Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

ПРОГРАМНИЙ МОДУЛЬ НЕЙРОМЕРЕЖЕВОГО ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ

Виконав: студент 4 курсу, групи 1КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Калінович Р.О.
(прізвище та ініціали)

Керівник: к.т.н., проф. каф.КН
Колесницький О.К.
(прізвище та ініціали)
«07» 06 2024 р.

Рецензент: к.т.н., проф. каф. КСУ
Биков М. М.
(прізвище та ініціали)
«07» 06 2024 р.

Допущено до захисту
Зав. кафедри Яровий А.А.
«07» 06. 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо - професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«07» 03 2024 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Каліновичу Ростиславу Олександровичу

1. Тема роботи: Програмний модуль нейромережевого прогнозування часових рядів.
керівник роботи: Колесницький Олег Костянтинович, к.т.н., доц.
затверджені наказом вищого навчального закладу "11" 03 2024 року №80

2. Термін подання студентом роботи 27.05.2024р.

3. Вихідні дані до роботи :
набір історичні даних з щоденними цінами на акції Apple Inc – не менше як за 3 роки, кількість епох навчання – не менше 5, кількість наступних прогнозів – не менше 3, використання об'єктно-орієнтованої мови програмування.

4. Зміст текстової частини (перелік питань, які потрібно розробити):
проаналізувати існуючі методи прогнозування часових рядів та вибрати найбільш ефективний; вибрати нейронну мережу, обрати її структуру та метод навчання; розробити алгоритм роботи модуля прогнозування часових рядів; спроектувати та реалізувати модуль прогнозування часових рядів за допомогою нейронної мережі; протестувати роботу програми прогнозування часових рядів на основі нейронної мережі.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
Схема алгоритму роботи програми
Структура нейронної мережі
Характеристики набору даних
Результати роботи програми

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Колесницький О.К., к.т.н., проф. каф. КН		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	06.05.2024	10.05.2024	
2	Обгрунтування методу розв'язання задачі, проектування програмного модуля нейромережевого прогнозування часових рядів	10.05.2024	15.05.2024	
3	Програмна реалізація модуля нейромережевого прогнозування часових рядів	15.05.2024	21.05.2024	
4	Проведення тестування модуля нейромережевого прогнозування часових рядів	21.05.2024	25.05.2024	
5	Аналіз результатів тестування модуля нейромережевого прогнозування часових рядів	26.05.2024	30.05.2024	
6	Оформлення матеріалів до захисту БДР	30.05.2024	08.06.2024	

Студентка _____ **Калінович Р.О.**
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ **Колесницький О.К.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Калінович Р.О. Програмний модуль нейромережевого прогнозування часових рядів. Бакалаврська кваліфікаційна робота скаладається з 66 сторінок формату А4, на яких є 13 рисунків, 1 таблиця, список використаних джерел містить 18 найменувань.

Метою роботи є підвищення точності прогнозування часових рядів програмними засобами за рахунок використання нейронних мереж.

У роботі було обґрунтовано вибір архітектури нейронної мережі LSTM з механізмом уваги. Для навчання цієї нейромережі використовується модифікація ADAM методу стохастичного градієнтного спуска. В процесі програмної реалізації модуля нейромережевого прогнозування часових рядів було обґрунтовано вибір мови програмування Python, середовища розробки GoogleColab та спеціалізованих бібліотек Keras, NumPy, Pandas та Matplotlib. Описано основні етапи програмної реалізація та функціонування програмного модуля нейромережевого прогнозування часових рядів, що складається з завантаження бібліотек, завантаження набору даних, попередньої обробки та підготовки даних, побудови моделі LSTM, навчання моделі, оцінювання роботи моделі, прогнозування та візуалізація прогнозів. Навчання модуля відбувалось з використанням набору історичні даних з Yfinance з щоденними цінами на акції Apple Inc. Набір даних було поділено на навчальну (80%) та тестову (20%) вибірки. Розроблений нейромережевий модуль прогнозування часових рядів має підвищену на 7-8% точність прогнозування порівняно з аналогом.

Ключові слова: часовий ряд, прогнозування, нейронна мережа, LSTM.

ABSTRACT

Kalinovich R.O. Software module for neural network forecasting of time series. The bachelor's qualification thesis consists of 66 pages of A4 format, on which there are 13 figures, 1 table, the list of used sources contains 18 names.

The purpose of the work is to increase the accuracy of time series forecasting by software tools due to the use of neural networks.

The work justified the choice of the LSTM neural network architecture with the attention mechanism. The ADAM modification of the stochastic gradient descent method is used to train this neural network. In the process of software implementation of the neural network forecasting module of time series, the choice of the Python programming language, the GoogleColab development environment, and the specialized libraries Keras, NumPy, Pandas, and Matplotlib were justified. The main stages of the software implementation and operation of the software module of neural network forecasting of time series are described, which consists of loading libraries, loading a dataset, preprocessing and preparing data, building an LSTM model, training the model, evaluating the performance of the model, forecasting and visualization of forecasts. The training of the module took place using a set of historical data from Yfinance with daily prices of shares of Apple Inc. The data set was divided into training (80%) and test (20%) samples. The developed neural network module for forecasting time series has a forecast accuracy increased by 7-8% compared to the analogue.

Key words: time series, forecasting, neural network, LSTM.

ЗМІСТ

ВСТУП

Актуальність досліджень.

Оскільки більшість досліджуваних фінансових, економічних або технічних реальних процесів є нестаціонарними, особливої актуальності набуває вирішення задач моделювання і прогнозування нестаціонарних процесів. До таких процесів відносяться, наприклад, процеси формування цін на біржах, значні випадкові коливання курсів валют. В останній час бурного розвитку отримали адаптивні методи короткострокового прогнозування. Перевага їх полягає у тому, що це саморегульовані моделі, і у разі появи нових даних, прогнози оновлюються із мінімальною затримкою в той час як економетрична модель з постійними параметрами буде екстраполювати істотно застарілі залежності. Адаптація до нових даних є перевагою нейронних мереж з їх здатністю до самонавчання.

Прогнозування часових рядів є важливою задачею в багатьох галузях, таких як економіка, фінанси, енергетика, метеорологія, медицина тощо. Традиційні методи прогнозування, такі як ARIMA, експоненційне згладжування та ковзні середні, можуть бути неточними для складних часових рядів, які мають нелінійні закономірності.

У світі фінансових ринків, що стрімко розвивається, точні прогнози не тільки мають важливе значення, а дозволяють зароюляти фінансові кошти. Оскільки ми шукаємо більш досконалі методи для інтерпретації ринкових тенденцій, машинне навчання є найбільш перспективним вибором. Серед різних моделей машинного навчання значну увагу привертають мережі Long Short-Term Memory (LSTM). У поєднанні з механізмом уваги ці моделі стають ще потужнішими, особливо в аналізі часових рядів даних, таких як ціни на акції. Ця робота присвячена використанню LSTM-мереж у поєднанні з механізмами уваги для прогнозування чотирьох наступних періодів ціни акцій Apple Inc. (AAPL), використовуючи дані з Yahoo Finance (yfinance).

Метою дослідження є підвищення точності прогнозування часових рядів програмними засобами за рахунок використання нейронних мереж. На відміну від аналогів, які використовують традиційні методи машинного навчання без використання нейромереж, в роботі пропонується застосувати рекурентні нейронні мережі для навчання прогнозуванню часових рядів.

Об'єктом дослідження є процес комп'ютеризованого прогнозування часових рядів на основі нейромереж.

Предметом дослідження є алгоритми та програмні засоби прогнозування часових рядів на основі рекурентної нейронної мережі та точність їх роботи.

Задачі дослідження:

1. Проаналізувати відомі методи прогнозування часових рядів та обрати напрямок досліджень;
2. Розробити структуру програмного модуля нейромережевого прогнозування часових рядів;
3. Обгрунтувати вибір архітектури нейромережі;
4. Розробити алгоритм роботи програмного модуля нейромережевого прогнозування часових рядів;
5. Здійснити програмну реалізацію модуля нейромережевого прогнозування часових рядів;
6. Провести тестування програмного модуля нейромережевого прогнозування часових рядів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ

1.1 Постановка задачі

Постановка задачі прогнозування часового ряду на основі рекурентної нейронної мережі передбачає наступні компоненти:

1. Вхідні дані:

Часовий ряд, який складається з послідовності значень змінної за певний період часу. Ці дані можуть включати часові марки, такі як дати або часові відмітки. Можливість включення додаткових екзогенних змінних, які можуть впливати на цільову змінну і поліпшити точність прогнозування [1].

2. Вихідні дані:

Прогнозовані значення часового ряду на певний горизонт прогнозу.

3. Виконувані функції:

Модель рекурентної нейронної мережі (RNN) навчається на вхідних даних для прогнозування майбутніх значень часового ряду. Здійснення прогнозування на основі вивчених зв'язків між попередніми значеннями часового ряду та його майбутніми значеннями.

4. Досяжні технічні параметри:

Варіанти архітектури RNN, такі як прості RNN, LSTM (Long Short-Term Memory) або GRU (Gated Recurrent Unit). Кількість шарів та кількість нейронів у кожному шарі. Розмір пакетів (batch size) та кількість епох навчання під час тренування моделі. Функція втрат (loss function), яка використовується під час тренування, наприклад, середньоквадратична помилка (MSE) або середня абсолютна помилка (MAE). Оптимізатори, такі як стохастичний градієнтний спуск (SGD), Adam або RMSProp (Root Mean Squared Propagation).

Результатом виконання цієї задачі є навчена модель, яка може бути використана для прогнозування майбутніх значень часового ряду на основі попередніх спостережень. Точність прогнозу може бути оцінена за допомогою

метрик, таких як середня абсолютна похибка (MAE) та середньоквадратична похибка (RMSE).

1.2 Огляд відомих методів прогнозування часових рядів

Прогнозування часових рядів залишається важливою науковою та прикладною задачею, що актуалізується з розвитком технологій збору та обробки інформації. Широке застосування нейронних мереж у цій сфері обумовлено складністю закономірностей, притаманних більшості часових рядів, які не піддаються адекватному опису лінійними методами.

Нейронні мережі стали домінантним підходом до прогнозування часових рядів завдяки здатності моделювати нелінійні та динамічні залежності, що лежать в основі багатьох реальних процесів.

Залежно від часового горизонту прогнозування розрізняють:

- Короткостроковий прогноз: охоплює найближчий період часу.
- Середньостроковий прогноз: фокусується на більш віддаленому періоді.
- Довгостроковий прогноз: націлений на прогнозування на значному часовому інтервалі.

Вибір типу прогнозу істотно впливає на методи та інструменти, що використовуються для його реалізації.

Прогнозування часового ряду полягає в обчисленні його майбутніх значень або характеристик, що дозволяють їх визначити, на основі аналізу доступних історичних даних.

Один з поширених підходів до прогнозування ґрунтується на припущенні про залежність прогнозованої величини від попередніх значень ряду. Теорема Такенса стверджує, що для динамічних систем існує певне число d (приблизно рівне ефективному числу степенів свободи), яке робить d попередніх значень ряду однозначною характеристикою наступного значення [2].

Огляд відомих методів прогнозування часових рядів [2]:

1. Прогнозування на основі ковзаючого середнього:

Метод: Цей метод використовує середнє значення за певний період часу (вікно) для прогнозування наступного значення. Існує кілька типів ковзаючого середнього, наприклад, просте, експоненціальне та зважене.

Переваги:

- Простий у розумінні та реалізації.
- Не потребує знання статистики.
- Швидкий та ефективний.

Недоліки:

- Не враховує сезонність та тренди.
- Може бути неточним для складних часових рядів.
- Вибір вікна може суттєво впливати на результат.

2. Експоненціальне згладжування:

Метод: Цей метод надає більшу вагу останнім значенням часового ряду, згладжуючи попередні значення експоненціально. Існує кілька типів експоненціального згладжування, наприклад, просте та з сезонністю.

Переваги:

- Більш гнучкий, ніж ковзаюче середнє.
- Може враховувати тренди.
- Простий у реалізації.

Недоліки:

- Може бути чутливим до викидів.
- Не враховує сезонність (за винятком моделей з сезонністю).
- Вибір параметра згладжування може бути складним.

3. ARIMA (Autoregressive Integrated Moving Average):

Метод: Цей метод використовує авторегресію та ковзне середнє для моделювання та прогнозування часових рядів, які мають сезонність та тренди. ARIMA моделі задаються трьома параметрами: p (порядок авторегресії), d (порядок інтегрування) та q (порядок ковзного середнього).

Переваги:

- Один з найпоширеніших методів прогнозування часових рядів.
- Може враховувати складні залежності в даних.
- Гнучкий та може бути адаптований до різних типів часових рядів.

Недоліки:

- Може бути складним для розуміння та реалізації.
- Потребує знання статистики.
- Може бути чутливим до вибору параметрів.
- Не завжди добре працює з нестаціонарними даними.

4. Прогнозування на основі нейронних мереж:

Метод: Цей метод використовує штучні нейронні мережі [3,4] для вивчення складних залежностей в даних та прогнозування часових рядів. Нейронні мережі можуть бути рекурентними (RNN) або згортковими (CNN).

Переваги:

- Може бути дуже точним для складних часових рядів.
- Може враховувати нелінійні залежності.
- Може бути використаний для прогнозування багатоваріантних часових рядів.

Недоліки:

- Може бути складним для навчання та потребує багато даних.

5. Прогнозування на основі екстремальних значень:

Метод: Цей метод використовує екстремальні значення (максимуми та мінімуми) часового ряду для прогнозування майбутніх екстремальних значень.

Переваги:

- Корисний для прогнозування ризиків та аномальних подій.
- Може бути використаний для прогнозування повеней, землетрусів та інших природних катаклізмів.

Недоліки:

- Може бути неточним для прогнозування звичайних значень часового ряду.
- Потребує багато даних про екстремальні значення.

6. Прогнозування на основі методу Монте-Карло:

Метод: Цей метод використовує стохастичне моделювання для генерації множинних можливих реалізацій часового ряду в майбутньому.

Переваги:

- Може враховувати невизначеність та ризики.
- Може бути використаний для прогнозування інтервалів довірчого інтервалу.

Недоліки:

- Може бути обчислювально витратним.
- Потребує багато даних для навчання моделі.

Для реалізації у цій роботі було обрано метод на основі штучних нейронних мереж

1.3 Огляд програм-аналогів до програмного модуля нейромережевого прогнозування часових рядів

Взагалі, існує кілька відомих програмних засобів, які здатні здійснювати прогнозування часових рядів. Однак, зазвичай їх важко порівнювати між собою через те, що вони використовують для тренування алгоритмів машинного навчання різні набори даних. Загальновідомо, що від якості підготовки тренувального набору даних суттєво залежить якість роботи програм прогнозування на основі машинного навчання. Якість підготовки набору даних для тренування визначається тим, наскільки різнорідні дані у нього вносять, наскільки повно ці дані репрезентують все можливе розмаїття випадків прогнозування. Багато залежить також від величини набору даних. Треба прагнути до створення наборів даних якомога більших обсягів, з рівномірним представництвом різних класів та з всеохопленням різних кейсів

і нюансів даних для різних представників кожного класу. Навіть при використанні одного і того ж набору даних, різні розробники по-різному розділяють їх на навчальну та тестову вибірку. А це також має вплив на результат прогнозування, тому що ті приклади, що у одних розробників знаходяться у навчальній вибірці, у інших розробників знаходяться у тестовій вибірці і навпаки. Якщо у навчальну вибірку залучаються більш складні та представницькі приклади, а у тестову вибірку потраплять більш прості та типові приклади, то достовірність прогнозування буде вищою, ніж у зворотному випадку.

Розглянемо програми-аналоги:

- Autodesk Prophet: Проста у використанні програма для прогнозування часових рядів, яка використовує метод експоненціального згладжування з сезонністю.
- Facebook Prophet: Більш складна програма для прогнозування часових рядів, яка використовує метод прогнозування на основі нейронних мереж.
- Statsmodels: Бібліотека Python для статистичного аналізу та прогнозування часових рядів, яка реалізує багато з вищезгаданих методів.

Недоліки програм-аналогів:

- Не всі програми можуть враховувати складні залежності в даних.
- Деякі програми можуть бути складними для розуміння та використання.
- Вибір методу прогнозування може бути складним завданням.

Переваги запропонованої в проекті програми:

- Враховує складні залежності в даних.
- Проста у використанні.

У даній роботі було вирішено в одній програмі промоделювати різні архітектури нейронної мережі LSTM для вирішення задачі прогнозування часових рядыв, а потім порівняти результати. Причому моделювання цих архытектур буде відбуватись на одному наборі даних ысторичних даних

часового ряду та при однаковому розбитті цього набору даних на навчальну та тестову вибірки.

Головними показниками якості програм прогнозування часових рядів є такі метрики, як середня абсолютна похибка (MAE) та середньоквадратична похибка (RMSE), тому порівнювати різні архітектури LSTM мереж будемо у розділі 4 саме за параметрами MAE та RMSE.

1.4 Висновок

У розділі описано детальну постановку задачі прогнозування часових рядів. Також розглядаються різні методи розв'язання задачі прогнозування часових рядів і оцінюється їх переваги та недоліки. Серед таких методів прогнозування часових рядів як прогнозування на основі ковзаючого середнього, експоненціальне згладжування, ARIMA (Autoregressive Integrated Moving Average), прогнозування на основі нейронних мереж, прогнозування на основі екстремальних значень, прогнозування на основі методу Монте-Карло для реалізації у цій роботі було обрано метод на основі штучних нейронних мереж як найбільш перспективний і застосовний до даної задачі. Крім цього, було проаналізовано програми-аналоги розроблюваного модуля, їх переваги та недоліки.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ НЕЙРОМЕРЕЖЕВОГО ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ

2.1 Обґрунтування вибору методу прогнозування часових рядів

Вибір методу розв’язання задачі є ключовим етапом дослідження, який визначає його ефективність та надійність. У цьому розділі буде обґрунтовано вибір методу розв’язання задачі прогнозування часових рядів на основі рекурентної нейронної мережі (РНМ) [5].

Існує широкий спектр методів прогнозування, які можна поділити на дві основні категорії:

1. Традиційні методи: експоненційне згладжування, ковзаючі середні, ARIMA та інші. Ці методи описують часовий ряд математичними моделями, що робить їх простими та зрозумілими. Проте, вони не завжди здатні адекватно враховувати складні нелінійні закономірності, притаманні реальним даним.

2. Методи машинного навчання: нейронні мережі, машини опорних векторів, методи на основі дерев рішень. Ці методи володіють більшою гнучкістю та можуть моделювати складні нелінійні залежності. Проте, вони потребують більшого обсягу даних для навчання та можуть бути складнішими для інтерпретації.

Рекурентні нейронні мережі мають ряд суттєвих переваг, які роблять їх ефективним інструментом для прогнозування часових рядів:

- Здатність враховувати динаміку часових рядів: РНМ можуть обробляти послідовності даних, враховуючи не лише поточне значення, але і його зв'язок з попередніми значеннями.
- Моделювання нелінійних залежностей: РНМ не обмежені лінійними моделями і можуть адекватно описувати складні нелінійні закономірності, притаманні багатьом часовим рядам.

– Висока точність прогнозування: РНМ, завдяки своїй гнучкості, можуть досягати високої точності прогнозування, що робить їх цінним інструментом для прийняття рішень.

Зважаючи на вищезазначені переваги, РНМ визнані одним з найефективніших методів прогнозування часових рядів. РНМ – це потужний інструмент для прогнозування часових рядів, який може ефективно моделювати складні нелінійні залежності та досягати високої точності прогнозування.

Однак, варто зазначити, що вибір методу також залежить від конкретної задачі та доступних даних. Наприклад, якщо обсяг даних обмежений, або наявні ресурси для навчання обмежені, то використання більш простих методів може бути виправданим.

Нейронні мережі мають ряд суттєвих переваг у порівнянні з іншими методами:

– Ефективність при вирішенні неформалізованих задач: не потребують чіткої математичної специфікації моделі.

– Стійкість до змін середовища: добре адаптуються до мінливих умов та динамічних процесів.

– Результативність з великими обсягами даних: автоматично враховують нелінійні та складні взаємодії між факторами.

– Здатність працювати з неповною інформацією: можуть обробляти "зашумлені" та неповні дані.

Незважаючи на переваги, нейронні мережі мають й певні обмеження:

– Необхідність великої кількості даних: для ефективного навчання зазвичай потрібно більше 50 спостережень.

– Високі часові витрати: процес навчання нейронної мережі може бути трудомістким та ресурсоємним.

– Складність інтерпретації результатів: потребує спеціальних знань та навичок.

Процес прогнозування часового ряду за допомогою нейронної мережі складається з наступних етапів: Попередні перетворення: зменшення помилки прогнозування за рахунок нормалізації, стандартизації, зменшення розмірності та усунення колінеарності даних; сструктурний синтез нейронної мережі: визначення архітектури нейрона та структури зв'язків між нейронами; параметричний синтез нейронної мережі: процес навчання нейронної мережі на основі навчальної вибірки; перевірка помилки прогнозу: оцінка якості прогнозу на контрольній вибірці.

Попередні перетворення відіграють важливу роль у підвищенні точності прогнозування. До них висуваються такі вимоги:

– Збереження інформативності прогнозованої величини: перетворення не повинні призводити до втрати суттєвої інформації, необхідної для відновлення майбутніх значень.

– Зменшення розмірності простору ознак: зниження складності моделі та прискорення навчання мережі. Методи, такі як згортки (convolutions), дозволяють описувати ситуацію меншою кількістю ознак без втрати точності.

– Зменшення колінеарності між входами: взаємозалежність вхідних даних може знижувати інформативність та погіршувати навчання. Деякі методи згортання лише частково вирішують цю проблему.

Стаціонарність та несуперечливість даних є важливими для коректного функціонування моделі.

Вимога стаціонарності вхідних значень: для ефективної екстраполяції майбутніх значень часового ряду необхідно забезпечити стаціонарність вхідних даних, тобто відсутність суттєвих змін їх статистичних властивостей з часом.

Вимога до несуперечливості наборів: набори навчальної та контрольної вибірок не повинні суперечити один одному. Це необхідно для

коректного відновлення функціональної залежності між вхідними даними та прогнозованою величиною.

Застосування нейронних мереж відкриває нові можливості для високоточного прогнозування часових рядів у різних сферах, включаючи економіку, фінанси, енергетику, метеорологію тощо. Проте, для досягнення оптимальних результатів необхідно ретельно підходити до попередньої обробки даних та враховувати обмеження нейронних мереж. Подальший розвиток технологій машинного навчання сприятиме підвищенню ефективності та розширенню сфери застосування нейронних мереж для прогнозування складних часових рядів.

Для поставленої задачі найкраще підходять нейронні мережі RNN та LSTM.

Рекурентні нейронні мережі (RNN) та довготривалаа короткочасна пам'ять (LSTM) – це два типи нейромереж, які застосовуються для обробки послідовних даних, таких як часові ряди, текст або аудіосигнали.

В рекурентних нейромережах (RNN) інформація передається через мережу у зворотньому напрямку, тобто вихід мережі у кожен момент часу також використовується як вхід мережі у наступний момент часу [5]. Проте, у звичайних RNN інформація швидко може втрачатися через проблему зниклого/вибувного градієнта.

Довга короткочасна пам'ять (LSTM) – це окремий тип рекурентних нейромереж, який має додаткові внутрішні блоки, що дозволяють зберігати і використовувати інформацію, яка має тривалі залежності у часі. Вони мають спеціальні ворота входу, виходу та забування, які регулюють потік інформації через нейромережу та дозволяють пам'ятати важливість різних аспектів вхідних даних [6].

Зберігання інформації в RNN може бути обмеженим через проблему зниклого/вибувного градієнта. Це означає, що нейромережа може забувати важливі залежності в дуже далекому минулому.

LSTM має можливість довготривалого зберігання інформації завдяки своїм спеціальним внутрішнім структурам, що дозволяють уникнути проблеми зниклого/вибувного градієнта. Це робить їх ефективними для обробки послідовних даних з тривалими залежностями [5].

RNN використовуються там, де важлива послідовність даних, але не обов'язково тривалість цієї залежності. Наприклад, у завданнях аналізу та розпізнавання мови, моделювання часових послідовностей тощо.

LSTM використовуються там, де потрібно обробляти тривалі залежності в даних, таких як курси валют, ціни акцій, прогнозуванні попиту тощо.

Отже, хоча RNN та LSTM подібні за своєю основною архітектурою, LSTM має додаткові механізми, які роблять її більш ефективною для обробки послідовних даних з тривалими залежностями, які часто зустрічаються у задачах прогнозування ціни акцій. Для реалізації у даній роботі була обрана саме нейромережа LSTM.

2.2 Структура та математична модель нейромережі LSTM

LSTM-мережі - це тип рекурентних нейронних мереж (RNN), спеціально розроблених для запам'ятовування та обробки послідовностей даних протягом тривалих періодів часу. Від традиційних RNN LSTM відрізняє здатність зберігати інформацію протягом тривалого часу завдяки унікальній структурі, що складається з трьох воріт: вхідних, забування та вихідних. Ці вентилі спільно керують потоком інформації, вирішуючи, що зберігати, а що відкидати, тим самим пом'якшуючи проблему зникаючих градієнтів - поширену проблему стандартних ШНМ.

У контексті фінансових ринків ця здатність запам'ятовувати і використовувати довгострокові залежності є безцінною. Наприклад, ціни на акції залежать не лише від нещодавніх тенденцій, але й від закономірностей, що склалися з плином часу. Мережі LSTM вправно фіксують ці часові залежності, що робить їх ідеальними для аналізу фінансових часових рядів.

Механізм уваги: Покращення LSTM [7].

Механізм уваги, спочатку популяризований в області обробки природної мови, знайшов своє застосування і в інших сферах, включаючи фінанси. Він працює на основі простої, але глибокої концепції: не всі частини вхідної послідовності є однаково важливими. Дозволяючи моделі зосередитися на певних частинах вхідної послідовності, ігноруючи інші, механізм уваги покращує здатність моделі розуміти контекст.

Включення уваги в LSTM-мережі призводить до того, що модель стає більш сфокусованою та контекстно-орієнтованою. При прогнозуванні цін на акції певні історичні дані можуть бути більш релевантними, ніж інші. Механізм уваги дає змогу LSTM зважувати ці точки більш вагомо, що призводить до більш точних і нюансованих прогнозів.

Релевантність у прогнозуванні фінансових моделей.

Поєднання LSTM з механізмами уваги створює надійну модель для прогнозування фінансових моделей. Фінансовий ринок - це складна адаптивна система, на яку впливає безліч факторів і яка має нелінійні характеристики. Традиційні моделі часто не в змозі відобразити цю складність. Однак мережі LSTM, особливо в поєднанні з механізмом уваги, здатні розгадати ці закономірності, пропонуючи глибше розуміння і точніші прогнози майбутніх рухів акцій.

У цій роботі ми будуємо і впроваджуємо LSTM з механізмом уваги для прогнозування наступних чотирьох періодів ціни акцій AAPL, ми застосуємо LSTM для фінансового аналізу, оскільки вона має адекватно реагувати на постійно мінливу динаміку фондового ринку.

Нейромережа з довгою короткочасною пам'яттю (LSTM) складається з різних компонентів, які дозволяють моделі зберігати та використовувати інформацію з тривалими залежностями в даних. На рисунку 2.1 показано структуру LSTM мережі.

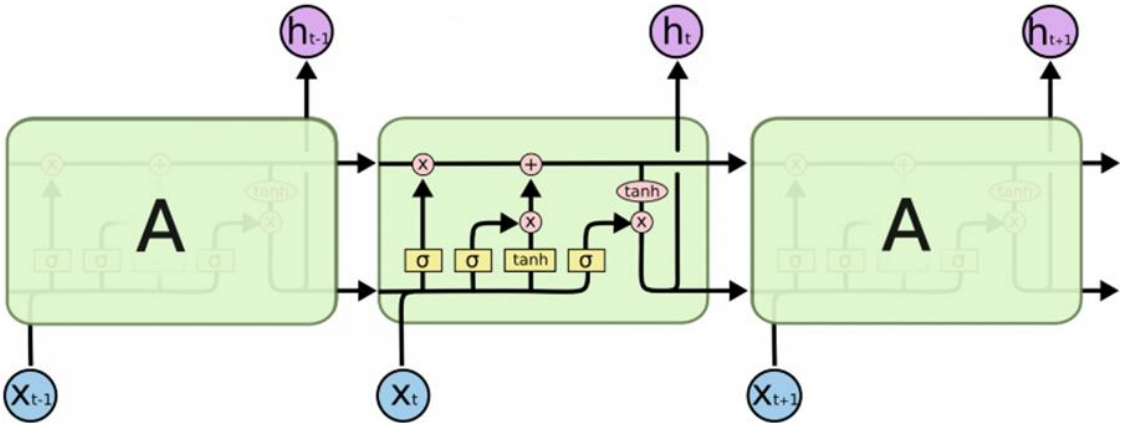


Рисунок 2.1 – Структура LSTM мережі

Основні компоненти LSTM включають:

– Комірка пам'яті (Memory Cell): Це головний елемент LSTM, який здійснює зберігання інформації протягом тривалого часу. Вона може зчитувати, записувати та стирати інформацію згідно зі своїми внутрішніми механізмами.

– Ворота входу (Input Gate): Ворота входу визначають, яка частина інформації буде збережена в комірці пам'яті. Вони регулюють потік нової інформації, що входить у пам'ять.

– Ворота забування (Forget Gate): Ворота забування визначають, яка частина інформації буде забута або видалена з комірки пам'яті. Вони регулюють здатність пам'яті зберігати старі значення.

– Ворота виходу (Output Gate): Ворота виходу визначають, яка частина інформації з пам'яті буде використана для виходу з LSTM. Вони регулюють, яка частина пам'яті буде передана до виходного шару мережі.

Математично модель LSTM можна виразити наступним чином.

Вирази для воріт:

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i)$$
$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f)$$
$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o)$$

де: i_t – ворота входу (input gate) на часі t;

f_t – ворота забування (forget gate) на часі t;

o_t – ворота виходу (output gate) на часі t;

x_t – вхідний вектор на часі t;

h_{t-1} – вектор прихованого стану на попередньому кроці;

W_{ix}, W_{fx}, W_{ox} – вагові матриці для входу на ворота i, f, o;

W_{ih}, W_{fh}, W_{oh} – вагові матриці для прихованого стану на ворота i, f, o;

b_i, b_f, b_o – зміщення для воріт i, f, o;

σ – функція активації (зазвичай сигмоїдна).

Оновлення комірки пам'яті c_t:

$$c_t = f_t c_{t-1} + i_t \tanh(W_{cx} x_t + W_{ch} h_{t-1} + b_c)$$

де: c_t – новий стан пам'яті на часі t.

Вихідний вектор h_t:

$$h_t = o_t \tanh(c_t)$$

Ці формули визначають роботу кожного компонента LSTM та оновлення стану пам'яті та вихідного вектора на кожному кроці часу t.

2.3 Розробка алгоритму роботи програмного модуля нейромережевого прогнозування часових рядів

Розглянемо ключові компоненти, структуру та алгоритм основного компонента програми прогнозування часових рядів за допомогою нейронної мережі LSTM.

Структура програми:

- Налаштування середовища,
- Отримання історичних даних з yfinance,
- Попередня обробка та підготовка даних:
    - Очищення даних,
    - Вибір ознак,
    - Нормалізація,
    - Створення послідовностей,
- Поділ на навчальний та тестовий набори,
- Перетворення даних для LSTM,
- Побудова моделі LSTM з механізмом уваги:
    - Створення шарів LSTM,
    - Інтеграція механізму уваги,
    - Оптимізація моделі,
    - Компіляція моделі,
- Виведення короткого опису моделі,
- Навчання моделі,
- Оцінювання роботи моделі за допомогою тестового набору,
- Обчислення показників ефективності,
- Прогнозування наступних 4 періодів часового ряду,
- Візуалізація прогнозів,
- Фінальна візуалізація для прогнозів.

Зобразимо роботу програми у вигляді алгоритму на рис. 2.2:

Першим кроком в програмному модулі нейромережевого прогнозування часових рядів буде завантаження історичних даних часового ряду (блок 1) з yfinance [8].

Далі переходимо до попередньої обробка та підготовки даних (блок 2), яка включає в себе: очищення даних, вибір ознак, нормалізацію, створення послідовностей.

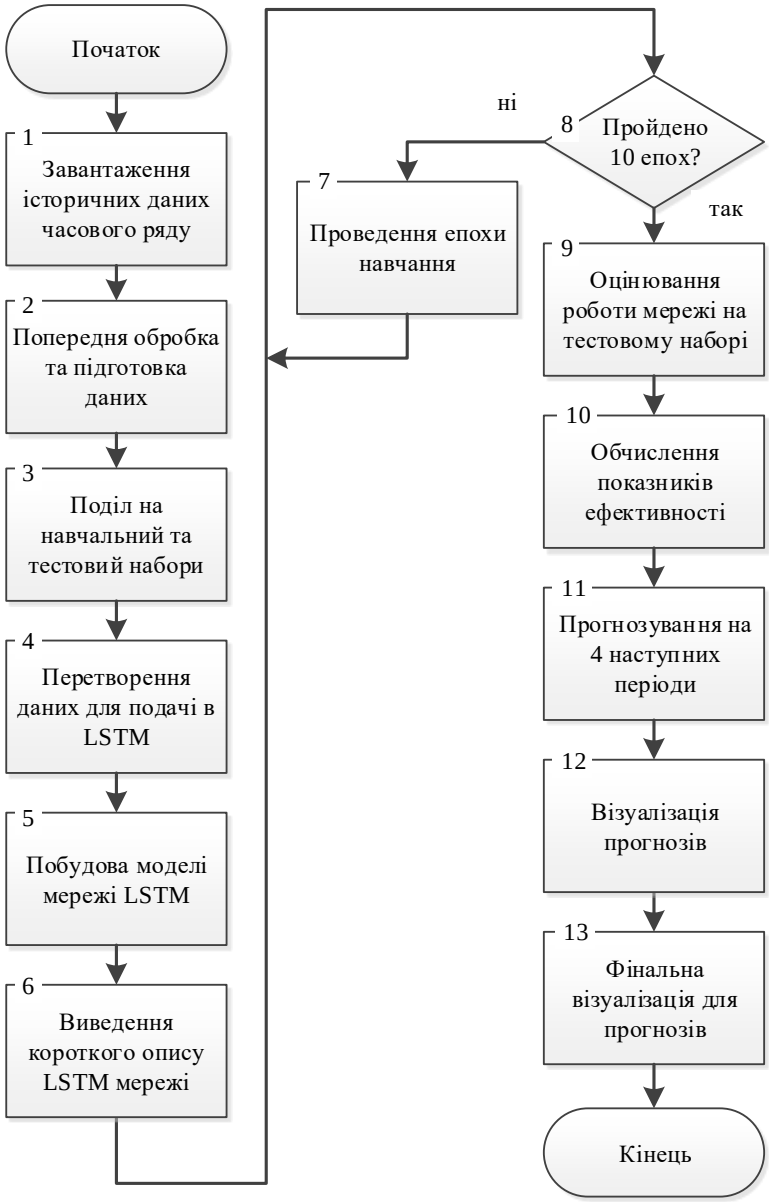


Рисунок 2.2 – Схема алгоритму роботи програмного модуля прогнозування часових рядів

Потім проводиться поділ всієї множини історичних даних часового ряда на навчальний та тестовий набори (блок 3).

Далі перетворюємо дані для подачі в LSTM мережу (блок 4). Потрібно перетворити дані у 3D-формат [samples, time steps, features], необхідний для шарів LSTM.

Після цього будуємо модель мережі LSTM з механізмом уваги (блок 5), куди входить: створення шарів LSTM, інтеграція механізму уваги, оптимізація

моделі, компіляція моделі. Потім виводимо короткий опис моделі мережі LSTM (блок 6), який можна побачити на рис. 3.3.

Далі відбувається навчання нейронної мережі LSTM протягом 10 епох (блоки 7 та 8). Після цього здійснюється оцінювання роботи моделі мережі LSTM за допомогою тестового набору (блок 9). При цьому обчислюється похибка мережі на тестовому наборі «Test loss». Потім обчислюються показники ефективності роботи мережі (блок 10). Точність прогнозу може бути оцінена за допомогою метрик, таких як середня абсолютна похибка (MAE) та середньоквадратична похибка (RMSE).

Далі переходимо до прогнозування наступних 4 періодів часового ряду (блок 11). Після цього відбувається візуалізація прогнозів (блок 12), тобто будується детальний графік чотирьох прогнозованих значень часового ряду.

І, нарешті, здійснюється фінальна візуалізація для прогнозів (блок 13).

2.6 Висновок

У розділі було обґрунтовано вибір типу нейронної мережі для програмного модуля нейромережевого прогнозування часових рядів. Було обрано архітектуру нейронної мережі LSTM з механізмом уваги та детально проаналізовано її математичну модель та принципи функціонування. Для навчання цієї нейромережі використовується модифікація ADAM методу стохастичного градієнтного спуска. Розроблено алгоритм роботи програмного модуля нейромережевого прогнозування часових рядів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ НЕЙРОМЕРЕЖЕВОГО ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ

3.1 Обґрунтування вибору мови програмування та спеціалізованих бібліотек

Обґрунтування вибору мови програмування та середовища розробки має велике значення для ефективного виконання проекту. У цьому випадку рекомендується використовувати мову програмування Python [9] та середовище розробки GoogleColab [10].

Переваги мови програмування Python:

- Простота використання: Python має чистий та лаконічний синтаксис, що дозволяє швидко розробляти та тестувати код.
- Велика спільнота та розширення: Python має активну спільноту розробників, яка постійно розвивається та підтримує широкий спектр бібліотек та інструментів для різних задач, включаючи роботу з нейронними мережами та аналіз даних.
- Підтримка штучного інтелекту та машинного навчання: Python є однією з найпопулярніших мов для розробки імплементацій нейронних мереж та інших алгоритмів машинного навчання через бібліотеки, такі як TensorFlow, PyTorch, scikit-learn тощо.

Google Colaboratory або Colab [11] - це хмарний сервіс від Google Research. Це IDE, яка дозволяє будь-якому користувачеві писати вихідний код у своєму редакторі та запускати його з браузера. Зокрема, він підтримує мову програмування Python і орієнтований на завдання машинного навчання, аналіз даних, навчальні проекти тощо.

Ця послуга, заснована на Jupyter Notebook [9], розміщена абсолютно безкоштовно за допомогою облікового запису Gmail, не вимагає налаштування, а також вам не доведеться завантажувати чи встановлювати Jupyter. Він запропонує вам обчислювальні ресурси для редагування та

тестування вашого коду, наприклад GPGPU його серверів тощо. Очевидно, що як щось безкоштовне, Google Colaboratory не має необмежених ресурсів і не гарантує їх, але вони змінюються залежно від використання, яке надається системі. Якщо ви хочете зняти ці обмеження і отримати більше, вам доведеться заплатити.

Важливо зазначити, що коли ви отримуєте доступ до Colab за допомогою свого облікового запису, ви отримуєте віртуальну машину, на якій ви можете запускати свій код, ізольований від інших користувачів і ресурсів. Тому ви можете відновити віртуальну машину до початкового стану, якщо у вас виникнуть проблеми. Це також означає, що якщо ви виконуєте деякий код у своїй віртуальній машині та закриваєте браузер, машини будуть видалені після певного періоду бездіяльності, щоб звільнити ресурси. Однак ви матимете свої блокноти в GDrive, якщо ви їх зберегли, або ви зможете завантажити їх локально (формат Jupyter з відкритим кодом .ipynb).

В цілому, використання Python та GoogleColab забезпечить зручне та ефективне середовище для розробки модуля прогнозування часового ряду на основі рекурентної нейронної мережі.

Оскільки ми використовуємо нейромережі, то для цієї роботи знадобляться бібліотеки Keras [12], NumPy [13], Pandas [14] та Matplotlib [15] (для побудови діаграм та графіків).

3.2 Опис набору даних часового ряду для навчання та тестування модуля

Як приклад часового ряду будемо використовувати часовий ряд цін акцій Apple Inc. (AAPL) – рис. 3.1 [16], використовуючи дані з Yahoo Finance (yfinance) [8].

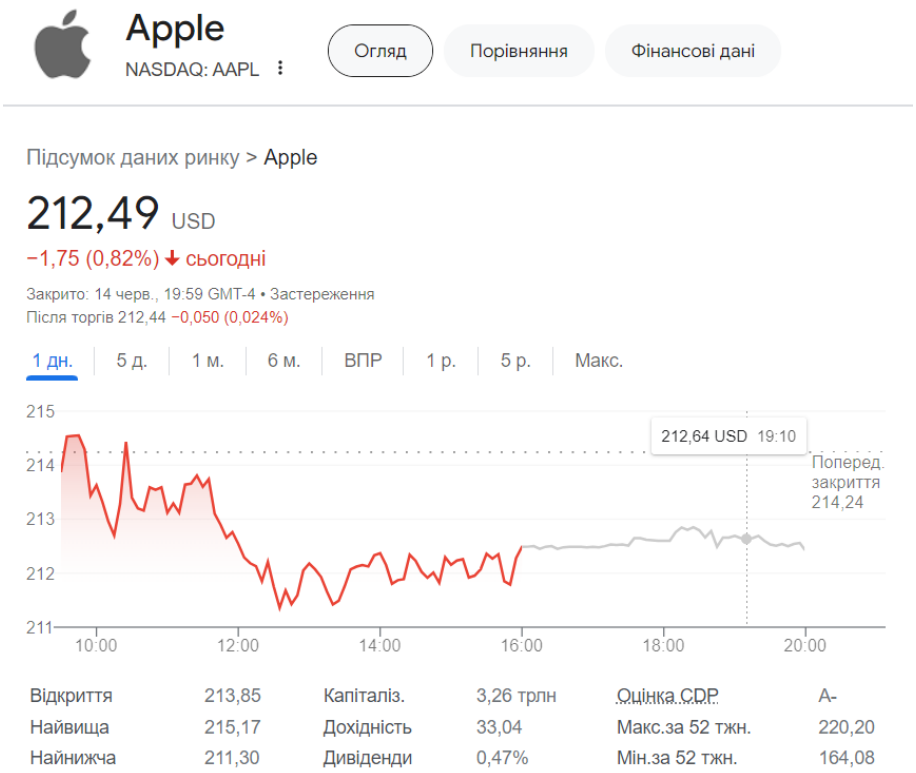


Рисунок 3.1 – Часовий ряд цін акцій Apple Inc. (AAPL)

Apple Inc. — американська корпорація з офісом у Купертіно, що проєктує та розробляє побутову електроніку, програмне забезпечення й онлайн-сервіси. Є першою американською компанією, чия капіталізація перевершила 1 трлн доларів США. Це сталося під час торгів акціями компанії 2 серпня 2018 року. Цього дня компанія також стала найдорожчою публічною компанією за всю історію, обійшовши капіталізацію попереднього рекордсмена — компанії PetroChina ($1,005 трлн у листопаді 2007 року). Станом на середину червня 2024 року, Apple продовжує утримувати титул найдорожчої компанії у світі, змістивши з першої позиції Microsoft.

Щоб проаналізувати динаміку акцій AAPL, нам потрібні історичні дані про ціни на акції, які можна отримати за допомогою Yfinance. Yfinance - це бібліотека, призначена для отримання історичних ринкових даних з Yahoo Finance. Вона містить щоденні ціни на акції Apple Inc. з 1 січня 2020 року по 1

січня 2024 року. Фрагмент набору даних часового ряду акцій Apple Inc. Показано на рис. 4.1.

У даних фондового ринку є такі ознаки: "відкриття", "максимум", "мінімум", "закриття" та "обсяг". Для нашої задачі ми будемо використовувати ціну закриття, але можна поекспериментувати з додатковими характеристиками, такими як "відкриття", "максимум", "мінімум" та "обсяг".

3.3 Програмна реалізація модуля прогнозування часових рядів на основі рекурентної нейронної мережі

Щоб почати будувати LSTM-модель з акцентом на прогнозування біржових патернів AAPL, першим кроком є налаштування нашого середовища кодування в Google Colab. Google Colab - це хмарний сервіс, який надає безкоштовне середовище для ноутбуків Jupyter з підтримкою GPU, що ідеально підходить для запуску моделей глибокого навчання.

```
!pip install tensorflow -qqq
!pip install keras -qqq
!pip install yfinance -qqq
```

Налаштування середовища. Після встановлення ми можемо імпортувати ці бібліотеки до нашого середовища Python. Запустіть наступний код:

```
import tensorflow as tf
import keras
import yfinance as yf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Check TensorFlow version
print("TensorFlow Version: ", tf.__version__)
```

Цей код не лише імпортує бібліотеки, але й перевіряє версію TensorFlow, щоб переконатися, що все актуально.

Отримання даних з Yfinance [8].

Отримання історичних даних.

Щоб проаналізувати динаміку акцій AAPL, нам потрібні історичні дані про ціни на акції, які можна отримати за допомогою Yfinance. Yfinance - це

бібліотека, призначена для отримання історичних ринкових даних з Yahoo Finance.

Щоб завантажити історичні дані AAPL потрібно запустити наступний код у ноутбуці Colab:

```
# Fetch AAPL data
aapl_data = yf.download('AAPL', start='2020-01-01', end='2024-01-01')

# Display the first few rows of the dataframe
aapl_data.head()
```

Цей скрипт отримує щоденні ціни на акції Apple Inc. з 1 січня 2020 року по 1 січня 2024 року. Дати початку і закінчення можна налаштувати за бажанням.

Важливість попередньої обробки даних та відбору ознак.

Після того, як дані отримано, важливими стають попередня обробка та відбір ознак. Попередня обробка передбачає очищення даних і приведення їх у відповідність до моделі. Це включає обробку відсутніх значень, нормалізацію або масштабування даних, а також потенційне створення додаткових функцій, таких як ковзні середні або відсоткові зміни, які можуть допомогти моделі ефективніше навчатися.

Вибір функцій полягає у виборі правильного набору функцій, які роблять найбільший внесок у прогнозовану змінну. Для прогнозування цін на акції зазвичай використовують такі ознаки, як ціна відкриття, ціна закриття, максимум, мінімум та обсяг. Важливо вибрати ознаки, які надають релевантну інформацію, щоб запобігти навчанню моделі на шумі.

Для цього проводиться попередня обробка цих даних і будується LSTM-модель з шаром уваги, щоб робити прогнози.

3.3.1 Попередня обробка та підготовка даних.

Першим важливим кроком перед побудовою LSTM-моделі є підготовка набору даних. Цей розділ охоплює основні етапи попередньої обробки даних,

щоб зробити дані про акції AAPL від yfinance готовими для нашої LSTM-моделі.

Очищення даних.

Набори даних фондового ринку часто містять аномалії або відсутні значення. Дуже важливо обробляти їх, щоб запобігти неточностям у прогнозах.

-- Визначення відсутніх значень: Перевіряємо, чи немає в наборі даних відсутніх значень. Якщо такі є, заповнюємо їх методом прямого або зворотного заповнення або повністю видаляємо ці рядки.

```
# Checking for missing values
aapl_data.isnull().sum()

# Filling missing values, if any
aapl_data.fillna(method='ffill', inplace=True)
```

-- Робота з аномаліями: Іноді набори даних містять помилкові значення через збої в процесі збору даних. Якщо ви помітили якісь аномалії (наприклад, екстремальні стрибки цін на акції, які є нереальними), їх слід виправити або видалити.

Вибір ознак.

У даних фондового ринку різні ознаки можуть бути важливими. Зазвичай використовуються "відкриття", "максимум", "мінімум", "закриття" та "обсяг".

-- Вирішальні ознаки: Для нашої моделі ми використаємо ціну закриття, але можна поекспериментувати з додатковими характеристиками, такими як "відкриття", "максимум", "мінімум" та "обсяг".

Нормалізація.

Нормалізація - це метод, який використовується для приведення значень числових стовпчиків у наборі даних до єдиного масштабу, не спотворюючи відмінності в діапазонах значень.

-- Застосування мінімально-максимального масштабування: Масштабує набір даних таким чином, що всі вхідні характеристики лежать у діапазоні між 0 і 1.

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0,1))
aapl_data_scaled = scaler.fit_transform(aapl_data['Close'].values.reshape(-
1,1))
```

Створення послідовностей.

LSTM-моделі вимагають, щоб вхідні дані були у форматі послідовностей. Ми перетворюємо дані в послідовності, щоб модель могла навчатися на них.

-- Визначення довжини послідовності: Обираємо довжину послідовності (наприклад, 60 днів). Це означає, що для кожного зразка модель буде переглядати дані за останні 60 днів, щоб зробити прогноз.

```
X = []
y = []

for i in range(60, len(aapl_data_scaled)):
    X.append(aapl_data_scaled[i-60:i, 0])
    y.append(aapl_data_scaled[i, 0])
```

Поділ на навчальні та тестові набори.

Розділяємо дані на навчальні та тестові набори, щоб правильно оцінити продуктивність моделі.

- Визначення співвідношення розділення: Зазвичай 80% даних використовується для навчання, а 20% - для тестування.

```
train_size = int(len(X) * 0.8)
test_size = len(X) - train_size

X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
```

Перетворення даних для LSTM.

Нарешті, нам потрібно перетворити наші дані у 3D-формат [samples, time steps, features], необхідний для шарів LSTM.

-- Перетворення даних:

```
X_train, y_train = np.array(X_train), np.array(y_train)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

У наступному розділі ми використаємо ці попередньо оброблені дані для побудови та навчання нашої LSTM-моделі з механізмом уваги.

3.3.2 Побудова моделі LSTM з увагою

Побудуємо нашу LSTM-модель з додатковим механізмом уваги, пристосованим для прогнозування біржових моделей AAPL. Для цього нам знадобляться TensorFlow і Keras, які вже налаштовані у вашому середовищі Colab.

Створення шарів LSTM.

Наша LSTM-модель складатиметься з декількох шарів, включаючи шари LSTM для обробки даних часових рядів. Базова структура виглядає наступним чином:

```
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout, AdditiveAttention, Permute,
Reshape, Multiply

model = Sequential()

# Adding LSTM layers with return_sequences=True
model.add(LSTM(units=50, return_sequences=True,
input_shape=(X_train.shape[1], 1)))
model.add(LSTM(units=50, return_sequences=True))
```

У цій моделі `units` представляють кількість нейронів у кожному шарі LSTM. `return_sequences=True` має вирішальне значення для перших шарів, щоб гарантувати, що на виході будуть послідовності, які необхідні для стекування шарів LSTM. Останній шар LSTM не повертає послідовності, оскільки ми готуємо дані для шару уваги.

Інтеграція механізму уваги.

Механізм уваги можна додати, щоб покращити здатність моделі фокусуватися на відповідних часових кроках:

```
# Adding self-attention mechanism
# The attention mechanism
attention = AdditiveAttention(name='attention_weight')
# Permute and reshape for compatibility
model.add(Permute((2, 1)))
model.add(Reshape((-1, X_train.shape[1])))
attention_result = attention([model.output, model.output])
multiply_layer = Multiply()([model.output, attention_result])
# Return to original shape
model.add(Permute((2, 1)))
model.add(Reshape((-1, 50)))

# Adding a Flatten layer before the final Dense layer
model.add(tf.keras.layers.Flatten())

# Final Dense layer
model.add(Dense(1))

# Compile the model
# model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
# history = model.fit(X_train, y_train, epochs=100, batch_size=25,
validation_split=0.2)
```

Цей користувацький шар обчислює зважену суму вхідної послідовності, що дозволяє моделі приділяти більше уваги певним часовим крокам.

Оптимізація моделі.

Щоб підвищити продуктивність моделі та зменшити ризик перенавчання, ми включаємо функцію відсіювання та пакетну нормалізацію.

```
from keras.layers import BatchNormalization

# Adding Dropout and Batch Normalization
model.add(Dropout(0.2))
model.add(BatchNormalization())
```

Відсіювання допомагає запобігти надмірному пристосуванню, випадково встановлюючи частину вхідних одиниць в 0 при кожному оновленні під час навчання, а пакетна нормалізація стабілізує процес навчання.

Компіляція моделі.

Нарешті, ми компілюємо модель з оптимізатором Adam [17] і функцією втрат, які підходять для нашої регресійної задачі.

```
model.compile(optimizer='adam', loss='mean_squared_error')
```

Оптимізатор Adam, як правило, є гарним вибором для рекурентних нейронних мереж, а середня квадратична похибка добре підходить як функція втрат для регресійних задач, подібних до нашої.

Короткий опис моделі.

Корисно переглянути резюме моделі, щоб зрозуміти її структуру та кількість параметрів (рис.3.2).

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 50)            10400

 lstm_1 (LSTM)               (None, 60, 50)            20200

 lstm_2 (LSTM)               (None, 50)                20200

 attention (Attention)       (50,)                     100

 dropout (Dropout)           (50,)                     0

 batch_normalization (Batch  (50,)                     200
 Normalization)

=================================================================
Total params: 51100 (199.61 KB)
Trainable params: 51000 (199.22 KB)
Non-trainable params: 100 (400.00 Byte)
_________________________________________________________________
```

Рисунок 3.2 – Структура та параметри нейромережі LSTM.

3.3.3 Навчання моделі.

Тепер, коли наша LSTM-модель з увагою побудована, настав час навчити її за допомогою підготовленого навчального набору. Цей процес включає в себе подачу навчальних даних до моделі та навчання її робити прогнози.

Навчальний код.

Використовуємо наступний код для навчання вашої моделі з `X_train` та `y_train`:

```
# Assuming X_train and y_train are already defined and preprocessed
history = model.fit(X_train, y_train, epochs=100, batch_size=25, validation_split=0.2)
```

Тут ми навчаємо модель на 100 епохах з розміром пакету 25. Параметр `validation_split` резервує частину навчальних даних для валідації, що дозволяє нам контролювати роботу моделі на невидимих даних під час навчання.

Перенавчання та як його уникнути.

Перенавчання відбувається, коли модель вивчає закономірності, характерні для навчальних даних, які не узагальнюються на нові дані. Ось способи, як уникнути перенавчання:

1. Валідаційний набір: Використання валідаційного набору (як ми зробили в навчальному коді) допомагає контролювати роботу моделі на непередбачуваних даних.

2. Рання зупинка: Ця техніка зупиняє навчання, коли продуктивність моделі на тестовому наборі починає погіршуватися. Реалізувати ранню зупинку у Keras дуже просто:

```
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10)
history = model.fit(X_train, y_train, epochs=100, batch_size=25, validation_split=0.2, callbacks=[early_stopping])
```

1. Тут `patience=10` означає, що навчання зупиниться, якщо втрата валідації не покращиться протягом 10 епох поспіль.

2. Методи регуляризації: Такі методи, як відсіювання та пакетна нормалізація, які вже включені в нашу модель, також допомагають зменшити надмірну пристосованість.

Необов'язково: Це більше зворотних дзвінків

```
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, TensorBoard,
CSVLogger

# Callback to save the model periodically
model_checkpoint = ModelCheckpoint('best_model.h5', save_best_only=True,
monitor='val_loss')

# Callback to reduce learning rate when a metric has stopped improving
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5)

# Callback for TensorBoard
tensorboard = TensorBoard(log_dir='./logs')

# Callback to log details to a CSV file
csv_logger = CSVLogger('training_log.csv')

# Combining all callbacks
callbacks_list = [early_stopping, model_checkpoint, reduce_lr, tensorboard,
csv_logger]

# Fit the model with the callbacks
history = model.fit(X_train, y_train, epochs=100, batch_size=25,
validation_split=0.2, callbacks=callbacks_list)
```

3.3.4 Оцінка роботи моделі.

Після навчання моделі наступним кроком є оцінка її продуктивності за допомогою тестового набору. Це дасть нам розуміння того, наскільки добре наша модель може узагальнювати нові, небачені дані.

Оцінювання за допомогою тестового набору.

Щоб оцінити модель, нам спочатку потрібно підготувати наші тестові дані `(X_test)` так само, як ми це робили для навчальних даних. Потім ми можемо використати функцію `evaluate` моделі:

```
# Convert X_test and y_test to Numpy arrays if they are not already
X_test = np.array(X_test)
y_test = np.array(y_test)

# Ensure X_test is reshaped similarly to how X_train was reshaped
# This depends on how you preprocessed the training data
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

# Now evaluate the model on the test data
test_loss = model.evaluate(X_test, y_test)
print("Test Loss: ", test_loss)
```

Показники ефективності.

На додаток до втрат, інші метрики можуть надати більше інформації про продуктивність моделі. Для регресійних задач, подібних до нашої, найпоширеніші метрики включають:

1. Середня абсолютна похибка (MAE): Вимірює середню величину помилок у наборі прогнозів без урахування їхнього напрямку.

2. Середньоквадратична похибка (RMSE): Це квадратний корінь із середнього квадрата різниці між прогнозом і фактичним спостереженням.

Щоб розрахувати ці показники, ми можемо робити прогнози за допомогою нашої моделі і порівнювати їх з фактичними значеннями:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Making predictions
y_pred = model.predict(X_test)

# Calculating MAE and RMSE
mae = mean_absolute_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)

print("Mean Absolute Error: ", mae)
print("Root Mean Square Error: ", rmse)
```

І MAE, і RMSE є мірами точності прогнозування для регресійної моделі. Ось що вони показують:

MAE вимірює середню величину помилок у наборі прогнозів, не враховуючи їх напрямок. Це середнє значення для тестової вибірки абсолютних відмінностей між прогнозом і фактичним спостереженням, де всі індивідуальні відмінності мають однакову вагу. MAE 0,0724 означає, що в середньому передбачення моделі віддалені від фактичних значень приблизно на 0,0724 одиниці.

RMSE - це квадратичне правило підрахунку балів, яке також вимірює середню величину помилки. Це квадратний корінь із середнього квадрата різниці між прогнозом і фактичним спостереженням. RMSE надає відносно велику вагу великим помилкам. Це означає, що RMSE має бути більш корисним, коли великі помилки особливо небажані. RMSE 0,0753 означає, що прогнози моделі в середньому на 0,0753 одиниці віддалені від фактичних значень, коли великі помилки караються сильніше.

Ці метрики допомагають зрозуміти, наскільки точною є ваша модель і де вона потребує вдосконалення. Аналізуючи ці показники, можна приймати

обґрунтовані рішення щодо подальшого налаштування моделі або зміни підходу.

3.3.5 Прогнозування наступних 4 періодів.

Після навчання та оцінки нашої моделі LSTM з механізмом уваги, останнім кроком буде використання її для прогнозування наступних 4 періодів (днів) цін на акції AAPL.

Здійснення прогнозів.

Щоб спрогнозувати майбутні ціни акцій, нам потрібно надати моделі найсвіжіші дані. Припустимо, у нас є дані за останні 60 днів, підготовлені в тому ж форматі, що і `X_train`:, і ми хочемо передбачити ціну на наступний день:

```
import yfinance as yf
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Fetching the latest 60 days of AAPL stock data
data = yf.download('AAPL', period='60d', interval='1d')

# Selecting the 'Close' price and converting to numpy array
closing_prices = data['Close'].values

# Scaling the data
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(closing_prices.reshape(-1,1))

# Since we need the last 60 days to predict the next day, we reshape the data
accordingly
X_latest = np.array([scaled_data[-60:].reshape(60)])


# Reshaping the data for the model (adding batch dimension)
X_latest = np.reshape(X_latest, (X_latest.shape[0], X_latest.shape[1], 1))

# Making predictions for the next 4 candles
predicted_stock_price = model.predict(X_latest)
predicted_stock_price = scaler.inverse_transform(predicted_stock_price)

print("Predicted Stock Prices for the next 4 days: ", predicted_stock_price)
```

Далі спрогнозуємо ціну на наступні 4 дні:

```
import yfinance as yf
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Fetch the latest 60 days of AAPL stock data
data = yf.download('AAPL', period='60d', interval='1d')

# Select 'Close' price and scale it
closing_prices = data['Close'].values.reshape(-1, 1)
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(closing_prices)

# Predict the next 4 days iteratively
predicted_prices = []
current_batch = scaled_data[-60:].reshape(1, 60, 1)  # Most recent 60 days

for i in range(4):  # Predicting 4 days
    # Get the prediction (next day)
    next_prediction = model.predict(current_batch)

    # Reshape the prediction to fit the batch dimension
    next_prediction_reshaped = next_prediction.reshape(1, 1, 1)

    # Append the prediction to the batch used for predicting
    current_batch = np.append(current_batch[:, 1:, :],
next_prediction_reshaped, axis=1)

    # Inverse transform the prediction to the original price scale
    predicted_prices.append(scaler.inverse_transform(next_prediction)[0, 0])

print("Predicted Stock Prices for the next 4 days: ", predicted_prices)
```

Візуалізація прогнозів.

Візуальне порівняння прогнозованих значень з фактичними цінами акцій може бути дуже корисним. Нижче наведено код для побудови графіків прогнозованих цін на акції проти фактичних даних:

```
!pip install mplfinance -qqq
import pandas as pd
import mplfinance as mpf
import matplotlib.dates as mpl_dates
import matplotlib.pyplot as plt

# Assuming 'data' is your DataFrame with the fetched AAPL stock data
# Make sure it contains Open, High, Low, Close, and Volume columns

# Creating a list of dates for the predictions
last_date = data.index[-1]
next_day = last_date + pd.Timedelta(days=1)
prediction_dates = pd.date_range(start=next_day, periods=4)

# Assuming 'predicted_prices' is your list of predicted prices for the next 4
days
predictions_df = pd.DataFrame(index=prediction_dates, data=predicted_prices,
columns=['Close'])

# Plotting the actual data with mplfinance
mpf.plot(data, type='candle', style='charles', volume=True)

# Overlaying the predicted data
plt.figure(figsize=(10,6))
plt.plot(predictions_df.index, predictions_df['Close'], linestyle='dashed',
marker='o', color='red')

plt.title("AAPL Stock Price with Predicted Next 4 Days")
plt.show()
```

Фінальна візуалізація для прогнозів:

```
# Creating a list of dates for the predictions
last_date = data.index[-1]
next_day = last_date + pd.Timedelta(days=1)
prediction_dates = pd.date_range(start=next_day, periods=4)

# Adding predictions to the DataFrame
predicted_data = pd.DataFrame(index=prediction_dates, data=predicted_prices,
columns=['Close'])

# Combining both actual and predicted data
combined_data = pd.concat([data['Close'], predicted_data['Close']])
combined_data = combined_data[-64:] # Last 60 days of actual data + 4 days of
predictions

# Plotting the actual data
plt.figure(figsize=(10,6))
plt.plot(data.index[-60:], data['Close'][-60:], linestyle='-', marker='o',
color='blue', label='Actual Data')

# Plotting the predicted data
plt.plot(prediction_dates, predicted_prices, linestyle='-', marker='o',
color='red', label='Predicted Data')

plt.title("AAPL Stock Price: Last 60 Days and Next 4 Days Predicted")
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```

Було розглянуто складну задачу використання LSTM-мереж з механізмом уваги для прогнозування цін на акції, зокрема для компанії Apple Inc. (AAPL). Ключові моменти включають:

- Здатність LSTM вловлювати довгострокові залежності в даних часових рядів.
- Додаткова перевага механізму уваги у фокусуванні на релевантних точках даних.
- Детальний процес побудови, навчання та оцінки LSTM-моделі.

Хоча LSTM-моделі з увагою є потужними, вони мають обмеження:

- Припущення, що історичні моделі повторюватимуться подібним чином, може бути проблематичним, особливо на волатильних ринках.
- Зовнішні фактори, такі як ринкові новини та глобальні події, не відображені в історичних цінових даних, можуть суттєво впливати на ціни акцій.

3.3 Висновок

У розділі було обґрунтовано вибір мови програмування Python, середовища розробки GoogleColab та спеціалізованих бібліотек Keras, NumPy, Pandas та Matplotlib для програмної реалізації програмного модуля нейромережевого прогнозування часових рядів. Проведено аналіз набору даних часового ряду для навчання та тестування модуля. Описано основні етапи програмної реалізація та функціонування програмного модуля нейромережевого прогнозування часових рядів, що складається з завантаження бібліотек, завантаження набору даних, попередньої обробки та підготовки даних, побудови моделі LSTM, навчання моделі, оцінювання роботи моделі, прогнозування та візуалізація прогнозів.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМНОГО МОДУЛЯ НЕЙРОМЕРЕЖЕВОГО ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ

4.1 Тестування програмного модуля нейромережевого прогнозування часових рядів

Розроблений програмний модуль є консольною програмою, не має інтерфейсу користувача і просто виконує послідовно всі дії, що в ньому запрограмовані та виводить на екран поьрібну інформацію.

Після запуску програми, вона імпортує бібліотеки і перевіряє версію TensorFlow, щоб переконатися, що все актуально та виводить повідомлення:

```
TensorFlow Version:  2.15.0
```

Потім програма завантажує історичні дані з Yfinance [8]. Відповідний скрипт отримує щоденні ціни на акції Apple Inc. з 1 січня 2020 року по 1 січня 2024 року та виводить кілька перших рядків цих даних, як показано на рис. 4.1.

```
[**********100%%**********]  1 of 1 completed
```

Date	Open	High	Low	Close	Adj Close	Volume
2020-01-02	74.059998	75.150002	73.797501	75.087502	73.059433	135480400
2020-01-03	74.287498	75.144997	74.125000	74.357498	72.349136	146322800
2020-01-06	73.447502	74.989998	73.187500	74.949997	72.925636	118387200
2020-01-07	74.959999	75.224998	74.370003	74.597504	72.582672	108872000
2020-01-08	74.290001	76.110001	74.290001	75.797501	73.750237	132079200

Рисунок 4.1 – Кілька перших рядків даних щоденних цін на акції Apple Inc. з 1 січня 2020 року

Після того, як дані отримано, виконується їх попередня обробка та відбір ознак. Далі будується LSTM-модель з шаром уваги, щоб робити прогнози.

Наша LSTM-модель складатиметься з декількох шарів, включаючи шари LSTM для обробки даних часових рядів. Програма виводить короткий опис моделі, який дозволяє зрозуміти її структуру та кількість параметрів (рис.4.2).

Layer (type)	Output Shape	Param #
lstm_3 (LSTM)	(None, 60, 50)	10400
lstm_4 (LSTM)	(None, 60, 50)	20200
permute (Permute)	(None, 50, 60)	0
reshape (Reshape)	(None, 50, 60)	0
permute_1 (Permute)	(None, 60, 50)	0
reshape_1 (Reshape)	(None, 60, 50)	0
flatten (Flatten)	(None, 3000)	0
dense_1 (Dense)	(None, 1)	3001
dropout (Dropout)	(None, 1)	0
batch_normalization (Batch Normalization)	(None, 1)	4

Total params: 33605 (131.27 KB)
Trainable params: 33603 (131.26 KB)
Non-trainable params: 2 (8.00 Byte)

Рисунок 4.2 – Структура та параметри нейромережі LSTM

Після того як наша LSTM-модель з увагою побудована, вона навчається за допомогою підготовленого навчального набору. В процесі навчання виводяться результати по кожній епосі навчання як показано на рис. 4.3.

```
Epoch 1/100
25/25 [==============================] - 24s 345ms/step - loss: 0.0444 - val_loss: 0.0066
Epoch 2/100
25/25 [==============================] - 5s 200ms/step - loss: 0.0043 - val_loss: 0.0026
Epoch 3/100
25/25 [==============================] - 5s 216ms/step - loss: 0.0030 - val_loss: 0.0033
Epoch 4/100
25/25 [==============================] - 4s 171ms/step - loss: 0.0026 - val_loss: 0.0036
Epoch 5/100
25/25 [==============================] - 4s 155ms/step - loss: 0.0029 - val_loss: 0.0036
Epoch 6/100
25/25 [==============================] - 4s 144ms/step - loss: 0.0026 - val_loss: 0.0029
```

……..

```
Epoch 96/100
25/25 [==============================] - 3s 114ms/step - loss: 4.1394e-04 - val_loss: 5.4733e-04
Epoch 97/100
25/25 [==============================] - 3s 118ms/step - loss: 4.6038e-04 - val_loss: 4.8601e-04
Epoch 98/100
25/25 [==============================] - 2s 77ms/step - loss: 4.4794e-04 - val_loss: 6.0891e-04
Epoch 99/100
25/25 [==============================] - 2s 86ms/step - loss: 4.1037e-04 - val_loss: 5.4728e-04
Epoch 100/100
25/25 [==============================] - 2s 80ms/step - loss: 4.2665e-04 - val_loss: 4.8331e-04
```

Рисунок 4.3 – Відображення процесу навчання, що виводиться модулем по кожній епосі навчання

Після навчання моделі наступним кроком є оцінка її продуктивності за допомогою тестового набору. Це дасть нам розуміння того, наскільки добре наша модель може узагальнювати нові, небачені дані. Програма виводить наступне повідомлення про оцінювання за допомогою тестового набору.

```
6/6 [==============================] - 0s 6ms/step - loss: 0.0057
Test Loss:  0.0056685502640903
```

А також виводить значення метрик

```
6/6 [==============================] - 1s 17ms/step
Mean Absolute Error:  0.06713882717183005
Root Mean Square Error:  0.07013579884276223
```

Виведенні дані означають, що:

- Середня абсолютна похибка (MAE): 0,0724 (приблизно)

- Середньоквадратична похибка (RMSE): 0,0753 (приблизно)

Після цього программа використовується для прогнозування наступних 4 періодів (днів) цін на акції AAPL та виводить ці прогнозовані значення:

```
[**********100%%**********]  1 of 1 completed1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step

1/1 [==============================] - 0s 18ms/step
Predicted Stock Prices for the next 4 days:  [172.72774, 172.89891, 173.70224, 174.45323]
```

Потім програма візуалізує прогнози. На рис.4.4 показано побудований прогноз на 4 дні.

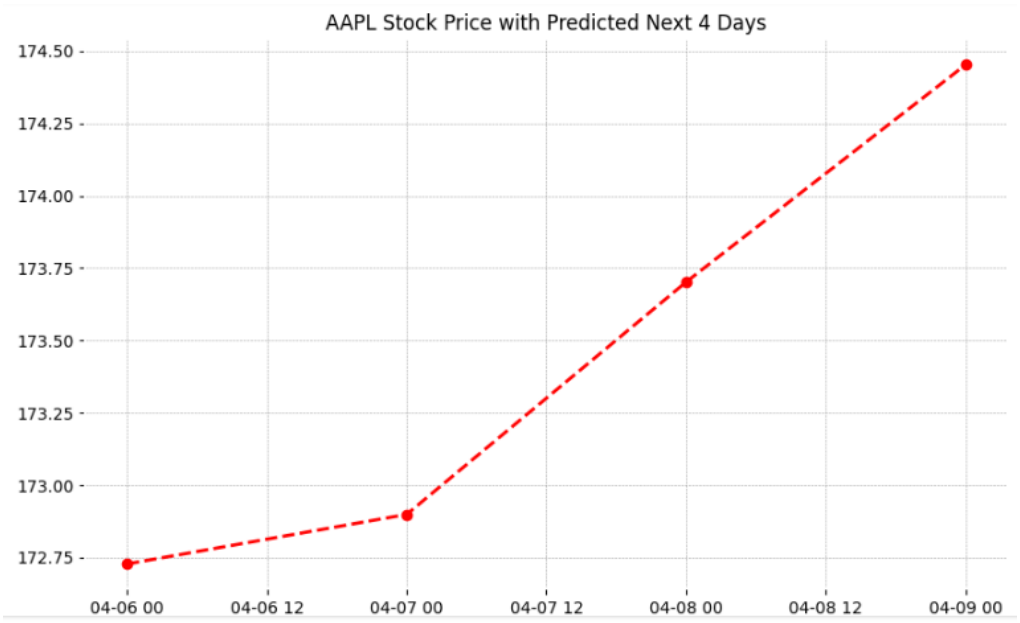


Рисунок 4.4 – Побудований прогноз на 4 дні

Візуальне порівняння прогнозованих значень з фактичними цінами акцій може бути дуже корисним. На рис. 4.5 наведено графік прогнозованих цін на акції проти фактичних даних.

Рисунок 4.5 – Графік прогнозованих цін на акції проти фактичних даних

Після цього програма будує фінальну візуалізацію для прогнозів, яка представлена на рис. 4.6.

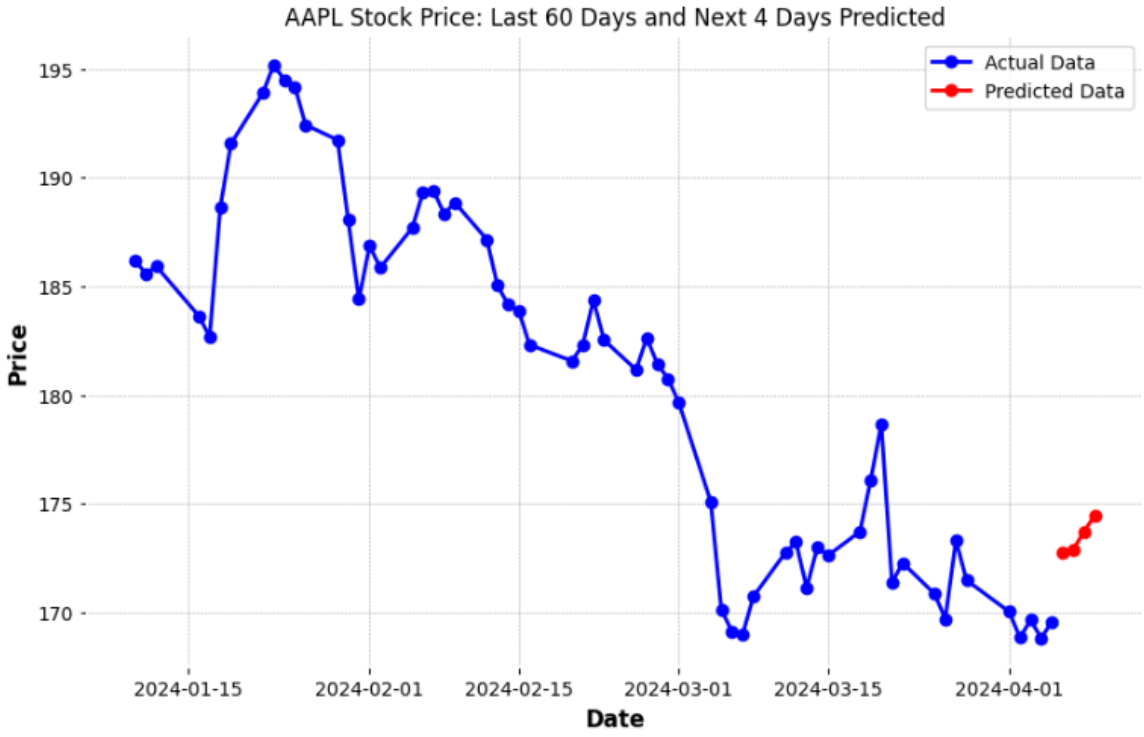
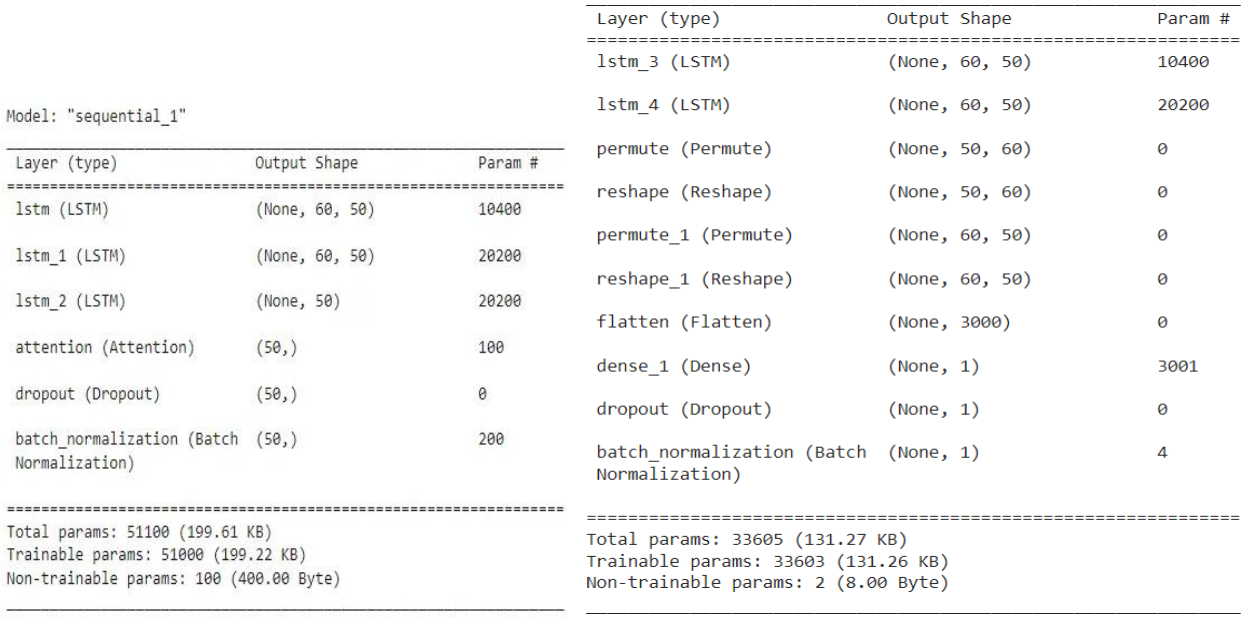


Рисунок 4.6 – Фінальна візуалізація для прогнозів

4.2 Аналіз результатів роботи програмного модуля нейромережевого прогнозування часових рядів

Для оцінки ефективності прогнозування розробленого модуля було порівняно результати його роботи з іншою структурою нейромережі LSTM. Інша структура нейромережі LSTM також була промодельована у цьому ж програмному модулі і на тому самому наборі даних, що дозволяє говорити про адекватність порівняння двох архітектур LSTM нейромереж.

На рис. 4.7 представлено структуру LSTM нейромережі-аналога та розробленої LSTM нейромережі.



Model: "sequential_1"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 60, 50)	10400
lstm_1 (LSTM)	(None, 60, 50)	20200
lstm_2 (LSTM)	(None, 50)	20200
attention (Attention)	(50,)	100
dropout (Dropout)	(50,)	0
batch_normalization (Batch Normalization)	(50,)	200

Total params: 51100 (199.61 KB)
Trainable params: 51000 (199.22 KB)
Non-trainable params: 100 (400.00 Byte)

Layer (type)	Output Shape	Param #
lstm_3 (LSTM)	(None, 60, 50)	10400
lstm_4 (LSTM)	(None, 60, 50)	20200
permute (Permute)	(None, 50, 60)	0
reshape (Reshape)	(None, 50, 60)	0
permute_1 (Permute)	(None, 60, 50)	0
reshape_1 (Reshape)	(None, 60, 50)	0
flatten (Flatten)	(None, 3000)	0
dense_1 (Dense)	(None, 1)	3001
dropout (Dropout)	(None, 1)	0
batch_normalization (Batch Normalization)	(None, 1)	4

Total params: 33605 (131.27 KB)
Trainable params: 33603 (131.26 KB)
Non-trainable params: 2 (8.00 Byte)

Рисунок 4.7 – Структура LSTM нейромережі-аналога (ліворуч) та розробленої LSTM нейромережі (праворуч)

Із рис. 4.7 видно, що структури цих LSTM нейромереж відрізняються номенклатурою і параметрами шарів, з яких вони побудовані.

Результати перевірки точності прогнозування на тестовій вибірці обох структур LSTM нейромереж (аналога та розробленої LSTM нейромережі) представлено на рис. 4.8.

Аналог:

```
6/6 [==============================] - 1s 9ms/step
Mean Absolute Error:  0.07238591910513067
Root Mean Square Error:  0.07528977958017959
```

Розроблена структура:

```
6/6 [==============================] - 1s 17ms/step
Mean Absolute Error:  0.06713882717183005
Root Mean Square Error:  0.07013579884276223
```

Рисунок 4.8 – Результати перевірки точності прогнозування на тестовій вибірці LSTM нейромережі-аналога (зверху) та розробленої LSTM нейромережі (знизу)

Трохи відрізняються і прогнози, що видають обидві цих LSTM нейромережі. На рис. 4.9 показано які прогнози на 4 дні видає LSTM нейромережа-аналог (зверху) та розроблена LSTM нейромережа (знизу).

Аналог:

```
[**********100%%**********]  1 of 1 completed1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step

1/1 [==============================] - 0s 18ms/step
Predicted Stock Prices for the next 4 days:  [172.72774, 172.89891, 173.70224, 174.45323]
```

Розроблена структура:

```
[**********100%%**********]  1 of 1 completed
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 34ms/step
Predicted Stock Prices for the next 4 days:  [172.07568, 172.70047, 173.35123, 173.74968]
```

Рисунок 4.9 – Результати прогнозу на 4 дні, які видаються LSTM нейромережею-аналогом (зверху) та розробленою LSTM нейромережею (знизу).

Показники похибок з рис.4.8 та розраховані значення виграшу по точності представлено у табл. 4.1

Таблиця 4.1 – Порівняння похибок прогнозування розробленої LSTM нейромережі та аналога

Вид похибки	Аналог	Розроблена структура	Виграш (%)
Середня абсолютна похибка (MAE)	0,072386	0,067139	7,8
Середньоквадратична похибка (RMSE)	0,075290	0,070136	6,9

Виграш по середній абсолютній похибці (MAE) розраховується так: ((0,072386-0,067139)/ 0,072386)*100%==7,82 %

Аналогічно розраховується і виграш по середньоквадратичній похибці (RMSE: ((0,075290-0,067139)/ 0,075290)*100%==6,85 %

Із таблиці 4.1 можна зробити висновок, що розроблена LSTM нейромережа є більш точною у прогнозуванні ціни акцій AAPL, ніж LSTM нейромережа-аналог. Причому, збільшення точності по середній абсолютній похибці (MAE) складає 7,8%, а по середньоквадратичній похибці (RMSE) – 6,9%.

Таким чином, розроблений нейромережевий модуль прогнозування часових рядів має підвищену на 7-8% точність прогнозування порівняно з аналогом. Тобто мета роботи досягнута – точність прогнозування часових рядів підвищена.

4.3 Висновок

У розділі в результаті тестування програми було доведено її працездатність та відповідність поставленому завданню. Навчання модуля відбувалось з використанням набору історичні даних з Yfinance з щоденними цінами на акції Apple Inc. Набір даних було поділено на навчальну (80%) та тестову (20%) вибірки. Загальний обсяг вибірки складає 1826 записів (5 років). Розроблена LSTM нейромережа є більш точною у прогнозуванні ціни акцій AAPL, ніж LSTM нейромережа-аналог. Причому, збільшення точності по середній абсолютній похибці (MAE) складає 7,8%, а по середньоквадратичній похибці (RMSE) – 6,9%. Таким чином, розроблений нейромережевий модуль прогнозування часових рядів має підвищену на 7-8% точність прогнозування порівняно з аналогом. Тобто мета роботи досягнута – точність прогнозування часових рядів підвищена.

ВИСНОВКИ

У роботі було розглянуто задачу прогнозування часових рядів на основі нейромереж. В ході аналізу предметної області було описано детальну постановку задачі прогнозування часових рядів. Розглянуто різні методи розв'язання задачі прогнозування часових рядів і оцінюються їх переваги та недоліки. Серед таких методів прогнозування часових рядів як прогнозування на основі ковзаючого середнього, експоненціальне згладжування, ARIMA (Autoregressive Integrated Moving Average), прогнозування на основі нейронних мереж, прогнозування на основі екстремальних значень, прогнозування на основі методу Монте-Карло для реалізації у цій роботі було обрано метод на основі штучних нейронних мереж як найбільш перспективний і застосовний до даної задачі. Крім цього, було проаналізовано програми-аналоги розроблюваного модуля, їх переваги та недоліки..

В процесі розробки було обґрунтовано вибір типу нейронної мережі для програмного модуля нейромережевого прогнозування часових рядів. Було обрано архітектуру нейронної мережі LSTM з механізмом уваги та детально проаналізовано її математичну модель та принципи функціонування. Для навчання цієї нейромережі використовується модифікація ADAM методу стохастичного градієнтного спуска. Розроблено алгоритм роботи програмного модуля нейромережевого прогнозування часових рядів.

В процесі програмної реалізації модуля нейромережевого прогнозування часових рядів було обґрунтовано вибір мови програмування Python, середовища розробки GoogleColab та спеціалізованих бібліотек Keras, NumPy, Pandas та Matplotlib. Проведено аналіз набору даних часового ряду для навчання та тестування модуля. Описано основні етапи програмної реалізація та функціонування програмного модуля нейромережевого прогнозування часових рядів, що складається з завантаження бібліотек, завантаження набору даних, попередньої обробки та підготовки даних, побудови моделі LSTM,

навчання моделі, оцінювання роботи моделі, прогнозування та візуалізація прогнозів.

В результаті тестування програми було доведено її працездатність та відповідність поставленому завданню. Навчання модуля відбувалось з використанням набору історичні даних з Yfinance з щоденними цінами на акції Apple Inc. Набір даних було поділено на навчальну (80%) та тестову (20%) вибірки. Загальний обсяг вибірки складає 1826 записів (5 років). Розроблена LSTM нейромережа є більш точною у прогнозуванні ціни акцій AAPL, ніж LSTM нейромережа-аналог. Причому, збільшення точності по середній абсолютній похибці (MAE) складає 7,8%, а по середньоквадратичній похибці (RMSE) – 6,9%. Таким чином, розроблений нейромережевий модуль прогнозування часових рядів має підвищену на 7-8% точність прогнозування порівняно з аналогом. Тобто мета роботи досягнута – точність прогнозування часових рядів підвищена.

..

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Аналіз та прогнозування часових рядів [Електронний ресурс]. - Режим доступу: https://drukarnia.com.ua/articles/analiz-ta-prognozuvannya-chasovikh-ryadiv-H6gg0

2 А. Т. Яровий, Є. М. Страхов АНАЛІЗ ЧАСОВИХ РЯДІВ / Навчально-методичний посібник для студентів математичних та економічних спеціальностей, Освіта України, 2019, Одеса.

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

4 О.М. Колодчак Інтелектуальний аналіз даних Lviv Polytechnic National University Institutional Repository [Електронний ресурс] – Режим доступу: http://ena.lp.edu.ua/bitstream/ntb/26402/1/10_49-58.pdf

5 Рекурентна нейронна мережа для обробки великих текстових даних [Електронний ресурс]. - Режим доступу: https://journals.nupp.edu.ua/sunz/article/view/1214

6 Застосування глибокої рекурентної нейронної мережі із використанням алгоритму LSTM у системах інтелектуальної взаємодії [Електронний ресурс]. - Режим доступу: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/22462/32.%20Застосування%20глибокої%20рекурентної%20нейронної%20мережі%20із%20використанням%20алгоритму%20LSTM%20у%20системах%20інтелектуальної%20взаємодії..pdf

7 Long Short-Term Memory [Електронний ресурс]. – Режим доступу: https://medium.com/@saba99/long-short-term-memory-lstm-fffc5eaebfdc

8 LSTM_plus_Attn_Adv.ipynb [Електронний ресурс]. - Режим доступу: https://colab.research.google.com/drive/1xdL4_9hkC5sqmM_YPWkbvrcL06_XA3dX?usp=sharing

9 Python [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Python

10 Google Colab або Google Colaboratory: що це таке [Електронний ресурс]. - Режим доступу: https://www.hwlibre.com/uk співробітництво-Google/

11 Colaboratory [Електронний ресурс]. – Режим доступу: https://research.google.com/colaboratory/faq.html#:~:text=Colab%20is%20a%20hosted%20Jupyter,%2C%20data%20science%2C%20and%20education.

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

13 NumPy [Електронний ресурс] – Режим доступу: https://numpy.org/

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

15 Matplotlib: Visualization with Python [Електронний ресурс] – Режим доступу: https://matplotlib.org/

16 Apple [Електронний ресурс]. - Режим доступу: https://www.google.com/finance/quote/AAPL:NASDAQ?sa=X&ved=2ahUKEwi_-suiq92GAxXgSvEDHRW6BhQQ3ecFegQIOBAZ

17 Дидерик П. Кінгма і Джиммі Лей Ба. Адам: метод стохастичної оптимізації, 2014. arXiv: 1412.6980v9

18 Методичні вказівки до виконання бакалаврської кваліфікаційної роботи для студентів денної та заочної форм навчання спеціальності 122 - «Комп’ютерні науки» / Укладачі А.А.Яровий, О.В.Сілагін, І.В.Богач. – Вінниця: ВНТУ, 2022. – 47 с.

Додаток А (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Програмний модуль нейромережевого прогнозування часових рядів

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

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

Показники звіту подібності Unicheck

Оригінальність 94,4% Схожість 5,6%

Аналіз звіту подібності (відмітити потрібне):

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

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

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

Особа, відповідальна за перевірку Озеранський В.С.

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

Автор роботи Калінович Р.О.

Керівник роботи Колесницький О.К.

Додаток Б (обов'язковий)

Лістинг програми

```
!pip install tensorflow -qqq
!pip install keras -qqq
!pip install yfinance -qqq
```

```
import tensorflow as tf
import keras
import yfinance as yf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


# Check TensorFlow version
print("TensorFlow Version: ", tf.__version__)

# Fetch AAPL data
aapl_data = yf.download('AAPL', start='2020-01-01', end='2024-01-01')

# Display the first few rows of the dataframe
aapl_data.head()

# Checking for missing values
aapl_data.isnull().sum()

# Filling missing values, if any
aapl_data.fillna(method='ffill', inplace=True)

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0,1))
aapl_data_scaled = scaler.fit_transform(aapl_data['Close'].values.reshape(-
1,1))

X = []
y = []

for i in range(60, len(aapl_data_scaled)):
    X.append(aapl_data_scaled[i-60:i, 0])
    y.append(aapl_data_scaled[i, 0])

train_size = int(len(X) * 0.8)
test_size = len(X) - train_size

X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

X_train, y_train = np.array(X_train), np.array(y_train)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))


from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout, AdditiveAttention, Permute,
Reshape, Multiply
```

```
model = Sequential()

# Adding LSTM layers with return_sequences=True
model.add(LSTM(units=50, return_sequences=True,
input_shape=(X_train.shape[1], 1)))
model.add(LSTM(units=50, return_sequences=True))


# Adding self-attention mechanism
# The attention mechanism
attention = AdditiveAttention(name='attention_weight')
# Permute and reshape for compatibility
model.add(Permute((2, 1)))
model.add(Reshape((-1, X_train.shape[1])))
attention_result = attention([model.output, model.output])
multiply_layer = Multiply()([model.output, attention_result])
# Return to original shape
model.add(Permute((2, 1)))
model.add(Reshape((-1, 50)))

# Adding a Flatten layer before the final Dense layer
model.add(tf.keras.layers.Flatten())

# Final Dense layer
model.add(Dense(1))

# Compile the model
# model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
# history = model.fit(X_train, y_train, epochs=100, batch_size=25,
validation_split=0.2)


from keras.layers import BatchNormalization

# Adding Dropout and Batch Normalization
model.add(Dropout(0.2))
model.add(BatchNormalization())

model.compile(optimizer='adam', loss='mean_squared_error')

model.summary()

# Assuming X_train and y_train are already defined and preprocessed
history    =    model.fit(X_train,    y_train,    epochs=100,    batch_size=25,
validation_split=0.2)

from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10)
history    =    model.fit(X_train,    y_train,    epochs=100,    batch_size=25,
validation_split=0.2, callbacks=[early_stopping])


from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, TensorBoard,
CSVLogger

# Callback to save the model periodically
model_checkpoint = ModelCheckpoint('best_model.h5', save_best_only=True,
```

```
monitor='val_loss')

# Callback to reduce learning rate when a metric has stopped improving
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5)

# Callback for TensorBoard
tensorboard = TensorBoard(log_dir='./logs')

# Callback to log details to a CSV file
csv_logger = CSVLogger('training_log.csv')

# Combining all callbacks
callbacks_list = [early_stopping, model_checkpoint, reduce_lr, tensorboard,
csv_logger]

# Fit the model with the callbacks
history = model.fit(X_train, y_train, epochs=100, batch_size=25,
validation_split=0.2, callbacks=callbacks_list)

# Convert X_test and y_test to Numpy arrays if they are not already
X_test = np.array(X_test)
y_test = np.array(y_test)

# Ensure X_test is reshaped similarly to how X_train was reshaped
# This depends on how you preprocessed the training data
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

# Now evaluate the model on the test data
test_loss = model.evaluate(X_test, y_test)
print("Test Loss: ", test_loss)



from sklearn.metrics import mean_absolute_error, mean_squared_error

# Making predictions
y_pred = model.predict(X_test)

# Calculating MAE and RMSE
mae = mean_absolute_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)

print("Mean Absolute Error: ", mae)
print("Root Mean Square Error: ", rmse)

import yfinance as yf
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Fetching the latest 60 days of AAPL stock data
data = yf.download('AAPL', period='60d', interval='1d')

# Selecting the 'Close' price and converting to numpy array
closing_prices = data['Close'].values

# Scaling the data
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(closing_prices.reshape(-1,1))

# Since we need the last 60 days to predict the next day, we reshape the data
```

```
accordingly
X_latest = np.array([scaled_data[-60:].reshape(60)])

# Reshaping the data for the model (adding batch dimension)
X_latest = np.reshape(X_latest, (X_latest.shape[0], X_latest.shape[1], 1))

# Making predictions for the next 4 candles
predicted_stock_price = model.predict(X_latest)
predicted_stock_price = scaler.inverse_transform(predicted_stock_price)

print("Predicted Stock Prices for the next 4 days: ", predicted_stock_price)



import yfinance as yf
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Fetch the latest 60 days of AAPL stock data
data = yf.download('AAPL', period='60d', interval='1d')

# Select 'Close' price and scale it
closing_prices = data['Close'].values.reshape(-1, 1)
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(closing_prices)

# Predict the next 4 days iteratively
predicted_prices = []
current_batch = scaled_data[-60:].reshape(1, 60, 1)  # Most recent 60 days

for i in range(4):  # Predicting 4 days
    # Get the prediction (next day)
    next_prediction = model.predict(current_batch)

    # Reshape the prediction to fit the batch dimension
    next_prediction_reshaped = next_prediction.reshape(1, 1, 1)

    # Append the prediction to the batch used for predicting
    current_batch = np.append(current_batch[:, 1:, :],
next_prediction_reshaped, axis=1)

    # Inverse transform the prediction to the original price scale
    predicted_prices.append(scaler.inverse_transform(next_prediction)[0, 0])

print("Predicted Stock Prices for the next 4 days: ", predicted_prices)



!pip install mplfinance -qqq
import pandas as pd
import mplfinance as mpf
import matplotlib.dates as mpl_dates
import matplotlib.pyplot as plt

# Assuming 'data' is your DataFrame with the fetched AAPL stock data
# Make sure it contains Open, High, Low, Close, and Volume columns

# Creating a list of dates for the predictions
last_date = data.index[-1]
next_day = last_date + pd.Timedelta(days=1)
```

```
prediction_dates = pd.date_range(start=next_day, periods=4)

# Assuming 'predicted_prices' is your list of predicted prices for the next 4
days
predictions_df = pd.DataFrame(index=prediction_dates, data=predicted_prices,
columns=['Close'])

# Plotting the actual data with mplfinance
mpf.plot(data, type='candle', style='charles', volume=True)

# Overlaying the predicted data
plt.figure(figsize=(10,6))
plt.plot(predictions_df.index, predictions_df['Close'], linestyle='dashed',
marker='o', color='red')

plt.title("AAPL Stock Price with Predicted Next 4 Days")
plt.show()
```

```
import pandas as pd
import mplfinance as mpf
import matplotlib.dates as mpl_dates
import matplotlib.pyplot as plt

# Fetch the latest 60 days of AAPL stock data
data = yf.download('AAPL', period='64d', interval='1d') # Fetch 64 days to
display last 60 days in the chart

# Select 'Close' price and scale it
closing_prices = data['Close'].values.reshape(-1, 1)
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(closing_prices)

# Predict the next 4 days iteratively
predicted_prices = []
current_batch = scaled_data[-60:].reshape(1, 60, 1)  # Most recent 60 days

for i in range(4):  # Predicting 4 days
    next_prediction = model.predict(current_batch)
    next_prediction_reshaped = next_prediction.reshape(1, 1, 1)
    current_batch = np.append(current_batch[:, 1:, :],
next_prediction_reshaped, axis=1)
    predicted_prices.append(scaler.inverse_transform(next_prediction)[0, 0])

# Creating a list of dates for the predictions
last_date = data.index[-1]
next_day = last_date + pd.Timedelta(days=1)
prediction_dates = pd.date_range(start=next_day, periods=4)

# Adding predictions to the DataFrame
predicted_data = pd.DataFrame(index=prediction_dates, data=predicted_prices,
columns=['Close'])

# Combining both actual and predicted data
combined_data = pd.concat([data['Close'], predicted_data['Close']])
combined_data = combined_data[-64:] # Last 60 days of actual data + 4 days of
predictions

# Plotting the actual data
```

```
    plt.figure(figsize=(10,6))
    plt.plot(data.index[-60:], data['Close'][-60:], linestyle='-', marker='o',
    color='blue', label='Actual Data')

    # Plotting the predicted data
    plt.plot(prediction_dates, predicted_prices, linestyle='-', marker='o',
    color='red', label='Predicted Data')

    plt.title("AAPL Stock Price: Last 60 Days and Next 4 Days Predicted")
    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.legend()
    plt.show()
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

«ПРОГРАМНИЙ МОДУЛЬ НЕЙРОМЕРЕЖЕВОГО ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ»

Виконав: студент 4 курсу,групи 1КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Калінович Р.О.
(прізвище та ініціали)

Керівник: к.т.н., проф. каф.КН
Колесницький О.К.
(прізвище та ініціали)
«07» 06 2024 р.

Вінниця ВНТУ - 2023 рік

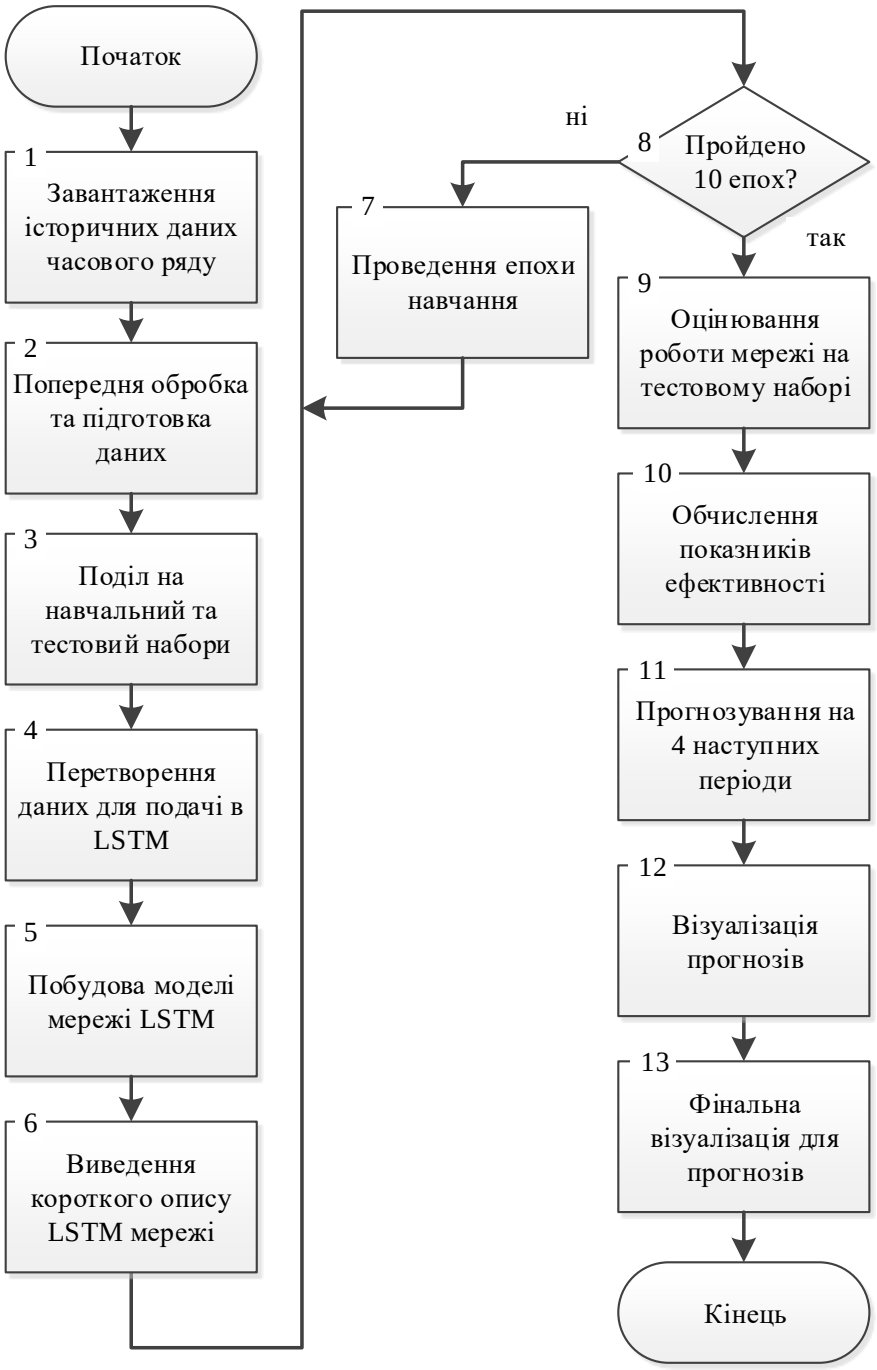


Рисунок В.1– Схема алгоритму роботи програмного модуля прогнозування часових рядів

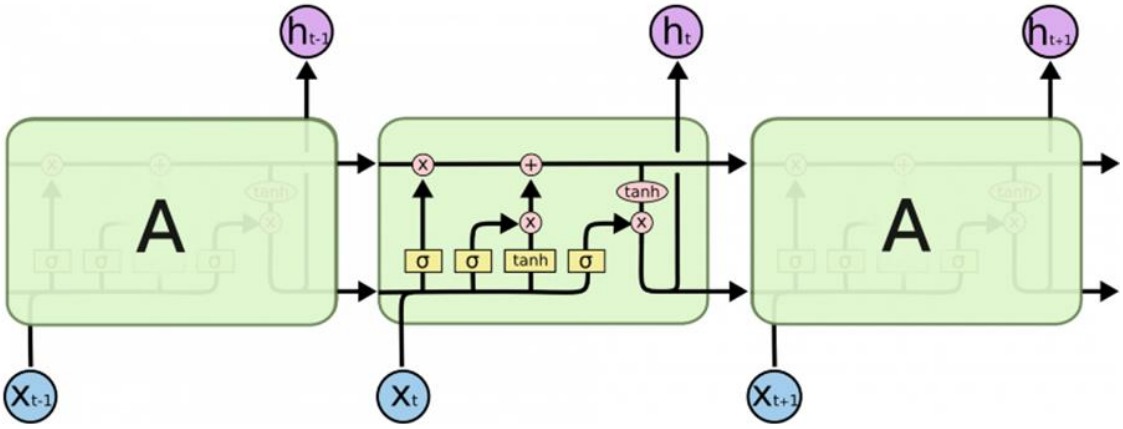


Рисунок В.2– Загальна структура LSTM мережі

Model: "sequential_1"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 60, 50)	10400
lstm_1 (LSTM)	(None, 60, 50)	20200
lstm_2 (LSTM)	(None, 50)	20200
attention (Attention)	(50,)	100
dropout (Dropout)	(50,)	0
batch_normalization (Batch Normalization)	(50,)	200

Total params: 51100 (199.61 KB)
Trainable params: 51000 (199.22 KB)
Non-trainable params: 100 (400.00 Byte)

Рисунок В.3 – Структура та параметри нейромережі LSTM.

```
[**********100%%**********]  1 of 1 completed
```

Date	Open	High	Low	Close	Adj Close	Volume
2020-01-02	74.059998	75.150002	73.797501	75.087502	73.059433	135480400
2020-01-03	74.287498	75.144997	74.125000	74.357498	72.349136	146322800
2020-01-06	73.447502	74.989998	73.187500	74.949997	72.925636	118387200
2020-01-07	74.959999	75.224998	74.370003	74.597504	72.582672	108872000
2020-01-08	74.290001	76.110001	74.290001	75.797501	73.750237	132079200

Рисунок В.4 – Фрагмент набору даних щоденних цін на акції Apple Inc. з 1 січня 2020 року

Рисунок В.5 – Графік прогнозованих цін на акції проти фактичних даних

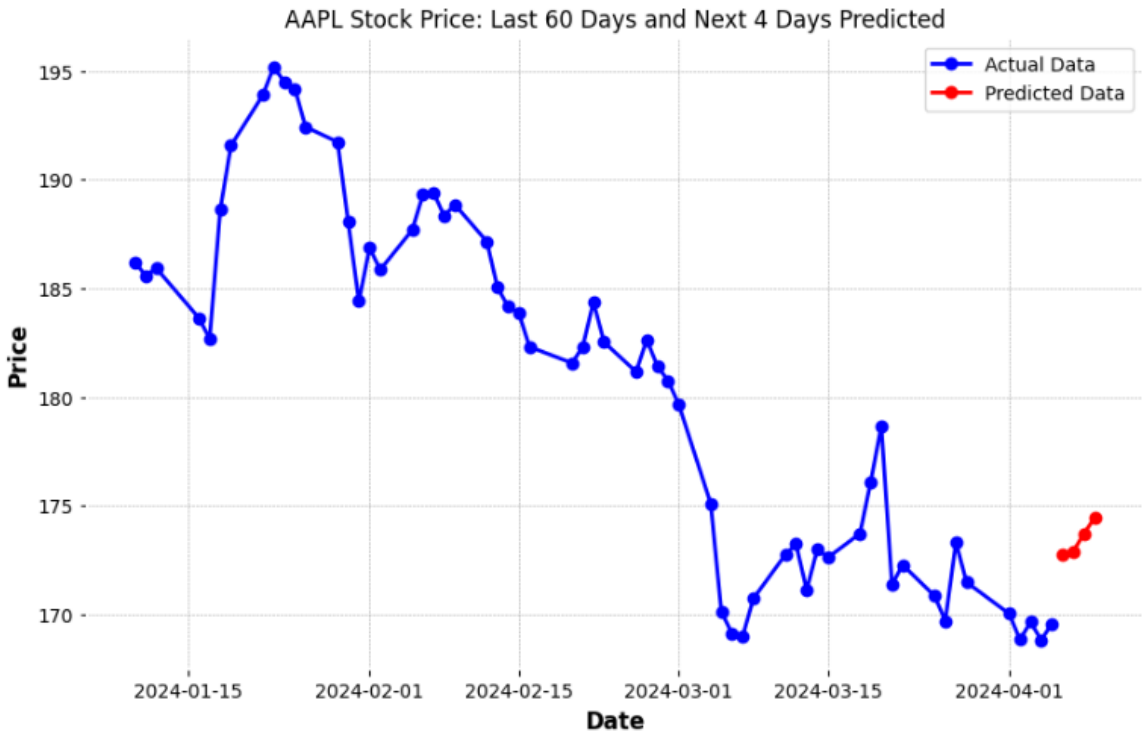


Рисунок В.6 – Фінальна візуалізація для прогнозів

Аналог:

```
6/6 [==============================] - 1s 9ms/step
Mean Absolute Error:  0.07238591910513067
Root Mean Square Error:  0.07528977958017959
```

Розроблена структура:

```
6/6 [==============================] - 1s 17ms/step
Mean Absolute Error:  0.06713882717183005
Root Mean Square Error:  0.07013579884276223
```

Рисунок В.7 – Результати перевірки точності прогнозування на тестовій вибірці LSTM нейромережі-аналога (зверху) та розробленої LSTM нейромережі (знизу)

Вид похибки	Аналог	Розроблена структура	Виграш (%)
Середня абсолютна похибка (MAE)	0,072386	0,067139	7,8
Середньоквадратична похибка (RMSE)	0,075290	0,070136	6,9

Рисунок В.8 – Порівняння похибок прогнозування розробленої LSTM нейромережі та аналога