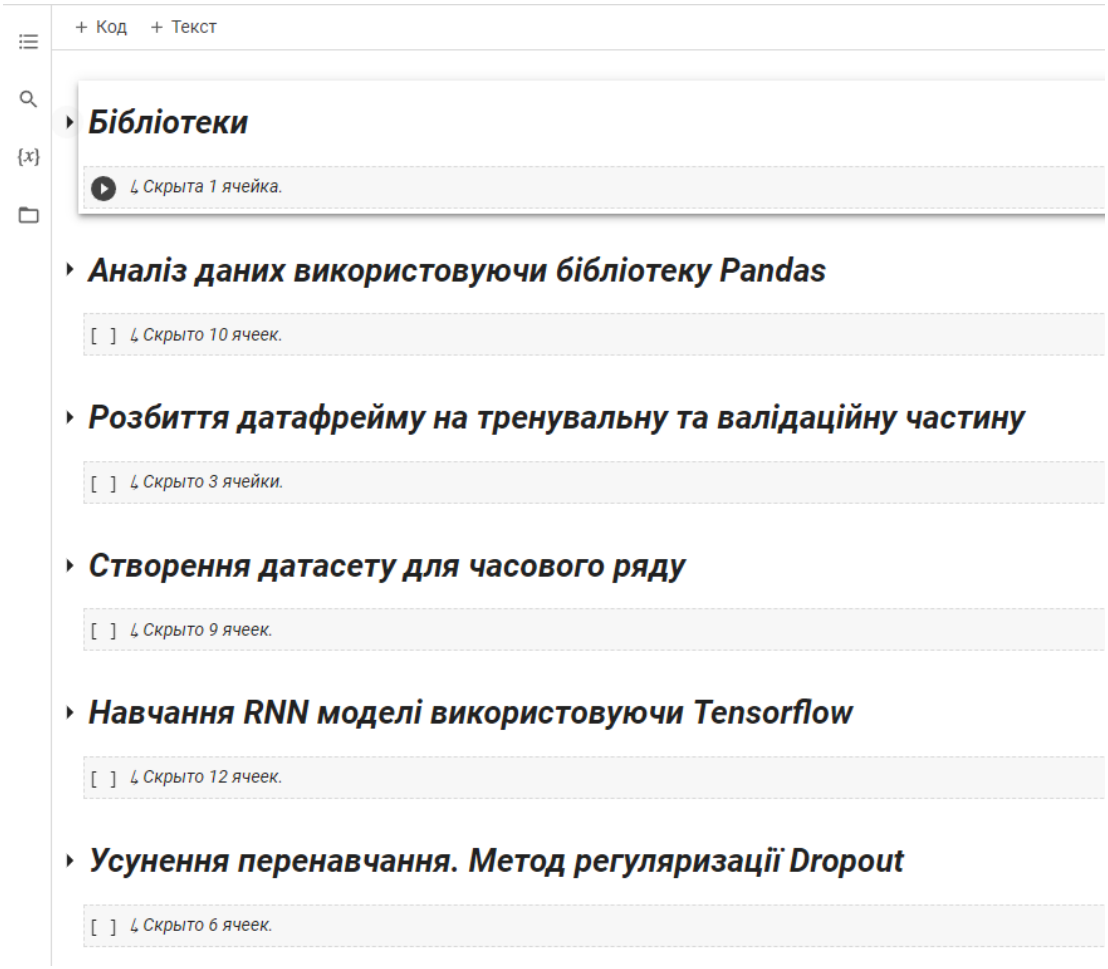


Рис. 3.3 – Поле з блоками коду

У першому блоці "Бібліотеки" знаходяться підключені до програми бібліотеки (Рис. 3.4)**:**

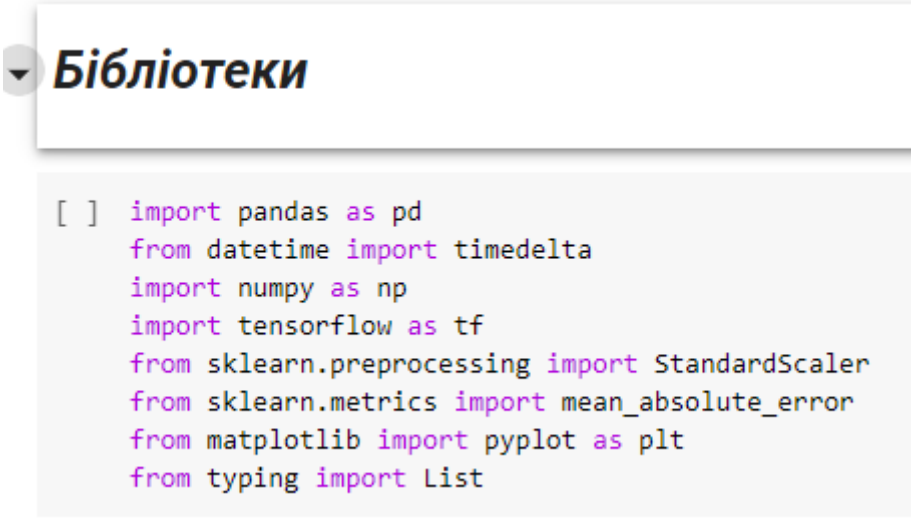


Рис. 3.4 – Поле блоку «Бібліотеки»

У блоці «Аналіз даних використовуючи бібліотеку Pandas» відбувається підключення файлу з даними, відображається розмір датафрейму, представлені перші п'ять колонок (для перевірки даних), також у програмі відображається період за яким є дані (Рис. 3.5).

```
[ ]  # читання файлу з даними
     df = pd.read_csv("aapl.us.txt", parse_dates=["Date"])
```

```
[ ]  # перегляд розміру датафрейму
     df.shape
```

```
(8364, 7)
```

```
[ ]  # перші п'ять колонок
     df.head(5)
```

	Date	Open	High	Low	Close	Volume	OpenInt
0	1984-09-07	0.42388	0.42902	0.41874	0.42388	23220030	0
1	1984-09-10	0.42388	0.42516	0.41366	0.42134	18022532	0
2	1984-09-11	0.42516	0.43668	0.42516	0.42902	42498199	0
3	1984-09-12	0.42902	0.43157	0.41618	0.41618	37125801	0
4	1984-09-13	0.43927	0.44052	0.43927	0.43927	57822062	0

```
[ ]  # період, за який є дані
     df["Date"].min(), df["Date"].max()
```

```
(Timestamp('1984-09-07 00:00:00'), Timestamp('2017-11-10 00:00:00'))
```

Рис. 3.5 – Аналіз даних підключених до програми

У цьому блоці також проведено побудова графіків, які відображають мінливість ціни. Ось y відповідає цінам на біржі, вісь x відповідає датам.

На початку було розглянуто мінливість ціни українських міжбанківських бірж на час відкриття торгів.

Бачимо, що з 1984 року по 2004 рік, на початок торгів ціни мало змінювалися рік від року (Рис.3.6.a). Хоча з 1999 по 2001 роки спостерігається невеликий сплеск ціни (Рис. 3.6.b). Це пояснюється форс-можором.

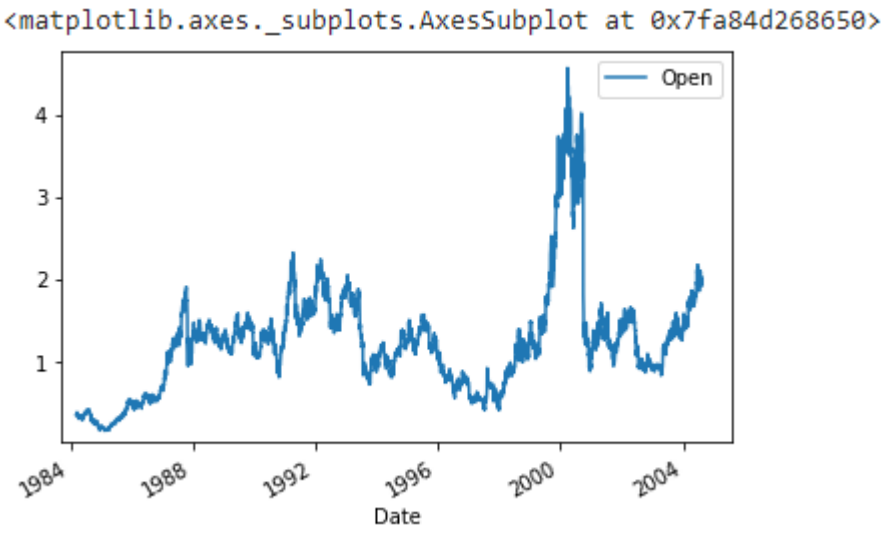


Рис. 3.6.a – Графіки зміни цін на акції на початку торгів

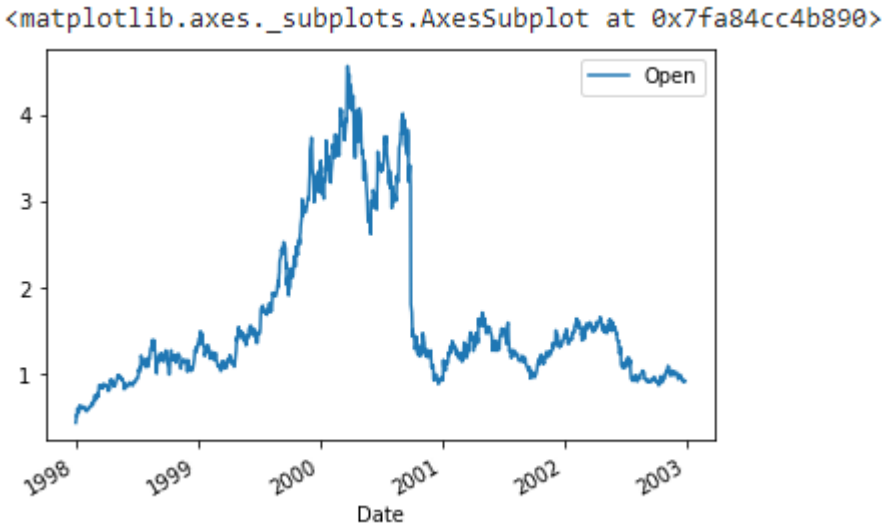


Рис. 3.6.b – Графіки зміни цін на акції на початку торгів у період з 1998 року по 2003 рік

З 2004 року відбувається значне зростання цін на біржі. Ціна безперервно зростає у 2004 році. Порівняно з 2004 роком у 2007 році ціна збільшилася на 686.52% відсотків.

У 2009 році відбувається різкий спад ціни на 103,48%. У 2009 році криза. У зв'язку із цим відбувається обвал цін (Рис.3.6.c).

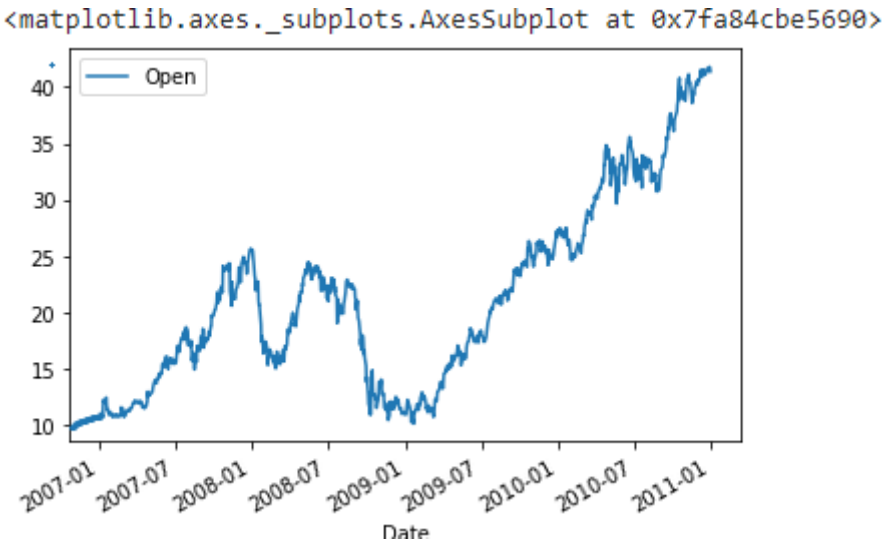


Рис. 3.6.c – Графіки зміни цін на акції на початку торгів у період з 2007 року до 2011 року

З 2010 до 2013 року ціни постійно зростають. У 2014 р. у зв'язку із кризою спостерігаємо черговий обвал цін.

Графік Рис. 3.6.d відображає максимальні ціни під час торгів.

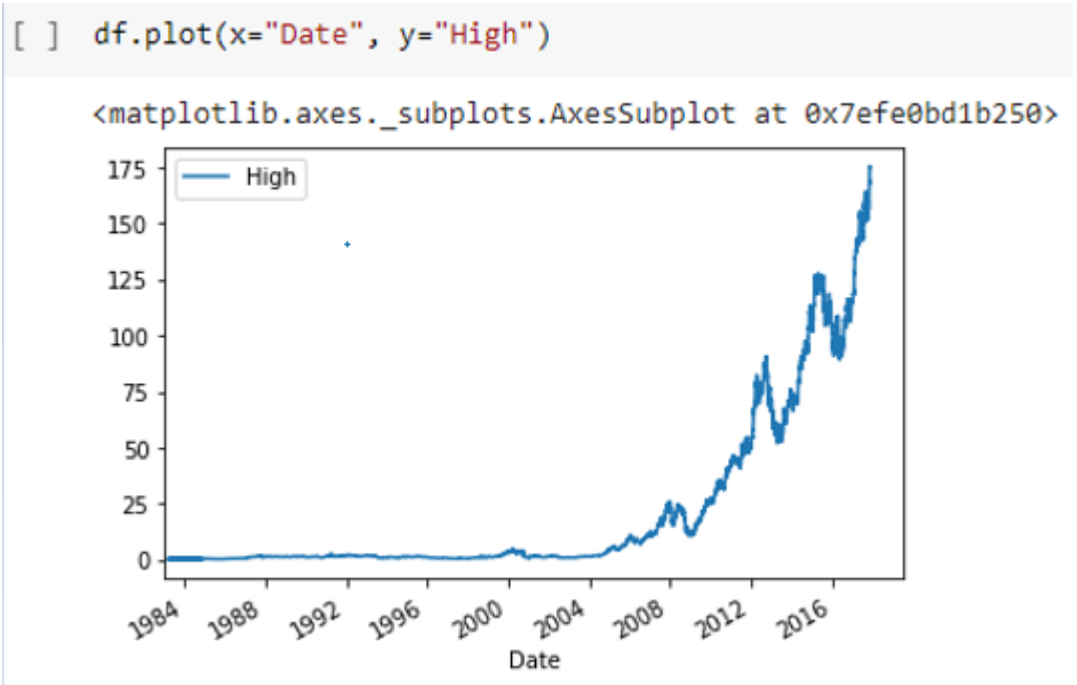


Рис. 3.6.d – Графіки максимальних цін під час торгів

Далі в цьому ж блоці проходить прорахунок попарної кореляції, щоб зрозуміти наскільки дані сильно залежать один від одного (Рис. 3.7).

```
[ ]  # прорахунок попарної кореляції
     df[["Open", "Close", "Low", "High"]].corr()
```

	Open	Close	Low	High
Open	1.000000	0.999902	0.999942	0.999956
Close	0.999902	1.000000	0.999955	0.999953
Low	0.999942	0.999955	1.000000	0.999928
High	0.999956	0.999953	0.999928	1.000000

Рис. 3.7 – Прорахунок попарної кореляції

У блоці «Розбиття датафрейму на тренувальну та валідаційну частину» відбувається підготовка даних для подальшої роботи з ними. Оскільки дані є за великий період часу, які були взяті, а саме ціни на акції Apple, то для зменшення часу навчання нейромережі було прийнято рішення взяти дані лише за останні 6 років. Далі відбувається розбиття наявних даних на тренувальну та валідаційну частину, за принципом 80% та 20% відповідно, і після розбиття виконуються кілька операцій для аналізу нових датафреймів (Рис. 3.8).

```
[ ]  # дані за останні 6 років
     df_6_yr = df[df["Date"] > df["Date"].max() - timedelta(days=365 * 6)]

     # перегляд нового розміру датафрейму
     df_6_yr.shape

     (1509, 7)

[ ]  # перегляд періоду нового датафрейму
     df_6_yr["Date"].min(), df_6_yr["Date"].max()

     (Timestamp('2011-11-14 00:00:00'), Timestamp('2017-11-10 00:00:00'))

[ ]  # 80% даних беремо у тренувальний датафрейм та 20% у валідаційний
     train_size = int(df_6_yr.shape[0] * 0.8)
     train_df = df_6_yr.iloc[:train_size]
     val_df = df_6_yr.iloc[train_size:]

[ ]  # перегляд розмірів тренувального та валідаційного датафреймів
     train_df.shape, val_df.shape

     ((1207, 7), (302, 7))

[ ]  # перегляд мінімальної та максимальної дати в кожному з датафреймів
     train_df["Date"].min(), train_df["Date"].max(), val_df["Date"].min(), val_df["Date"].max()

     (Timestamp('2011-11-14 00:00:00'),
      Timestamp('2016-08-31 00:00:00'),
      Timestamp('2016-09-01 00:00:00'),
      Timestamp('2017-11-10 00:00:00'))
```

Рис. 3.8 – Блок «Розбиття датафрейму на тренувальну та валідаційну частину»

У наступному блоці відбувається створення датасету для часового ряду. Починається блок із створення скейлера для нормалізації фічів та з функції створення датасету (Рис. 3.9).

```
# scaler - класи бібліотеки sklearn для нормалізації фічів
# для передбачення n+1 елемента буде використовуватися n попередніх як фічі
scaler = StandardScaler()
scaler.fit(train_df[["Low"]])

# на вхід функція приймає датафрейм, кількість елементів тимчасового ряду,
# кількість елементів в батчі для навчання, скейлер, змішування елементів череш шафл
def make_dataset(
    df,
     window_size,
     batch_size,
     use_scaler=True,
     shuffle=True
     ):
  features = df[["Low"]].iloc[:-window_size]
# нормалізація якщо потрібно
  if use_scaler:
    features = scaler.transform(features)
  data = np.array(features, dtype=np.float32)
  ds = tf.keras.preprocessing.timeseries_dataset_from_array(
      data=data,
      targets=df["Low"].iloc[window_size:],
      sequence_length=window_size,
      sequence_stride=1,
      shuffle=shuffle,
      batch_size=batch_size)
  return ds
```

Рис. 3.9 – Функція створення датасету для часового ряду

Після функції створюється датасет для прикладу, а також відбувається аналіз отриманого датасету (Рис. 3.10).

```
[ ] # дата сет для прикладу, вхідні дані - тренувальний датасет
    example_ds = make_dataset(df=train_df, window_size=3, batch_size=2, use_scaler=False, shuffle=False)

[ ] # отримання елементів з датасету по одному за допомогою iterator
    example_feature, example_label = next(example_ds.as_numpy_iterator())

[ ] # розмір фіч - багатовимірний вектор (розмір батча, довжина послідовності елементів, розмірність кожної фічі)
    example_feature.shape

    (2, 3, 1)

[ ] # розмірність лейблів, так як в батчі 2 елементи то 2 лейбла
    example_label.shape

    (2,)

[ ] # перші 6 елементів з датафрейму
    train_df["Low"].iloc[:6]

    6855    48.432
    6856    48.592
    6857    49.217
    6858    48.086
    6859    48.009
    6860    46.860
    Name: Low, dtype: float64

[ ] # елементи з батча
    print(example_feature[0])
    print(example_label[0])

    [[48.432]
     [48.592]
     [49.217]]
    48.086

[ ] print(example_feature[1])
    print(example_label[1])

    [[48.592]
     [49.217]
     [48.086]]
    48.009
```

Рис. 3.10 – Створення та аналіз створеного для прикладу датасету

Після створення датасета для прикладу створюється вже справжній датасет для тренувального та для валідаційного датафрейму, за яким і відбуватиметься навчання моделі (Рис. 3.11).

```
[ ] # створення датасету для тренерувального та валідаційного датафрейму
    window_size = 10
    batch_size = 8
    train_ds = make_dataset(df=train_df, window_size=window_size, batch_size=batch_size, use_scaler=True, shuffle=True)
    val_ds = make_dataset(df=val_df, window_size=window_size, batch_size=batch_size, use_scaler=True, shuffle=True)
```

Рис. 3.11 – Створення датасету для тренувального та валідаційного датафрему

Далі йде створення моделі самої нейромережі за допомогою класу sequential (Рис. 3.12).

```
[ ]  # Sequential - всі шари виконуються послідовно
     # LSTM - блок для рекурентної нейромережі
     lstm_model = tf.keras.models.Sequential([
         tf.keras.layers.LSTM(32, return_sequences=False),
         tf.keras.layers.Dense(1)
     ])
```

Рис. 3.12 – Створення моделі нейромережі

Після створення моделі нейронної мережі йде функція компіляції та навчання моделі (Рис. 3.13).

```
# функція компіляції та навчання моделі
def compile_and_fit(model, train_ds, val_ds, num_epochs: int = 20):
  model.compile(
        loss=tf.losses.MeanSquaredError(),
        optimizer=tf.optimizers.Adam(),
        metrics=[tf.metrics.MeanAbsoluteError()]
      )
  history = model.fit(
      train_ds,
      epochs=num_epochs,
      validation_data=val_ds,
      verbose=0
      )
  return history
```

Рис. 3.13 – Функція компіляції та навчання моделі

Далі відбувається безпосередньо навчання створеної моделі нейронної мережі з кількістю епох, що дорівнює 100 (Рис. 3.14).

```
[ ]  # навчання моделі
     history =  compile_and_fit(lstm_model, train_ds, val_ds, num_epochs=100)
```

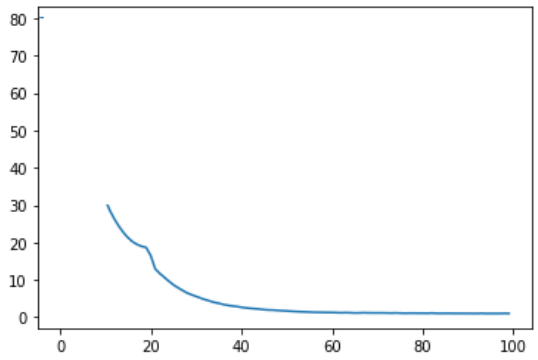
Рис. 3.14 – Навчання нейромережі (100 епох)

Після навчання за допомогою графіків було роздруковано значення

метрик для тренувального та валідаційного датасету (Рис. 3.15).

```
[ ]  # значення метрик для тренувального та валідаційного датасету
     plt.plot(history.history['mean_absolute_error'])
```

```
[<matplotlib.lines.Line2D at 0x7efe0d6e4810>]
```




```
[ ]  plt.plot(history.history['val_mean_absolute_error'])
```

```
[<matplotlib.lines.Line2D at 0x7efe0cfa4dd0>]
```

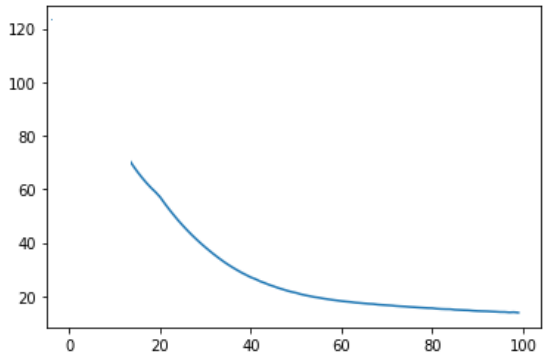


Рис. 3.15 – Значення метрик для тренувального та валідаційного датасету (100 епох)

Судячи з графіків значення метрики для тренувального датасету менше, ніж для валідаційного, це виглядає, ніби є перенавчання. Тому далі йде точніший розрахунок метрик для тренувального та валідаційного датасету (Рис. 3.16).

```
[ ]  # розрахуємо метрики точно
     lstm_model.evaluate(train_ds)
```

```
149/149 [==============================] - 0s 3ms/step - loss: 2.5877 - mean_absolute_error: 1.1258
[2.5876500606536865, 1.1257712841033936]
```

```
[ ]  lstm_model.evaluate(val_ds)
```

```
36/36 [==============================] - 0s 2ms/step - loss: 321.6702 - mean_absolute_error: 13.8332
[321.6701965332031, 13.833154678344727]
```

Рис. 3.16 – Точний розрахунок метрик (100 епох)

Завдяки метрикам видно велике перенавчання. Для того, щоб точно побачити перенавчання, було прийнято рішення збільшити кількість епох до 500 (Рис. 3.17).

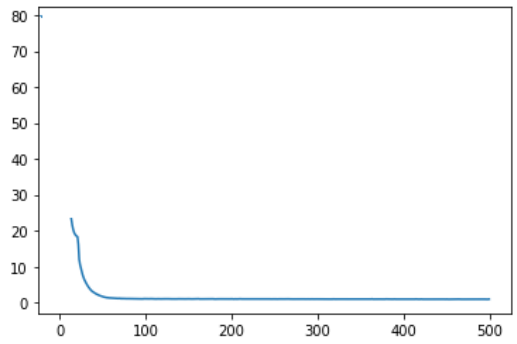
```
[ ]  # завдяки розрахункам видно перенавчання
     # спробуємо зробити більше епох
     lstm_model = tf.keras.models.Sequential([
         tf.keras.layers.LSTM(32, return_sequences=False),
         tf.keras.layers.Dense(1)
     ])

     history =  compile_and_fit(lstm_model, train_ds, val_ds, num_epochs=500)
```

Рис. 3.17 – Навчання нейромережі (500 епох)

Також як і при 100 епох були роздруковані графіки значень метрик і був зроблений точніший розрахунок метрик (Рис. 3.18 та Рис. 3.19).

```
[ ]  plt.plot(history.history['mean_absolute_error'])
```

```
[<matplotlib.lines.Line2D at 0x7efe0bb464d0>]
```




```
[ ]  plt.plot(history.history['val_mean_absolute_error'])
```

```
[<matplotlib.lines.Line2D at 0x7efe0c1a6cd0>]
```

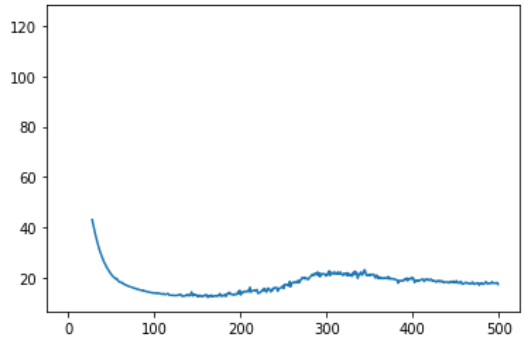


Рис. 3.18 – Значення метрик для тренувального та валідаційного датасету (500 епох)

```
[ ]  lstm_model.evaluate(train_ds)

     149/149 [==============================] - 1s 3ms/step - loss: 1.7565 - mean_absolute_error: 0.9134
     [1.7564606666564941, 0.9134067296981812]

[ ]  # значення для тренувального датасету зменшилося а для валідаційного датасету збільшилося, в наявності перенавчання
     lstm_model.evaluate(val_ds)

     36/36 [==============================] - 0s 3ms/step - loss: 502.9006 - mean_absolute_error: 17.4285
     [502.9006042480469, 17.428464889526367]
```

Рис. 3.19 – Точний розрахунок метрик (500 епох)

Видно що значення для тренувального датасету зменшилося а для валідаційного датасету збільшилося, в наявності перенавчання.

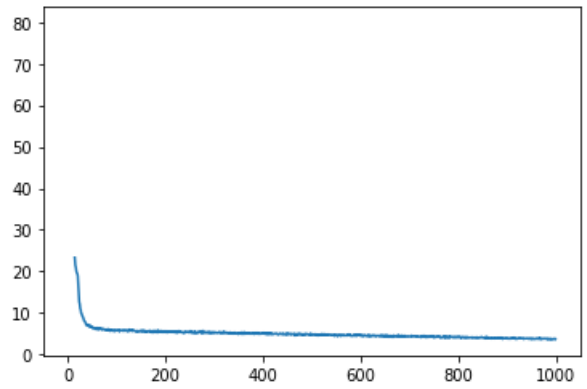
Для боротьби з перенавчанням було прийнято рішення використати метод регуляризації Dropout. Для цього було додано ще один шар у модель, а саме шар Dropout (Рис. 3.20).

```
[ ]  # додаємо ще один шар, шар dropout
     lstm_model = tf.keras.models.Sequential([
         tf.keras.layers.LSTM(32, return_sequences=False),
         tf.keras.layers.Dropout(0.2),
         tf.keras.layers.Dense(1)
     ])

     history =  compile_and_fit(lstm_model, train_ds, val_ds, num_epochs=1000)
```

Рис. 3.20 – Додавання нового шару у модель

Після навчання моделі з новим шаром за допомогою графіків було роздруковано значення метрик для тренувального та валідаційного датасету, а також було збільшено кількість епох до 1000 (Рис. 3.21).

```
[ ]  plt.plot(history.history['mean_absolute_error'])   о датасету

[<matplotlib.lines.Line2D at 0x7efe0c3c19d0>]
```




```
[ ]  plt.plot(history.history['val_mean_absolute_error']
                                                        )

[<matplotlib.lines.Line2D at 0x7efe0e9f40d0>]
```

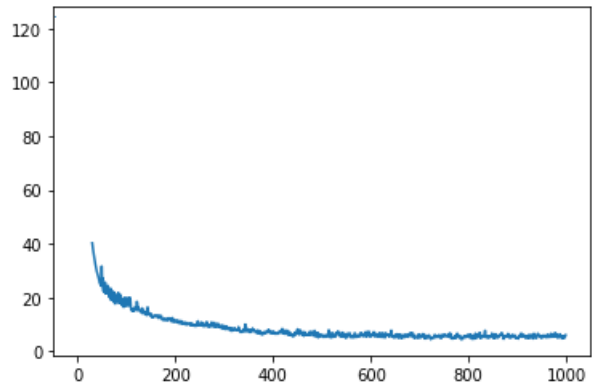


Рис. 3.21 – Значення метрик для тренувального та валідаційного датасету (1000 епох з Dropout)

А також були розраховані метрики для тренувального та валідаційного датасету для більш точної картини (Рис. 3.22).

```
[ ]  lstm_model.evaluate(train_ds)

149/149 [==============================] - 0s 3ms/step - loss: 2.3021 - mean_absolute_error: 1.0739
[2.3021280765533447, 1.0738967657089233]

[ ]  # видно що значно покращилася метрика для валідаційного датасету
     lstm_model.evaluate(val_ds)

36/36 [==============================] - 0s 3ms/step - loss: 37.6870 - mean_absolute_error: 4.7495
[37.68696212768555, 4.7494659423828125]
```

Рис. 3.22 – Точний розрахунок метрик (1000 епох з Dropout)

Видно, що значно покращилася метрика за валідаційним датасетом, тобто перенавчання усунуто.

Висновки до третього розділу

Результати обчислювальних експериментів дали цікавий результат.

Аналіз динаміки цін показав, що ціна досить довго рухається в одному напрямі. Після навчання мережі було опрацьовано дані за 5 років. Зростання цін мережа дає досить впевнено. Але на практиці важливішими є прогноз прогнозування падіння цін, тобто зміна ціни з високого рівня на нижчий.

Пробували перед навчанням мережі провести нормування даних, але це не покращило результат прогнозу.

Середня відносна похибка, розрахована для тестової множини, становить 3.7%. Це не є задовільний результат для короткострокового прогнозу. У разі, коли йдеться про фінансові торги щодня, мережу не можна вважати придатною. Така погрішність пояснюється тим, що в денних даних багато шуму. Але дана мережа вловлює тенденцію за довгострокового прогнозу в умовах політичної стабільності (відсутності «форс-мажору»)

4 ДОСЛІДЖЕННЯ НЕЙРОМЕРЕЖ

4.1 Підготовка до експерименту

Актуальність та доцільність застосування методів машинного навчання на фінансових ринках продиктована їхньою ефективністю у вирішенні завдань прогнозування часових рядів. Нейронні мережі вирішують такого роду завдання набагато ефективніше за класичні алгоритми статистичного аналізу, побудованих на ковзних середніх, авторегресії та умовній гетероскедастичності.

Для прогнозування цін за допомогою нейромереж використовують багато способів аналізів. Розглянемо декілька способів:

1) Морфологічний аналіз – метод прогнозування, в основу якого покладена побудова матриці характеристик ринку і їхніх можливих значень. Далі на основі відбору характеристик ринку і їхніх значень одержують різні варіанти прогнозу.

2) Фундаментальний аналіз – метод аналізу фінансових ринків, що базується на вивчені інформації. Ця ймовірно, впливає на динаміку активу або фінансового інструменту.

3) Незважаючи на те, що методи та прийоми фундаментального аналізу досить об'єктивні, його дуже складно формалізувати, оскільки об'єм інформації та її різна інтерпретація учасниками ринку надають елементи мисте

Актуальність·та·доцільність·застосування·методів·машинного·навчання·на·фінансових·ринках·продиктована·їхньою·ефективністю·у·вирішенні·завдань·прогнозування·часових·рядів.·Нейронні·мережі·вирішують·такого·роду·завдання·набагато· ефективніше· за· класичні· алгоритми· статистичного· аналізу,· побудованих·на·ковзних·середніх,·авторегресії·та·умовній·гетероскедастичності.¶

Для·прогнозування·цін·за·допомогою·нейромереж·використовують·багато·способів·аналізів.·Розглянемо·декілька·способів:¶

1) → Морфологічний· аналіз· –· метод· прогнозування,· в· основу· якого· покладена·побудова·матриці·характеристик·ринку·і·їхніх·можливих·значень.·Далі· на·основі·відбору·характеристик·ринку·і·їхніх·значень·одержують·різні·варіанти· прогнозу.·¶

2) → Фундаментальний· аналіз· –· метод· аналізу· фінансових· ринків,· що· базується·на·вивчені·інформації.·Ця··ймовірно,·впливає·на·динаміку·активу·або· фінансового·інструменту.¶

3) → Незважаючи·на·те,·що·методи·та·прийоми·фундаментального·аналізу· досить·об'єктивні,·його·дуже·складно·формалізувати,·оскільки·об'єм·інформації· та·її·різна·інтерпретація·учасниками·ринку·надають·елементи·мистецтва.·¶

цтва.

4) Технічний аналіз – прогнозування майбутнього курсу ціни на основі аналізу динаміки цієї ціни у встановлений відрізок минулого часу. Ціна в даному випадку яка представлена у вигляді часового ряду – залежності показника від конкретного моменту часу. Окрім ціни, об’єктом аналізу може бути інша інформація, наприклад об'єми торгів та інші статистичні дані.

Технічний аналіз містить в собі такі популярні методи:

– Рекурентний аналіз – полягає у побудові спеціальної рекурентної діаграми часового ряду та подальшого аналізу побудованої діаграми.

– Нейроні мережі – обирається певна модель (архітектура) мережі, яка приймає на вхід часовий ряд, та на основі циклу навчання та тренування, прогнозує ряд. Точність прогнозу залежить від обраної архітектури нейронної

мережі, розміру вхідного та прихованого шарів, об’єму вхідного часового ряду, тощо.

Серед усіх варіантів, самими ефективними та точними є рекурентний та нейромережевий аналізи, адже вони здатні виявити характеристику вхідних даних, забезпечити достатню точність обчислення та спрогнозувати динаміку ціни на обраний момент майбутнього часу. Отже, саме ці методи й будуть використовуватися у подальшому.

4.2 Проведення експерименту

Перш ніж розпочати працювати із прогнозуванням ціноутворення, переконаємося, що часові ряди цін на ці метали є хаотичними, або, іншими словами, невпорядкованими. Саме цей фактори є ознакою того, що часовий ряд цін є дуже важкопередбачуваним, та для його прогнозування потрібне застосування алгоритмів нейронної мережі.

У якості часових рядів візьмемо дані курсу цін за 2011-2017 р. та дослідимо їх на предмет хаотичності.

Побудуємо графіки динаміки цін за 2011-2017 р.:

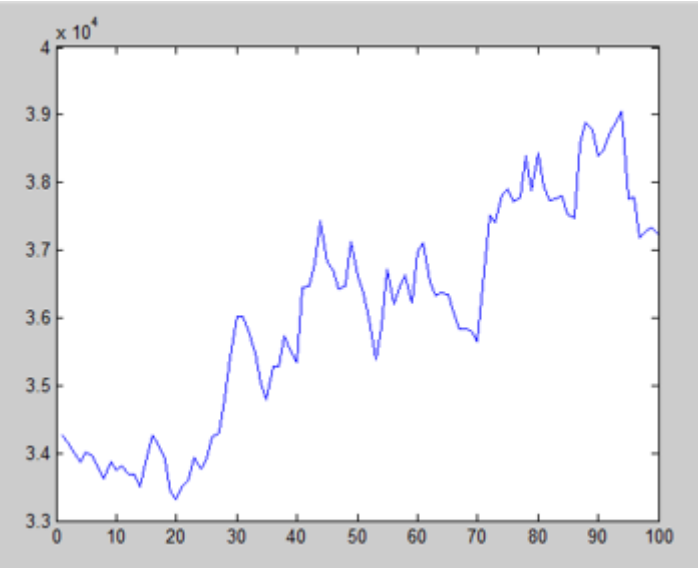
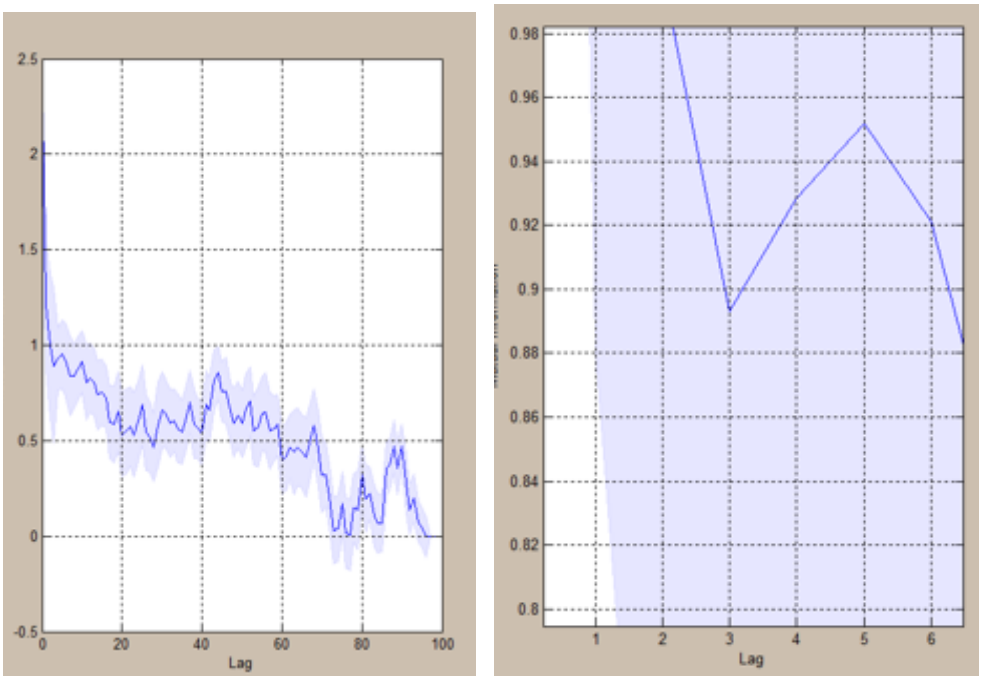


Рис. 4.1 – Графік курсу ціни

Для визначення хаотичності об’єктів дослідження необхідно визначити розмір запізнювання часового ряду. Запізнювання такого часового ряду показано на Рис. 4.2 як перший мінімум графіку загальної інформації ряду:



а) *б)*

Рис. 4.2 – а) загальна інформація часового ряду в залежності від запізнювання;

б) Запізнювання ряду (*Lag*=3).

Наступним кроком визначаємо розмірність простору вкладення (або іншими словами, фазового простору. Результат процедури за методом помилкових сусідів зображено на Рис. 4.3:

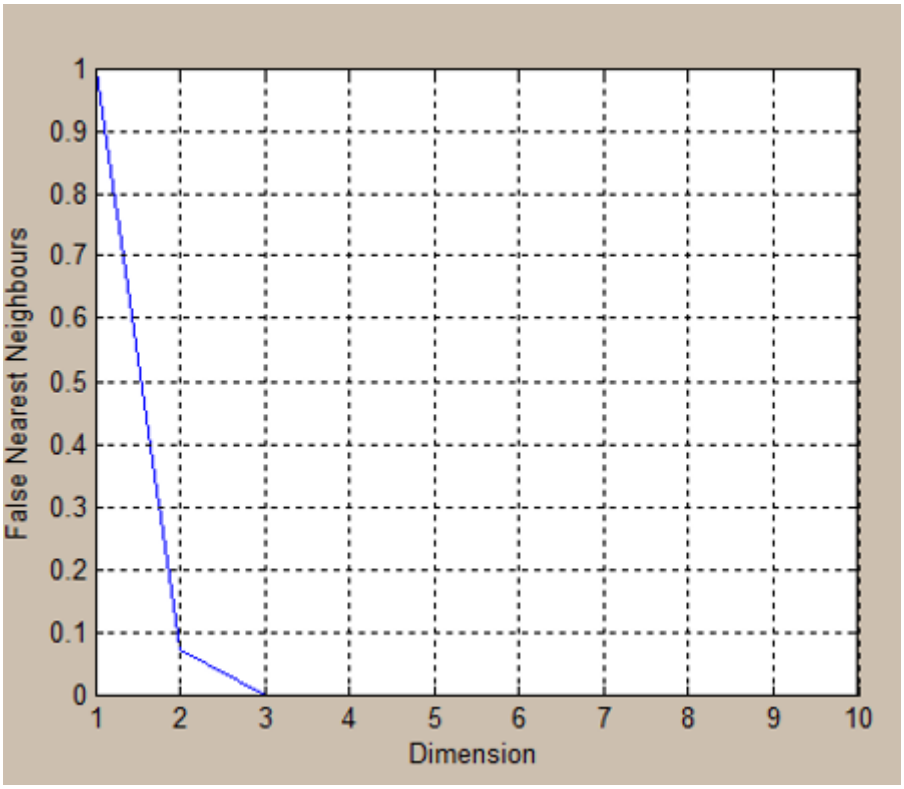


Рис. 4.3 – Розмірність фазового простору часового ряду
(на графіку це значення дорівнює 3),
із вказаним запізнюванням (*Delay*=3).

Знайдена розмірність є більшою за 2, що свідчить про наявність хаотичності у розглянутому курсі, а це означає, що часовий ряд цього курсу відповідає заявленим вимогам та може бути експлуатований у обробці нейронними мережами для здійснення прогнозу на майбутній період часу.

Для більш детальної інформації побудуємо рекуренту діаграму цін Рис. 4.4:

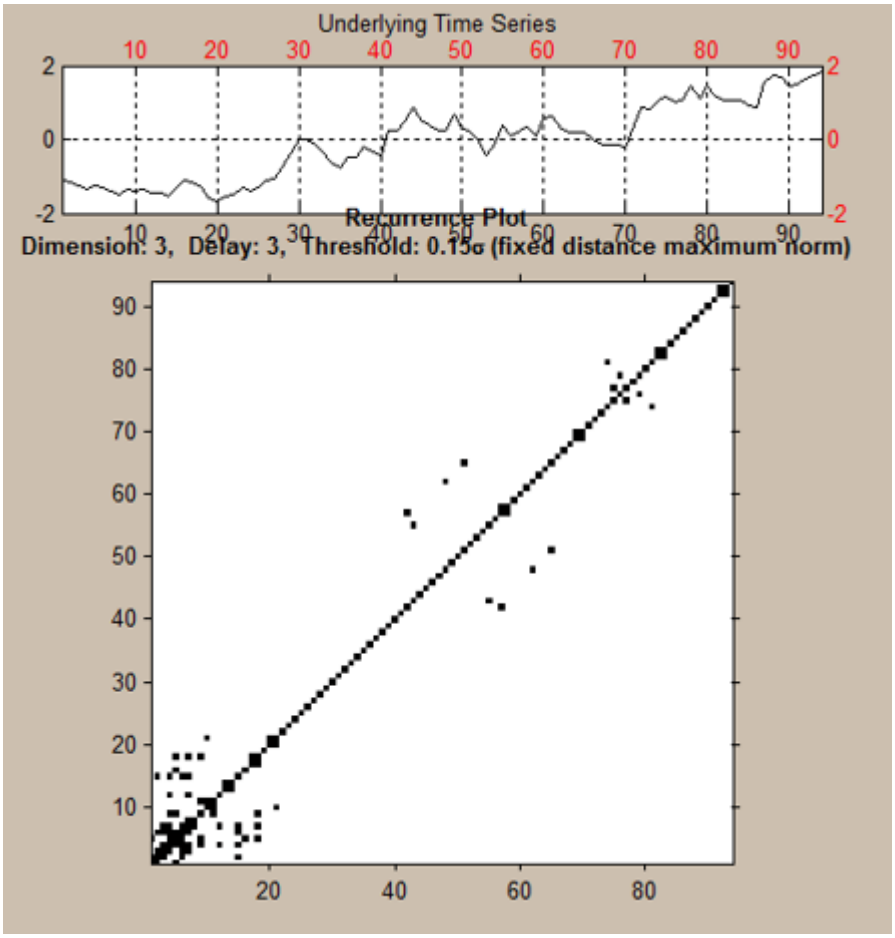


Рис. 4.4 – Рекурентна діаграма часового ряду
(*Dimension*=3 – розмірність фазового простору,
Delay=3 – запізнювання,
Threshold=0.15 – порогова відстань між точками ряду).

На Рис. 4.5 нижче наведені результати кількісного рекурентного аналізу курсу ціни у вигляді графіків різноманітних характеристик:

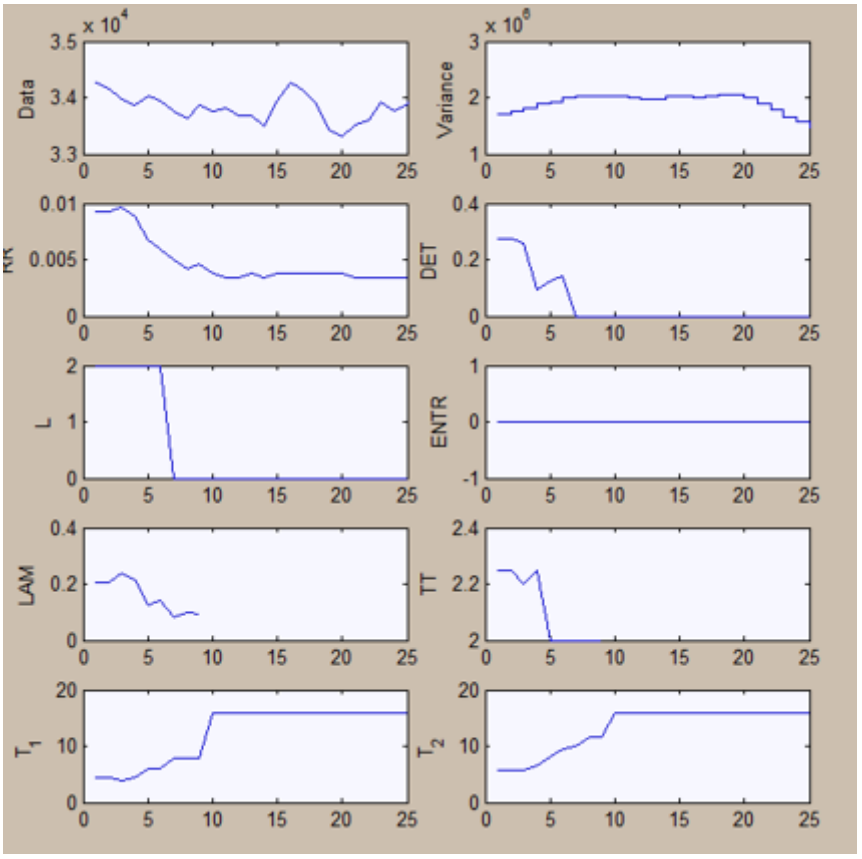


Рис 4.5 – Результати кількісного рекурентного аналізу курсу ціни

Визначивши, що часові ряди курсу відповідає критерію хаотичності, можемо перейти до розв’язання задачі прогнозування із використанням штучної нейронної мережі.

У якості нейронні мережі, розглянемо та використаємо у прогнозуванні часових рядів такі архітектури, як LSTM .

Структура LSTM також нагадує ланцюжок. Замість одного шару нейронної мережі вони містять чотири, і ці шари взаємодіють особливим чином.

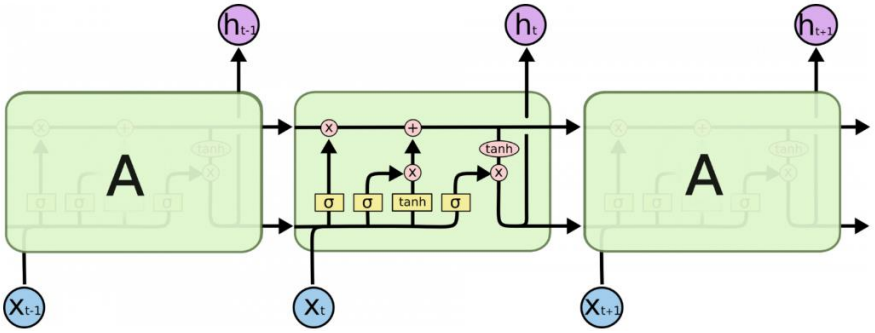


Рис 4.6 – Структура LSTM

На вхід нейронної мережі подається часовий ряд *y(t)*. У прихованому шарі мережа здійснює навчання. Опираючись на часовий ряд, відкидаєм його перших **b** елементів, та потім у вихідний шар виводить навчений часовий ряд – масив пробних даних, генерованих мережею під час навчання. Навчений ряд повинен збігатися із оригінальним рядом із допустимим порогом похибки.

Слід зазначити, що перед початком процедури навчання, в налаштуваннях нейромережі потрібно визначити наступні параметри:

- ***Training*** – число елементів ряду *y(t)*, яке мережа застосує безпосередньо для навчання;
- ***Validation*** – кількість елементів *y(t)*, яке мережа буде використовувати для перевірки узагальнення. При припиненні узагальнення мережі, процес навчання переривається;
- ***Testing*** – частина ряду *y(t)* для перевірки результатів навчання.

Усі вищезазначені значення задаються в нейронної мережі у відсотковому співвідношенні. Налаштування параметрів навчання нейронної мережі:

Training=70%, *Validation*=15%, *Testing*=15%. Тут і далі будуть використовуватися саме такі значення.

Налаштував усі необхідні компоненти, нейрона мережа обробляє вхідні дані часового ряду та формує навчений ряд. Після цього, LSTM-модель нейронної мережі замикається, та часовий зсувається на один елемент вліво.

Після замикання, мережа безпосередньо здійснює прогнозування зміщеного ряду *y(t)* на 1 елемент вперед та відає на вихід передбачене значення *y(t+1)*.

Дана операція виконує прогноз ряду всього на 1 крок вперед. Але це застосовується ще й для того, щоб навчальний часовий ряд, перетворений при замкненні мережі, був конвертований назад до істинних прогнозованих значень цієї мережі.

Тепер, можемо спрогнозувати майбутні значення на будь-яку кількість елементів вперед. Ця кількість значень може бути будь-яким додатним цілим числом.

Для цього буде застосовуватись алгоритм обчислення, наданий на Рис. 4.7:

```
% 5. Multi-step ahead prediction
N=10;
targetSeriesPred = [targetSeries(end-1:end), con2seq(nan(1,N))];
[Xs,Xi,Ai,Ts] = preparets(netc,{},{},targetSeriesPred);
yPred = netc(Xs,Xi,Ai);
perf = perform(net,Ts,yPred);
% yPred = [cell2mat(targetSeries),cell2mat(yPred)];
yPred = [ys(1:end-1),cell2mat(yPred)];
```

Рис. 4.7 – Вихідний код прогнозування часового ряду (*N*=10 – кількість кроків).

Прогнозування із використанням нейронної мережі LSTM

Першим кроком треба задати у мережу вхідну навчальну вибірку. Такою вибіркою буде служити досліджуваний часовий ряд.

Візьмемо часовbq ряд цінового банківського курсу та порівняємо передбачені значення цього ряда із фактичними даними майбутнього періоду кроком на 10 днів для кожного ряду.

На Рис. 4.8 нижче зображено вхідний ряд значень цін періодом 100 днів та разом з ним навчений на його основі нейронною мережею часовий ряд:

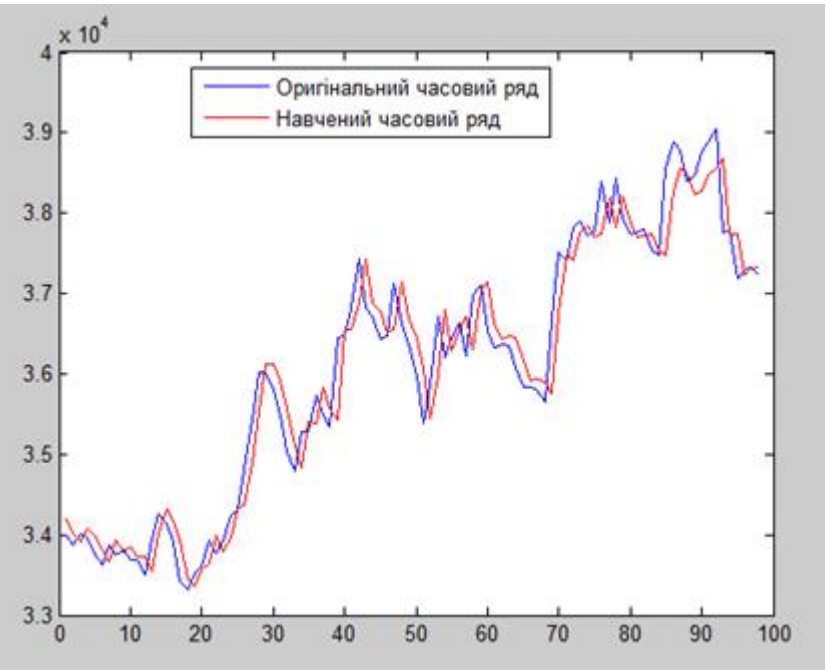


Рис. 4.8 – Порівняння графіків вхідних даних до та після навчання.

Навчений часовий ряд співпадає із фактичним рядом за формою, але має розбіжності у числових значеннях із помітною похибкою передбачення.

Додамо до ряду обчислені прогнозовані значення та оцінимо, наскільки відхиляється передбачення від реальних даних. Результат зображено на Рис. 4.9:

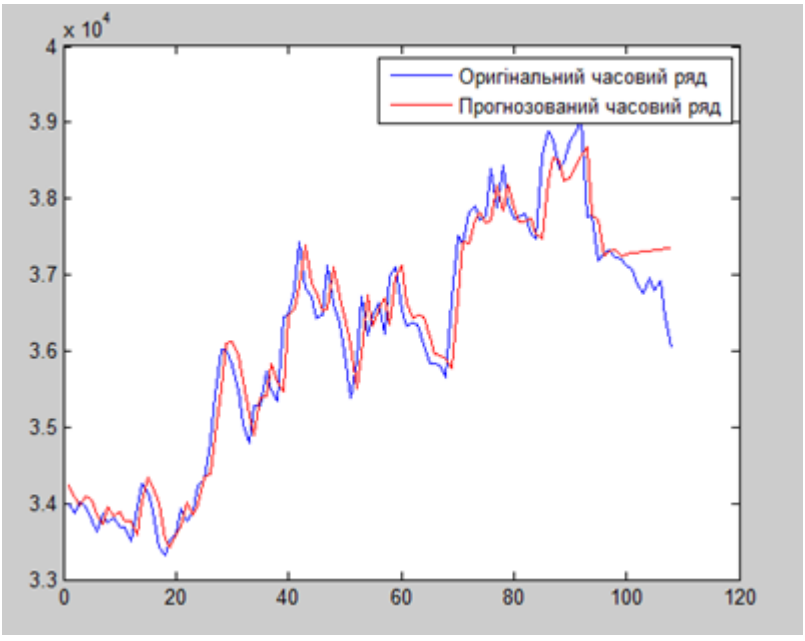


Рис. 4.9: Порівняння графіків реального та прогнозованого курсу цін.

Легко бачити, що передбачені ціни золота у наступні 10 пунктів абсолютно не відповідають дійсним значенням курсу наступних 10 кроків.

Алгоритм процедури передбачення здійснюється із великою похибкою та не може визначити тенденцію розвитку часового ряду.

Розглянутє у цьому експерименті, прогнозування нейронною мережею подальшого курсу зміни цін значно відрізняється від реального курсу цін на обраний майбутній термін. Передбачений інтервал значень графіку не відповідає самим точкам фактичних цін, а динаміка також разюче відхиляється від інтервалу справжніх цін.

Отже, як показало тестування, нейронна мережа моделі **LSTM** добре здійснює навчання вхідного часового ряду, навчений масив значень співпадає за функціональними та хаотичними характеристиками збігається із фактичним набором точок ряду, але відхиляється від нього у абсолютних значеннях цих точок. Алгоритм багатокрокового передбачення не дає прийнятні результати. Така розбіжність отриманих даних із істинними показниками цін може виникати через те, що невпорядкований часовий ряд не сформовано ані за яким математичним правилом. Хаотична послідовність даних ряду створює фактор невизначеності утворення цих даних та ускладнює можливість для максимально приблизного передбачення нейронної мережі наступних значень даного ряду.

Спробуємо змінити алгоритм прогнозування, видалив процедуру багатокрокового передбачення та замість неї передбачати часовий ряд на **один** елемент вперед (із відповідним зсувом самого ряду) задану кількість разів (у нашому випадку – 10). Перетворений алгоритм прогнозування виглядає наступним чином (див. Рис. 4.10):

```
for x=1.0:1:N

targetSeries = tonndata(numbers,true,false);

[inputs,inputStates,layerStates,targets] = preparets(net,{},{},targetSeries);

% Time series partitions
net.divideParam.trainRatio = 75/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;


% Train
[net,tr] = train(net,inputs,targets,inputStates,layerStates);

% Test
outputs = net(inputs,inputStates,layerStates);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs);

% View the Network
% view(net);

```

Рис. 4.10 – Початок циклу однокрокового передбачення.

```
% Closed Loop Network
netc = closeloop(net);
[xc,xic,aic,tc] = preparets(netc,{},{},targetSeries);
yc = netc(xc,xic,aic);
perfc = perform(net,tc,yc);
% view(netc);

% One Step Ahead Prediction Network
nets = removedelay(net);
[xs,xis,ais,ts] = preparets(nets,{},{},targetSeries);
ys = nets(xs,xis,ais);
closedLoopPerformance = perform(net,tc,yc);
% view(nets);

stepAheadPerformance = perform(nets,ts,ys);
ys = cell2mat(ys);

numbers=[numbers(2),ys];
end
```

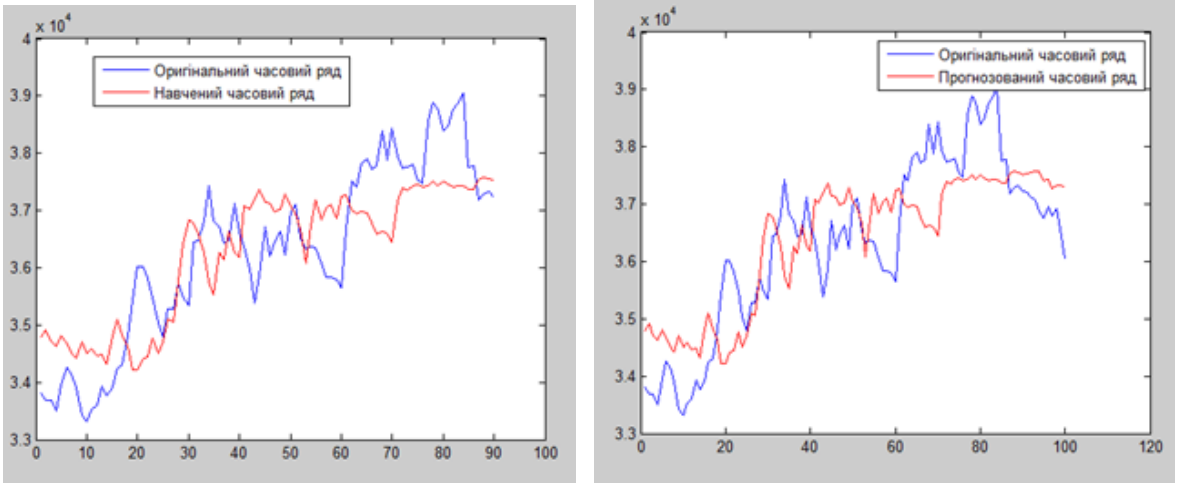
Рис. 4.11 – Кінець циклу однокрокового передбачення.

Перевіримо результати прогнозування такого варіанту алгоритму та порівняємо їх із попередніми отриманими.

Оскільки тепер нейронна мережа здійснює десятикратне однокрокове передбачення, то й процедура навчання часового ряду буде повторюватися відповідну кількість разів, що може вплинути на результати навчання цієї мереж.

На Рис. 4.12 (а–б) показано навчений та передбачений курс цін

інтервалом на 10 точок вперед. Для експеременту взяли дані, у ті годи, коли була не стабільна сітуація (2014):

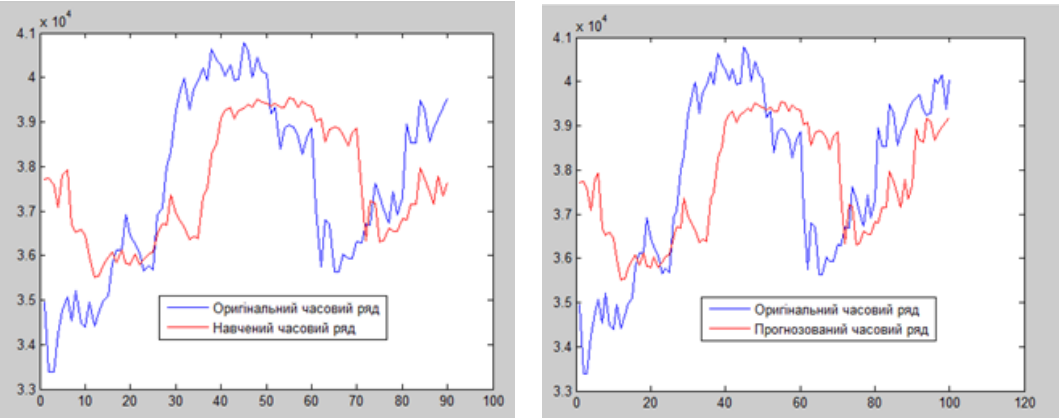


а) *б)*

Рис. 4.12 – Результат навчання (а) та прогнозування(б) нейронної мережі за новим алгоритмом для 2014 року.

Бачимо, що навчання **LSTM**-мережі значно погіршилось внаслідок багаторазового повторення постійно зміщеного часового ряду, проте передбачений інтервал результуючого графіку за тенденцією починає нагадувати справжні значення курсу на 10 кроків вперед.

Перевіримо цю закономірність у випадку стабільної політичної сітуації. На Рис. 4.13 (а–б) зображені навчений та прогнозований часові ряди курсу ціни **у стабільній сітуації**:



а) *б)*

Рис. 4.13: Реальний, навчений (а) та прогнозований (б) **LSTM**-мережею часові ряди цін.

Тут тенденція прогнозу ряду також дуже значно співпадає із тенденцію справжніх даних курсу ціни.

Висновки до четвертого розділу

У підсумку роботи із нейронною мережею **NAR** можна заключити, що ця модель здатна навчатися на основі вибірки вхідних даних та видавати наступні значення часового ряду на же кулька кроків вперед. Але якщо на вхід до мережі йде часовий ряд невпорядкованих, хаотичних, не основаних ані на якій математичної асоціації даних, то мережі становиться складно передбачати наступне можливе значення такого ряду.

Навчання часового ряду у таких випадках проходить в рамках певної похибки та вихідний навчений часовий ряд виходить подібним вхідному, але зміщеному на газомір цієї похибки. Однак, спроба передбачити майбутні значення ряду приводить до графіку виродженої кривої, який не має спільних рис із справжніми даними.

При переходу від стандартних процедур до зацикленого одно крокового прогнозування зі зміщенням часового ряду, час виконання алгоритму стає помітно довшим. Якість навчання нейронної мережі суттєво погіршується через

фактору багаторазового зсуву ряду та перенавчання, проте динаміка зімни передбаченого часового ряду починає проявляти схожі ознаки із фактичними значеннями майбутніх елементів ряду, що свідчить про вдосконалення процесу прогнозування, хоча й за рахунок перенавчання та довшого часу на передбачення.

ВИСНОВКИ

В результаті роботи над дипломною роботою було розроблено реккуретну мережу та метод усунення її перенавчання. Навчання нейронної мережі зроблено за допомогою методу регресії.

Для навчання нейронної мережі використовувався набір цін акцій компанії Apple за п’ять років. Прогнози відрізняються від вірних значень в середньому на 4%. Нейронна мережа не тількі вловила загальну тенденцію акцій Apple до росту, але в один із днів навіть припустила їх падіння, що також вельми цікавий результат.

Було проведено аналіз сучасних нейронних мереж та областей їх застосування. Дослідження показали, що предметних областей чи мала кількість, а нейронних мереж для їх запровадження у якусь область ще більще.

Були визначені програмні мови та середовища для роботи з нейронними мережами, та в процесі роботи над дипломною роботою були обрані самі підходящи для конкретної роботи. Була обрана програмна мова Python, а обране середовище, де була розроблена нейронна мережа це хмарний сервіс від компанії Google – Google Colaboratory.

Проаналізував результати роботи, можно зробити висновок, що поставлені цілі і задачи виконані в повному обсязі.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с.

2. Єжов А. А. Нейрокомп'ютинг та його застосування в економіці та бізнесі / А. А. Єжов, С. А. Шумський. - М.: МІФІ, 1998. - 222 с.

3. Галушкін А. І. Теорія нейронних мереж. Кн.1: Навчальний посібник для вузів. / А. І. Галушкін. // Видавниче підприємство редакції журналу "Радіотехніка". - 2000. - С. 215

4. Мкртчян С. О. Нейрони та нейронні мережі / С. О. Мкртчян. - М.: Енергія, 1970. - 230 с.

5. Deepmind [Електронний ресурс]: Новини високих технологій - Режим доступу: https://hinews.ru/technology/ii-alphago-ot-deepmind-obygral-chempiona-mira-pologicheskoj-igre- go.html

6. Новіков В.А. Організація та навчання штучних нейронних мереж: Експериментальне навч. допомога. / В.А.Новіков, Л.В.Калацкая, В.С.Садов - Мінськ: БДУ, 2003. - 72 с.

7. Хайкін С. Нейронні мережі/С. Хайкін. - М.: Вільямс, 2006. - 1103 с.

8. Степанов В. А. Фондовий ринок та нейромережі [Електронний ресурс] / В. А. Степанов // Світ ПК. - 1998. - Режим доступу до ресурсу: http://www.osp.ru/pcworld/1998/12/159835/.

9. Герасименко Н. А. Нейросетельні технології в аналізі фондового ринку [Електронний ресурс] / 7. Герасименко Н. А.. – 1998. – Режим доступу до ресурсу: http://fakit.ru/main_dsp.php?top_id=1086

10. Міцель А. А. Прогнозування динаміки цін на фондовому ринку / А.

А. Міцель, Є. А. Єфремова. // Вісті Томського політехнічного університету. - 2007. - №8.

11. Андрієнко В. М. Аналіз та моделювання динаміки українського фондового ринку / В. М. Андрієнко, О. Ш. Тулякова. // Науковий журнал "Аспект". - 2011. - №2. - С. 34.

12. Іванов Д. В. Прогнозування фінансових ринків з використанням штучних нейронних мереж [Електронний ресурс] / Д. В. Іванов - Режим доступу до ресурсу: forex-mmcis.ru. / D. Ivanov.

13. Уоссермен Ф. [Текст]/Ф. Уоссермен. Нейрокомп'ютерна техніка: Теорія та практика. - пров. з англ., 1992. – 118 с.

14. Уоссермен Ф. Нейронні мережі. Модифіковані базові індикатори NeuroShell DayTrader. Частина 1. Травень 21, 2012 – 00:25 [Електронний ресурс]. - Режим доступу: http://iworkclub.net/Iskusstvennie NejronnieSeti/uossermen-nejronnie-seti

15. E. Karakoyun and A. Cibikdiken, “Comparison of arima time series model and lstm deep learning algorithm for bitcoin price forecasting,” in The 13th Multidisciplinary Academic Conference in Prague, vol., pp. 171–180, 2018.

16. Mohammad J. Hamayel , Amani Yousef Owda “A Novel Cryptocurrency Price Prediction Model Using GRU, LSTM and bi-LSTM Machine Learning Algorithms”// vol. 2, p. 477–496, 2021.

17. T. Zoumpekas, E. Houstis, and M. Vavalis, “Eth analysis and predictions utilizing deep learning,” Expert Systems with Applications, vol. 162, p. 113866, 2020.

18. Dutta, S. Kumar, and M. Basu, “A gated recurrent unit approach to bitcoin price prediction,” Journal of Risk and Financial Management, vol. 13, p. 23, Feb 2020.

19. S. Corbet, C. Larkin, B. M. Lucey, A. Meegan, and L. Yarovaya, “The impact of macroeconomic news on bitcoin returns,” The European Journal of Finance, vol. 26, no. 14, pp. 1396–1416, 2020.

20. Z. Qiang, “Bitcoin High-Frequency Trend Predictionwith Convolutional

and Recurrent NeuralNetworks,” stanford winter report 70308950, Dtanford, Mar. 2021

21. S. Alonso-Monsalve, A. L. Suárez-Cetrulo, A. Cervantes, and D. Quintana, “Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators,” Expert Systems with Applications, vol. 149, p. 113250, 2020.

22. Singh N. Automatic Speaker Recognition: Current Approaches and Progress in Last Six Decades [Text] / Singh N., Agrawal A., R. A. Khan // Global Journal of Enterprise Information System. – 2017. – V. 9, №3. – P. 38–45.

23. Stylianou Y. Continuous probabilistic transform for voice conversion. Speech and Audio Processing [Text] / Stylianou Y., Cappé O., Moulines E. // Proceedings. IEEE Transactions on Acoustics, Speech, and Signal Processing. – Vol. 6, no. 2 (Mar). – 1998. – P. 131–142.

24. Toda T. Spectral conversion based on maximum likelihood estimation considering global variance of converted parameter [Text] / Toda T., Black A. W., Tokuda K. // Proceedings. (ICASSP’05). IEEE International Conference on Acoustics, Speech, and Signal Processing. – Vol. 1 (Mar). – 2005. – P. 9–12.

25. Kobayashi K. sprocket : Open-Source Voice Conversion Software / Kobayashi K., Toda T. – Proc. Odyssey. –2018. – P. 203–210.

26. Dempster A.P. Maximum Likelihood from Incomplete Data via the EM Algorithm [Text] / Dempster A. P.; Laird N. M.; Rubin D. B. // Journal of the Royal Statistical Society, Series B (Methodological). – Vol. 39 (1). – 1977. – P 1– 38.

27. Godoy E. Voice conversion using dynamic frequency warping with amplitude scaling, for parallel or nonparallel corpora [Text] / Godoy E., Rosec O., Chonavel T. // Proceedings. IEEE Transactions on Acoustics, Speech, and Signal Processing. – Vol. 20, no. 4 (May). – 2012. – P. 1313–1323.

28. Agiomyrgiannakis Y. Voice morphing that improves TTS quality using an optimal dynamic frequency warping-and-weighting transform [Електронний

ресурс]/Y.Agiomyrgiannakis, Z. Roupakia. – Режим доступу: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/44861.pdf

29. Erro D. Voice conversion based on weighted frequency warping [Text] / D. Erro, A. Moreno, A. BonafoLante // Proceedings. IEEE IEEE 56 Transactions on Acoustics, Speech, and Signal Processing. – Vol. 18, no. 5 (July). – 2010. – P. 922–931.

30. Macon M. W. Applications of sinusoidal modeling to speech and audio signal processing [Електронний ресурс] / M. W. Macon, D. D. J. Blumenthal, D. M. A. Clements, D. R. M. Mersereau. – Режим доступу : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.51.2790&rep=rep1&type=pdf

31. Kain A. High resolution voice transformation [Text] : PhD dissertation in Computer Science and Engineering; Oregon Graduate Institute / Alexander Kain. – Portland, OR. – 2001. – 129 p.

32. Ye H. Perceptually Weighted Linear Transformation for Voice Conversion [Text] / Ye H. Young S // Eurospeech. – 2003. – P. 2409–2412.

33. Young S. High quality voice morphing [Електронний ресурс] / S. Young. – Режим доступу: https://www.researchgate.net/publication/4087475_High_quality_voice_morphing

34. Zen H. Review: Statistical parametric speech synthesis [Text] / H. Zen, K. Tokuda, A. W. Black // Speech Communication. – Vol. 51, no. 11 (Nov). – 2009. – P. 1039–1064.

35. Yamagishi J. Analysis of speaker adaptation algorithms for hmmbased speech synthesis and a constrained smaplr adaptation algorithm [Text] / J. Yamagishi, T. Kobayashi, Y. Nakano, K. Ogata, J. Isogai // IEEE Transactions on Audio, Speech, and Language Processing. – Vol. 17, no. 1 (Jan). – 2009. – P. 66–83.

36. Hsu C. Voice Conversion from Non-parallel Corpora Using Variational

Auto-encoder [Електронний ресурс] / C. Hsu, H. Hwang, Y. Wu, Y. Tsao, H. Wang. – Режим доступу : https://arxiv.org/pdf/1610.04019.pdf – 13.10.2016.

37. Nakashika T. Voice conversion in high-order eigen space using deep belief nets [Text] / T. Nakashika, R. Takashima, T. Takiguchi, Y. Ariki // Interspeech, ISCA. – 2013. – P. 369–372. 57

38. Karaali O. Speech synthesis with neural networks [Text] / O. Karaali, G. Corrigan, I. A. Gerson // CoRR. – Vol. cs.NE/9811031. – 1998. – P. 45–50.

39. Zen H. Statistical parametric speech synthesis using deep neural networks [Text] / H. Zen, A. Senior, M. Schuster // Proceedings. ICASSP. IEEE, 2013. – P. 7962–7966.

40. Wu Z. A study of speaker adaptation for DNN-based speech synthesis [Text] / Z. Wu, P. Swietojanski, C. Veaux, S. Renals, S. King // International Speech Communication Association. – 2015; Date of Acceptance: 01/06/2015.

41. Mehta A. A Complete Guide to Types of Neural Networks [Електронний ресурс] /A. Mehta. – Режим доступу : https://www.digitalvidya.com/blog/types-of-neural-networks/ - 25.01.2019 p.

42. Arık S. Ö. Neural Voice Cloning with a Few Samples [Електронний ресурс] / S. Ö. Arık. J. Chen, K. Peng, W. Ping, Y. Zhou. – Режим доступу : https://papers.nips.cc/paper/8206-neural-voice-cloning-with-a-few-samples.pdf - 14.02.2018 p.

43. Cooley J. W. An algorithm for the machine calculation of complex Fourier series [Text] / Cooley J. W., Tukey J. W. Math // Comput. – 19 (90). – 1965. – P. 297–301.

44. Bogert B. P. The Quefrency Alanysis [sic] of Time Series for Echoes: Cepstrum, Pseudo Autocovariance, Cross-Cepstrum and Saphe Cracking [Text] / B. P. Bogert, M. J. R. Healy, J. W. Tukey // Proceedings of the Symposium on Time Series Analysis (M. Rosenblatt, Ed). – Chapter 15. – New York: Wiley, 1963. – P. 209–243.

45. Davis S. Comparison of Parametric Representations for Monosyllabic

Word Recognition in Continuously Spoken Sentences [Text] / Davis S., Mermelstein P. // Proceedings. IEEE Transactions on Acoustics, Speech, and Signal Processing. – Vol. 28, No. 4. – 1980. – P. 357– 366.

46. Griffin D. Signal estimation from modified short-time fourier transform [Text] / D. Griffin, J. Lim // Proceedings. IEEE Transactions on Acoustics, Speech, and Signal Processing. – 1984. – P. 236–243.

47. Sutskever I. Sequence to Sequence Learning with Neural Networks [Електронний ресурс] / I. Sutskever, O. Vinyals, Q. V. Le. – Режим доступу : https://arxiv.org/pdf/1409.3215.pdf -14.12.2014 p.

48. Kingma D. P. Auto-Encoding Variational Bayes [Електронний ресурс] / D. P. Kingma, M. Welling. – Режим доступу : https://arxiv.org/abs/1312.6114 - 01. 05. 2014 p.

49. Vaswani A. Attention Is All You Need [Електронний ресурс] / N. Shazeer, N. Parmar, J. Uszkoreit, L Jones, A. N. Gomez, Ł. Kaiser. – Режим доступу : https://arxiv.org/abs/1706.03762 - 06.12.2017 p.

50. Hahnloser R. Permitted and Forbidden Sets in Symmetric ThresholdLinear Networks [Text] / R. Hahnloser, H. S. Seung // NIPS Conference. – 2001. – P. 217–223.

51. The LJ Speech Dataset [Електронний ресурс]. – Режим доступу : https://keithito.com/LJ-Speech-Dataset/ - 08.07.2017 p.

52. CSTR VCTK Corpus. English Multi-speaker Corpus for CSTR Voice Cloning Toolkit [Електронний ресурс]. – Режим доступу : https://homepages.inf.ed.ac.uk/jyamagis/page3/page58/page58.html

53. Kingma D. P Adam: A Method for Stochastic Optimization [Text] / D. P Kingma, J. Ba // Proceedings of the 3rd International Conference on Learning Representations (ICLR). – arXiv preprint arXiv:1412.6980, 2014. P. 1–15.

54. Nagrani A. VoxCeleb: a large-scale speaker identification dataset [Text] / A. Nagrani, J. S. Chung, A. Zisserman // Proceedings of INTERSPEECH, 2017. – P.

2616–2620.

55. Bernd Fritzke. A Growing Neural Gas Network Learns Topologies. [Електронний ресурс]. – Режим доступу: https://papers.nips.cc/paper/893-agrowing-neural-gas-network-learns topologies.pdf.

56. Jinwon An, Sungzoon Cho. Variational Autoencoder based Anomaly Detection using Reconstruction Probability. 2015. P. 18

57. Hawkins, Simon; He, Hongxing; Williams, Graham; Baxter, Rohan (2002). "Outlier Detection Using Replicator Neural Networks". Data Warehousing and Knowledge Discovery. Lecture Notes in Computer Science. pp. 170–180.

58. Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly Detection: A Survey. Department of Computer Science and Engineering University of Minnesota 4-192 EECS Building 200 Union Street SE Minneapolis, MN 55455-0159 USA. 2007. P. 7

ДОДАТКИ

Діаграма варіантів використання

ДОДАТОК А

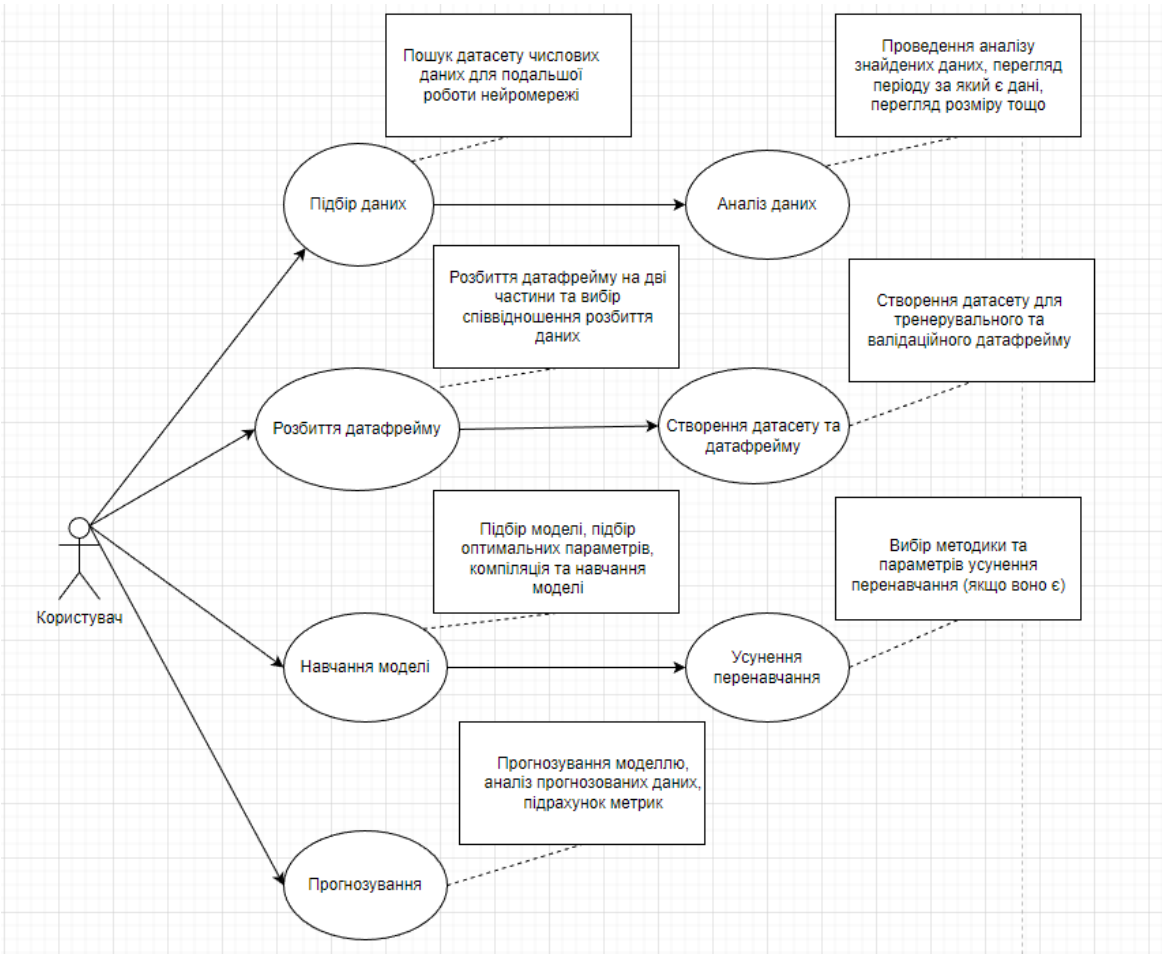


Рис А.1 - Діаграма варіантів використання

ДОДАТОК Б

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Перший проректор
Українського державного
університету науки і технології
Анатолій РАДКЕВИЧ

ПРОГНОЗУВАННЯ ДИНАМІКИ ЦІН НА УКРАЇНСЬКІЙ МІЖБАНКІВСЬКІЙ БІРЖІ ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖЕВИХ ТЕХНОЛОГІЙ

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01260–01–ЛЗ

Завідувач кафедри КІТ

_____Вадим ГОРЯЧКІН

Керівник розробки

_____ Світлана ВОЛКОВА

Виконавець

_____ Денис ГІЛЯВСЬКИЙ

Нормоконтролер

_____Світлана ВОЛКОВА

2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

1116130.01260–01

ПРОГНОЗУВАННЯ ДИНАМІКИ ЦІН НА УКРАЇНСЬКІЙ МІЖБАНКІВСЬКІЙ БІРЖІ ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖЕВИХ ТЕХНОЛОГІЙ

Технічне завдання

Аркушів 24

2022

АНОТАЦІЯ

Документ 1116130.01260–01 «Прогнозування динаміки цін на українській міжбанківській біржі за допомогою нейромережевих технологій» входить до складу програмної документації на програму, яка виконує побудову нейронної мережі.

Об’єкт дослідження – нейронні мережі за визначеними критеріями.

Для вирішення задачі, використовується метод регресії, який передбачає те, що минулі значення тимчасового ряду будуть використовуватися для передбачення наступного значення.

Програма являється додатком, з використанням мови Python. Конфігурація комп'ютера стандартна. Комплекс функціонує як в середовищі MS WINDOWS, так і на сімействі систем Unix.

В даному документі вказано призначення і область застосування програмного продукту, технічні, техніко-економічні показники, що пред'являються до програми, терміни розробки проекту.

ЗМІСТ

ВВЕДЕННЯ

Безліч сфер діяльності людей, такі як економіка, постійно вимагають удосконалення. З часом швидко збільшується розмір інформації та швидкість її зміни. Обробка і управління таким обсягом інформації розумом людини неефективне, і використання стандартних обчислень є дуже затратним і трудомістким заняттям. Тому розумно замість людських ресурсів використати сучасні технології для обробки такої інформації. Сьогодні для аналізу та обробки даних часто застосовують різні інтелектуальні методи, наприклад, нейронні мережі. Нейронна мережа (вона ж штучна нейронна мережа, ШНС) — математична модель, і навіть її програмне чи апаратне втілення, побудована за принципом організації та функціонування біологічних нейронних мереж — мереж нервових клітин живого організму. Це поняття виникло щодо процесів, які у мозку, і за спробі змоделювати ці процеси. Після розробки алгоритмів навчання одержувані моделі стали використовувати в практичних цілях: завдання прогнозування, для розпізнавання образів, завдання управління і ін.

Дослідження у цій сфері нейронних мереж проводили різні вчені: Хайкін С., Руденко Є. Г., Бодянський Є. В., Осовський С. та інші. Це дуже цікавий продукт, який надає розробникам багато можливостей для конкретних результатів.

Розробники алгоритмів машинного навчання використовують різні підходи у створенні прогнозних інструментів. Найбільш популярні з них – рекурентна нейронна мережа та модель з довгою короткостроковою пам'яттю (LSTM).

LSTM-модель – це різновид рекурентної нейронної мережі, яка може запам'ятовувати довгострокові залежності. Подібно до того, як ми використовуємо попередній досвід для прогнозування майбутніх подій, нейромережа здатна запам'ятовувати інформацію протягом тривалих періодів і швидко знаходити закономірності.

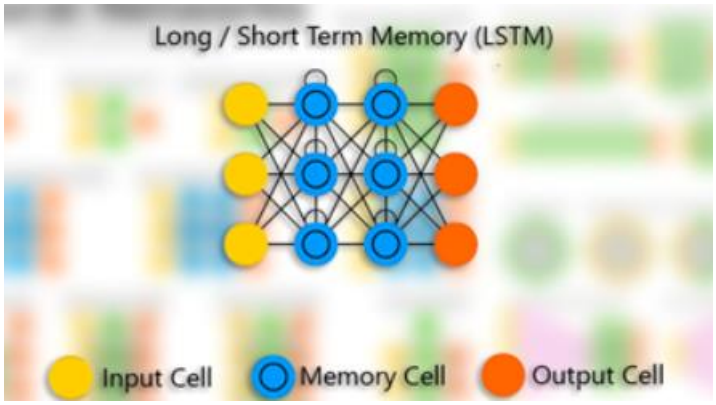


Рис. 1.1 – Принцип роботи моделі Long-Short Term Memory (LSTM).

Система нейронної мережі – програмний продукт, для економічного прогнозування цін на фондовій біржі, з використанням методу регресії.

Необхідність розробки програмного застосунку зумовлена шаленим попитом у даній сфері.

Область застосування програмного застосунку – фондова біржа. Основна аудиторія – будь яка компанія, якій потрібно прогнозувати ціни на акції.

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Основою для розробки є наказ ректора Українського державного університету науки і технології Радкевич А.В. «Про затвердження тем та призначення керівників дипломних проектів» №173 ст. від 11.02.2022 р.

Тема дипломної роботи: Прогнозування динаміки цін на українській міжбанківській біржі за допомогою нейромережевих технологій.

Керівник дипломної роботи – Волкова С. А.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

2.1 Функціональне призначення:

Для роботи з чисельними даними, за визначеним набором критерій, для їх аналізу та прогнозування.

Програма, що розробляється, повинна надавати зручний призначений для користувача інтерфейс і виконувати наступні функції:

1) імпорт даних;
2) обробка даних;
3) нормалізація даних;
4) налаштування параметрів нейронної мережі;
5) навчання нейронної мережі;
6) візуалізація даних;
7) експеримент;
8) збереження навченої нейронної мережі;
9) експорт даних.

2.2 Експлуатаційне призначення:

Основним експлуатаційним напрямом програми є вдосконалення якості створюваного продукту, спрощення взаємодії користувача та розробника та установка точних цілей для розробника стосовно виконуваних задач, які формуються на основі потреб користувача.

3 ВИМОГИ ДО ПРОГРАМИ

3.1 Вимоги до функціональних характеристик

Програма повинна виконувати наступні функції:

1) надавати можливість імпорту, обробки та нормалізації даних;
2) надавати можливість налаштування параметрів та навчання нейронної мережі;
3) надавати можливість збереження навченої нейронної мережі та експорту даних.

Введення і формування даних:

Введення початкових даних здійснюється в діалоговому режимі, з повідомленням системи, якщо дані були введені некоректно.

Вимоги до вхідних даних:

До початкових даних належить такий набір таких даних:

- статистичні дані;
- параметри нормалізації;
- кількість шарів та нейронів;
- параметри навчання нейронної мережі;
- умови завершення навчання нейронної мережі;
- параметри візуалізації даних;
- дані для експерименту.

Очікувані результати :

- створено систему для передбачення курсу акцій методом регресії;
- в результаті роботи системи нейронна мережа повертає у відсотках дельту зміни курсу;
- відповідь нейронної мережі вважається правильною якщо вибрано правильний напрямок, і передбачене значення відхиляється від правильного не більше ніж на 5% в будь-який бік;

– точність передбачення залежить від кількості повідомлень про акції;

– під час навчання використовувалося приблизно 500 ітерацій (епох), що є малою кількістю такої складної логіки. В результаті чого отримано такі дані про точність передбачення курсу акцій.

3.2 Вимоги до надійності

– передбачити контроль інформації, що вводиться;

– передбачити блокування некоректних дій користувача під час роботи із системою;

– забезпечити цілісність інформації, що зберігається;

– вимоги до складу та параметрів технічних засобів;

– система має працювати на сумісних комп'ютерах.

3.3 Умови експлуатації

Даний програмний продукт може бути застосований в будь-яких установах і приміщеннях, призначених для роботи з ЕОМ і відповідних вимогам по пожежо та електро-безпеки.

Умови для експлуатації програмного продукту повинні бути схожими з умовами на обчислювальному центрі (температура повітря 21-25°С, відносна вогкість 60 ± 15%, атмосферний тиск 630-800 мм.рт.ст.).

Для обслуговування програмного продукту достатньо одного програміста. Роботу з програмою може виконувати людина яка має досвід роботи з персональним комп’ютером та ознайомлена з керівництвом користувача.

3.4 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація комп'ютера для забезпечення працездатності програмного продукту:

–тип процесора - Intel core i3 3220 і вище;

–частота процесора - 3000Mhz і більше;

–обсяг оперативного запам'ятовуючого пристроя (залежить кількості шарів і нейронів у мережі) - 4096 Мб і більше.

–дисковод 3.5", 1.4 Мбайт;

–CD-ROM;

–дисплей;

–стандартна клавіатура (104/105 клавіші);

–маніпулятор "миша".

3.5 Вимоги до інформаційної і програмної сумісності

Для функціонування програмного продукту необхідна наявність встановлених:

1) операційної системи Windows 8/10/11 системи UNIX;
2) будь-який браузер та інтернет з'єднання.

3.6 Вимоги до маркування і упаковки

Приклад маркування приведений на рис. 4.1

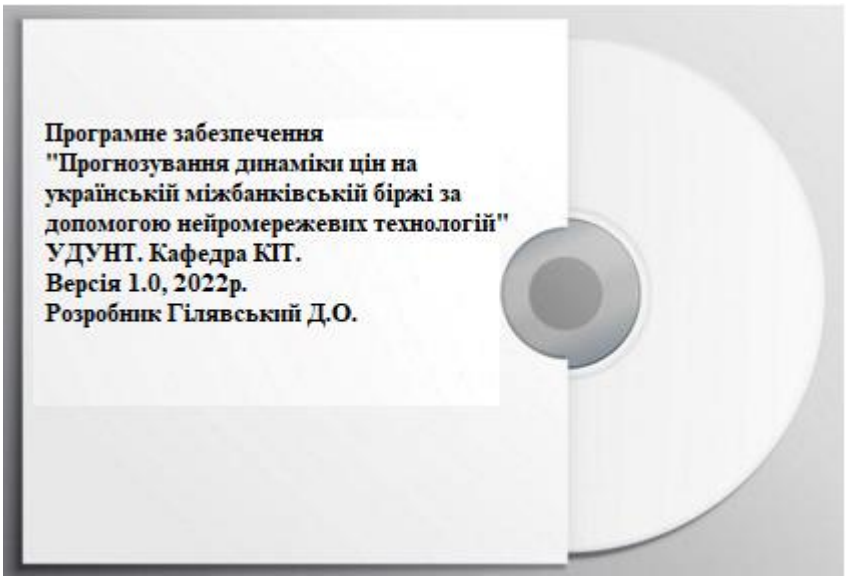


Рис. 3.1 – Приклад маркування

3.7 Вимоги до транспортування і зберігання

Копія програмного продукту повинна зберігатися на жорсткому диску. Вимоги до зберігання повинні відповідати вимогам по експлуатації гнучких/жорстких дисків.

Транспортування програмного продукту може здійснюватися за допомогою CD-дисків. Транспортування повинно забезпечувати збереження програмного продукту, його цілісність та запобігання несанкціонованого доступу до нього. Транспортування фізичної упаковки програмного продукту повинно проводитись довіреною особою та здійснюватися комплектами в упаковці з термоплівки та піни, яка захищає носій від різного виду пошкоджень. Місце зберігання програмного продукту повинне бути сухим з відсутністю пилу при температурі 21-25°С і вологості 40-60%, з уникненням впливу вологи та шкідників.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація на дану програму повинна включати:

– специфікація;

– текст програми;

– опис програми;

– керівництво користувача.

Вся документація до програми повинна задовольняти вимогам державного стандарту до оформлення програмних документів ГОСТ 19.101-77 «Єдина система програмної документації. Види програм та програмних документів».

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Стадії і етапи розробки програмного продукту представлені в табл. 6.1.

Таблиця 6.1 - Стадії і етапи розробки

Стадії розробки	Етапи робіт	Зміст робіт	Терміни
1. Технічне завдання	Обґрунтування необхідності розробки програми	Постановка задачі. Збір висхідних матеріалів. Визначення структури вхідних і вихідних даних. Визначення вимог до технічних засобів. Вибір мови програмування.	25.02.2022 - 03.03.2022
	Розробка і затвердження технічного завдання	Визначення вимог до програми. Розробка техніко-економічного обгрунтовування розробки програми. Визначення стадій, етапів, термінів розробки програми і документації на неї. Узгодження і затвердження технічного завдання.	06.03.2022 12.03.2022 13.03.2022 26.02.2022
2. Робочий проект	Розробка проекту і програми	Розробка структур даних. Розробка специфікацій. Розробка призначеного для користувача інтерфейсу. Розробка програми. Тестування і відлагодження програми.	20.03.2022 20.03.2022 24.03.2022 24.04.2022 29.04.2022
		Розробка, узгодження і затвердження програмних документів.	25.05.2022 - 08.06.2022
3. Впровадження		Підготовка і передача програми і програмної документації для супроводу і (або) виготовлення	25.06.2022 - 11.07.2022

7 ПОРЯДОК ВИКОНАННЯ І ПРИЙМАННЯ

Контроль здійснюється за допомогою виконання набору тестів з метою виявлення помилок в програмному продукті та його специфікаціях. Контроль за виконанням роботи здійснює головним керівником з розробки. Прийом здійснюється державною комісією.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО
1116130.01260-01

ДОДАТОК «ПРОГНОЗУВАННЯ ДИНАМІКИ ЦІН НА УКРАЇНСЬКІЙ МІЖБАНКІВСЬКІЙ БІРЖІ ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖЕВИХ ТЕХНОЛОГІЙ»

Специфікація

Аркушів 2

2022

Позначення	Найменування	Примітка
	Документація	
1116130.01260-01-ЛЗ	Лист затвердження	
1116130.01260-01 12 01-ЛЗ	Лист затвердження	
1116130.01260-01 12 01	Текст програми	
1116130.01260-01 13 01-ЛЗ	Лист затвердження	
1116130.01260-01 13 01	Опис програми	
1116130.01260-01 І3 01-ЛЗ	Лист затвердження	
1116130.01260-01 І3 01	Керівництво користувача	

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО
1116130.01260-01 12 01-ЛЗ

ПРОГНОЗУВАННЯ ДИНАМІКИ ЦІН НА УКРАЇНСЬКІЙ МІЖБАНКІВСЬКІЙ БІРЖІ ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖЕВИХ ТЕХНОЛОГІЙ

Текст програми

1116130.01260-01 12 01

Аркушів 10

2022

АНОТАЦІЯ

Документ 1116130.01260-01 12 01 ««Прогнозування динаміки цін на українській міжбанківській біржі за допомогою нейромережевих технологій». Текст програми» входить до складу програмної документації додатку, який надає можливість прогнозування цін на акції за допомогою нейромережі.

Документ містить текст програми. Програма реалізована на мові Python в хмарному сервісі Google Colaboratory.

ЗМІСТ

1116130.01260-01 12 01

1 СТРУКТУРА ПРОГРАМИ

Завдяки хмарному сервісу Google Colaboratory, програма має блокову структуру. Виглядає це так: пишеться блок коду та відразу виконується. Тому структура програми це блоки коду та результати цих блоків (Рис. 1.1).

```
[ ]  # дані за останні 6 років
     df_6_yr = df[df["Date"] > df["Date"].max() - timedelta(days=365 * 6)]

     # перегляд нового розміру датафрейму
     df_6_yr.shape

     (1509, 7)

[ ]  # перегляд періоду нового датафрейму
     df_6_yr["Date"].min(), df_6_yr["Date"].max()

     (Timestamp('2011-11-14 00:00:00'), Timestamp('2017-11-10 00:00:00'))

[ ]  # 80% даних беремо у тренувальний датафрейм та 20% у валідаційний
     train_size = int(df_6_yr.shape[0] * 0.8)
     train_df = df_6_yr.iloc[:train_size]
     val_df = df_6_yr.iloc[train_size:]

[ ]  # перегляд розмірів тренувального та валідаційного датафреймів
     train_df.shape, val_df.shape

     ((1207, 7), (302, 7))

[ ]  # перегляд мінімальної та максимальної дати в кожному з датафреймів
     train_df["Date"].min(), train_df["Date"].max(), val_df["Date"].min(), val_df["Date"].max()

     (Timestamp('2011-11-14 00:00:00'),
      Timestamp('2016-08-31 00:00:00'),
      Timestamp('2016-09-01 00:00:00'),
      Timestamp('2017-11-10 00:00:00'))
```

Рисунок 1.1 – Структура програмного продукту

Файлова структура проекту відображена на рис. 1.2.

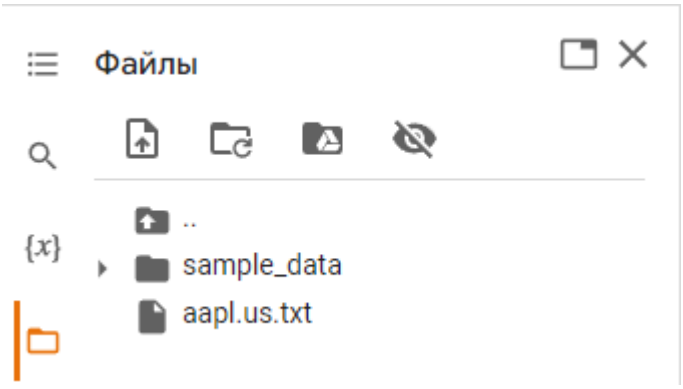


Рисунок 1.2 – Файлова структура проекту програмного продукту

aapl.us.txt – файл з даними на ціни компанії Apple.

Блокова структура проекту відображена на рис. 1.3.

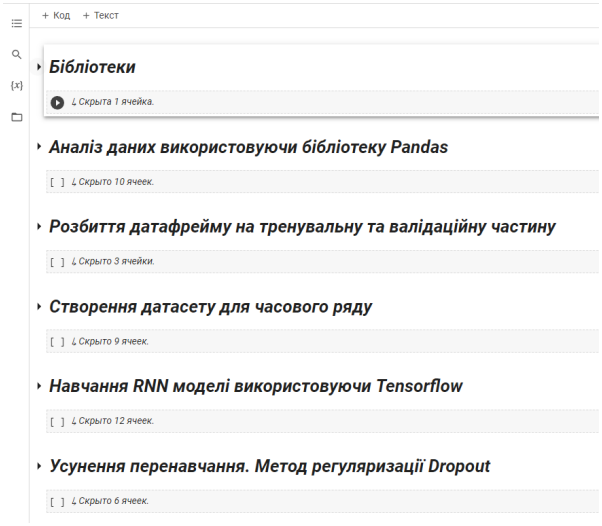


Рисунок 1.3 – Файлова структура проекту програмного продукту

Наведемо опис кожного блоку з структури:

Бібліотеки – знаходяться підключені до програми бібліотеки.

Аналіз даних використовуючи бібліотеку Pandas – відбувається підключення файлу з даними, відображається розмір датафрейму, побудова графіків.

Розбиття датафрейму на тренувальну та валідаційну частину – відбувається підготовка даних для подальшої роботи з ними.

Створення датасету для часового ряду – відбувається створення датасету для часового ряду за допомогою скейлера для нормалізації фічів та з функції створення датасету.

Навчання RNN моделі використовуючи Tensorflow – творення моделі самої нейромережі за допомогою класу sequential, компіляція та навчання моделі.

Усунення перенавчання. Метод регуляризації Dropout – усунення перенавчання нейромережі за допомогою методу регуляризації Dropout

1116130.01260-01 12 01

2 ТЕКСТ ПРОГРАМИ

Блок коду «Бібліотеки»

```
import pandas as pd
from datetime import timedelta
import numpy as np
import tensorflow as tf
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error
from matplotlib import pyplot as plt
from typing import List
```

Блок коду «Аналіз даних використовуючи бібліотеку Pandas»

```
# читання файлу з даними
df = pd.read_csv("aapl.us.txt", parse_dates=["Date"])
# перегляд розміру датафрейму
df.shape
# перші п'ять колонок
df.head(5)
# період, за який є дані
df["Date"].min(), df["Date"].max()
# мінливість ціни за час
df.plot(x="Date", y="Open")
df.plot(x="Date", y="High")
# прорахунок попарної кореляції
df[["Open", "Close", "Low", "High"]].corr()
# дані за останні 6 років
df_6_yr = df[df["Date"] > df["Date"].max() - timedelta(days=365 * 6)]
# перегляд нового розміру датафрейму
df_6_yr.shape
# перегляд періоду нового датафрейму
df_6_yr["Date"].min(), df_6_yr["Date"].max()
```

Блок коду «Розбиття датафрейму на тренувальну та валідаційну частину»

```
# 80% даних беремо у тренувальний датафрейм та 20% у валідаційний
train_size = int(df_6_yr.shape[0] * 0.8)
train_df = df_6_yr.iloc[:train_size]
val_df = df_6_yr.iloc[train_size:]
# перегляд розмірів тренувального та валідаційного датафреймів
train_df.shape, val_df.shape
# перегляд мінімальної та максимальної дати в кожному з датафреймів
train_df["Date"].min(), train_df["Date"].max(), val_df["Date"].min(), val_df["Date"].max()
```

Блок коду «Створення датасету для часового ряду»

```
# scaler - класи бібліотеки sklearn для нормалізації фічів
# для передбачення n+1 елемента буде використовуватися n попередніх як фічі
scaler = StandardScaler()
scaler.fit(train_df[["Low"]])

# на вхід функція приймає датафрейм, кількість елементів тимчасового ряду, кількість елементів в батчі дл
я навчання, скейлер, змішування елементів череш шафл
def make_dataset(
    df,
     window_size,
     batch_size,
     use_scaler=True,
     shuffle=True
     ):
  features = df[["Low"]].iloc[:-window_size]
# нормалізація якщо потрібно
  if use_scaler:
    features = scaler.transform(features)
  data = np.array(features, dtype=np.float32)
  ds = tf.keras.preprocessing.timeseries_dataset_from_array(
      data=data,
      targets=df["Low"].iloc[window_size:],
      sequence_length=window_size,
      sequence_stride=1,
      shuffle=shuffle,
      batch_size=batch_size)
  return ds
# дата сет для прикладу, вхідні дані - тренувальний датасет
example_ds = make_dataset(df=train_df, window_size=3, batch_size=2, use_scaler=False, shuffle=False)
# отримання елементів з датасету по одному за допомогою iterator
example_feature, example_label = next(example_ds.as_numpy_iterator())
# розмір фіч - багатовимірний вектор (розмір батча, довжина послідовності елементів, розмірність кожної
фічі)
example_feature.shape
# розмірність лейблів, так як в батчі 2 елементи то 2 лейбла
example_label.shape
# перші 6 елементів з датафрейму
train_df["Low"].iloc[:6]
# елементи з батча
print(example_feature[0])
print(example_label[0])
print(example_feature[1])
print(example_label[1])
# створення датасету для тренерувального та валідаційного датафрейму
window_size = 10
batch_size = 8
```

train_ds = make_dataset(df=train_df, window_size=window_size, batch_size=batch_size, use_scaler=True, shuffle=True)
val_ds = make_dataset(df=val_df, window_size=window_size, batch_size=batch_size, use_scaler=True, shuffle=True)

Блок коду «Навчання RNN моделі використовуючи Tensorflow»

```
# Sequential - всі шари виконуються послідовно
# LSTM - блок для рекурентної нейромережі
lstm_model = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(32, return_sequences=False),
    tf.keras.layers.Dense(1)
])
# функція компіляції та навчання моделі
def compile_and_fit(model, train_ds, val_ds, num_epochs: int = 20):
  model.compile(
        loss=tf.losses.MeanSquaredError(),
        optimizer=tf.optimizers.Adam(),
        metrics=[tf.metrics.MeanAbsoluteError()]
      )
  history = model.fit(
      train_ds,
      epochs=num_epochs,
      validation_data=val_ds,
      verbose=0
      )
  return history
# навчання моделі
history = compile_and_fit(lstm_model, train_ds, val_ds, num_epochs=100)
# значення метрик для тренувального та валідаційного датасету
plt.plot(history.history['mean_absolute_error'])
plt.plot(history.history['val_mean_absolute_error'])
# судячи з графіків, значення для тренувального датасету менше, ніж значення для валідаційного, вигляда
є як перенавчання
# розрахуємо метрики точно
lstm_model.evaluate(train_ds)
lstm_model.evaluate(val_ds)
# завдяки розрахункам видно перенавчання
# спробуємо зробити більше епох
lstm_model = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(32, return_sequences=False),
    tf.keras.layers.Dense(1)
])
history = compile_and_fit(lstm_model, train_ds, val_ds, num_epochs=500)
lstm_model.evaluate(train_ds)
# значення для тренувального датасету зменшилося а для валідаційного датасету збільшилося, в наявності
перенавчання
lstm_model.evaluate(val_ds)
plt.plot(history.history['mean_absolute_error'])
plt.plot(history.history['val_mean_absolute_error'])
```

Блок коду «Усунення перенавчання. Метод регуляризації Dropout»

```
# додаємо ще один шар, шар dropout
lstm_model = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(32, return_sequences=False),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(1)
])

history =  compile_and_fit(lstm_model, train_ds, val_ds, num_epochs=1000)
lstm_model.evaluate(train_ds)
# видно що значно покращилася метрика для валідаційного датасету
lstm_model.evaluate(val_ds)
plt.plot(history.history['mean_absolute_error'])
plt.plot(history.history['val_mean_absolute_error'])
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО
1116130.01260-01 13 01-ЛЗ

ПРОГНОЗУВАННЯ ДИНАМІКИ ЦІН НА УКРАЇНСЬКІЙ МІЖБАНКІВСЬКІЙ БІРЖІ ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖЕВИХ ТЕХНОЛОГІЙ

Опис програми

1116130.01260-01 13 01

Аркушів 17

2022

АНОТАЦІЯ

Документ· 1116130.01260-01· 13· 01· ««Прогнозування· динаміки· цін· на· українській· міжбанківській· біржі· за· допомогою· нейромережевих· технологій».· Опис·програми»·входить·до·складу·програмної·документації·на·програму,·яка· виконує·побудову·нейронної·мережі.¶

У· даному· документі· представлений· опис· програми· системи:· функціональне· призначення,· опис· логічної· структури,· використані· технічні· засоби,· виклик· і· завантаження,· вхідні· і· вихідні· дані,· опис· інтерфейсу· користувача,·порядок·роботи·з·програмою.·Програма·реалізована·на·мові·Python.· Для· роботи· з· додатком· потрібен· комп'ютер· який· відповідає· мінімальним· системним·вимогам·до·програми.¶

.

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

Програмний додаток «Прогнозування динаміки цін на українській міжбанківській біржі за допомогою нейромережевих технологій» являє собою прогнозування цін на акції за допомогою нейромережі.

Для коректного функціонування програми необхідно щоб на персональному комп’ютері було встановлено операційну систему Windows 10 або вище, та необхідно щоб був встановлений будь-який браузер.

Програма реалізована на мові програмування Python.

2 ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Метою програмного додатку є прогнозування цін на акції за допомогою нейромережі та її програмною реалізацією на мові програмування Python.

Функціональні вимоги до додатку:

- імпорт даних;
- обробка даних;
- нормалізація даних;
- настроювання параметрів нейронної мережі;
- навчання нейронної мережі;
- візуалізація даних;
- експеримент;
- збереження навченої нейронної мережі;
- експорт даних.

Відповідно до функціональних вимог додатку було визначено наступні вимоги до даних програми:

- програмний код має бути на мові програмування Python;
- нейромережа має бути розроблена та протестована за допомогою хмарного сервіса від Google – Google Collaboratory;
- також необхідне середовище для розробки, для експериментів із нейронними мережами прийнято використовувати Jupyter Notebook.

3 ОПИС ЛОГІЧНОЇ СТРУКТУРИ

3.1 Алгоритм та методи програми

Програмний додаток складається з трьох логічних блоків:

– обробка та аналіз поданого датасету до нейромережі;

– розбиття датасету та навчання нейромережі;

– проведення прогнозування за допомогою тренувального та валідаційного датасетів.

Навчання нейромережі складається з наступних етапів:

– розбиття наявних даних на тренувальну та валідаційну частину, за принципом 80% та 20% відповідно;

– створення датасету для часового ряду;

– створюється датасет для прикладу;

– створення датасета для прикладу створюється вже справжній датасет для тренувального та для валідаційного датафрейму, за яким і відбуватиметься навчання моделі;

– створення моделі самої нейромережі за допомогою класу sequential;

– відбувається безпосередньо навчання створеної моделі нейронної мережі.

3.2 Структура програми з описом функцій складових частин і зв’язки між ними

На рис. 3.1 відображено поле з блоками коду.

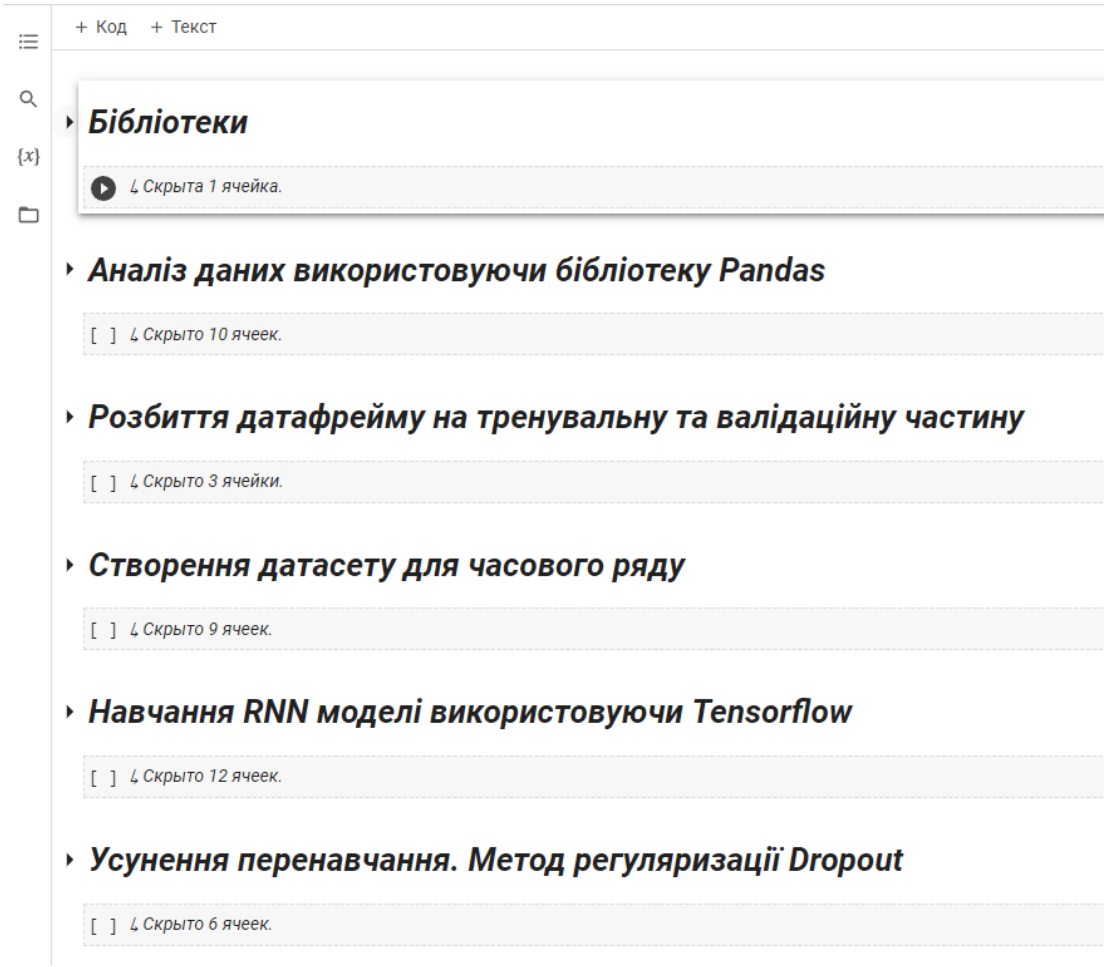


Рисунок 3.1 – Поле з блоками коду

Нижче представлений опис представлених програмних модулів. Бібліотеки – знаходяться підключені до програми бібліотеки.

Аналіз даних використовуючи бібліотеку Pandas – відбувається підключення файлу з даними, відображається розмір датафрейму, побудова графіків.

Розбиття датафрейму на тренувальну та валідаційну частину – відбувається підготовка даних для подальшої роботи з ними.

Створення датасету для часового ряду – відбувається створення датасету для часового ряду за допомогою скейлера для нормалізації фічів та з функції створення датасету.

Навчання RNN моделі використовуючи Tensorflow – творення моделі

самої нейромережі за допомогою класу sequential, компіляція та навчання моделі.

Усунення перенавчання. Метод регуляризації Dropout – усунення перенавчання нейромережі за допомогою методу регуляризації Dropout.

3.3 Зв’язки програми з іншими програмами

Для коректної роботи програмного додатку необхідні такі програми:

- операційна система Windows 10 або більше;
- framework .NET 5.0;

Google Chrome або інший аналогічний браузер.

4 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Для коректного функціонування програмного забезпечення достатньо матиробочий персональний комп’ютер, що має наступні технічні характеристики:

– тип процесора - Intel core i3 3220 і вище;

– частота процесора - 3000Mhz і від;

– обсяг оперативного запам'ятовуючого пристроя (залежить кількості шарів і нейронів у мережі) - 4096 Мб і більше.

– дисковод 3.5", 1.4 Мбайт;

– CD-ROM;

– дисплей;

– стандартна клавіатура (104/105 клавіші);

– маніпулятор "миша".

5 ВИКЛИК ТА ЗАВАНТАЖЕННЯ

Програма викликається шляхом відкриття файлу Neural.ipynb за допомогою хмарного сервісу Google Collaboratory. Після першого відкриття файлу, його копія автоматично зберігається в Google Drive.

Сам файл Neural.ipynb, який містить програмний код може зберігатися у будь-якому місці зручному користувачеві

6 ВХІДНІ ДАНІ

Вхідні дані програмного продукту:

- дані про ціни на акції;
- період для аналізу;
- різноманітних кореляційні показники взаємодії цінних паперів;
- неекономічні складові, що впливають зміну ціни;
- інша інформація, необхідна побудови моделі.

7 ВИХІДНІ ДАНІ

Вихідні дані програмного продукту:

- дані які доступні для перегляду звичайному користувачеві;
- графіки зміни цін на акції з часом;
- функції аналізу даних;
- результат навчання нейронної мережі.

8 ОПИС ІНТЕРФЕЙСУ КОРИСТУВАЧА

8.1 Опис станів програми

Стани програми наведено у табл. 8.1.

Таблиця 8.1 – Стани програми

№ стану	Назва стану	Опис стану	Рекомендовані дії
1	Запуск додатку	Додаток запускається.	Очікування клієнтського вікнадодатку
2	Завантаження файлуз програмним кодомна мові Python	Вибір та завантаження файлуз формату .ipynb	Вибрати необхіднийфайл з розширення .ipynb
3	Введення програмного коду на мові Python	Введення програмного кодуна мові Python за допомогою вбудованого текстового редактору	Ввести програмнийкод на мові Python
4	Завантаження нового файлу з даними	Завантаження нового файлу з даними про ціни на акції біржового ринку	Вибрати вкладку файли та натиснути кнопку завантаження. Після чого підключити новий файл в коді

8.2 Опис переходів між станами програми

Схема переходів програми наведена на рис. 8.1, де цифри є номером стану програми (табл. 8.1). Вийти з програми можливо під час усіх станів крім першого стану.

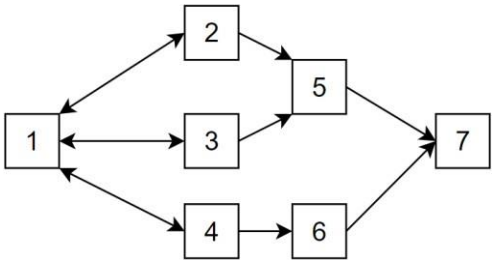


Рисунок 8.1 – Схема переходів між станами програми

8.3 Опис керування діалогом

Керування діалогом відбувається за допомогою вбудованих функцій керування Jupiter Notebook.

8.4 Форматування екрана

Головне вікно програми можна форматувати, а саме змінювати його розмір.

Інші засоби форматування не передбачені

9 ПОРЯДОК РОБОТИ З ПРОГРАМОЮ

Порядок роботи з програмою передбачає наступну послідовність операцій:

– введення або завантаження програмного коду програми на мові Python;

– завантаження датасету у форматі txt або csv;

– виконання програмного коду;

– навчання нейромережі;

– прогнозування.

10 ПОВІДОМЛЕННЯ

У табл. 10.1 наведені повідомлення користувачу, що можуть з’явитися під часроботи з програмою.

Таблиця 10.1 – Повідомлення користувачу

Текст повідомлення	Опис ситуації	Рекомендовані дії
Помилка відкриття файлу з програмнимкодом	Файл з програмним кодом не було відкрито	Обрати коректний файл зпрограмним кодом у форматі .ipynb
Помилка відкриття файлу з даними	Файл з даними небуло відкрито	Обрати коректний файл з даними форматі txt або csv
Програмний код невведено!	Розпізнавання програмного коду не виконано оскільки вхідні дані не коректні	Ввести або відрити файл зпрограмним кодом на мовіPython
Програмний код виконано!	Програмний код виконано успішно	Натиснути кнопку «ОК»

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО
1116130.01260-01 ІЗ 01-ЛЗ

ПРОГНОЗУВАННЯ ДИНАМІКИ ЦІН НА УКРАЇНСЬКІЙ МІЖБАНКІВСЬКІЙ БІРЖІ ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖЕВИХ ТЕХНОЛОГІЙ

Керівництво користувача

1116130.01260-01 ІЗ 01

Аркушів 13

2022

АНОТАЦІЯ

Документ 1116130.01260-01 І3 01 «Прогнозування динаміки цін на українській міжбанківській біржі за допомогою нейромережевих технологій». Керівництво користувача. Керівництво з використання прогнозування динаміки цін на українській міжбанківській біржі за допомогою нейромережевих технологій» входить до складу програмної документації додатку, який надає можливість прогнозування цін на акції за допомогою нейромережі.

У даному документі представлено призначення та умови застосування програмного продукту, інструкції з підготовки до роботи, а також опис основних операцій та аварійних ситуацій програмного додатку.

ЗМІСТ

ВСТУП

Програмний додаток «Прогнозування динаміки цін на українській міжбанківській біржі за допомогою нейромережевих технологій» використовується компаніями, що використовують нейромережі, для аналізу та передбачення коливання цін на акції.

Програмний додаток надає можливість аналізу даних, використання даних для навчання, прогнозування цін.

Для роботи з додатком потрібен комп’ютер який відповідає мінімальним системним вимогам до програми.

Для коректної роботи з програмним додатком необхідно ознайомитися з описом програми та керівництвом викладача.

1 ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

1.1 Функціональне призначення програми

Основною метою програмного додатку є прогнозування цін на акції за допомогою нейромережі та її програмною реалізацією на мові програмування Python.

До функціональних можливостей програмного додатку входить:

- імпорт даних;
- обробка даних;
- нормалізація даних;
- настроювання параметрів нейронної мережі;
- навчання нейронної мережі;
- візуалізація даних;
- експеримент;
- збереження навченої нейронної мережі;
- експорт даних.

1.2 Вимоги до складу і параметрів технічних засобів

Для коректного функціонування програмного забезпечення достатньо матиробочий персональний комп’ютер, що має наступні технічні характеристики:

- тип процесора - Intel core i3 3220 і вище;
- частота процесора - 3000Mhz і від;
- обсяг оперативного запам'ятовуючого пристроя (залежить кількості шарів і нейронів у мережі) - 4096 Мб і більше.
- дисковод 3.5", 1.4 Мбайт;
- CD-ROM;
- дисплей;
- стандартна клавіатура (104/105 клавіші);
- маніпулятор "миша".

1.3 Вимоги до інформаційної і програмної сумісності

Для коректного функціонування програми необхідно щоб на персональному комп’ютері було встановлено операційну систему Windows 10 або вище, Framework .NET 5 та Google Chrome або інший аналогічний браузер.

2 ПІДГОТОВКА ДО РОБОТИ

Для запуску додатку потрібно відкрити файл Neural.ipynb через сервіс Google Colaboratory. Для цього потрібно відкрити Google Colaboratory, вибрати вкладку Файл – Завантажити блокнот, та вибрати потрібний файл.

Також, якщо файл присутній на сервісі Google Drive, то при відкритті Google Colaboratory сервіс запропонує відкрити вже наявний файл.

3 ОПИС ОПЕРАЦІЙ

Після старту роботи з програмним додатком можливі наступні операції:

1. Завантаження програмного коду на мові програмування Python: завантажте файл з розширенням .ipynb за допомогою пункту головного меню «Файл–>Завантажити блокнот–>Файл»з програмним кодом;

2. Введення програмного коду на мові Python: введіть програмний код за допомогою вбудованого текстового редактору;

3. Відкриття вже існуючого програмного коду на мові програмування Python: відкрийте файл з розширенням .ipynb за допомогою пункту головного меню «Файл–>Відкрити блокнот–>Файл» з програмним кодом або за допомогою комбінації клавіш Ctrl+O;

4. Виконати весь код: для цього оберіть пункт головного меню «Середовище виконання–>Виконати все» або натисніть комбінацію клавіш Ctrl+F9;

Перегляд довідки: оберіть пункт головного меню «Довідка–>Перегляд довідки» або натисніть клавішу F1.

4 АВАРІЙНІ СИТУАЦІЇ

У разі виявлення помилок у даних програми, необхідно закрити програмнийдодаток та перевірити коректність вхідних даних.

У разі виявлення інших аварійних ситуацій, необхідно закрити програмний додаток та зв’язатися з персоналом, що супроводжує даний додаток.

5 РЕКОМЕНДАЦІЇ ЩОДО ЗАСВОЄННЯ

Інтерфейс програмного додатку зображено на рис. 5.1.

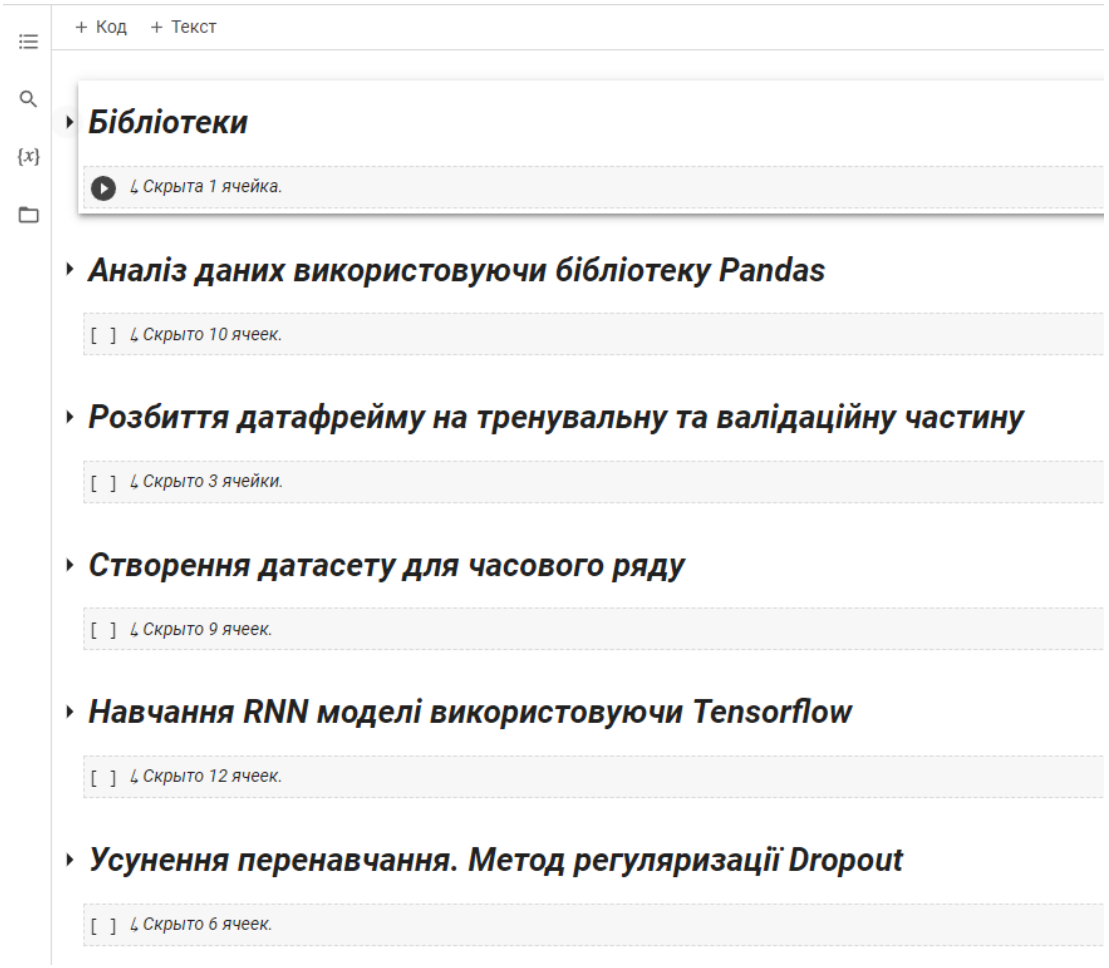


Рисунок 5.1 – Інтерфейс програмного додатку

Для завантаження файлів з програмним кодом на мові Python та оберіть пункт головного меню Файл–>Завантажити блокнот (рис. 5.2).

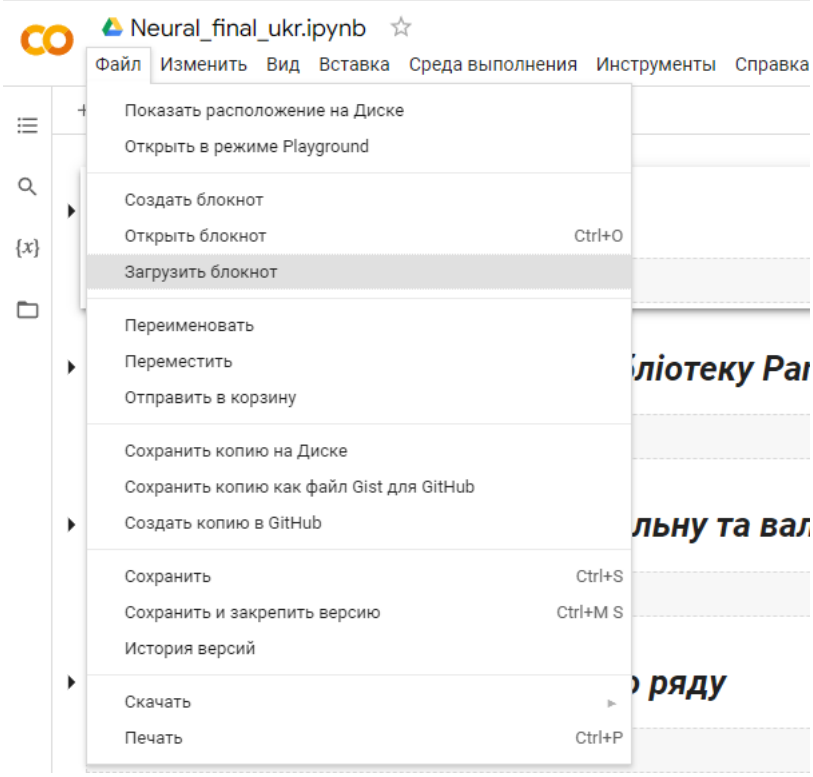


Рисунок 5.2 – Пункт головного меню Файл–> Завантажити блокнот

Щоб відредагувати вхідні дані скористайтеся вбудованим текстовим редакторомдодатку (рис. 5.3).

Рисунок 5.3 – Вбудований текстовий редактор додатку з вхідними даними

Для виконання всього коду або лише вибраної ділянки, натисніть пункт головного меню «Середовище виконання» (рис. 5.4).

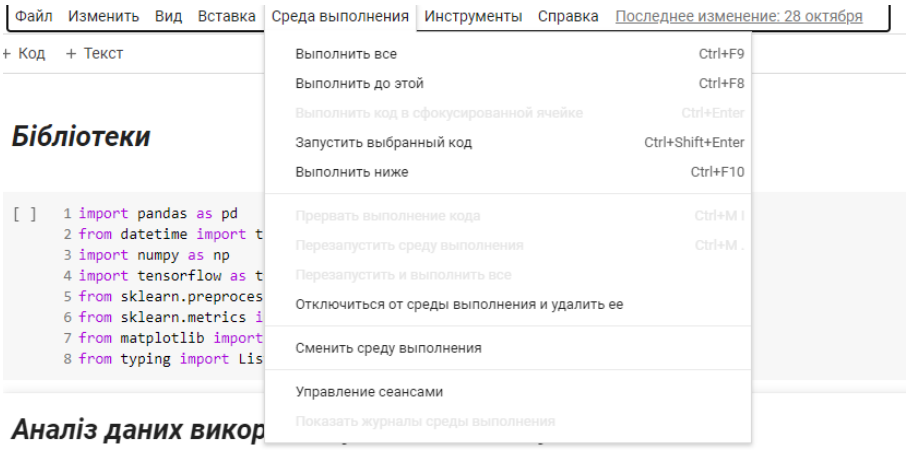


Рисунок 5.4 – Пункт головного меню «Середовище виконання»

Для того щоб нейронна мережа почала своє функціонування, потрібно завантажити файл з даними у форматі txt або csv, для цього потрібно вибрати вкладку «Файли» та натиснути кнопку завантаження. Після чого підключити

новий файл в коді (рис. 5.5).

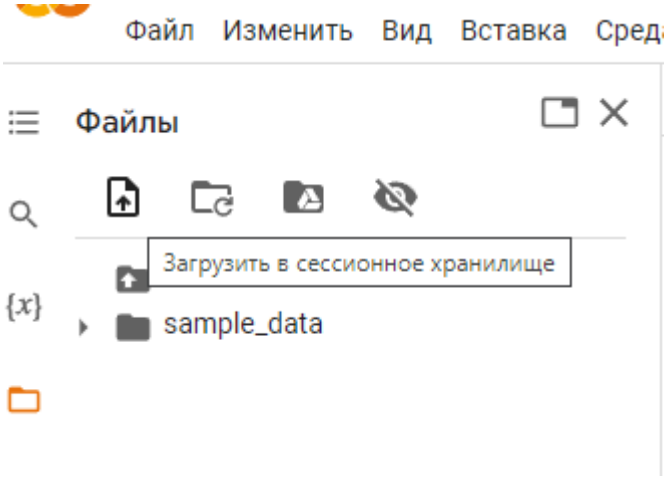


Рисунок 5.5 – Вкладка «Файли»

CERTIFICATE

is awarded to

Gilyavski Denis

for being an active participant in
VII International Scientific and Practical Conference

“MODERN RESEARCH IN WORLD SCIENCE”

24 Hours of Participation
(0,8 ECTS credits)

LVIV

2-4 October 2022

INTERNATIONAL SCIENTIFIC AND PRACTICAL CONFERENCE
MODERN RESEARCH IN WORLD SCIENCE
SCI-CONF.COM.UA
• LVIV •
• 2-4 OCTOBER 2022 •

sci-conf.com.ua

CERTIFICATE

is awarded to

Gilyavski Denis

for being an active participant in

VI International Scientific and Practical Conference

"MODERN RESEARCH IN WORLD SCIENCE"

24 Hours of Participation

(0,8 ECTS credits)

LVIV

4-6 September 2022

INTERNATIONAL SCIENTIFIC AND PRACTICAL CONFERENCE
MODERN RESEARCH IN WORLD SCIENCE
SCI-CONF.COM.UA
• LVIV •
• 4-6 SEPTEMBER 2022 •

sci-conf.com.ua


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДНТБ УКРАЇНИ
ІНТЕІ
ORCID
Connecting Research and Researchers
BENTHAM SCIENCE
4SCIENCE
Share your knowledge
MDClinic
SPRINGER NATURE
Clarivate Analytics
ELSEVIER
AJE
cost
EUROPEAN COOPERATION IN SCIENCE AND TECHNOLOGY
НМШ
СЕРТИФІКАТ
УЧАСНИКА
ЦЕЙ СЕРТИФІКАТ ЗАСВІДЧУЄ, ЩО
Денис ГІЛЯВСЬКИЙ
Брав (ла) участь у роботі Першої міжнародної конференції
“ВІДКРИТА НАУКА ТА ІННОВАЦІЇ В УКРАЇНІ 2022"
(в онлайн режимі, платформа ZOOM)
27-28 жовтня 2022 року
в якості вільного слухача
В.о. директора ДНТБ України, д.е.н.
Алла Жарінова

CERTIFICATE

OF PARTIPIATION

THIS IS TO CERTIFY THAT

Denis Gilyavski

took online part in the

14-th INTERNATIONAL SCIENTIFIC AND PRACTICAL CONFERENCE OF STUDENTS AND YOUNG SCIENTISTS NAMED AFTER HEORHII KIRPA «MODERN TRANSPORT TECHNOLOGIES»

Lviv, Ukraine, 08 December 2022

Chairman of the conference scentific committee

M. Kuzin

ET №22005