НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ

Кафедра математичних методів системного аналізу

До захисту допущено:

Завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____2024 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

на тему: «Прогнозування часових рядів економічної природи. Порівняльний аналіз методів прогнозування»

Виконав:
студент IV курсу, групи КА-02
Ярко Андрій Юрійович _____

Керівник:
Доцент, к.т.н.
Селін Юрій Миколайович _____

Консультант з економічного розділу:
Професор кафедри ЕК, д.е.н.,
Семенченко Наталія Віталіївна _____

Консультант з нормоконтролю:
к.ф.-м.н. Статкевич Віталій Михайлович _____

Рецензент:
Професор кафедри ІСТ, д.т.н.
Корнієнко Богдан Ярославович _____

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

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 124 «Системний аналіз»
Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Оксана ТИМОЩУК

«__» травня 2024 р.

ЗАВДАННЯ
на дипломну роботу студенту
Ярку Андрію Юрійовичу

1. Тема роботи «Прогнозування часових рядів економічної природи. Порівняльний аналіз методів прогнозування», керівник роботи доцент, к.т.н. Селін Ю.М. затверджені наказом по університету від «31»травня 2024 р. №2240-с.
2. Термін подання студентом роботи 11.06.2024.
3. Вихідні дані до роботи: моделі прогнозування, часові ряди, що описують зміни акції ціни компанії
4. Зміст роботи: аналітичний огляд задач прогнозування економічних процесів, застосування авторегресій та нейронних мереж для побудови прогнозів, порівняльний аналіз різних методів прогнозування.
5. Перелік ілюстративного матеріалу: актуальність дослідження, постановка задачі, авторегресії, результати точностей, нейромережі, висновки.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Семенченко Н.В., професор кафедри ЕК		

7. Дата видачі завдання

Календарний план

/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
	Затвердження теми БДР. Ознайомлення зі структурою БДР згідно з Положення про випускну атестацію студентів КПІ ім. Ігоря Сікорського та з державним стандартом України ДСТУ 3008:2015 та стандарти ЄСПД	21.04.2024	виконано
	Проведений аналіз існуючих підходів, досліджена література, сформульовані етапи роботи, обраний за затвердженою темою та опрацьований набір даних під керівництво керівника БДР. Підготовлений перший розділ дослідження предметної області.	28.04.2024	виконано
	Робота над математичними основами роботи: алгоритмами МН в цілому, попередня обробка даних, критерії якості тощо.	05.05.2024	виконано
	Завершення роботи над другим розділом, сформований перший варіант основної частини БДР. Продовжена робота над експериментальною частиною і розробкою програмного продукту	12.05.2024	виконано
	Доопрацювання основної частини БДР, розробка програмного продукту, аналіз результатів	25.05.2024	виконано
	Написання функціонально-вартісного аналізу розробленої практичної частини, аналіз результатів, оформлення дипломної роботи	31.05.2024	виконано

Студент Андрій ЯРКО

Керівник Юрій СЕЛІН

РЕФЕРАТ

Дипломна робота: 119 с., 36 рис., 11 табл., 2 дод. та 15 джерел.

ПРОГНОЗУВАННЯ ФІНАНСОВИХ РИНКІВ, ЧАСОВІ РЯДИ, МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, ARIMA, RNN, LSTM.

Об’єкт дослідження – методи прогнозування цін фінансових інструментів з використанням сучасних алгоритмів машинного навчання.

Мета роботи – дослідження та застосування сучасних методів машинного навчання для прогнозування фінансових ринків, з метою підвищення точності прогнозів та ефективності прийняття економічних рішень.

Методи дослідження – аналіз та порівняння різних моделей прогнозування часових рядів (ARIMA, RNN, LSTM, GRU, WaveNet), попередня обробка даних (заповнення пропусків, масштабування), статистичний аналіз результатів.

Програмний продукт розроблено у середовищі Google Colaboratory мовою програмування Python.

Проведено порівняльний аналіз моделей, виявлено їх переваги та недоліки. Визначено кращу модель для нашої задачі. В роботі використовувалися дійсні дані часових рядів з сайту yahoo.finance.

ABSTRACT

Master's thesis: 119 p., 36 fig., 11 tables, 2 appendices,15 references.

FINANCIAL MARKET FORECASTING, TIME SERIES, MACHINE LEARNING, NEURAL NETWORKS, ARIMA, RNN, LSTM.

Object of the research – methods of forecasting the prices of financial instruments using modern machine learning algorithms.

The purpose of research is to investigate and apply modern machine learning methods for financial market forecasting to improve the accuracy of forecasts and the efficiency of economic decision-making.

Methods of research – analysis and comparison of various time series forecasting models (ARIMA, RNN, LSTM, GRU, WaveNet), data preprocessing (missing value imputation, scaling), and statistical analysis of results.

The software product was developed in Google Colaboratory using the Python programming language.

A comparative analysis of the models was conducted, identifying their advantages and disadvantages. The best model for our task was determined. Real time series data from the yahoo.finance website were used in the study.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ARMA – авторегресія з ковзним середнім

ARIMA – авторегресія з інтегрованим ковзним середнім

SARIMA – сезонна авторегресія з інтегрованим ковзним середнім

RNN – рекурентна нейронна мережа

CNN – згорткова нейронна мережа

LSTM – довга короткострокова пам’ять

ReLU – зрізаний лінійний вузол

GRU - вентильний рекурентний вузол

АКФ – автокореляційна функція

ЧАКФ – часткова автокореляційна функція

ВСТУП

Прогнозування економічних показників є важливою складовою частиною прийняття управлінських рішень в різних сферах діяльності. Точне прогнозування майбутніх значень економічних індикаторів дозволяє підприємствам планувати свою діяльність, державним органам розробляти ефективну економічну політику, а інвесторам приймати обґрунтовані рішення про вкладення коштів.

Для прогнозування використовують часові ряди, що представляють послідовність значень у часі. Складність в прогнозуванні полягає в тому, що зазвичай часові ряди економічної природи є нестаціонарними та залежать від зовнішніх факторів. Існують різноманітні методи прогнозування часових рядів. В даній роботі для аналізу були обрані методи основані на нейронних мережах. Для порівняння в даній роботі в якості базової моделі використовується ARIMA, яка є однією з традиційних моделей.

Об’єктом дослідження було обрано ринок акцій компанії Microsoft. Предметом дослідження є моделі прогнозування цін на акції Microsoft.

Метою дослідження є розробка моделей для прогнозування з застосуванням сучасних методів, таких як RNN, CNN та інших. Їх порівняльний аналіз на основі даних про продажі кукурудзи.

Для досягнення цієї мети поставлені наступні завдання:

1. Вивчення предметної області;
2. Огляд існуючих методів прогнозування;
3. Вибір критеріїв оцінки моделей;
4. Збір статистичних даних;
5. Розробка, навчання та тестування моделей;
6. Порівняльний аналіз результатів;

7. Формування висновків.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ЕКОНОМІЧНОГО ПРОГНОЗУВАННЯ

1.1 Актуальність економічного прогнозування

Прогнозування вартості інструментів фінансового ринку відіграє вагому роль при проведенні економічної політики держави, та становить інтерес для великих компаній при здійсненні економічної діяльності. Окремий інтерес до прогнозування вартості фінансових інструментів пов’язаний із можливістю спекулятивної торгівлі. Тому розробка нових та вдосконалення існуючих теоретико-методологічних підходів до прогнозування вартості інструментів фінансового ринку є актуальним питанням економічної науки [1].

Основні області застосування економічного прогнозування включають прогнозування економічного зростання, інфляції, безробіття, курсів валют, цін на товари та послуги, а також вивчення впливу різних економічних політик на економіку.

Завдяки економічному прогнозуванню компанії можуть розробляти стратегії для максимізації прибутку та мінімізації ризиків, уряди можуть приймати ефективні рішення щодо бюджету, податків та грошової політики, а індивідуальні споживачі можуть планувати свої фінанси і інвестиції.

1.2 Особливості економічного прогнозування

Економічне прогнозування – це сукупність наукових методик, які використовуються фахівцями для розробки оптимальних алгоритмів подальшого розвитку різноманітних сфер економіки [2]. Прогнози можуть здійснюватися на агрегованому рівні, наприклад, щодо ВВП, інфляції, безробіття або бюджетного дефіциту, або на більш дезагрегованому рівні, для окремих секторів економіки або навіть окремих фірм. Економічне прогнозування дозволяє визначити майбутню стратегію та модель інвестування і є ключовим фактором в економічному аналізі.

1.2.1 Мета економічного прогнозування

Економічне прогнозування використовується в різноманітних структурах та організаціях: національні уряди, банки та центральні банки, консультанти та суб'єкти приватного сектору, такі як аналітичні центри, компанії та міжнародні організації, наприклад, Міжнародний валютний фонд, Світовий банк і ОЕСР. Широкий діапазон прогнозів збирається та укладається "Consensus Economics". Прогнози можуть складатися щорічно, щоквартально, щомісячно та навіть щоденно.

При прогнозуванні, як правило, враховуються ризики (тобто події або умови, які можуть призвести до відхилення результату від початкових оцінок). Ці ризики допомагають при прийнятті рішення щодо вибору моделі, що використовується для отримання остаточних прогнозів. Економісти зазвичай використовують звіти з візуалізованими даними, такими як таблиці та графіки, для демонстрації свого прогнозу.

При підготовці економічних прогнозів використовувалася різноманітна інформація для підвищення точності. Для досягнення кращих прогнозів застосовується все: від макроекономічних, мікроекономічних моделей,

майбутніх ринкових даних, машинного навчання (штучних нейронних мереж) до досліджень людської поведінки.

Прогнози використовуються для досягнення різноманітних цілей. Економічні прогнози допомагають уряду та підприємствам розробляти плани, багаторічні плани та бюджети на наступний рік. Прогнози допомагають аналітикам фондового ринку оцінити вартість компанії та її акцій, а також спрогнозувати їх зростання або падіння.

Економісти обирають змінні, які є важливими для об’єкта дослідження. Економісти можуть використовувати статистичний аналіз історичних даних, щоб визначити очевидні зв'язки між певними незалежними змінними та їхнім зв'язком із залежною змінною, що досліджується. Наприклад, наскільки зміни цін на житло вплинули на статки населення загалом у минулому? Цей зв'язок потім можна використовувати для прогнозування майбутнього. Тобто, якщо ціни на житло, як очікується, зміняться певним чином, який вплив це матиме на майбутні статки населення? Прогнози, як правило, базуються на вибіркових даних, а не на повній сукупності, що призводить до невизначеності.

Економіст проводить статистичні тести та розробляє статистичні моделі (часто використовуючи регресивний аналіз), щоб визначити, які зв'язки найкраще описують або прогнозують поведінку досліджуваних змінних. Історичні дані та припущення щодо майбутнього застосовуються до моделі для отримання прогнозу щодо конкретних змінних.

1.2.2 Методи прогнозування

Загалом методи прогнозування можна розділити на три широкі класи:

1. Прогнозування на основі суджень, тобто, прогнозування, що ґрунтується на суб'єктивних судженнях (оцінках), інтуїції, поглиблених

знаннях конкретної області та іншій інформації, що має відношення прогнозованого процесу;

2. Методи прогнозування на основі використання часового ряду однієї змінної, тобто, на основі авторегресії, авторегресії з ковзним середнім (АРКС), АРКС плюс тренд і т. ін.;

3. Методи прогнозування на основі використання часових рядів декількох змінних (векторні процеси). В останньому випадку ендогенна змінна, що прогнозується, залежить від декількох регресорів або екзогенних змінних у правій частині рівняння. Очевидно, що в загальному випадку процедура прогнозування може поєднувати в собі 2-3 наведених вище методи [3].

Основні завдання прогнозування:

- виявлення усіх можливих функціональних зв'язків, що можуть бути включені в модель;
- збір необхідних статистичних даних та виявлення помилок вимірювання;
- виявлення закону зміни одного економічного фактора під впливом іншого;
- побудова моделі, яка б описувала виявлені закономірності;
- обгрунтування якості та адекватності моделі;
- аналіз та прогнозування на основі одержаної моделі.

На сьогоднішній день в спеціальній літературі описано досить велика кількість методів прогнозування лінійних та нелінійних процесів на основі використання часових рядів. Найбільш поширеними серед них є метод групового врахування аргументів (МГВА), авторегресія (АР), АРКС, авторегресія з інтегрованим ковзним середнім (АРІКС), лінійна та нелінійна множинна регресія, квантильна регресія, регресійні дерева, нейромережі, байєсівські мережі, нечіткі множини, нечіткі нейромережі та інші. Відносно „універсальними" методами моделювання та прогнозування є МГВА і нечіткі нейромережі. Однак, практика показує, що одного, навіть досить

універсального методу, недостатньо для досягнення повноти аналізу процесу [3].

1.3 Характеристики часового ряду

1.3.1 Стаціонарність

Стаціонарний часовий ряд - це часовий ряд, у якого статистичні характеристики, такі як середнє значення і дисперсія, залишаються постійними в часі.

У стаціонарного часового ряду немає систематичних змін або трендів (сильно виражених), і його характеристики залишаються стабільними на протязі всього спостереження.

Більш формально, стаціонарний часовий ряд повинен відповідати таким умовам:

- Постійність середнього значення: середнє значення ряду залишається постійним для всіх моментів часу.

- Постійність дисперсії: дисперсія ряду залишається постійною для всіх моментів часу.

- Відсутність сезонних ефектів: відсутні сезонні або циклічні коливання, які періодично впливають на ряд.

Наприклад щоденні курси акцій або інших фінансових інструментів можуть бути стаціонарними, коли немає систематичних трендів або сезонних змін у цінах. Деякі короткострокові фінансові дані можуть відповідати стаціонарним рядам.

Взаємозв'язок між стаціонарністю часового ряду і можливістю його аналізу та прогнозування полягає у спрощенні структури ряду та полегшенні роботи з ним для більшості аналітичних методів та моделей прогнозування.

Проте важливо відзначити, що існують методи та моделі, які можуть працювати і з нестаціонарними рядами. Наприклад, деякі моделі машинного навчання, такі як рекурентні нейронні мережі (RNN) або деякі варіації алгоритмів ARIMA, можуть ефективно прогнозувати нестаціонарні ряди.

Для визначення стаціонарності можна використовувати графічний аналіз. Візуально оцінивши часовий ряд виявити тренд або сезонність. Так на рис. 1.1 можна чітко побачити сезонність і тренд.

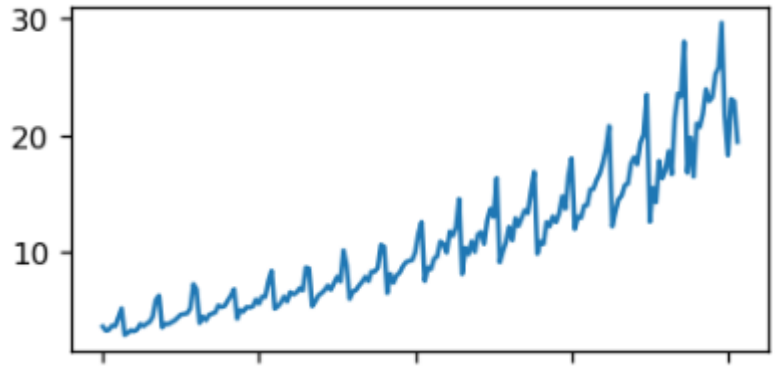


Рисунок 1.1 Нестаціонарний часовий ряд

Але далеко не завжди можна візуально оцінити стаціонарність часового ряду. Тому існує більш надійний спосіб – описові статистики, за допомогою яких ми можемо проводити тести на стаціонарність. Розглянемо декілька з них.

1. ADF.

Тест Дікі-Фуллера з поправкою (ADF-тест) використовується для перевірки нульової гіпотези про наявність одиничного кореня у вибірці часового ряду. Альтернативна гіпотеза залежить від конкретної версії тесту, але зазвичай означає стаціонарність або трендову стаціонарність. На відміну від звичайного тесту Дікі-Фуллера, який використовується лише для AP(1) моделей, ADF-тест призначений для ширшого класу моделей часових рядів.

Статистика ADF-тесту є від'ємним числом. Чим більше воно за модулем (абсолютне значення), тим сильніше відхиляється гіпотеза про наявність одиничного кореня на певному рівні значимості.

ADF припускає модель часового ряду AR(p)

$$y_t = \mu + \sum_{i=1}^{p} \gamma_i y_{t-i} + \varepsilon_t,$$

Візьмемо першу різницю

$$\Delta y_t = \mu + \delta y_{t-1} + \sum_{i=1}^{p} \beta_i \Delta y_{t-i} + \varepsilon_t,$$

Статистика для перевірки:

$$t_{\hat{\beta}_l} = \frac{\hat{\beta}_l}{SE(\hat{\beta}_l)}, \quad (1.1)$$

Нульова гіпотеза (H0): Існує одиничний корінь у часовому ряді, і він є нестаціонарним. Одиничний корінь = 1 або $\delta = 0$

Альтернативна гіпотеза (H1): Не існує одиничного кореня в часовому ряді, і він є стаціонарним. Одиничний корінь < 1 або $\delta < 0$

Якщо тестова статистика (1.1) менша за критичне значення або якщо p-значення менше попередньо заданого рівня значущості (наприклад 0,05), тоді нульова гіпотеза відхиляється, а часовий ряд вважається стаціонарним.

Якщо тестова статистика перевищує критичне значення, нульову гіпотезу не можна відхилити, а часовий ряд вважається нестаціонарним.

2. KPSS

KPSS-тест (Kwiatkowski-Phillips-Schmidt-Shin test) використовується для виявлення наявності тренда в часовому ряді. Він має іншу нульову гіпотезу порівняно з ADF-тестом. Нульова гіпотеза KPSS-тесту припускає, що часовий ряд є стаціонарним, тобто не має тренда. Альтернативна гіпотеза передбачає, що в часовому ряді присутній тренд. Якщо результат KPSS-тесту вказує на відхилення від нульової гіпотези, це свідчить про наявність тренда у часовому ряді. З іншого боку, якщо результат не відхиляє нульову гіпотезу, це може вказувати на відсутність тренда в часовому ряді, що робить його стаціонарним.

$$y_t = c_t + \delta t + e_t, c_t = c_{t-1} + u_t,$$
$$c_t = \mathrm{c}_{t-1} + \mathrm{u}_t,$$

де

δ — коефіцієнт тренда,

e_t — якийсь стаціонарний процес,

u_t — якийсь незалежний і одинаково розподілений з e_t процесом з математическим очікуванням 0 і дисперсією σ^2.

Висувається дві конкурентні гіпотези:

H_0: тимчасовий ряд є стаціонарним (або аналогічно $\sigma^2 = 0$),

H_1: тимчасовий ряд не є стаціонарним ($\sigma^2 \neq 0$).

Вичислюємо статистику:

$$\frac{\sum_{t=1}^{T} S_t^2}{s^2 T^2}, \tag{1.2}$$

де

T — розмір вибору,

$S_t = e_1 + e_2 + ... + e_t$,

s^2 — стандартна помилка у формі Ньюі-Уеста (оцінка Ньюі-Вест).

Якщо статистика (1.2) перевищує критичне значення, тоді нульова гіпотеза відхиляється; ряд нестаціонарний.

Основним недоліком тесту KPSS є те, що він має високий рівень помилок 1-го роду (він надто часто відхиляє нульову гіпотезу). Якщо ж збільшувати p, то це негативно впливає на потужність тесту.

Один із способів боротьби з потенційними помилками 1-го роду — поєднання KPSS із тестом ADF. Якщо результат обох тестів вказує на те, що часовий ряд є стаціонарним, то, ймовірно, це так.

1.3.2 Гетероскедастичність

Гетероскедастичність - це термін, який використовується в статистиці та економетриці для опису ситуації, коли дисперсія помилок (залишкових членів) в регресійній моделі залежить від значень незалежних змінних. Іншими словами, розкид помилок не є сталим по всьому діапазону значень змінних.

Коли гетероскедастичність присутня, це може вплинути на ефективність та конзистентність оцінок параметрів моделі. Також, оцінки стандартних помилок можуть бути неточними, що може призвести до неправильних висновків при статистичних тестах.

Причини гетероскедастичності.

- Систематичні зміни в розкиді: якщо в реальних даних спостерігаються систематичні зміни в розкиді даних вздовж значень незалежних змінних, це може призвести до гетероскедастичності.

- Викиди або виняткові значення: наявність викидів або виняткових значень в даних може призвести до нерівномірного розкиду залишкових членів та, таким чином, до гетероскедастичності.
- Неадекватна функціональна форма моделі: якщо функціональна форма моделі невірно вибрана і не враховує структуру дисперсії вздовж значень змінних, це може призвести до гетероскедастичності
- Нестаціонарність: нестаціонарність в ряді може також призводити до гетероскедастичності.

Виявити гетероскедастичність можна за допомогою візуалізації або статистичних тестів, таких як: тест Парка, тест Глейзера, тест Бройша-Паґана, тест Вайта.

Найпростіший спосіб виявити гетероскедастічність - використовувати графік зіставлення значення та залишку, як на рис. 1.2.

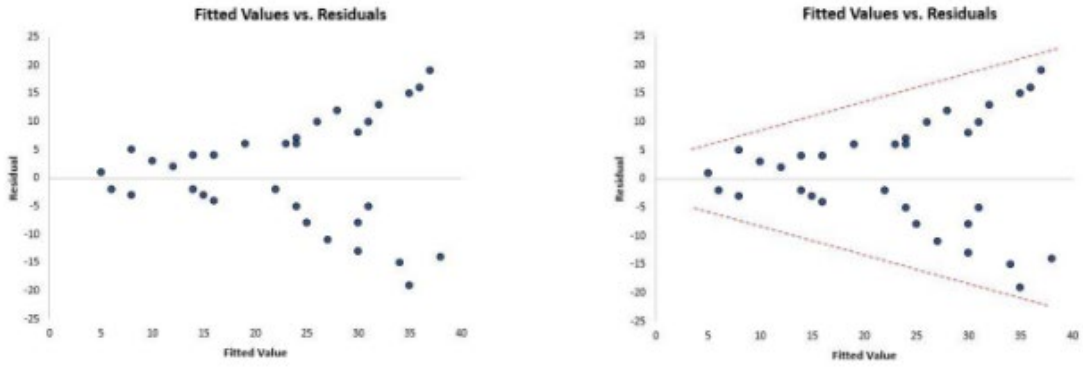


Рисунок 1.2 Зіставлення Value vs Residuals

Тест Бреуша-Пагана-Годфрі (іноді скорочений до тесту Бреуша-Пагана) є тестом на гетероскедастичність помилок у регресії.

Нульова гіпотеза (H0): Гетероскедастичності немає. Розкид залишкових членів є сталим.

Альтернативна гіпотеза (H1): Є гетероскедастичність. Розкид залишкових членів є нерівномірним.

P-значення: Якщо p-значення менше визначеного рівня значущості (зазвичай 0.05), ми можемо відкинути нульову гіпотезу і припустити наявність гетероскедастичності.

Критерій Уайта використовується для перевірки гетероскедастичних («різнорозподілених») помилок у регресійному аналізі. Це окремий випадок (простішого) тесту Бреуша-Пагана.

Для того щоб позбутися гетероскедастичності існують наступні способи.

- Декомпозиція: Якщо є сезонність або циклічність в часовому ряді, можна спробувати використовувати декомпозицію для стабілізації дисперсії.
- Моделі GARCH (Generalized Autoregressive Conditional Heteroskedasticity): ці моделі призначені спеціально для роботи з гетероскедастичністю в часових рядах. GARCH-моделі дозволяють моделювати змінність в часі.
- Нормалізація даних: подібно до звичайних методів обробки гетероскедастичності, можна використовувати різні нормалізації для зміни розподілу даних.
- Вікно згладжування: застосування згладжування на основі ковзного вікна або інших методів може допомогти стабілізувати дисперсію.

1.4 Огляд програмних засобів для прогнозування часових рядів

На сьогоднішній день існує широкий вибір програмних засобі, що дозволяють аналізувати та прогнозувати часові ряди економічної природи. Всі вони поділяються на програми, які надають певний готовий набір інструментів та бібліотеки для мов програмування, в яких реалізований певний функціонал, який користувач може поєднувати з логікою власної програми. Обидва варіанти мають свої переваги: готові продукти зазвичай зручніші та легші для

використання, в той час як другий варіант є більш гнучким, користувач може налаштовувати його відповідно до своїх задач комбінуючи функціонал бібліотек та власних функцій та класів.

1.4.1Eveiws

EViews — це статистичний пакет для Windows, який використовується переважно для економетричного аналізу, орієнтованого на часові ряди. Він розроблений компанією Quantitative Micro Software (QMS), яка зараз є частиною IHS. Версія 1.0 була випущена в березні 1994 року і замінила MicroTSP [1]. Програмне забезпечення та мова програмування TSP спочатку були розроблені Робертом Холлом у 1965 році. Поточна версія EViews — 13, випущена в серпні 2022 року.

EViews можна використовувати для загального статистичного аналізу та економетричного аналізу, наприклад аналізу поперечних і панельних даних, а також для оцінки та прогнозування часових рядів.

EViews поєднує технологію електронних таблиць і реляційних баз даних із традиційними завданнями, які можна знайти в статистичному програмному забезпеченні, і використовує графічний інтерфейс Windows. Це поєднується з мовою програмування, яка відображає обмежену об’єктну орієнтацію [4].

1.4.2 Python

Часові ряди це дуже важливий тип даних для розуміння динаміки та закономірностей у різних сферах, таких як фінанси, економіка, наукові

дослідження та багато інших. Python пропонує широкий спектр інструментів для аналізу та обробки часових рядів. Ці інструменти представлені у вигляді бібліотек, які надають набір функцій та алгоритмів для виконання поширених завдань, пов'язаних з часовими рядами. Розглянемо основні бібліотеки для роботи з часовими рядами:

Бібліотека Pandas

Pandas — це бібліотека, яка активно використовується для всіх видів обробка даних на Python. Ця бібліотека в основному використовується для структуровані дані, а його функції інтуїтивно зрозумілі та можуть легко використовувати розробники для різноманітної обробки даних випадки використання.

Pandas використовується для обробки структурованих наборів даних природи. Основною особливістю цієї бібліотеки є кадри даних, які є табличним представленням даних. Ми можемо завантажити набори даних різних форматів у фрейми даних Pandas і тим самим виконувати різні операції над даними, в т.ч очищення, агрегування або перетворення даних. Ми також можемо переглянути структуру даних, виконати тип даних перетворення та проведення дослідницького аналізу даних.

Pandas також підтримує операції часових рядів над даними. Pandas було розроблено в контексті фінансового моделювання, тому, як і слід було очікувати, він містить досить великий набір інструментів для роботи з датами, часом та даними, індексованими за часом. Дані про дату й час доступні в кількох варіантах.

- Мітки часу стосуються певних моментів часу (наприклад, 4 липня 2015 року о 7:00).
- Інтервали часу та періоди вказують на проміжок часу між конкретною початковою та кінцевою точкою; наприклад, 2015 рік. Періоди зазвичай посилаються на особливий випадок часових інтервалів, у яких кожен інтервал має однакову довжину та не збігається (наприклад, 24-годинні періоди, що складаються з днів).

- Дельти часу або тривалості вказують на точний проміжок часу (наприклад, тривалістю 22,56 секунди).

Бібліотека statsmodels

В даній бібліотеці є спеціальний модуль для аналізу часових рядів, який називається tsa.

statsmodels.tsa містить класи моделей і функції, корисні для аналізу часових рядів. Основні моделі включають однофакторні авторегресійні моделі (AR), векторні авторегресійні моделі (VAR) і однофакторні авторегресійні моделі ковзного середнього (ARMA). До нелінійних моделей відносяться динамічна регресія з перемиканням Маркова та авторегресія. Він також містить описову статистику для часових рядів, наприклад автокореляцію, часткову автокореляційну функцію та періодограму, а також відповідні теоретичні властивості ARMA або пов'язаних процесів. Він також містить методи роботи з авторегресійними та ковзними середніми лаг-поліномами. Крім того, доступні відповідні статистичні тести та деякі корисні допоміжні функції.

Оцінка виконується за допомогою точного або умовного методу максимальної правдоподібності або умовного методу найменших квадратів з використанням фільтра Калмана або прямих фільтрів.

Бібліотека Prophet

Prophet розроблена Facebook і широко використовується для прогнозування часових рядів. Ця бібліотека має ефективні методи для роботи з різними складовими часового ряду, такими як тренд та сезонність. Prophet можна використовувати для визначення сезонних патернів у даних. Цю бібліотеку можна використовувати для створення індивідуальних моделей з різними вхідними параметрами за потреби для різних випадків використання. Він також може ефективно працювати з великими наборами даних. Він має

простий API, подібний до scikit-learn щоб адаптувати модель до даних, якщо бути ознайомленим з scikit learn.

Prophet можна використовувати для наступного.

- Прогнозування майбутніх значень часових рядів. Наприклад, прогнозування майбутніх продажів або прогнозування запасів.
- Визначення викидів або аномалій в даних часових рядах. Для прикладу, визначення землетрусів з сейсмографічних показників.
- Візуалізація часових рядів і проведення розвідувального аналізу за допомогою функцій для побудови графіків, доступних у бібліотеці.

Бібліотека TensorFlow

TensorFlow — це безкоштовна бібліотека програмного забезпечення з відкритим кодом для машинного навчання та штучного інтелекту . Його можна використовувати в ряді завдань, але він зосереджений на навчанні та висновках глибоких нейронних мереж [5]. TensorFlow стає потужним інструментом для дослідників, які виконують аналіз часових рядів завдяки можливості використовувати методи глибокого навчання. Впроваджуючи глибоке навчання в аналіз часових рядів, ми можемо досягти значного прогресу як у глибині, так і в точності наших прогнозів. TensorFlow знаходиться в авангарді цього трансформаційного середовища, пропонуючи надійну та універсальну платформу для створення, навчання та розгортання цих глибоких нейронних мереж.

1.5 Висновки до розділу 1

У даному розділі було розглянуто актуальність задачі прогнозування часових рядів економічної природи. Розглянуто поняття економічного прогнозування, його мета та особливості.

Також були досліджені характеристики часових рядів, такі як стаціонарність та гетероскедастичність. Розглянуто причини, наслідки та методи визначення стаціонарності, основними з яких є статистичні тести ADF, KPSS та гетероскедаскичності (тест Вайта та тест Бреуша-Пагана-Годфрі). Також були показані приклади нестаціонарних та гетероскедастичних часових рядів.

Далі було зроблено огляд основних програмних засобів для прогнозування часових рядів. Серед них розглянуто пакет програм EViews та бібліотеки для аналізу та прогнозування часових рядів в мові програмування Python. Ці бібліотеки будуть в подальшому використані для моделювання та прогнозування в розділі 3.

РОЗДІЛ 2 МАТЕМАТИЧНІ МОДЕЛІ ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ

У розділі 2 буде описано математичні моделі прогнозування часових рядів, які потім будуть використані в розділі 3. Також будуть розглянуті критерії оцінювання моделей. Можна виділити три цілі побудови математичних моделей процесів та об'єктів:

- поглиблене вивчення процесів;
- прогнозування значень змінних за допомогою моделі;
- використання моделі для створення (синтезу) систем керування.

Прогнозування значень змінних виконується, як правило, на основі набагато простіших моделей ніж поглиблене вивчення процесів. Для цієї мети дуже зручними є дискретні моделі у вигляді авторегресійних рівнянь (АР) та авторегресії з ковзним середнім (АРКС). Часто використовують також стандартизоване представлення в просторі станів неперервних чи дискретних моделей. Тип моделі залежить також від того, який прогноз необхідний короткостроковий, середньостроковий чи довгостроковий. Задача прогнозування може бути дуже складною, якщо стохастичний процес нелінійний та нестаціонарний [3].

2.1 Підготовка даних до прогнозування

Попередній аналіз та обробка даних є важливим кроком для подальшого прогнозування. Це дає змогу краще зрозуміти дані – дослідити розподіл, виявити пропущені значення, викиди та інші проблеми з якістю даних. Також на цьому етапі відбувається очистка даних та їх підготовка до моделювання.

2.1.1Заповнення пропусків

Заповнення пропусків даних - це процес відновлення відсутніх значень у даних. У реальних даних часто зустрічаються пропуски, які можуть виникати з різних причин, таких як помилки в зборі даних, технічні проблеми, відмови обладнання тощо. Заповнення цих пропусків є важливим етапом у підготовці даних для аналізу та моделювання. На це є багато причин, по-перше, пропуски можуть призвести до втрати інформації та спотворення результатів аналізу. Також багато алгоритмів машинного навчання вимагають повних даних для ефективного навчання та прогнозування. Окрім цього неправильна обробка пропусків може призвести до викривлення статистичних властивостей набору даних та неправильних висновків. Правильне заповнення пропусків може покращити якість та точність аналізу даних.

Пропуски можуть виникати з багатьох причин, головні з них наступні.

- Помилки при зборі даних: під час збору даних можуть виникати помилки, такі як технічні несправності, неправильне введення даних або втрата даних під час передачі.
- Відсутність даних: іноді певні ознаки можуть бути відсутніми з навмисного пропуску, наприклад, якщо людина вирішила не надавати певну інформацію у опитувальній формі.
- Неспроможність отримати дані: деякі дані можуть бути важкодоступними або неможливими для отримання з технічних, легальних або етичних причин.

- Природні катастрофи або непередбачені обставини: природні катастрофи, кібератаки або інші непередбачені обставини також можуть призвести до втрати або пошкодження даних.

Методи обробки пропусків даних можна умовно поділити на три основні категорії: ігнорування, видалення та заповнення.

При першому методі пропуски просто ігноруються. Відмінність полягає в тому, що жодна додаткова обробка даних не проводиться. Цей метод може бути прийнятним, коли для роботи використовують моделі, які можуть працювати з пропусками. Наприклад моделі засновані на деревах рішень, байєсівські моделі та деякі алгоритми нейронних мереж.

При використанні видалення рядки або стовпці з пропусками видаляються з набору даних. Цей метод може бути ефективним, якщо пропуски становлять невеликий відсоток від загального обсягу даних і не призводять до великих втрат інформації. Видалення даних з часових рядів може вплинути на структуру та характеристики даних. Видалення пропусків є лише одним із можливих підходів до обробки пропущених значень в часових рядах.

Під час заповнення пропусків відсутні значення замінюються певним шляхом. Цей метод дозволяє зберегти обсяг даних та зменшити вплив пропусків на результати аналізу, проте він може призвести до спотворення реальних залежностей в даних. В такому випадку ми маємо розуміти, що похибка отриманих результатів буде накопичуватись. Заповнення пропусків - це важливий крок в обробці даних, оскільки він дозволяє зберегти цілісність часового ряду та забезпечити правильність аналізу та моделювання.

Найпростіший спосіб заповнення пропусків це заповнення нулями або константою. Із переваг способу - його простота. Але це може вплинути на характеристики часового ряду, особливо якщо нулі не відображають реальних значень, а просто позначають пропуски. А також це може викликати перекіс у великих обсягах даних, особливо якщо пропущених значень дуже багато.

Іншим способом є заповнення пропусків медіаною. Такий спосіб зберігає центральну тенденцію даних на відміну від заповнення константою. Також він менше вразливий до викидів у даних порівняно з середнім арифметичним. Із недоліків даного методу можна виділити те, що він може втратити важливість випадкових коливань.

Заповнення пропусків середнім арифметичним є простим у використанні та зберігає центральну тенденцію даних. Але такий метод є вразливим до викидів в даних, а також як і заповнення медіаною може втратити важливість випадкових коливань. Існують також методи заповнення пропусків медіаною, мінімальним або максимальним значенням, але ці методи використовуються вкрай рідко.

Заповнення пропусків методами LOCF/NOCF (Last/Next Observation Carried Forward). LOCF заповнює пропущені значення останнім відомим значенням. Це означає, що для кожного пропущеного значення використовується останнє відоме значення у часовому ряді. NOCF заповнює пропущені значення наступним відомим значенням. Це означає, що для кожного пропущеного значення використовується наступне відоме значення у часовому ряді. Такий метод використовується, коли є припущення, що значення залишаються постійними після останнього відомого значення, є прийнятним. Придатний для даних, де значення змінюються рідко або повільно. Може бути використаний для даних, де між значеннями існує плавний перехід або низька змінність.

Розглянемо ситуацію, коли дані представляють собою часовий ряд температури в приміщенні, який фіксується кожну годину, і в деяких випадках значення може бути пропущено через технічні або інші причини. В цьому випадку метод може бути виправданим, оскільки температура в приміщенні зазвичай змінюється повільно, і припущення, що значення залишаються постійними протягом тривалого періоду, може бути прийнятним. На рис. 2.1 приведений приклад використання методу.

Час	Температура	Час	Температура	Час	Температура
12:00	20°C	12:00	20°C	12:00	20°C
13:00	21°C	13:00	21°C	13:00	21°C
14:00	NaN	14:00	21°C (LOCF)	14:00	22°C (NOCF)
15:00	NaN	15:00	21°C (LOCF)	15:00	22°C (NOCF)
16:00	22°C	16:00	22°C	16:00	22°C
17:00	23°C	17:00	23°C	17:00	23°C

Рисунок 2.1 Заповнення пропусків методом LOCF/NOCF

2.1.2Масштабування даних

Масштабування даних - це процес перетворення значень у масиві даних так, щоб вони попадали в певний діапазон. Зазвичай виконується шляхом масштабування значень в певному діапазоні. Це дозволяє уникнути перекосу у впливі великих значень на аналіз та забезпечити порівнянність між різними ознаками.

Перерахуємо коли потрібно застосовувати масштабування даних.

- Коли числові ознаки в наборі даних мають різний масштаб (наприклад, одна ознака вимірюється у тисячах, а інша - у одиницях), то це може призвести до проблем у багатьох алгоритмах машинного навчання. Обробка даних допомагає зробити ознаки порівнюваними за масштабом.
- Якщо розподіл значень ознак відрізняється від нормального розподілу (наприклад, є великі викиди або асиметрія), то нормалізація може полегшити аналіз та покращити роботу багатьох алгоритмів, які передбачають нормальний розподіл даних.
- Деякі алгоритми машинного навчання, такі як методи градієнтного спуску, k-середніх, підтримки векторів, нейронні мережі, можуть бути

чутливими до масштабу та розподілу даних. Обробка допомагає їм працювати ефективніше та швидше.

- Великі значення в даних можуть мати надмірний вплив на результати моделей: масштабування допомагає зменшити цей вплив та забезпечити рівноцінну обробку всіх ознак.

Масштабування потрібне наприклад, якщо аналізувати дані про дохід людей, вони можуть бути в дуже різних масштабах, наприклад, від декількох тисяч до мільйонів доларів. Нормалізація може допомогти зробити ці дані порівнюваними. Розглянемо деякі популярні методи масштабування.

Десяткове масштабування. Основна ідея полягає в тому, що значення оригінальних даних діляться на певну ступінь десятки, щоб зменшити масштаб. У цьому методі ми переміщуємо десяткову точку значень атрибута. Це переміщення десяткових знаків повністю залежить від максимального значення серед всіх значень атрибута.

Мінімаксна нормалізація. У цьому методі масштабування даних лінійне перетворення виконується на вихідних даних. З даних витягується мінімальне і максимальне значення, і кожне значення замінюється у відповідності з формулою (2.1).

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}}, \tag{2.1}$$

Цей метод масштабує значення в діапазон від 0 до 1.

При нормалізації іншими методами є наявність аномальних значень даних, які «розтягують» діапазон, що призводить до того, що нормалізовані значення знову ж концентруються в деякому вузькому діапазоні поблизу нуля. Щоб уникнути цього, можна визначати діапазон не за допомогою максимальних і мінімальних значень, а за допомогою - середнього і дисперсії. Такий метод називається z-нормалізація.

Він стандартизує дані так, що середнє значення стає рівним 0, а стандартне відхилення - 1. Формула для перетворення показана в рівнянні (2.2):

$$Z = \frac{X - \mu}{\sigma}, \tag{2.2}$$

де

μ - середнє арифметичне,

σ - середньоквадратичне відхилення.

У випадку часових рядів, особливо якщо використовується масштабування даних, демасштабування може бути потрібним для отримання прогнозів або результатів в їх оригінальному масштабі. Коли прогнозуються майбутні значення часового ряду або оцінюються результати моделі, отримані на масштабованих даних, результати також будуть масштабовані. Щоб зрозуміти ці результати у контексті оригінальних даних, потрібно виконати демасштабування.

2.2 Моделі ARIMA

Авторегресійне інтегроване ковзне середнє, або ARIMA(англ. autoregressive integrated moving average), — це модель статистичного аналізу, яка використовує дані часових рядів, щоб або краще зрозуміти набір даних, або передбачити майбутні тенденції. Статистична модель є авторегресійною, якщо вона передбачає майбутні значення на основі минулих значень. Наприклад, модель ARIMA може передбачити майбутні ціни акцій на основі їх минулих показників або спрогнозувати прибутки компанії на основі минулих періодів [3].

Модель ARIMA складається з 3 основних компонентів:

- Авторегресійна частина: враховує минулі значення з різними лагами;

- Інтегрована частина: показує скільки разів потрібно взяти різницю, щоб ряд став стаціонарним;

- Ковзне середнє: середнє значення з певної підмножини часового ряду, яке називається вікном.

Модель можна записати як в формулі (2.3):

$$y_t^{'} = c + \phi_1 y_{t-1}^{'} + \cdots + \phi_p y_{t-p}^{'} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t, \qquad (2.3)$$

де

$y_t^{'}$ - різниця d-го порядку,

p - порядок авторегресійної частини,

d - порядок інтегрованої частини,

q - порядок КС.

Модель ARIMA(2.3) може добре прогнозувати нестаціонарні ряди через наявність інтегрованої компоненти. Але така модель взагалі не враховує сезонність.

SARIMA – це розширення моделі ARIMA з урахування сезонної компоненти. Сезонна модель ARIMA формується шляхом включення додаткових сезонних термінів у моделі ARIMA, які ми бачили досі. Це записується так:

$$\text{ARIMA} \quad \underbrace{(p,d,q)}_{\text{Non-seasonal part of the model}} \quad \underbrace{(P,D,Q)_m}_{\text{Seasonal part of of the model}}$$

де m - кількість спостережень на рік. Ми використовуємо позначення у верхньому регістрі для сезонних частин моделі та у нижньому регістрі для несезонних частин моделі.

Сезонна частина моделі складається з термінів, подібних до несезонних компонентів моделі, але включають зворотні зсуви сезонного періоду. Наприклад, $ARIMA(1,1,1)(1,1,1)_4$ модель (без константи) для квартальних даних (м=4), і її можна записати як:

$$(1-\phi_1 B)\,(1-\Phi_1 B^4)(1-B)(1-B^4)y_t = (1+\theta_1 B)\,(1+\Theta_1 B^4)\varepsilon_t, \quad (2.4)$$

Додаткові сезонні змінні просто помножуються на несезонні змінні як в (2.4) [4].

2.3 Нейронні мережі

Нейромережа (Штучна нейронна мережа) — це математична модель, яка імітує структуру та функціонування біологічних нейронних мереж з метою вирішення різноманітних задач, таких як класифікація, регресія, прогнозування та генерація. В основі нейромереж лежать штучні нейрони, які об’єднуються в графові структури і передають сигнали один одному через ваги зв’язків. Завдяки процесу навчання, під час якого ваги та зміщення між нейронами оптимізуються, нейромережі стають здатними до виявлення закономірностей та залежностей у вхідних даних [6].

Глибоке навчання - це галузь машинного навчання, де кілька рівнів нейронних мереж використовуються для розробки моделей, які можуть працювати виконувати різні завдання прогнозування на даних. Є кілька варіації цих алгоритмів глибокого навчання. Ми будемо розглядати відповідні алгоритми, які можна налаштувати та застосувати до даних часових рядів для задач прогнозування. Деякі з популярні алгоритми глибокого навчання включають рекурентні нейронні мережі (RNN), мережі з довгою короткочасною пам’яттю (LSTM) і згорткові нейронні мережі (CNN) [7].

2.3.1 Рекурентні нейронні мережі

Рекурентні нейронні мережі або RNN — це особливий тип нейронних мереж, розроблений для вирішення проблем прогнозування послідовностей. Враховуючи стандартну багатошарову мережу перцепторів прямого зв’язку, рекурентну нейронну мережу можна розглядати як додавання петель до архітектури мережі. Наприклад, у певному шарі кожен нейрон може передавати свій сигнал вбік на додаток до наступного шару. Вихід мережі може подаватись як вхід до мережі з наступним вхідним вектором. І так далі. Повторювані з'єднання додають стан або пам'ять мережі та дозволяють їй вивчати ширші абстракції з вхідних послідовностей.

Розглянемо найпростішу RNN, що складається з одного нейрона, який отримує вхідні дані, дає певні дані на вихід та приймає на вхід свій же вихід назад, як показано на рис. 2.2

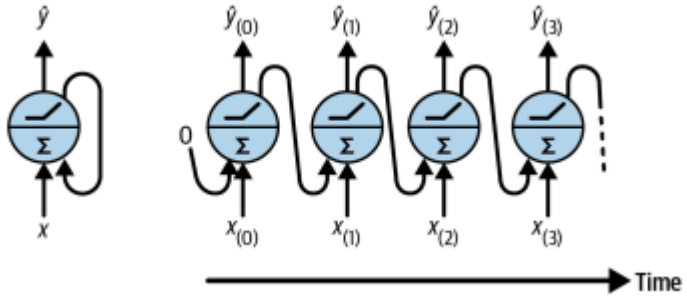


Рисунок 2.2 Структура RNN

Кожен рекурентний нейрон має два набори ваг: один для входів x(t) а інший для вихідних даних попереднього кроку часу, ŷ(t–1). Назвемо ці вагові вектори w_x і $w_{\hat{y}}$. Якщо ми розглянемо весь рекурентний шар замість одного рекурентного нейрона, ми зможемо помістіть усі вектори ваг у дві вагові матриці: W_x і $W_{\hat{y}}$.

Вихідний вектор усього рекурентного шару може бути обчислений як показано в рівнянні (2.5)

$$\hat{y}(t) = \phi\left(W_x{}^T x(t) + W_{\hat{y}}{}^T \hat{y}(t-1) + b\right), \quad (2.5)$$

де

b є вектором зсуву,

ϕ(·) є функція активації (наприклад ReLU).

Ми можемо обчислити вихід рекурентного шару за один раз для цілого міні-пакету, помістивши всі входи на кроці часу t в вхідну матрицю X(t), як показано в (2.6)

$$\hat{Y}(t) = \phi\left(X(t)W_x + \hat{Y}(t-1)W_{\hat{Y}} + b\right) = \phi\left(\left[X_{(t)}\hat{Y}_{(t-1)}\right]W + b\right), \quad (2.6)$$

де

$$W = \begin{bmatrix} W_x \\ W_{\hat{y}} \end{bmatrix},$$

Ŷ(t) є m × n матриця, що містить виходи шару на кроці часу t для кожного екземпляра у міні-пакеті(m – кількість екземплярів у міні-пакеті та n — кількість нейронів),

X(t) це матриця m × n, що містить входи для всіх екземплярів (n це кількість вхідних функцій),

W_x є матрицею $n_{\text{входів}} \times n_{\text{нейронів}}$, що містить ваги з'єднань для входів поточного кроку часу,

$W_{\hat{Y}}$ є $n_{\text{нейронів}} \times n_{\text{нейронів}}$ матриця, що містить вагові коефіцієнти з'єднання для виходи попереднього кроку часу,

b - вектор розміру $n_{\text{нейронів}}$, який містить зсув кожного нейрона.

Оскільки вихід рекурентного шара на кроці часу t є функцією всіх входів з попередніх часових кроків, можна сказати, що він має певну форму пам'яті. Частина нейронної мережі, яка зберігає певний стан протягом кроків

у часі, називається коміркою пам’яті. Один рекурентний нейрон або шар рекурентних нейронів є дуже простою клітиною, здатний вивчати лише короткі шаблони (зазвичай близько 10 кроків, але це значення відрізняється в залежності від задачі).

Щоб навчити RNN, хитрість полягає в тому, щоб розгорнути його в часі, а потім використовувати регулярне зворотне поширення похибки (див. Рис. 2.3). Ця стратегія називається зворотним поширенням похибки через час(BPTT). Так само, як у звичайному зворотному поширенні похибки, є перший прохід вперед через розгорнуту мережу (позначено пунктирними стрілками). Потім оцінюється вихідна послідовність використовуючи функцію втрат $\mathscr{L}$(Y(0), Y(1), …, Y(T); Ŷ(0), Ŷ(1), …, Ŷ(T)) (де Y(i) є і-та ціль, Ŷ(i) є і-те прогнозоване значення і T – максимальний крок за часом). Зазначимо, що ця функція втрат може ігнорувати деякі виходи. Наприклад, у RNN послідовність до вектора всі виходи ігноруються, за винятком останнього. На рис. 2.3 функція втрат обчислюється лише на основі останніх трьох виходів. Градієнти цієї функції втрат потім поширюється назад через розгорнуту мережу (представлену суцільним тілом стрілки). У цьому прикладі, оскільки вихідні дані Ŷ(0) і Ŷ(1) не використовуються для обчислення втрати, градієнти не течуть назад через них; вони лише протікають через Ŷ(2), Ŷ(3) і Ŷ(4). Крім того, оскільки на кожному з них використовуються однакові параметри W і b за кожен часовий крок, їхні градієнти будуть налаштовані кілька разів під час проходження. Коли зворотня фаза завершена, і всі градієнти обчислено, BPTT може виконайти крок градієнтного спуску, щоб оновити параметри (це нічим не відрізняється від звичайного методу).

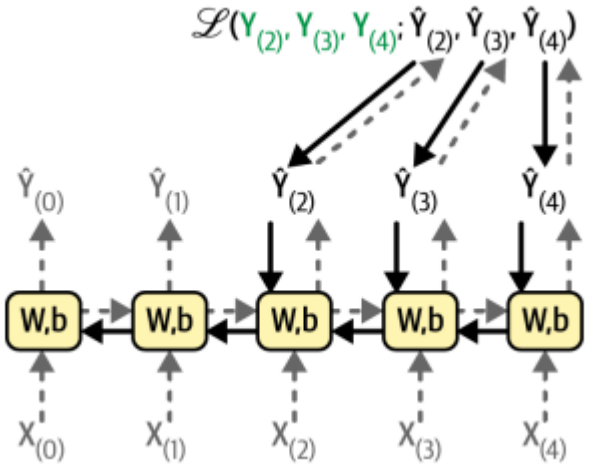


Рисунок 2.3 Зворотне поширення похибки через час

2.3.2 LSTM

Завдяки перетворенням, через які проходять дані під час проходження RNN, деяка інформація втрачається на кожному кроці часу. Через деякий час стан RNN практично не містить слідів перших входів. Це може бути шоустоппером. Щоб вирішити цю проблему, були запропоновані різні типи комірок з довгостроковою пам’яттю. Вони виявилися настільки успішними, що основні клітини більше не використовуються. Спочатку розглянемо найпопулярнішу з цих довгострокових комірок пам'яті: комірка LSTM [8].

LSTM, яка найбільш часто використовується в літературі було описано Грейвсом і Шмідхубером [9]. Архітектуру LSTM показано на рис. 2.4

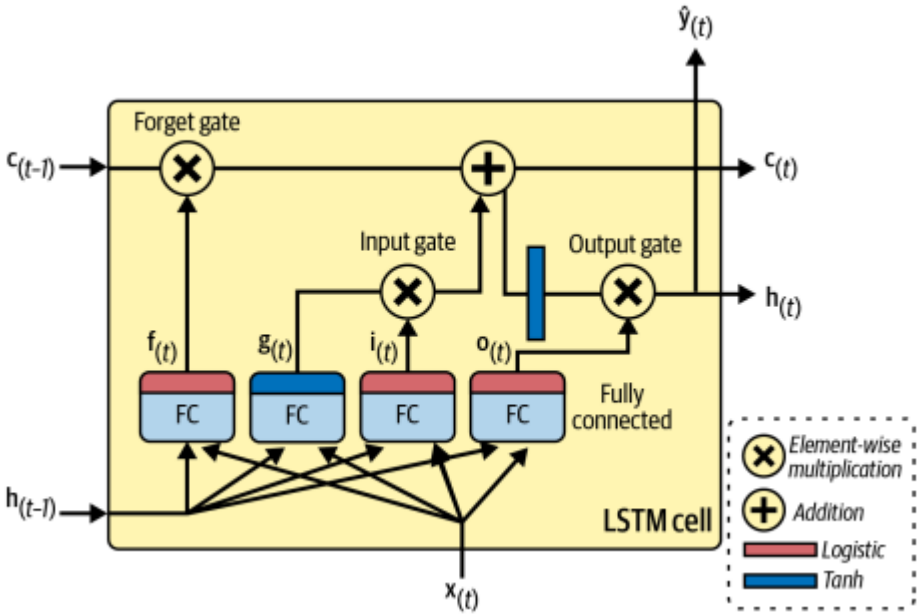


Рисунок 2.4 Архітектура LSTM

Ключова ідея полягає в тому, що мережа може дізнатися, що зберігати в довгостроковому стані, що викидати, а що читати з нього. Коли довгостроковий стан c(t–1) перетинає мережу зліва направо, можна побачити, що він спочатку проходить через забувальний вентиль, відкидаючи деякі дані з пам'яті, а потім додає кілька нових даних за допомогою операції додавання (яка додає дані, які були вибрані вхідними воротами). Результат c(t) надсилається безпосередньо без будь-яких подальших перетворень. Отже, на кожному часовий крок, деякі спогади видаляються, а деякі додаються. Крім того, після операції додавання довгостроковий стан копіюється та передається через функцію tanh, а потім результат фільтрується вихідним вентилем. Це створює короткочасний стан h(t) (що дорівнює виходу комірки за цей часовий крок, y(t)) [7].

Основний шар - це той, який виводить g(t). Його роль полягає в аналізі поточних входів x(t) і попереднього (короткострокового) стану h(t–1). У базовій комірці немає нічого, окрім цього шару, і його вихідні дані переходять безпосередньо до y(t) і h(t). Але в комірці LSTM вихід цього шару не виходить прямо, натомість його найважливіші частини зберігаються в довгостроковому стані (а решта викидається). Три інші шари це:

- Забувальний вентиль (керована f(t)) контролює, які частини довгострокового стану слід стерти;
- Вхідний вентиль (керований i(t)) контролює, які частини g(t) слід додати до довгострокового стану;
- Вихідний вентиль (керований o(t)) контролює, які частини довгострокового стану слід зчитувати та виводити на цьому кроці часу, як для h(t), так і для y(t).

LSTM описується наступними рівняннями (2.7)-(2.11):

$$f_t = \sigma_g\big(W_f x_t + U_f h_{t-1} + b_f\big), \tag{2.7}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i), \tag{2.8}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o), \tag{2.9}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c), \tag{2.10}$$

$$h_t = o_t \circ \sigma_h(c_t), \tag{2.11}$$

де

x_t – вхідний вектор,

h_t – вихідний вектор,

c_t – вектор стану комірки,

$W, U\ i\ b$ – матриці та вектор параметрів,

f_t, i_t, o_t – вектори вентилів,

$\circ$ позначає добуток Адамара (поелементний добуток).

Функції активації: σ_g - сигмоїдна функція активації; σ_h, σ_c - гіперболічний тангенс(tanh).

Існує кілька варіантів LSTM. Одним з популярних варіантів є GRU, який ми зараз і розглянемо.

2.3.3 GRU

Вентильні рекурентні вузли(англ. Gated recurrent units,GRU) була запропонована Kyunghyun Cho та ін. у статті [10]. На рис. 2.5 представлено структуру одного вузла:

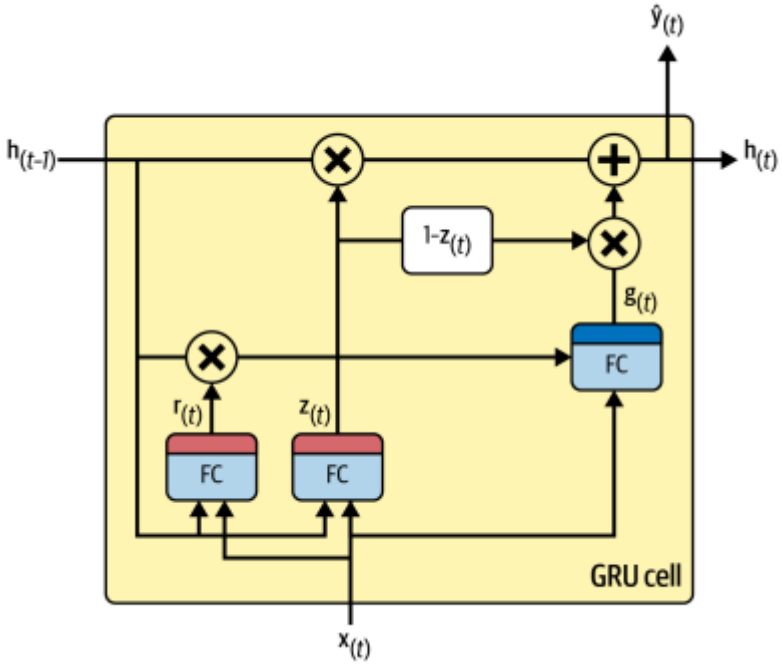


Рисунок 2.5 Структура вузла GRU

GRU є спрощеною версією LSTM, але показує такі ж хороші результати, що пояснює його зростаючу популярність. Основні спрощення в порівняння з LSTM:

- Обидва вектори стану об’єднані в один вектор h(t)
- Один контролер z(t) керує як вентилем забуття, так і вентилем введення. Якщо контролер видає 1, це означає, що вентиль забуття відкритий (= 1), а вхідний вентиль закритий (1 – 1 = 0). Якщо він виводить 0, відбувається навпаки. Іншими словами, щоразу, коли необхідно зберегти пам’ять, спочатку стирається місце, де вона буде зберігатися.
- Відсутній вихідний вентиль; повний вектор стану виводиться на кожному кроці. Однак існує новий контролер r(t), який контролює, яка частина попереднього стану буде показана на основному шарі (g(t)).

Формули для обчислення стану вузла на момент часу t:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z),$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r),$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h),$$

Функції активації: σ_h - tanh, σ_g - сигмоїдна функція.

LSTM та GRU є одними з головних причин успіху рекурентних нейронних мереж. Хоча вони можуть працювати з набагато довшими послідовностями, ніж прості RNN, вони все ще мають досить обмежену короткочасну пам’ять, і їм важко вивчати довгострокові шаблони в послідовностях із 100 часових кроків або більше, наприклад аудіо зразки, довгі часові ряди або довгі речення. Один із способів вирішити це - скоротити вхідні послідовності; наприклад, за допомогою 1D згорткових шарів.

2.3.4 WaveNet

Одновимірний згортковий шар розміщує кілька ядер по послідовності, створюючи одновимірну карту функцій для кожного ядра. Кожне ядро навчиться виявляти один дуже короткий послідовний шаблон (не довший за розмір ядра). Якщо використовувати 10 ядер, вихід шару складатиметься з 10 1D-послідовностей (усі однакової довжини), або, еквівалентно, можна переглядати цей вихід як одну 10D-послідовність. Це означає, що можна побудувати нейронну мережу, що складається з суміші рекурентних шарів і одновимірних згорткових шарів.

У статті 2016 року Аарон ван ден Оорд та інші дослідники DeepMind представили нову архітектуру під назвою WaveNet [11]. Вони склали одновимірні згорткові шари, подвоївши швидкість розширення (наскільки

рознесені вхідні дані кожного нейрона) на кожному шарі: перший згортковий шар бачить лише два часові кроки за раз, тоді як наступний бачить чотири часові кроки (його поле містить чотири часові кроки), наступне поле містить вісім часових кроків і так далі (див. рис. 2.6). Таким чином, нижчі рівні засвоюють короткострокові шаблони, а вищі – довгострокові. Завдяки швидкості подвоєння розширення мережа може дуже ефективно обробляти надзвичайно великі послідовності.

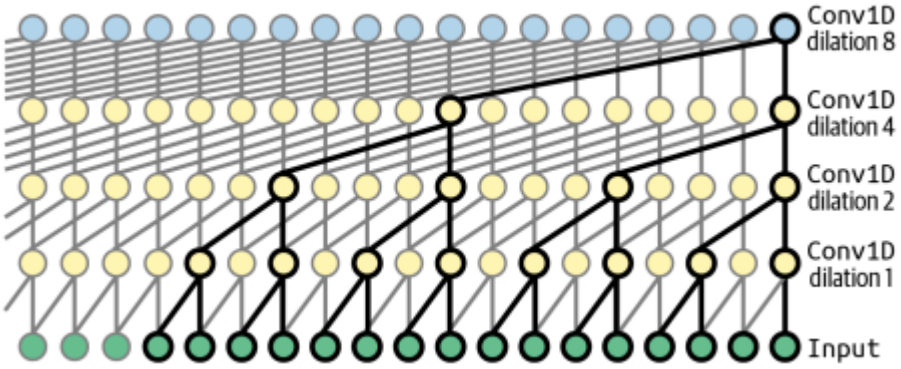


Рисунок 2.6 Структура WaveNet

Автори статті фактично склали 10 згорткових шарів зі швидкістю розширення 1, 2, 4, 8, …, 256, 512, потім вони склали іншу групу з 10 ідентичних шарів (також зі швидкостями розширення 1, 2, 4, 8, …, 256, 512), потім ще одну ідентичну групу з 10 шарів. Вони вказали, що один стек із 10 згорткових шарів із такими темпами розширення діятиме як надефективний згортковий шар з ядром розміром 1024 (за винятком того, що він значно швидший, потужніший і використовує значно менше параметрів). Вони також доповнили вхідні послідовності кількістю нулів, що дорівнює швидкості розширення перед кожним шаром, щоб зберегти однакову довжину послідовності по всій мережі.

Архітектура моделі (див. рис. 2.7) складається зі стека блоків WaveNet, де вихідні дані з попереднього блоку передаються в наступний, а також до решти мережі для отримання остаточного результату. Вихідні дані є набором категоріальних розподілів ймовірностей, який має таку саму часову

розмірність, як і вхідні дані, тобто для кожного вхідного часового ряду (xt−n,..., xt−1) ми отримуємо розподіли ймовірностей для наступних часових кроків (xt−n+1,..., xt) (див. рис. 2.8). Розподіли створюються з активацією softmax як останнього рівня в мережі, а параметри моделі оптимізовано для максимізації логарифмічної ймовірності прогнозів [12].

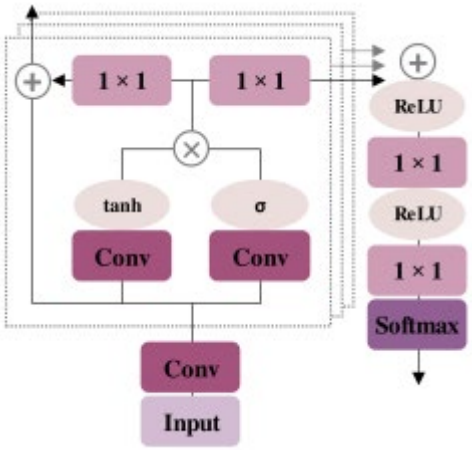


Рисунок 2.7 Архітектура WaveNet

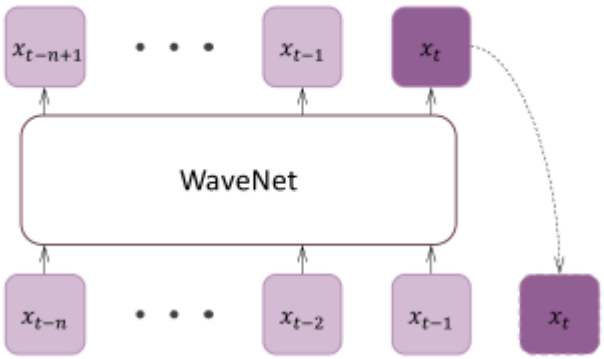


Рисунок 2.8 Вихідні дані WaveNet

2.4 Критерії оцінювання моделей

Для вибору кращої моделі з множини побудованих моделей виконується перевірка цих моделей на адекватність. Така перевірка включає наведені нижче кроки.

1. Аналіз графіка похибок моделі. На графіку не повинно бути значних викидів та довгих інтервалів, в яких похибка приймає великі значення(інтервали суттєвої неадекватності).

2. Похибки моделі не повинні корелювати між собою. Для перевірки кореляції необхідно обчислити АКФ та ЧАКФ для похибок і за допомогою Q-статистики визначити ступінь корельованості. Також корельованість можна визначити за допомогою статистики Дарбіна-Уотсона, яка розраховується за наступною формулою:

$$DW = 2 - 2\rho$$

де

$\rho = E[e(k)e(k-1)] / \sigma_e^2$ - коефіцієнт кореляції між сусідніми значеннями похибки,

σ_e^2 - дисперсія похибок.

Якщо кореляція між похибками відсутня то DW=2

3. Коефіцієнт множинної детермінації R^2, що обчислюється так:

$$R^2 = \frac{var(\hat{y})}{var(y)} = 1 - \frac{SSE}{SST}$$

де

$\text{var}(\hat{y})$ - дисперсія залежної змінної, оціненою за допомогою моделі,

$\text{var}(y)$ - дисперсія вимірів залежної змінної,

SSE - сума квадратів похибок,

SST - загальна сума квадратів.

Найкращим значенням є $R^2 = 1$, тобто дисперсія вимірів змінної та дисперсія оцінена за моделлю збігаються.

4. Сума квадратів похибок повинна бути мінімальною у порівнянні з іншими моделями:

$$\sum_{k=1}^{N} e^2(k) = \sum_{k=1}^{N} [\hat{y}(k) - y(k)]^2 \to \min_{\hat{\theta}}$$

5. Для того щоб оцінити адекватність моделі використовується критерій Акайке:

$$AIC = N\, ln\left(\sum_{k=1}^{N} e^2(k)\right) + 2n$$

де

n – кількість параметрів моделі, що оцінюються за допомогою статистичних даних.

В правій частині міститься сума квадраті похибок, тому кращою буде модель з найменшим значенням. Якщо потрібно оцінити адекватність моделі в цілому, то можна використати статистику Фішера.

Також серед популярних критеріїв можна виділити

- RMSE(Корінь середньоквадратичної похибки)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (\hat{y} - y)^2}$$

- MAE (Середня абсолютна похибка)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y} - y|$$

- MAPE (Середня абсолютна відносна похибка)

$$MAPE = \left(\frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{|y_i|}\right) * 100\%$$

2.5 Висновки до розділу 2

У даному розділі було розглянуто методи підготовки даних до прогнозування, такі як очищення даних та маштабування. Були описані різні моделі для прогнозування часових рядів, а саме:

- Моделі ARIMA та її модифікації;
- Найпростіші рекурентні нейронні мережі;
- LSTM;
- GRU;
- WaveNet.

Також було побудовано алгоритм оцінювання якості моделі та вибору кращої моделі серед кандидатів. Розглянуто різні критерії якості моделей. Детально описано особливості навчання рекурентних нейронних мереж.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ

3.1 Вибір мови програмування

Для реалізації програмного продукту мною було обрано мову програмування Python. Python широко використовується для прогнозування часових рядів завдяки численним перевагам.

- Простота використання. Синтаксис Python простий і зрозумілий, що робить його доступним як для початківців, так і для досвідчених програмістів. Велика спільнота користувачів Python полегшує пошук допомоги та рішень проблем.
- Широкий спектр бібліотек. Існує безліч бібліотек Python, спеціально розроблених для аналізу та прогнозування часових рядів, таких як: Statsmodels(надає широкий спектр інструментів для статистичного аналізу часових рядів, включаючи ARIMA, GARCH та інші моделі), Pandas(потужна бібліотека для роботи з даними, що включає функції для маніпулювання часовими рядами, такі як передискретизація, агрегування та зсув), scikit-learn(містить алгоритми машинного навчання, які можна використовувати для прогнозування часових рядів, такі як лінійні регресії, дерева рішень та нейронні мережі).
- Гнучкість. Python дозволяє інтегрувати код з іншими мовами програмування, такими як C++ або R, що робить його гнучким інструментом для складних завдань прогнозування. Python підтримує різні середовища розробки, включаючи Jupyter Notebooks та PyCharm, що робить його зручним для інтерактивної роботи з даними.

- Ефективність. Python має оптимізовані бібліотеки для чисельних обчислень, що робить його ефективним для роботи з великими наборами даних часових рядів.

Зважаючи на всі ці фактори, Python є чудовим вибором для прогнозування часових рядів завдяки простоті використання, широкому спектру бібліотек, гнучкості, ефективності, широкому застосуванню та постійному розвитку.

В якості середовища для програмування було обрано Colaboratory, головною перевагою якого те, що записники Colab виконують код на хмарних серверах Google, тобто можна застосувати можливості апаратного забезпечення Google, зокрема графічних і тензорних процесорів, що є дуже корисним при навчанні глибоких нейронних мереж.

3.2 Інформація про дані

В якості даних для прогнозування було взято денну ціну на акції компанії Microsoft Corporation за період з 2019-05-08 по 2024-05-07. На рис. 3.1 показано перші 7 значень.

	Date	Open	High	Low	Close	Adj Close	Volume
0	2019-05-08	125.440002	126.370003	124.750000	125.510002	119.471992	28419000
1	2019-05-09	124.290001	125.790001	123.570000	125.500000	119.462502	27235800
2	2019-05-10	124.910004	127.930000	123.820000	127.129997	121.014053	30915100
3	2019-05-13	124.110001	125.550003	123.040001	123.349998	117.415916	33944900
4	2019-05-14	123.870003	125.879997	123.699997	124.730003	118.729538	25266300
5	2019-05-15	124.260002	126.709999	123.699997	126.019997	120.401512	24722700
6	2019-05-16	126.750000	129.380005	126.459999	128.929993	123.181755	30112200

Рисунок 3.1 Ціни на акції MSFT

Розберемо що означає кожна зі змінних:

- Date – Дата в форматі yyyy-mm-dd;
- Open – Ціна, за якою акція відкрилася для торгівлі на початку дня;
- High – Найвища ціна, за якою акція торгувалася протягом дня;
- Low – Найнижча ціна, за якою акція торгувалася протягом дня;
- Close – Ціна, за якою акція закрилася для торгівлі наприкінці дня;
- Adj Close – Ціна закриття з урахуванням корпоративних дій, таких як дроблення акцій або виплати дивідендів;
- Volume – це кількість акцій, що були продані за день.

Після проведення попереднього аналізу виявилось, що пропусків даних немає, дані є повними. Для прогнозування було обрано змінну High. Найвища ціна може бути раннім сигналом про потенційний прорив, коли акція може різко зрости в ціні. Це може бути цікаво для трейдерів, які шукають короткострокові можливості для отримання прибутку. Також за її допомогою можна відслідковувати тренди. На рис. 3.2 показано графік змін змінної High у часі.

Рисунок 3.2 Візуалізація часового ряду

Нескладно помітити, що ряд є нестаціонарним, перевіримо це за допомогою тестів ADF та KPSS. Результати показані на рис. 3.3.

```
ADF test:
Test statistic: -0.49720831946754984
p-value: 0.8925601239887193
Series is not stationary


KPSS test:
Test statistic: 4.547493959603245
p-value: 0.01
Series is not stationary
```

Рисунок 3.3 Тести на стаціонарність

Як бачимо обидва тести вказали на нестаціонарність ряду. Для того, щоб позбутися нестаціонарності спробуємо взяти першу різницю.

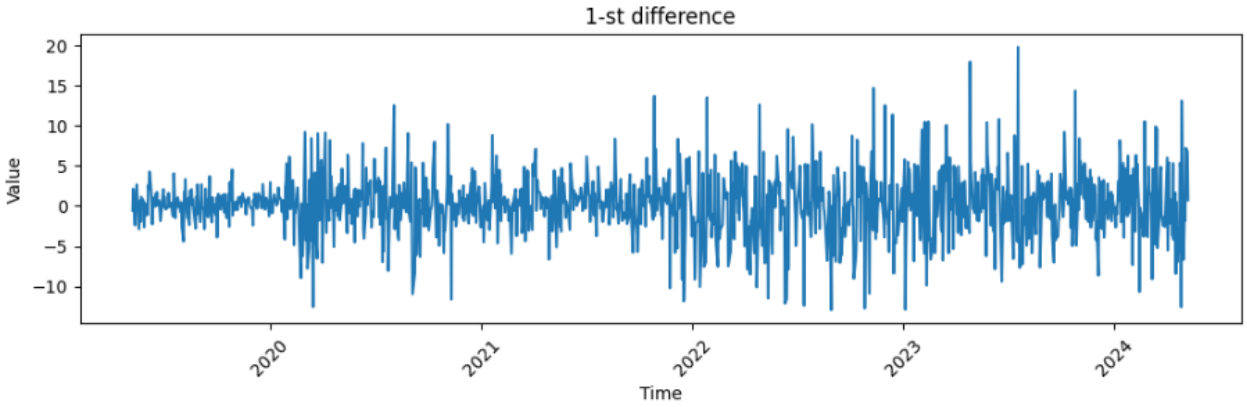


Рисунок 3.4 Графік 1-ої різниці

```
ADF test:
Test statistic: -15.177472723676189
p-value: 6.265242081314771e-28
Series is stationary


KPSS test:
Test statistic: 0.07762806191082204
p-value: 0.1
Series is stationary
```

Рисунок 3.5 Тести на стаціонарність

Обидва тести вказали на стаціонарність, отже можна зробити висновки, що 1-ої різниці достатньо, щоб привести ряд до стаціонарного. Тепер на

новому ряді побудуємо АКФ та ЧАКФ. Графіки для перших 40 лагів зображено на рис. 3.6-3.7.

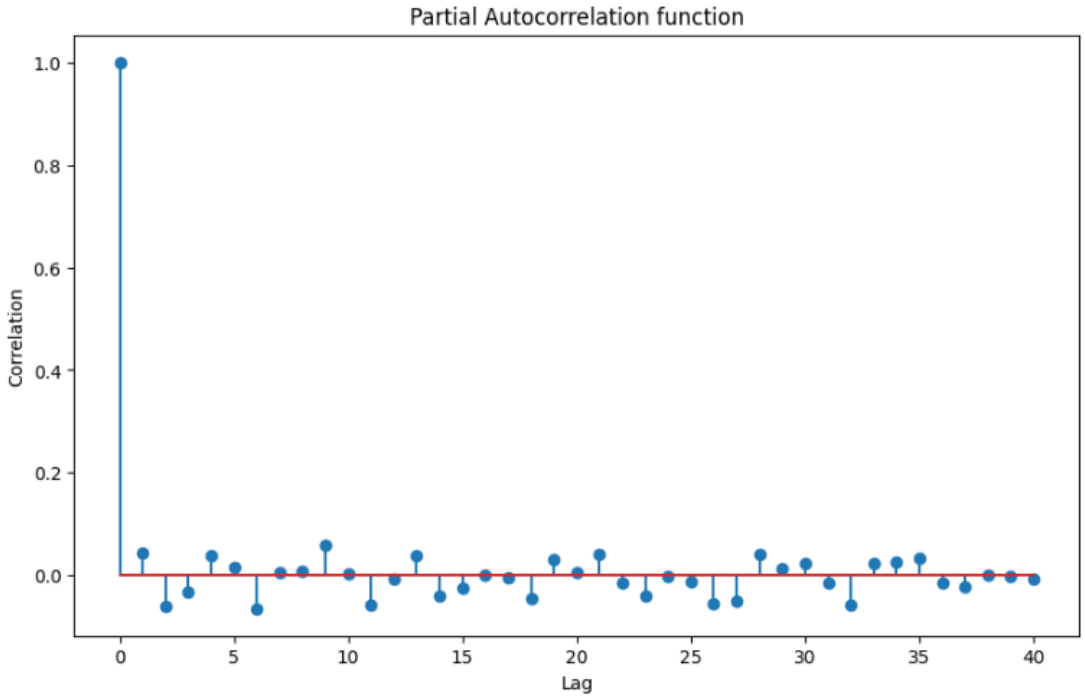


Рисунок 3.6 ЧАКФ

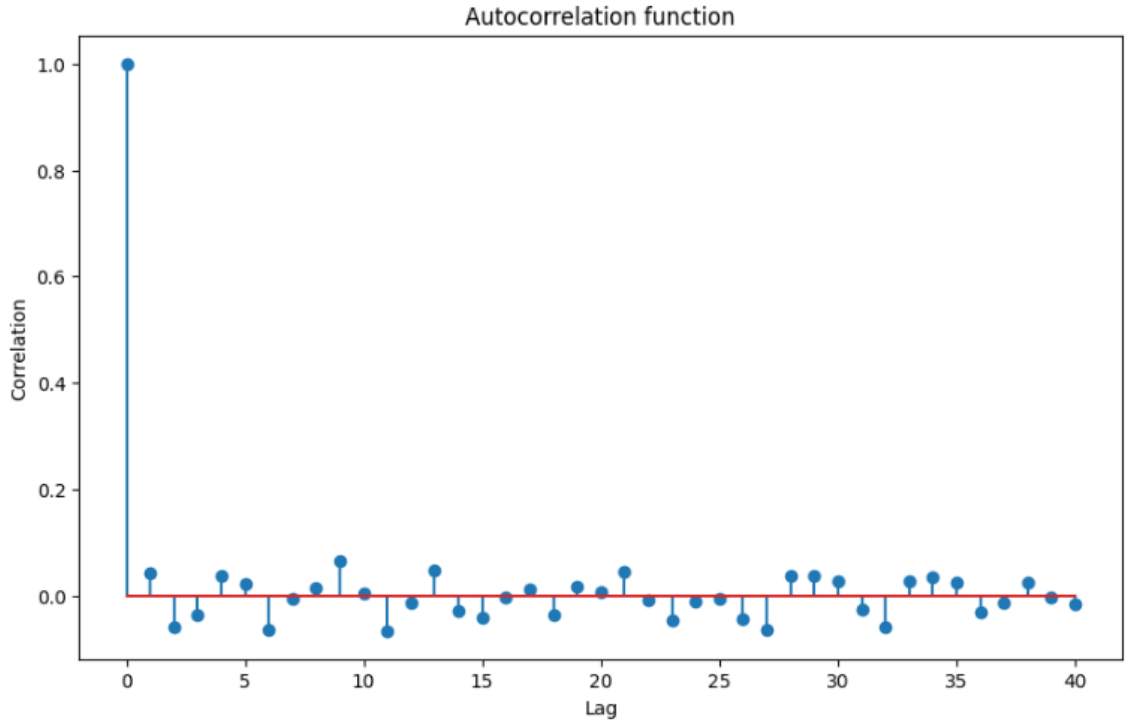


Рисунок 3.7 АКФ

З даних графіків можна зробити висновки, що лише 1 лаг явно впливає на змінну, ця інформація знадобиться при виборі порядку авторегресійної частини та порядку ковзного середнього в моделях ARIMA.

Для підготовки даних до прогнозування було проведено масштабування за допомогою z-нормалізації. Результати показані на рис. 3.8.

Date	Open	High	Low	Close	Volume
2019-05-08	-1.718755	-1.730431	-1.705662	-1.719640	-0.053972
2019-05-09	-1.733825	-1.737987	-1.721244	-1.719771	-0.149288
2019-05-10	-1.725700	-1.710108	-1.717943	-1.698401	0.147108
2019-05-13	-1.736183	-1.741114	-1.728243	-1.747959	0.391182
2019-05-14	-1.739328	-1.736815	-1.719528	-1.729866	-0.307946
...	...	...	...	...	...
2024-05-01	1.782197	1.856724	1.801028	1.812753	-0.445200
2024-05-02	1.848371	1.833405	1.858338	1.850774	-0.916712
2024-05-03	1.908911	1.927464	1.953545	1.966409	-0.937874
2024-05-06	1.993824	2.015791	2.013099	2.056610	-0.974133
2024-05-07	2.071137	2.025432	2.049016	2.001546	-0.731969

Рисунок 3.8 Дані після нормалізації

3.3 Побудова моделей

Для моделі ARIMA проаналізувавши АКФ і ЧАКФ було обрано порядок (1, 1, 1). На рис. 3.9 наведено результати моделювання.

SARIMAX Results

Dep. Variable:	value	No. Observations:	1133
Model:	ARIMA(1, 1, 1)	Log Likelihood	-3109.053
Date:	Sat, 18 May 2024	AIC	6224.107
Time:	17:37:09	BIC	6239.202
Sample:	0	HQIC	6229.809
	- 1133		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.2559	0.280	-0.915	0.360	-0.804	0.292
ma.L1	0.3419	0.272	1.256	0.209	-0.192	0.875
sigma2	14.2270	0.396	35.934	0.000	13.451	15.003

Ljung-Box (L1) (Q):	0.09	Jarque-Bera (JB):	359.52
Prob(Q):	0.76	Prob(JB):	0.00
Heteroskedasticity (H):	2.56	Skew:	0.20
Prob(H) (two-sided):	0.00	Kurtosis:	5.73

Рисунок 3.9 Модель ARIMA(1,1,1)

Цю модель будемо використовувати як базову. Її недоліком є те, що вона не враховує сезонність, тож на її основі побудуємо модель SARIMA, параметри якої будуть підібрані за допомогою решітчастого пошуку. Критерієм для вибору найкращої моделі було обрано інформаційний критерій Акаіке. На рис. 3.10 можемо побачити результати для найкращої моделі.

SARIMAX Results

Dep. Variable:	value	No. Observations:	1133
Model:	SARIMAX(1, 1, 1)x(0, 1, 1, 25)	Log Likelihood	-3083.019
Date:	Sat, 18 May 2024	AIC	6174.039
Time:	20:13:46	BIC	6194.077
Sample:	0	HQIC	6181.617
	- 1133		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.1809	0.296	-0.611	0.541	-0.761	0.399
ma.L1	0.2672	0.291	0.918	0.358	-0.303	0.837
ma.S.L25	-0.9733	0.020	-48.721	0.000	-1.012	-0.934
sigma2	14.4073	0.428	33.626	0.000	13.568	15.247

Ljung-Box (L1) (Q):	0.01	Jarque-Bera (JB):	273.99
Prob(Q):	0.91	Prob(JB):	0.00
Heteroskedasticity (H):	2.34	Skew:	0.15
Prob(H) (two-sided):	0.00	Kurtosis:	5.42

Рисунок 3.10 Модель SARIMA(1,1,1)(0,1,1,25)

Порівняємо моделі ARIMA та SARIMA за такими показниками як MAPE та MSE. Результати порівняння наведені в табл. 3.1

Таблиця 3.1

	ARIMA	SARIMA
MSE	2393.64	20.5589
MAPE	0.1088	0.0087

Як бачимо модель SARIMA показала значно кращі результати. Проведемо прогнозування поза вибіркою, тобто коли модель прогнозує значення для дня t і використовує це значення для прогнозування дня t+1. Результати зображено на рис. 3.11.

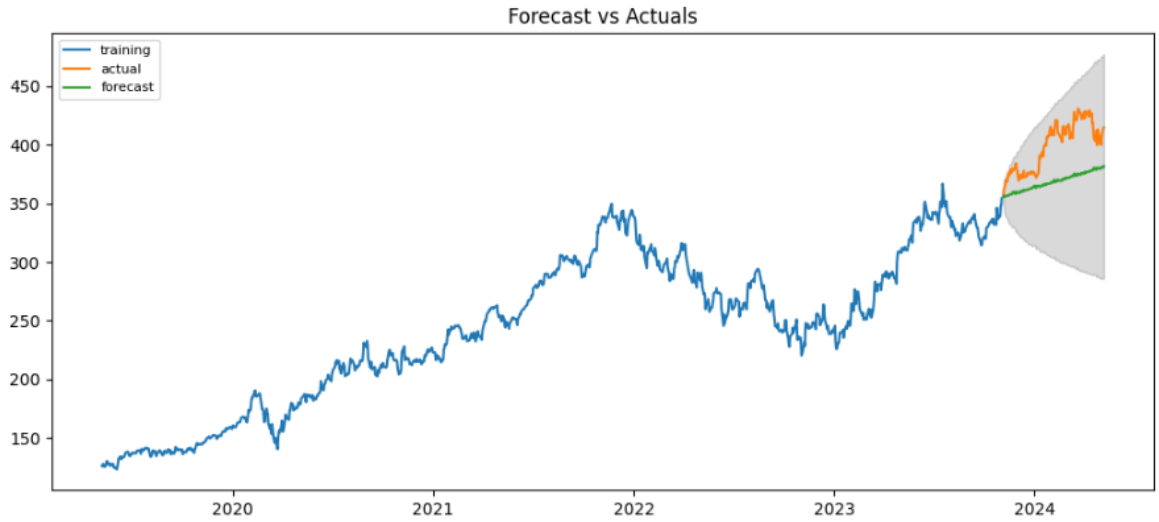


Рисунок 3.11 Прогнозування поза вибіркою

Далі перейдемо до нейронних мереж. Почнемо з найпростіших моделей. На вхід моделей подаються значення High, Low, Open, Close, Volume за останні 12 днів, на виході отримаємо змінну High на наступний день. На рис. 3.12 – 3.14 показано структуру моделей Simple LSTM, Stacked LSTM та BiLSTM.

```
Model: "Simple LSTM"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 64)                17920

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 17985 (70.25 KB)
Trainable params: 17985 (70.25 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Рисунок 3.12 Проста LSTM

```
Model: "Stacked_LSTM"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_7 (LSTM)               (None, 12, 32)            4864

 lstm_8 (LSTM)               (None, 32)                8320

 dense_5 (Dense)             (None, 1)                 33

=================================================================
Total params: 13217 (51.63 KB)
Trainable params: 13217 (51.63 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Рисунок 3.13 Багатошарова LSTM

```
Model: "BiLSTM"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional (Bidirection  (None, 128)               35840
 al)

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 35969 (140.50 KB)
Trainable params: 35969 (140.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Рисунок 3.14 Двонаправлена LSTM

Графічно оцінимо прогнози моделей на тестовій вибірці на рис. 3.15 – 3.17.

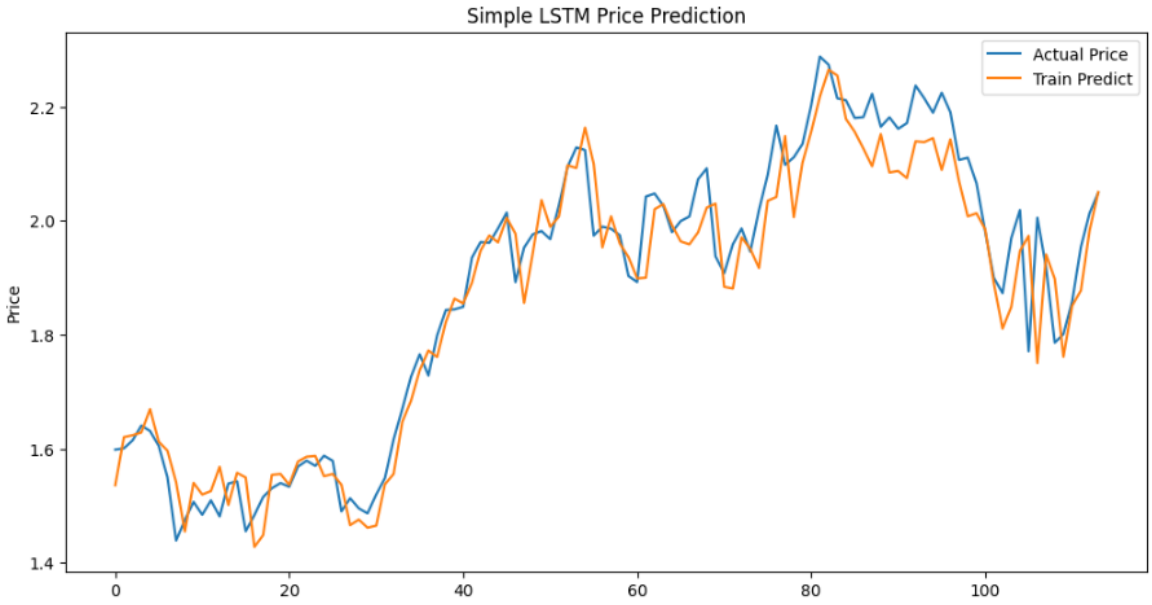


Рисунок 3.15 Проста LSTM

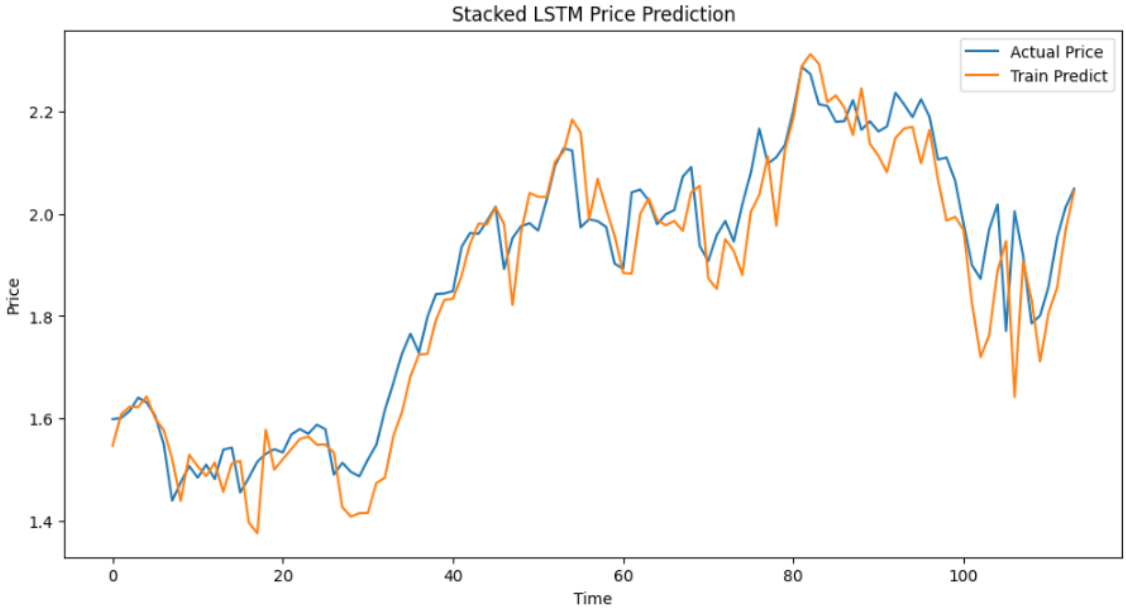


Рисунок 3.16 Багатошарова LSTM

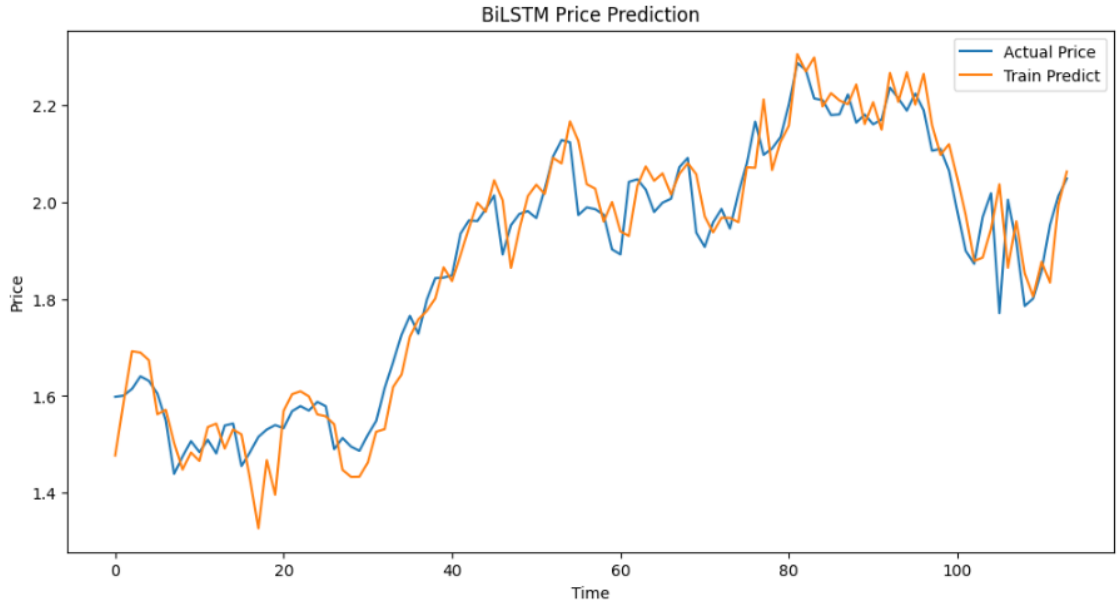


Рисунок 3.17 Двонаправлена LSTM

Порівняємо основні критерії даних моделей в таблиці 3.2

Таблиця 3.2

Модель	MAPE	MAE	MSE	RMSE
Simple LSTM	0.0091	3.7097	24.2945	4.9289
Stacked LSTM	0.0112	4.523	37.8378	6.1512
BiLSTM	0.0095	3.8211	24.5267	4.9524

Як бачимо найкращий результат показала проста LSTM з 64 комірками, що свідчить про те, що Simple LSTM модель прогнозує значення з найменшою середньою абсолютною відносною помилкою. BiLSTM показала практично ідентичні результати, а ось LSTM з двома послідовними шарами з 32 комірками показала гірший результат, що свідчить про те, що додавання ще одного шару LSTM не покращило точність прогнозування. Проста структура LSTM може бути більш ефективною для цього завдання, ніж більш складні моделі завдяки своїй простоті та ефективності.

Далі поєднаємо CNN з LSTM. Поєднання CNN та LSTM дозволяє моделі враховувати як локальні, так і довгострокові закономірності в часових рядах. Було побудовано дві моделі. Перша модель використовує 1D згортку для виявлення локальних закономірностей, а потім LSTM для довгострокових залежностей. Друга модель використовує потужніший ConvLSTM2D шар, який одночасно враховує локальні закономірності в часі та ознаках. Їх структуру можна побачити на рис. 3.18 – 3.19.

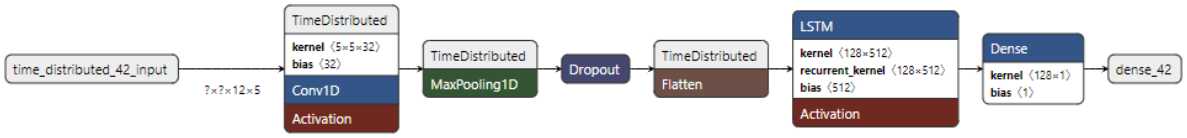


Рисунок 3.18 Структура Conv1D+LSTM

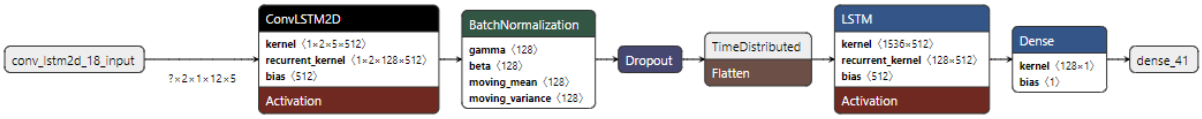


Рисунок 3.19 Структура Conv2D+LSTM

Подивимося на результати прогнозування в таблиці 3.3.

Таблиця 3.3

Модель	MAPE	MAE	MSE	RMSE
Conv1D+LSTM	0.0124	5.0163	37.2705	6.1049
Conv2D+LSTM	0.0098	3.9551	26.0453	5.1034

На основі всіх чотирьох метрик оцінювання (MAPE, MAE, MSE та RMSE), Conv2D+LSTM перевершує Conv1D+LSTM в даній задачі. Вона досягає нижчих рівнів похибки та свідчить про більш точне прогнозування цільових значень. Хоча варто відмітити, що навчання Conv2D+LSTM займає більше часу. Подивимось на графіки співставлення реальних даних і прогнозованих Conv2D+LSTM для тестової вибірки на рис. 3.20

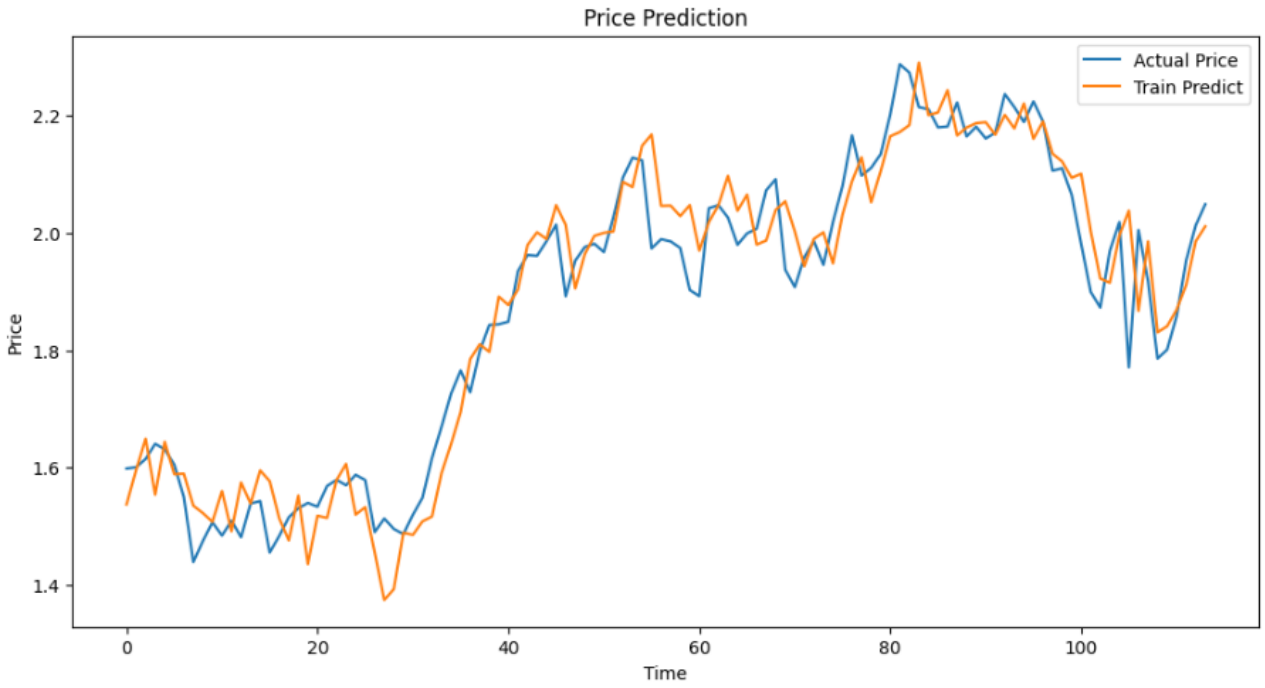


Рисунок 3.20 Прогноз Conv2D+LSTM

Далі побудуємо модель GRU. На рис. 3.21 зображено структурупобудованої моделі.

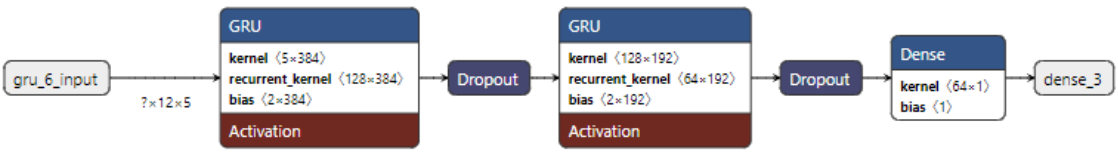


Рисунок 3.21 Структура моделі GRU

В таблиці 3.4 наведені основні метрики для прогнозу моделі на тестових даних.

Таблиця 3.4

MAPE	MAE	MSE	RMSE
0.008	3.2456	19.8908	4.4599

Дана модель показала найкращі результати серед інших моделей. Вона майже ідеально підходить для вирішення поставленої задачі. Також варто відмітити високу швидкість навчання в порівнянні з поєднанням CNN і LSTM.

3.4 Аналіз результатів

З результатів прогнозування видно, що всі вісім моделей, які були навчені, досягли хороших показників у завданні прогнозування. Кожна модель має свої сильні та слабкі сторони, тому вибір найкращої моделі залежить від конкретних потреб та пріоритетів. В таблиці 3.5 зібрані результати всіх моделей. Розберемо їх результати.

Таблиця 3.5

Модель	MAPE	MAE	MSE	RMSE
ARIMA	0.1088	-	2393.64	-
SARIMA	0.0087	-	20.5589	-
Simple LSTM	0.0091	3.7097	24.2945	4.9289
Stacked LSTM	0.0112	4.523	37.8378	6.1512
BiLSTM	0.0095	3.8211	24.5267	4.9524
Conv1D+LSTM	0.0124	5.0163	37.2705	6.1049
Conv2D+LSTM	0.0098	3.9551	26.0453	5.1034
GRU	**0.008**	**3.2456**	**19.8908**	**4.4599**

Модель ARIMA має найвищий MAPE серед усіх моделей, що свідчить про те, що вона має найгіршу точність прогнозування в абсолютних відсотках. Її MSE також є значним, що вказує на більшу похибку в прогнозуванні.

Модель SARIMA показала одні з найкращих результатів, поступившись лише GRU. Але прогнозування виконувалося на 1 день вперед, тобто модель знала значення за попередній день, а отже її доводилося перенавчати з кожним новим днем, що є дуже трудомістким процесом, що займає купу часу. Тож це треба враховувати при виборі моделі.

Моделі LSTM загалом показали хороші результати. Прогнози простої LSTM з 64 комірками виявилися найточнішими, що свідчить про те, що додавання додаткових шарів LSTM або додавання зворотних зв'язків не є ефективними для даної задачі.

Поєднання LSTM та CNN також покращень відносно простої LSTM не дають, так як показують або значно гірші(у випадку з Conv1D) або співставні(у випадку з Conv2D) результати, але вимагають набагато більше ресурсів і відповідно часу на навчання. Тому для нашої задачі таке поєднання не показало бажаних результатів.

Найкращою ж виявилась модель GRU, результати прогнозування якої є кращими по всім показникам. А також модель досить швидко навчається, її швидкість навчання співставна з моделями LSTM. Структура моделі складається з двох послідовних шарів GRU 128 і 64 комірок відповідно з використанням Dropout. Можна зробити висновки, що такі покращення досягаються саме за допомогою використання комірок GRU, так як послідовні шари LSTM давали гірші результати.

3.5 Висновки до розділу 3

В даному розділі було обґрунтовано вибір мови програмування Python для реалізації програмного продукту з прогнозування часових рядів. Python володіє численними перевагами для цієї задачі, такими як: простота використання, широкий спектр бібліотек, гнучкість та ефективність.

В якості даних для прогнозування було використано денні ціни на акції компанії Microsoft Corporation за період з 2019-05-08 по 2024-05-07. Було проведено попереднє очищення та аналіз даних, яке включало: перевірку на пропуски, вибір змінної для прогнозування, перевірку на стаціонарність, маштабування даних та аналіз автокореляції та часткової автокореляції.

Було побудовано та порівняно декілька моделей для прогнозування часових рядів цін на акції.

- ARIMA: авторегресійна інтегрована модель з ковзним середнім - класична модель для прогнозування стаціонарних часових рядів.
- SARIMA: сезонна ARIMA - розширення ARIMA, яке враховує сезонність в даних.
- LSTM: нейронна мережа, яка добре підходить для прогнозування часових рядів з довгостроковими залежностями.
- Stacked LSTM: багатошарова LSTM - модель, яка складається з декількох послідовних шарів LSTM.
- BiLSTM: двонаправлена LSTM - модель, яка може враховувати залежності як в прямому, так і в зворотному напрямку часового ряду.
- Conv1D+LSTM: поєднання 1D згортки та LSTM - модель, яка використовує згортку для виявлення локальних закономірностей в даних, а потім LSTM для довгострокових залежностей.
- Conv2D+LSTM: поєднання 2D згортки та LSTM - модель, яка використовує 2D згортку для виявлення локальних закономірностей в даних як в часі, так і в ознаках, а потім LSTM для довгострокових залежностей.

- GRU: Gated Recurrent Unit - нейронна мережа, схожа на LSTM, але з меншою кількістю параметрів.

Всі вісім побудованих моделей показали хороші результати в прогнозуванні цін на акції. ARIMA має найгіршу точність прогнозування, але є найпростішою моделлю. SARIMA покращує точність прогнозування ARIMA, але потребує значних обчислювальних ресурсів. Моделі LSTM показали хороші результати, проста LSTM з 64 комірками виявилась найефективнішою, додавання додаткових шарів LSTM не покращило точність прогнозування. Поєднання 1D згортки та LSTM не дало значних покращень в порівнянні з простою LSTM, а потребує значно більше ресурсів. Поєднання 2D згортки та LSTM показало кращі результати, ніж Conv1D+LSTM, але поступається GRU за точністю. Модель GRU з двома послідовними шарами 128 і 64 комірок виявилась найкращою моделлю для прогнозування цін на акції. Вона має найменші значення MAPE, MAE, MSE та RMSE, а також швидко навчається.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ

Цей розділ присвячений аналізу ринку для програмного продукту, який використовує машинне навчання та аналіз даних для прогнозування цін на акції Microsoft. Мета аналізу:

- Описати основні характеристики продукту;
- Визначити цільову аудиторію та їхні потреби;
- Оцінити потенціал ринку та конкурентне середовище.

Функціонально-вартісний аналіз (ФВА) - це метод системного дослідження функцій об'єкта з метою пошуку балансу між його собівартістю і корисністю. Іншими словами, ФВА прагне оптимізувати конструкцію, технологію та інші аспекти продукту, щоб максимально зменшити його витрати, зберігаючи при цьому його ключові функції та корисність для користувача.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи прогнозу стійкості фінансових показників. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Технічні вимоги до програмного продукту є наступні:

- функціонування на персональних комп’ютерах із стандартним набором компонентів;

- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція $F0$ – розробка можливого програмного продукту, яка дозволяє аналізувати різні характеристики, що безпосередньо впливають на стійкість підприємства. Беручи за основу цю функцію, можна виділити наступні:

$F1$ – вибір середовища розробки.

$F2$ – якісний аналіз даних.

$F3$ – графічні показники.

Кожна з цих функцій має декілька варіантів реалізації:

Функція $F1$:

а) Eviews;

б) Jupiter Notebook.

Функція $F2$:

а) Застосування вбудованих функцій та бібліотек;

б) Створення своїх функцій для обчислень та перетворень.

Функція $F3$:

а) Використання шаблонних графіків;

б) Створення власних.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1).

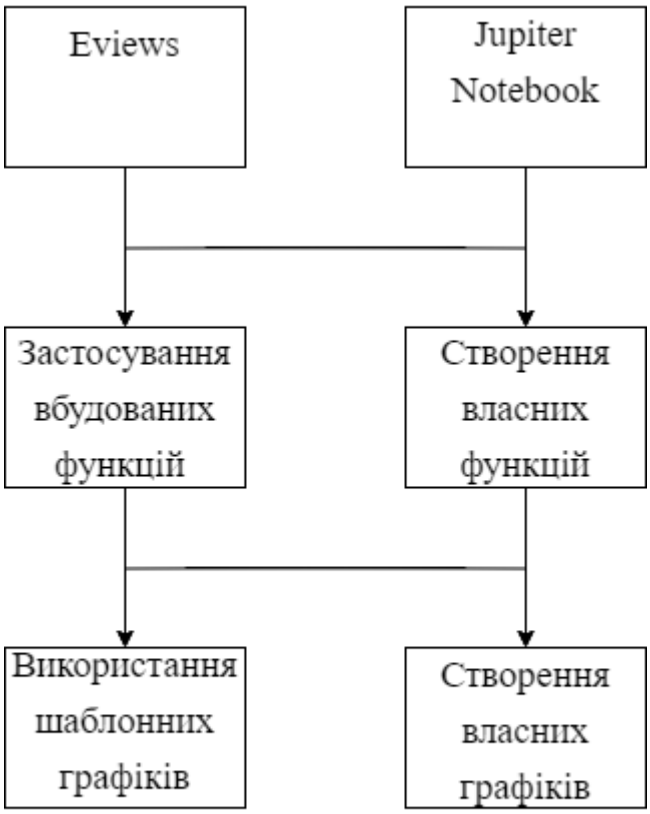


Рисунок 4.1 Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

Таблиця 4.1

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	А	Має зручний графічний інтерфейс. Простий у використанні для початківців.	Менш гнучкий, ніж інші програмні пакети.
	Б	Дозволяє створювати власні коди та візуалізації. Гнучкий та потужний.	Потребує більше ресурсів
F2	А	Знижує необхідність писати власний код. Забезпечує надійність та точність результатів.	Може бути менш гнучким, ніж створення власних функцій. Не завжди доступні всі необхідні функції.
	Б	Повна гнучкість та контроль над процесом. Можливість створити функції, які відповідають вашим конкретним потребам.	Може бути більш часозатратним та трудомістким.
F3	А	Швидкий та простий спосіб створити графіки.	Може бути менш гнучким, ніж створення власних графіків. Не завжди доступні всі необхідні типи графіків.
	Б	Повна гнучкість та контроль над зовнішнім виглядом графіка.	Може бути більш часозатратним та трудомістким.

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція $F1$: Перевагу даємо гнучкості та кількості вбудованих бібліотек. Для зручного коду під конкретні задачі варіант А має бути відкинутий.

Функція *F*2: Програма допускає обрання обох варіантів. Можливо використати варіанти А чи Б.

Функція *F*3: Реалізація першого варіанту є сприйнятливою для програми. Це варіант А. Таким чином, будемо розглядати такий варіанти реалізації ПП:

*F*1б – *F*2а – *F*3а

*F*1б – *F*2б – *F*3а

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, які були розглянуті раніше, ми визначаємо основні параметри вибору, які будуть використовуватися для розрахунку коефіцієнта технічного рівня програмного продукту. Для характеристики програмного продукту ми використовуємо наступні параметри:

X1 – Швидкодія мови програмування;

X2 – Обсяг пам'яті для обчислень та збереження даних;

X3 – Час навчання даних;

X4 – Потенційний обсяг програмного коду.

Гірші, середні і кращі значення цих параметрів вибираються на основі вимог замовника та умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 4.2.

Таблиця 4.2

Назва параметра	Умовні позначення	Одиниці вимірювання	Значення параметра		
			Гірші	Середні	Кращі
Швидкодія мови програмування	X1	оп/мс	100	1000	10000

Продовження таблиці 4.2

Обсяг пам'яті для обчислень та збереження даних	X2	Мб	100	50	10
Час навчання даних	X3	мс	100000	50000	25000
Потенційний обсяг програмного коду	X4	кількість рядків коду	1200	500	300

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5

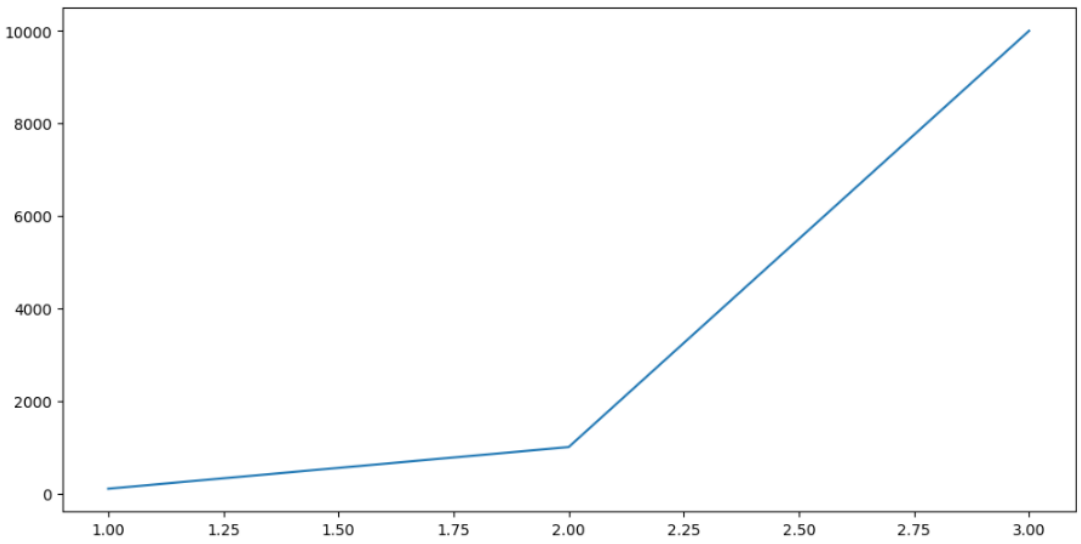


Рисунок 4.2 Швидкодія мови програмування (оп/мс), X1

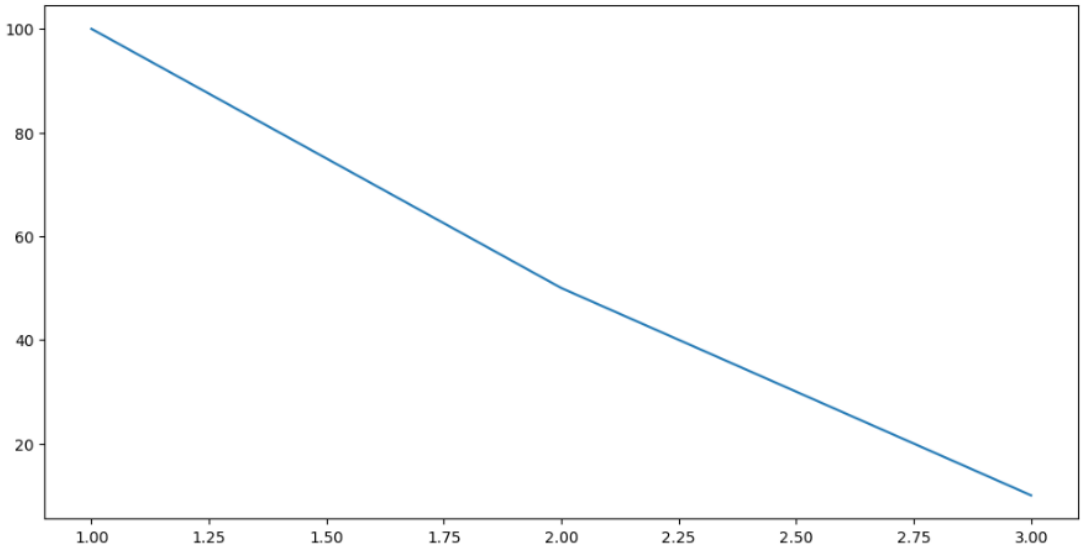


Рисунок 4.3 Об’єм обчислювальних ресурсів (мб), X2

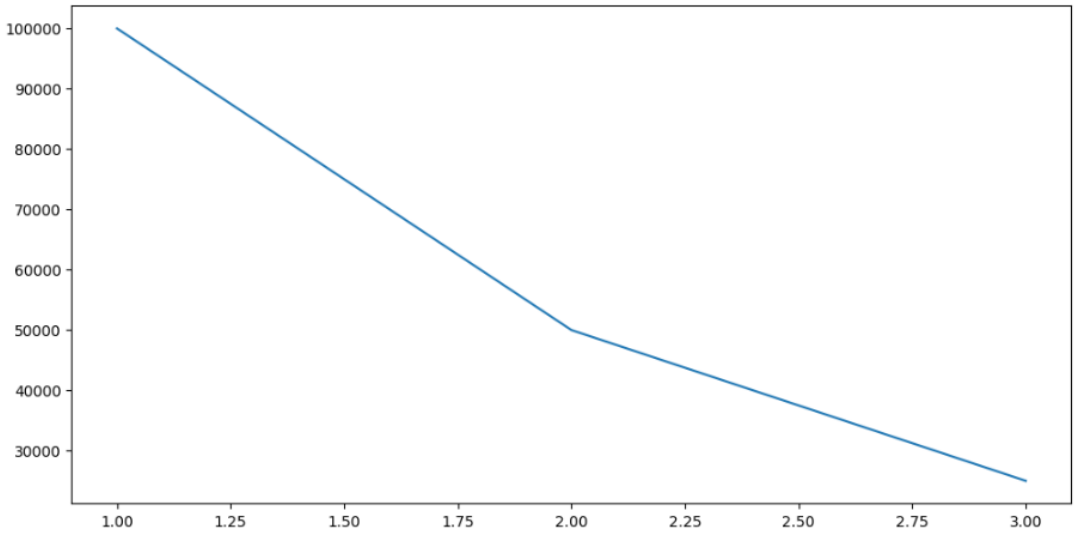


Рисунок 4.4 Час навчання моделі (хв), X3

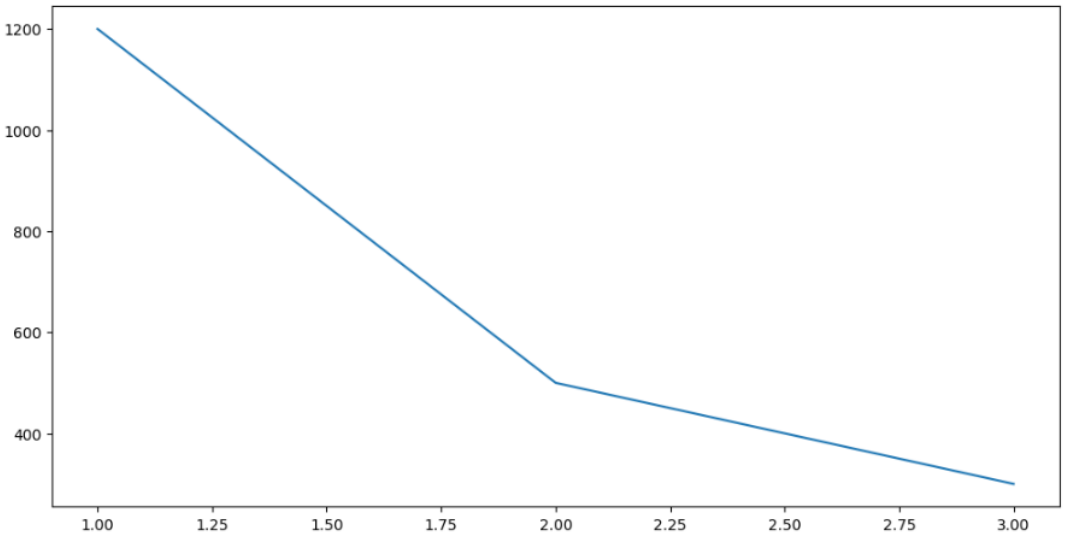


Рисунок 4.5 Потенційний об’єм програмного коду, X4

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3

Назва параметра	**Ранг параметра за оцінкою експерта**							**Сума рангів,R_i**	**Відхи-лення, Δ_i**	**Δ_i^2**
	1	**2**	**3**	**4**	**5**	**6**	**7**			
X1	1	2	2	3	2	2	2	13	-4.5	20.25
X2	3	3	4	2	3	3	1	19	1.5	2.25
X3	2	1	1	1	1	1	3	10	-7.5	56.25
X4	4	4	3	4	4	4	4	27	9.5	90.25
Разом	10	10	10	10	10	10	10	70	0	

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

a) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70$$

де N – число експертів,

n – кількість параметрів.

b) Середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17.5$$

c) Відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0

d) Загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 169$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 * 169}{7^2(4^3 - 4)} = 0.69 > W_k = 0.67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0.67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4

Таблиця 4.4

Параметри	**Експерти**							**Підсумкова оцінка**	**Числове значення коефіцієнтів переваги (x_{ij})**
	1	**2**	**3**	**4**	**5**	**6**	**7**		
x_1 та x_2	>	>	>	<	>	>	<	>	1,5
x_1 та x_3	>	<	<	<	<	<	>	<	0,5
x_1 та x_4	>	>	>	>	>	>	>	>	1,5
x_2 та x_3	<	<	<	<	<	<	>	<	0,5
x_2 та x_4	>	>	<	>	>	>	>	>	1,5
x_3 та x_4	>	>	>	>	>	>	>	>	1,5

Числове значення, що визначає ступінь переваги і–го параметра над j–тим, аіj визначається по формулі:

$$a_{ij} = \begin{cases} 1.5, X_i > X_j \\ 1.0, X_i = X_j \\ 0.5, X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю A=‖aij‖.

Для кожного параметра зробимо розрахунок вагомості Кві за наступними формулами:

$$K_{ві} = \frac{b_i}{\sum_{i=1}^{n} b_i}$$

$$b_i = \sum_{i=1}^{n} a_{ij}$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{ві} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}$$

$$b_i' = \sum_{i=1}^{n} a_{ij} b_j$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 5%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5

X_i	**Параметри Xj,**				**Перша ітерація**		**Друга ітерація**	
	X_1	X_2	X_3	X_4	**$в_i$**	**$К_{ві}$**	**$в'_i$**	**$К'_{ві}$**
X_1	1,0	1,5	0,5	1,5	4,5	0,28125	16,25	0,2754
X_2	0,5	1,0	0,5	1,5	3,5	0,21875	12,25	0,2076
X_3	1,5	1,5	1,0	1,5	5,5	0,34375	21,25	0,35
X_4	0,5	0,5	0,5	1,0	2,5	0,15625	9,25	0,156
Всього					16,0	1,0	59	1,0

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо. Абсолютні значення параметрів X2 (об'єм пам'яті), X3 (час навчання) та X4 (потенційний об'єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП. Абсолютне значення параметра X1 (швидкість роботи мови програмування) обрано не найгіршим. Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi} B_{ij},$$

де n – кількість параметрів;

Kвi – коефіцієнт вагомості і–го параметра;

Bi – оцінка і–го параметра в балах.

Таблиця 4.6

Основна функція,	**Варіант реалізації функції**	**Параметри, які беруть участь у реалізації функцій**	**Абсолютне значення параметра**	**Оцінка параметра в балах, Bi**	**Коефіцієнт вагомості, $К_{вi}$**	**Показник технічного рівня, $К_{тр}$ [Fi]**
F1	б	X_1	900	4	0,28	1,25
F2	а	X_2	35	8	0,21	1,68
	б	X3	25000	10	0,35	3,5
Fi	а	X4	450	5	0,16	0,8

Показник рівня якості *k*-го варіанта реалізації основних функцій виробу

$$K_к = K_{тр}\,[F_{1k}] + K_{тр}[F_{2k}] + \ldots + K_{тр}\,[F_{zk}]\ ,$$

де $K_{тр}\,[F_{1k}]$ - показник технічного рівня першої функції K-го варіанта реалізації основних функцій виробу.

Визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 1,25 + 1,68 + 0,8 = 3,73$$
$$K_{K2} = 1,25 + 3,5 + 0,8 = 5,55$$

Як видно з розрахунків, кращим є варіант 2, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості. Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних. Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як:

$$T_o = T_p \bullet K_n \bullet K_{ck} \bullet K_M \bullet K_{cm} \bullet K_{cm.M}$$

де ТР – трудомісткість розробки ПП;

КП – поправочний коефіцієнт;

КСК – коефіцієнт на складність вхідної інформації;

КМ – коефіцієнт рівня мови програмування;

КСТ – коефіцієнт використання стандартних модулів і прикладних програм;

КСТ.М – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: ТР = 92 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: КП = 1,45. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх завдань рівний 1: КСК = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта КСТ = 0.8. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 92 * 1.45 * 0.8 = 106,7$$

Проведемо аналогічні розрахунки для подальших дій. Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто ТР =69 людино-днів, КП =0,72, КСК = 1, КСТ =0.9:

$$T_2 = 69 * 0.72 * 0.9 = 44,7$$

Складаємо трудомісткість відповідних завдань щоб отримати їх трудомісткість:

$$T_I = (106,7 + 44,7 + 44,7) * 8 = 1568,8$$

$$T_{II} = (106,7 + 106,7 + 44,7) * 8 = 2064,8$$

В розробці беруть участь два програмісти з окладом 30000 грн., один аналітик в області даних з окладом 35000. Визначимо середню зарплату за годину за формулою:

$$C_ч = \frac{M}{T_m t} грн.,$$

де М – місячний оклад працівників;

Tm – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$C_{ч} = \frac{30000 + 30000 + 35000}{3 * 21 * 8} = 188,49 грн.$$

Тоді, розрахуємо заробітну плату за формулою:

$$C_{ЗП} = C_{ч} T_i K_{Д},$$

де СЧ– величина погодинної оплати праці програміста;

Тi – трудомісткість відповідного завдання;

КД – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$I.C_{ЗП} = 188.49 * 1568.8 * 1.2 = 354843.7$$

$$II.C_{ЗП} = 188.49 * 2064.8 * 1.2 = 467032.98$$

Відрахування на єдиний соціальний внесок становить 22%:

$$I.C_{ВІД} = C_{ЗП} * 0,22 = 354843.7 * 0,22 = 78065,6$$

$$II.C_{ВІД} = C_{ЗП} * 0,22 = 467032.98 * 0,22 = 102747,26$$

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 30000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 30000 \cdot 0,2 = 72000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 72000 \cdot (1 + 0.2) = 86400 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 48960 \cdot 0,22 = 19008 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 10000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.4 \cdot 0.12 \cdot 10000 = 1680 \text{ грн.},$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.4 \cdot 10000 \cdot 0.08 = 1120 \text{ грн.},$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 12 – 16) \cdot 8 \cdot 0.35 =$$

$$= 627,2 \text{ години},$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 627,2 \cdot 0,2 \cdot 0,3 \cdot 5,22 = 196,44 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 10000 \cdot 0,67 = 6700 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$С_{ЕКС} = 86400 + 19008 + 1680 + 1120 + 196,44 + 6700 = 115104,44$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 115104,44 / 627,2 = 183,52 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М-Г} \cdot Т, \qquad (4.18)$$

I. $С_М = 183,52 \cdot 1568,8 = 287796,06$ грн.

II. $C_М = 183,52 \cdot 2064,8 = 378932,1$ грн.

Накладні витрати складають 67% від заробітної плати:

$$C_Н = C_{ЗП} \cdot 0,67, \quad (4.19)$$

I. $C_Н = 354843.7 \cdot 0,67 = 237745,28$грн.

II. $C_Н = 467032.98 \cdot 0,67 = 312912,1$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н$$

I. $C_{ПП} = 354843.7 + 78065,6 + 287796,06 + 237745,28 = 958450,64$ грн.

II. $C_{ПП} = 467032.98 + 102747,26 + 378932,1 + 312912,1 = 1261624,44$ грн.

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР_j} = К_{Кj} / C_{Фj},$$

$К_{ТЕР1} = 3,73 / 958450,64 = 3,891 \cdot 10^{-6}$,

$К_{ТЕР2} = 5,55 / 1261624,44 = 4,399 \cdot 10^{-6}$.

Як бачимо, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1}= 4,399 \cdot 10^{-6}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є другий варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 4,399 \cdot 10^{-6}$.

Цей варіант реалізації програмного продукту має такі параметри:

– Вибір середовища розробки – Jupiter notebook;

– Якісний аналіз даних за допомогою власних створених функцій;

– Використання шаблонних варіантів для побудови графіків.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, швидку реалізацію програми та доступний функціонал для роботи.

4.8 Висновки до розділу 4

В даному розділі було проведено повний функціонально-вартісний аналіз програмного продукту. Також було знайдено оцінку основних функцій програмного продукту. Також в даному дослідженні показано різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що має вплив на економічні фактори та сумісність з майбутнім програмним продуктом. В результаті виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, було визначено та проведено оцінку основних функцій програмного продукту, а також знайдено

параметри, які його характеризують. На основі аналізу вибрано варіант реалізації програмного продукту.

ВИСНОВКИ

У даній роботі досліджено теоретичні та практичні аспекти прогнозування економічних показників за часовими рядами. Було проаналізовано актуальність економічного прогнозування, особливості та методи, що використовуються в цій сфері. Виявлено, що прогнозування вартості фінансових інструментів має велике значення для економічної політики держави та діяльності великих компаній. Проаналізовано характеристики часових рядів, такі як стаціонарність та гетероскедастичність. Оглянуто програмні засоби для прогнозування часових рядів, такі як Eveiws та Python. Описано математичні моделі прогнозування часових рядів, включаючи ARIMA, SARIMA, нейронні мережі (RNN, LSTM, GRU, WaveNet). Запропоновано підходи до підготовки даних для прогнозування, включаючи заповнення пропусків та масштабування. Розглянуто критерії оцінювання моделей прогнозування. Реалізовано програмний продукт на мові програмування Python для прогнозування економічних показників за часовими рядами. Виконано порівняльний аналіз результатів роботи різних моделей на основі обраних критеріїв оцінювання. Виявлено, що сучасні методи, такі як нейронні мережі, показують кращі результати в порівнянні з традиційними методами, такими як ARIMA.

В результаті дослідження отримали наступне.

- Розроблено програмний продукт для прогнозування цін на акції за часовими рядами.
- Здійснено порівняльний аналіз різних моделей прогнозування та визначено найбільш ефективну модель.
- Запропоновані моделі можуть бути використані для прогнозування цін на акції, що є актуальним для інвесторів, компаній та державних органів. Це дозволить більш ефективно планувати діяльність та приймати обґрунтовані економічні рішення.

Таким чином, проведене дослідження підтвердило ефективність сучасних методів прогнозування на основі нейронних мереж та їх переваги у порівнянні з традиційними методами, що відкриває нові перспективи для подальших досліджень та розробок в цій галузі.

Для подальших досліджень можна розглянути наступні напрямки.

- Використання альтернативних джерел даних, таких як соціальні мережі, новини та економічні індикатори для покращення прогностичних можливостей моделей.
- Розробка функціоналу для автоматизованого оновлення моделей у реальному часі з врахуванням нових даних дозволить підтримувати актуальність прогнозів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Рибачук Ю.О. Особливості застосування технічного аналізу фінансового ринку при прогнозування його інструментів. Науковий вісник Міжнародного гуманітарного університету, 2015, 12. С. 223–226.
2. Аналіз господарської діяльності. Г. Даценко та ін.; Київ. нац. торг.-екон. ун-т. Вінниц. торг.-екон. ін-т. Вінниця, 2021. 416 с.
3. Бідюк П.І., Романенко В.Д., Тимощук О.Л. Аналіз часових рядів (навчальний посібник) Київ: Політехніка, 2010. 317 с.
4. Radzikhovska Larisa, Husak Lyudmila, Panchuk Yulia. Побудова багатофакторної регресійної моделі засобами програмного забезпечення Eviews. Computer-integreted technologies, science, production, 2021. P. 54-59.
5. Що таке Tensorflow?. URL: https://www.oksim.ua/2023/08/07/shho-take-tensorflow/
6. Autoregressive Integrated Moving Average (ARIMA) Prediction Model. URL: https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp
7. Seasonal ARIMA models. URL: https://otexts.com/fpp2/seasonal-arima.html
8. Нейромережа — що це таке, як працює та навіщо потрібна. URL: https://termin.in.ua/neyromerezha/
9. Sulekha Aloorravi Mastering Time Series Analysis and Forecasting with Python: Bridging Theory and Practice Through Insights, Techniques, and Tools for Effective Time Series Analysis in Python (English Edition), Orange Education Pvt Ltd, 2024. 322 p.

10. Geron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques. O'Reilly Media, 2019. 856 p.
11. Alex Graves, Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks, July 2005, 18(5–6). P. 602–610.
12. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. K. Cho, B. van Merriënboer, C. Gulcehre et. al. Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014. P. 1724–1734.
13. WaveNet: A Generative Model for Raw Audio. Aaron van den Oord et. al., 2016. 15 p.
14. Pankka Hanna, Lehtinen, Jaakko, Ilmoniemi Risto, Roine Timo. Forecasting EEG time series with WaveNet, 2024. 16 p.
15. Heteroscedasticity Definition: Simple Meaning and Types Explained. URL: https://www.investopedia.com/terms/h/heteroskedasticity.asp

ДОДАТОК А ЛІСТИНГИ ПРОГРАМНОГО ПРОДУКТУ

ARIMA.py

```
# -*- coding: utf-8 -*-

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1kuAJEGKown-
WF_NVgibiR-utaahnG3u-
"""

# Mounting GDrive.
import sys
from google.colab import drive

MOUNT_POINT = "/content/drive"
drive.mount(MOUNT_POINT)

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.tsa.arima.model import ARIMA

df = pd.read_csv('/content/drive/MyDrive/TS/MSFT.csv')
df.head()

def plot_data(index, data,  title):
    plt.plot(index, data)
    plt.xlabel('Time')
    plt.ylabel('Value')
    plt.xticks(rotation = 45)
    plt.title(title)
```

```
        plt.show()

    #df['high']       =       df['Цена'].apply(lambda       x:
float(x.replace(',', '.')))
    date_rng = df['Date'].apply(lambda x: pd.to_datetime(x))
    data = df['High'].to_numpy()
    ts = pd.DataFrame(data, index=date_rng, columns=['value'])
    ts.sort_index(inplace=True)

    from statsmodels.tsa.stattools import adfuller

    result = adfuller(ts.value.diff().dropna())
    print('ADF Statistic: %f' % result[0])
    print('p-value: %f' % result[1])

    plot_data(ts.index, ts['value'], 'Maximum price in a trading
day')

    plot_data(ts.index[15:47],
ts['value'].diff().diff(14).dropna()[:32], 'Maximum price in a
trading day')

    import statsmodels.api as sm

    pacf = sm.tsa.pacf(ts.diff().dropna(), nlags=20)

    # Візуалізація автокореляційної функції
    plt.figure(figsize=(10, 6))
    plt.stem(pacf)
    plt.title('Partial Autocorrelation function')
    plt.xlabel('Lag')
    plt.ylabel('Correlation')
    plt.show()

    acf = sm.tsa.acf(ts.diff().dropna(), nlags=40)
```

```
# Візуалізація автокореляційної функції
plt.figure(figsize=(10, 6))
plt.stem(acf)
plt.title('Autocorrelation function')
plt.xlabel('Lag')
plt.ylabel('Correlation')
plt.show()

def sumSquares(Y, Ymodeled) -> float:
      return np.sum((Y - Ymodeled)**2)

def r2(Y, Ymodeled) -> float:
    return np.var(Ymodeled)/np.var(Y)

model = ARIMA(ts['value'], order=(1, 1, 1))

model_fit = model.fit()
print(model_fit.summary())

from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_percentage_error

pred = model_fit.predict()
mape = mean_absolute_percentage_error(ts, pred)
mse = mean_squared_error(ts, pred)
print(f"MSE: {mse}, MAPE: {mape}")

plt.plot(ts.index[-50:], ts['value'][-50:])
plt.plot(pred.index[-50:], pred[-50:])
plt.xlabel('Time')
plt.ylabel('Value')
plt.xticks(rotation = 45)
plt.title("ARIMA(1,1,1)")
plt.show()
```

```
sumSquares(ts['value'], pred)

r2(pred, ts['value'])

train_size = 0.9
train = ts.value[:int(len(ts.value)*train_size)]
test = ts.value[int(len(ts.value)*train_size):]

model = ARIMA(train, order=(1, 1, 1))
fitted = model.fit()
print(fitted.summary())

# Forecast
fcst    =    fitted.get_forecast(int(len(ts.value)*(1    -
train_size)) + 1)
fc = fcst.predicted_mean
se = fcst.se_mean
conf = fcst.conf_int(0.95).to_numpy()  # 95% conf

# Make as pandas series
fc_series = pd.Series(fc, index=test.index)
lower_series = pd.Series(conf[:, 0], index=test.index)
upper_series = pd.Series(conf[:, 1], index=test.index)

# Plot
plt.figure(figsize=(12,5), dpi=100)
plt.plot(train, label='training')
plt.plot(test, label='actual')
plt.plot(test.index, fc, label='forecast')
plt.fill_between(lower_series.index,          lower_series,
upper_series,
                 color='k', alpha=.15)
plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=8)
```

```
plt.show()

def forecast_accuracy(forecast, actual):
    mape = np.mean(np.abs(forecast - actual)/np.abs(actual))
# MAPE
    me = np.mean(forecast - actual)             # ME
    mae = np.mean(np.abs(forecast - actual))    # MAE
    mpe = np.mean((forecast - actual)/actual)   # MPE
    rmse = np.mean((forecast - actual)**2)**.5  # RMSE
    corr = np.corrcoef(forecast, actual)[0,1]   # corr
    mins = np.amin(np.hstack([np.array(forecast)[:,None],
                              np.array(actual)[:,None]]),
axis=1)
    maxs = np.amax(np.hstack([np.array(forecast)[:,None],
                              np.array(actual)[:,None]]),
axis=1)
    minmax = 1 - np.mean(mins/maxs)             # minmax
    mse = mean_squared_error(actual, forecast)            #
MSE
    return({'mape':mape, 'me':me, 'mae': mae,
            'mpe': mpe, 'rmse':rmse,
            'corr':corr, 'minmax':minmax, 'mse': mse})

forecast_accuracy(fc, test.values)

ts

mod_sarimax = sm.tsa.SARIMAX(train, order=(1,1,1),
                             seasonal_order=(0,1,1,25))
res_sarimax = mod_sarimax.fit()

# Show the summary of results
print(res_sarimax.summary())
```

```
      fcst   =   res_sarimax.get_forecast(int(len(ts.value)*(1   -
train_size)) + 1)
      fc = fcst.predicted_mean
      forecast_accuracy(fc, test.values)

      fc = fcst.predicted_mean
      se = fcst.se_mean
      conf = fcst.conf_int(0.95).to_numpy()  # 95% conf

      # Make as pandas series
      fc_series = pd.Series(fc, index=test.index)
      lower_series = pd.Series(conf[:, 0], index=test.index)
      upper_series = pd.Series(conf[:, 1], index=test.index)

      # Plot
      plt.figure(figsize=(12,5), dpi=100)
      #plt.plot(train, label='training')
      plt.plot(test, label='actual')
      plt.plot(test.index, fc, label='forecast')
      plt.fill_between(lower_series.index,          lower_series,
upper_series,
                       color='k', alpha=.15)
      plt.title('Forecast vs Actuals')
      plt.legend(loc='upper left', fontsize=8)
      plt.show()

      se = fcst.se_mean
      conf = fcst.conf_int(0.95).to_numpy()  # 95% conf

      # Make as pandas series
      fc_series = pd.Series(fc, index=test.index)
      lower_series = pd.Series(conf[:, 0], index=test.index)
      upper_series = pd.Series(conf[:, 1], index=test.index)

      # Plot
```

```
      plt.figure(figsize=(12,5), dpi=100)
      plt.plot(train, label='training')
      plt.plot(test, label='actual')
      plt.plot(test.index, fc, label='forecast')
      plt.fill_between(lower_series.index,          lower_series,
upper_series,
                       color='k', alpha=.15)
      plt.title('Forecast vs Actuals')
      plt.legend(loc='upper left', fontsize=8)
      plt.show()

      pred = res_sarimax.predict(start=len(train)+1, end=len(ts))
      test.index.shape

      plt.figure(figsize=(12,5), dpi=100)
      #plt.plot(train, label='training')
      plt.plot(test, label='actual')
      plt.plot(test.index, pred, label='predictions')

      plt.title('Forecast vs Actuals')
      plt.legend(loc='upper left', fontsize=8)
      plt.show()

      res_sarimax.predict(len(ts)+1)

      forecast = []
      for i in range(len(test)):
          model        =        sm.tsa.SARIMAX(ts[:len(train)+i],
order=(1,1,1),
                                  seasonal_order=(0,1,1,25))
          res = model.fit()
          pred = res.predict(len(train)+i)
          forecast.append(pred)
      forecast
```

```
arr = [i.to_numpy()[0] for i in forecast]

forecast_accuracy(arr, test.values)

plt.figure(figsize=(12,5), dpi=100)

plt.plot(test, label='actual')
plt.plot(test.index, arr, label='predictions')

plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=8)
plt.show()

mod_sarimax = sm.tsa.SARIMAX(ts, order=(1,1,1),
                             seasonal_order=(0,1,1,25))
res_sarimax = mod_sarimax.fit()

pred = res_sarimax.predict()
mape = mean_absolute_percentage_error(ts, pred)
mse = mean_squared_error(ts, pred)
print(f"MSE: {mse}, MAPE: {mape}")
```

LSTM.py

```
# -*- coding: utf-8 -*-
"""new_lstm.ipynb

Automatically generated by Colab.

Original file is located at

https://colab.research.google.com/drive/15HDHFMiCMAnPgsmauK2E60EoF13TUwnl
"""
```

```
# Mounting GDrive.
import sys
from google.colab import drive

MOUNT_POINT = "/content/drive"
drive.mount(MOUNT_POINT)

import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
import math
from sklearn.metrics import mean_squared_error
tf.keras.utils.set_random_seed(42)
tf.random.set_seed(42)

df = pd.read_csv('/content/drive/MyDrive/TS/MSFT.csv')
df.head(7)

df.drop(['Date'], axis=1)

from sklearn.preprocessing import StandardScaler
X_columns = ['Open', 'High', 'Low', 'Close', 'Volume']
y_columns = ['High']

scaler_x = StandardScaler()
scaler_y = StandardScaler()
x_scaled = scaler_x.fit_transform(df[X_columns])
y_scaled = scaler_y.fit_transform(df[y_columns])
```

```
    df[X_columns] = x_scaled
    df.drop('Adj Close', axis=1)

    '''open  =  df['Open'].to_numpy().reshape((len(df['Open']),
1))
    low = df['Low'].to_numpy().reshape((len(df['Low']), 1))
    close  =  df['Close'].to_numpy().reshape((len(df['Close']),
1))
    volume                                                  =
df['Volume'].to_numpy().reshape((len(df['Volume']), 1))
    high = df['High'].to_numpy().reshape((len(df['High']), 1))
    dataset = np.hstack((open, low, close, volume, high))'''


    def split_sequences(sequence, n_steps):
        X, y = list(), list()
        for i in range(len(sequence)):
        # find the end of this pattern
            end_ix = i + n_steps
        # check if we are beyond the sequence
            if end_ix > len(sequence)-1:
                break
        # gather input and output parts of the pattern
            seq_x,    seq_y    =    sequence[i:end_ix,    :],
sequence[end_ix, 2]
            X.append(seq_x)
            y.append(seq_y)
        return np.array(X), np.array(y)

    TRAIN_SIZE = int(len(df) * 0.9)

    # split a univariate sequence into samples
    def split_sequence(sequence, n_steps):
        X, y = list(), list()
```

```
            for i in range(len(sequence)):
            # find the end of this pattern
                end_ix = i + n_steps
            # check if we are beyond the sequence
                if end_ix > len(sequence)-1:
                    break
            # gather input and output parts of the pattern
                seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
                X.append(seq_x)
                y.append(seq_y)
            return np.array(X), np.array(y)

        data = df['High']

        n_steps = 12
        # split into samples
        #X, y = split_sequence(data, n_steps)
        #  reshape from [samples, timesteps] into [samples,
timesteps, features]
        X, y = split_sequences(x_scaled, n_steps)
        n_features = 5

        X = X.reshape((X.shape[0], X.shape[1], n_features))

        train_X = X[:TRAIN_SIZE]
        test_X = X[TRAIN_SIZE:]

        train_y = y[:TRAIN_SIZE]
        test_y = y[TRAIN_SIZE:]

        def forecast_accuracy(forecast, actual):
            mape = np.mean(np.abs(forecast - actual)/np.abs(actual))
# MAPE
            me = np.mean(forecast - actual)             # ME
            mae = np.mean(np.abs(forecast - actual))    # MAE
```

```
        mpe = np.mean((forecast - actual)/actual)   # MPE
        mse = np.mean((forecast - actual)**2)   # MSE
        rmse = np.mean((forecast - actual)**2)**.5  # RMSE


        return({'mape':mape, 'me':me, 'mae': mae,
                'mpe': mpe, 'mse': mse, 'rmse':rmse})

    """#Simple LSTM"""

    # define model
    model = Sequential(name='Simple_LSTM')
    model.add(LSTM(64, activation='relu', input_shape=(n_steps,
n_features)))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    model.summary()

    # fit model
    model.fit(train_X,    train_y,    epochs=100,    verbose=1,
batch_size=32, validation_data=[test_X, test_y])

    pred = model.predict(train_X)
    mean_squared_error(scaler_y.inverse_transform(pred),
scaler_y.inverse_transform(train_y.reshape(pred.shape[0],
pred.shape[1])))

    plt.figure(figsize=(12,6))
    plt.plot(train_y, label='Actual Price')
    plt.plot(pred, label='Train Predict')

    plt.title('Price Prediction')
    plt.xlabel('Time')
    plt.ylabel('Price')
    plt.legend()
```

```
      plt.show()

      pred = model.predict(test_X)
      forecast_accuracy(scaler_y.inverse_transform(pred),
scaler_y.inverse_transform(test_y.reshape(pred.shape[0],
pred.shape[1])))

      plt.figure(figsize=(12,6))
      plt.plot(test_y, label='Actual Price')
      plt.plot(pred, label='Train Predict')

      plt.title('Simple LSTM Price Prediction')
      plt.xlabel('Time')
      plt.ylabel('Price')
      plt.legend()
      plt.show()

      """#Stacked LSTM"""

      model = Sequential(name='Stacked_LSTM')
      model.add(LSTM(32,                        activation='relu',
return_sequences=True, input_shape=(n_steps, n_features)))
      model.add(LSTM(32, activation='relu'))
      model.add(Dense(1))
      model.compile(optimizer='adam', loss='mse')
      model.summary()

      # fit model
      model.fit(train_X,    train_y,    epochs=200,    verbose=1,
batch_size=64, validation_data=[test_X, test_y])

      pred = model.predict(train_X)
      mean_squared_error(scaler_y.inverse_transform(pred),
scaler_y.inverse_transform(train_y.reshape(pred.shape[0],
pred.shape[1])))
```

```
      plt.figure(figsize=(12,6))
      plt.plot(train_y, label='Actual Price')
      plt.plot(pred, label='Train Predict')

      plt.title('Price Prediction')
      plt.xlabel('Time')
      plt.ylabel('Price')
      plt.legend()
      plt.show()

      pred = model.predict(test_X)
      forecast_accuracy(scaler_y.inverse_transform(pred),
scaler_y.inverse_transform(test_y.reshape(pred.shape[0],
pred.shape[1])))

      plt.figure(figsize=(12,6))
      plt.plot(test_y, label='Actual Price')
      plt.plot(pred, label='Train Predict')

      plt.title('Stacked LSTM Price Prediction')
      plt.xlabel('Time')
      plt.ylabel('Price')
      plt.legend()
      plt.show()

      """#BiLSTM"""

      from keras.layers import Bidirectional

      model = Sequential(name='BiLSTM')
      model.add(Bidirectional(LSTM(64,        activation='relu'),
input_shape=(n_steps, n_features)))
      model.add(Dense(1))
      model.compile(optimizer='adam', loss='mse')
```

```
        model.summary()

        # fit model
        model.fit(train_X,    train_y,    epochs=200,    verbose=1,
batch_size=64, validation_data=[test_X, test_y])

        pred = model.predict(train_X)
        mean_squared_error(scaler_y.inverse_transform(pred),
scaler_y.inverse_transform(train_y.reshape(pred.shape[0],
pred.shape[1])))

        plt.figure(figsize=(12,6))
        plt.plot(train_y, label='Actual Price')
        plt.plot(pred, label='Train Predict')

        plt.title('Price Prediction')
        plt.xlabel('Time')
        plt.ylabel('Price')
        plt.legend()
        plt.show()

        pred = model.predict(test_X)
        forecast_accuracy(scaler_y.inverse_transform(pred),
scaler_y.inverse_transform(test_y.reshape(pred.shape[0],
pred.shape[1])))

        plt.figure(figsize=(12,6))
        plt.plot(test_y, label='Actual Price')
        plt.plot(pred, label='Test Predict')

        plt.title('BiLSTM Price Prediction')
        plt.xlabel('Time')
        plt.ylabel('Price')
        plt.legend()
        plt.show()
```

```
"""#CNNLSTM"""

n_seq = 2
n_steps = 12
#X, y = split_sequence(data, n_steps)
X, y = split_sequences(x_scaled, n_steps)

n_steps = 6
X = X.reshape((X.shape[0], n_seq, 6, n_features))
#X = X.reshape((X.shape[0], X.shape[1], n_features))
train_X = X[:TRAIN_SIZE]
test_X = X[TRAIN_SIZE:]

train_y = y[:TRAIN_SIZE]
test_y = y[TRAIN_SIZE:]

# define model
model = Sequential()
model.add(TimeDistributed(Conv1D(filters=32, kernel_size=5,
activation='relu'), input_shape=(None, n_steps, n_features)))
model.add(TimeDistributed(MaxPooling1D(pool_size=2)))
model.add(Dropout(0.1))
model.add(TimeDistributed(Flatten()))
model.add(LSTM(128, activation='relu'))
model.add(Dense(1))

optimizer = tf.keras.optimizers.SGD(lr=1e-4, momentum=0.9)
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mse"])

# fit model
model.fit(train_X,    train_y,    epochs=290,    verbose=1,
batch_size=32, validation_data=[test_X, test_y])
```

```
      from keras.layers import Flatten
      from keras.layers import TimeDistributed
      from keras.layers import Conv1D
      from keras.layers import MaxPooling1D
      from keras.layers import Dropout

      # define model
      model = Sequential()
      model.add(TimeDistributed(Conv1D(filters=32, kernel_size=5,
activation='relu'), input_shape=(None, n_steps, n_features)))
      model.add(TimeDistributed(MaxPooling1D(pool_size=2)))
      model.add(Dropout(0.1))
      model.add(TimeDistributed(Flatten()))
      model.add(LSTM(128, activation='relu'))
      model.add(Dense(1))

      optimizer = tf.keras.optimizers.SGD(lr=1e-4, momentum=0.9)
      model.compile(loss=tf.keras.losses.Huber(),
                    optimizer=optimizer,
                    metrics=["mse"])

      # fit model
      model.fit(train_X,    train_y,    epochs=250,    verbose=1,
batch_size=32, validation_data=[test_X, test_y])

      model.summary()

      from sklearn.metrics import mean_absolute_percentage_error
      pred = model.predict(train_X)
      mean_squared_error(scaler_y.inverse_transform(pred),
scaler_y.inverse_transform(train_y.reshape(pred.shape[0],
pred.shape[1]))),
mean_absolute_percentage_error(scaler_y.inverse_transform(pred),
```

```
scaler_y.inverse_transform(train_y.reshape(pred.shape[0],
pred.shape[1])))

      scaler_y.inverse_transform(pred),
scaler_y.inverse_transform(train_y.reshape(pred.shape[0],
pred.shape[1]))

      plt.figure(figsize=(12,6))
      plt.plot(train_y, label='Actual Price')
      plt.plot(pred, label='Train Predict')

      plt.title('Price Prediction')
      plt.xlabel('Time')
      plt.ylabel('Price')
      plt.legend()
      plt.show()

      pred = model.predict(test_X)
      forecast_accuracy(scaler_y.inverse_transform(pred),
scaler_y.inverse_transform(test_y.reshape(pred.shape[0],
pred.shape[1])))

      plt.figure(figsize=(12,6))
      plt.plot(test_y, label='Actual Price')
      plt.plot(pred, label='Test Predict')

      plt.title('Price Prediction')
      plt.xlabel('Time')
      plt.ylabel('Price')
      plt.legend()
      plt.show()

      """#ConvLSTM"""

      n_seq = 2
```

```
n_steps = 12
X, y = split_sequences(x_scaled, n_steps)

n_steps = 6
X = X.reshape((X.shape[0], n_seq, 1, 6, n_features))

train_X = X[:TRAIN_SIZE]
test_X = X[TRAIN_SIZE:]

train_y = y[:TRAIN_SIZE]
test_y = y[TRAIN_SIZE:]



from keras.layers import ConvLSTM2D, BatchNormalization

# define model
model = Sequential()
model.add(ConvLSTM2D(filters=128,        kernel_size=(1,2),
activation='relu',  input_shape=(n_seq,  1,  n_steps,  n_features),
padding='same', return_sequences=True))
model.add(BatchNormalization())
model.add(Dropout(0.1))
model.add(TimeDistributed(Flatten()))
model.add(LSTM(128, activation='relu'))
model.add(Dense(1))

model.compile(loss=tf.keras.losses.Huber(),
              optimizer='adam',
              metrics=["mse"])
# fit model
model.fit(train_X,    train_y,    epochs=100,    verbose=1,
batch_size=32, validation_data=[test_X, test_y])

pred = model.predict(train_X)
```

```
      mean_squared_error(scaler_y.inverse_transform(pred),
scaler_y.inverse_transform(train_y.reshape(pred.shape[0],
pred.shape[1])))

      plt.figure(figsize=(12,6))
      plt.plot(train_y, label='Actual Price')
      plt.plot(pred, label='Train Predict')

      plt.title('Price Prediction')
      plt.xlabel('Time')
      plt.ylabel('Price')
      plt.legend()
      plt.show()

      pred = model.predict(test_X)
      forecast_accuracy(scaler_y.inverse_transform(pred),
scaler_y.inverse_transform(test_y.reshape(pred.shape[0],
pred.shape[1])))

      plt.figure(figsize=(12,6))
      plt.plot(test_y, label='Actual Price')
      plt.plot(pred, label='Test Predict')

      plt.title('Price Prediction')
      plt.xlabel('Time')
      plt.ylabel('Price')
      plt.legend()
      plt.show()

      """#Encoder-Decoder LSTM"""

      # split a univariate sequence into samples
      def split_sequence_uni(sequence, n_steps):
          X, y = list(), list()
          for i in range(len(sequence)):
```

```
            # find the end of this pattern
                end_ix = i + n_steps
                out_end_ix = end_ix + n_steps_out
            # check if we are beyond the sequence
                if out_end_ix > len(sequence)-1:
                    break
            # gather input and output parts of the pattern
                seq_x,      seq_y      =      sequence[i:end_ix],
sequence[end_ix:out_end_ix]
                X.append(seq_x)
                y.append(seq_y)
            return np.array(X), np.array(y)

      n_steps_in, n_steps_out = 12, 3
      # split into samples
      X, y = split_sequence_uni(data, n_steps_in)
      #  reshape  from  [samples,  timesteps]  into  [samples,
timesteps, features]
      n_features = 1
      X = X.reshape((X.shape[0], X.shape[1], n_features))
      y = y.reshape((y.shape[0], y.shape[1], n_features))

      train_X = X[:TRAIN_SIZE]
      test_X = X[TRAIN_SIZE:]

      train_y = y[:TRAIN_SIZE]
      test_y = y[TRAIN_SIZE:]

      from keras.layers import RepeatVector

      # define model
      model = Sequential()
      model.add(LSTM(128,                        activation='relu',
input_shape=(n_steps_in, n_features)))
      model.add(RepeatVector(n_steps_out))
```

```
      model.add(LSTM(128,                        activation='relu',
return_sequences=True))
      model.add(TimeDistributed(Dense(1)))
      model.compile(optimizer='adam', loss='mse')
      # fit model
      model.fit(train_X, train_y, epochs=100, verbose=0)

      pred = model.predict(test_X[:1])
      #mean_squared_error(pred, test_y[:3])

      test_y[0] - pred[0]

      plt.figure(figsize=(12,6))
      plt.plot(test_y[0], label='Actual Price')
      plt.plot(pred[0], label='Train Predict')

      plt.title('Price Prediction')
      plt.xlabel('Time')
      plt.ylabel('Price')
      plt.legend()
      plt.show()



      """#GRU"""

      def fit_model(model, epochs=100):
          early_stop = tf.keras.callbacks.EarlyStopping(monitor =
'val_loss',
                                                        patience   =
10)
          history = model.fit(train_X, train_y, epochs = epochs,
                              validation_split = 0.1,
                              shuffle = False,
                              callbacks = [early_stop])
```

```
        return history

    n_steps = 12
    # split into samples
    X, y = split_sequences(x_scaled, n_steps)
    # reshape from [samples, timesteps] into [samples,
timesteps, features]
    n_features = 5
    X = X.reshape((X.shape[0], X.shape[1], n_features))

    train_X = X[:TRAIN_SIZE]
    test_X = X[TRAIN_SIZE:]

    train_y = y[:TRAIN_SIZE]
    test_y = y[TRAIN_SIZE:]

    from tensorflow.keras.layers import Dropout, GRU

    model = Sequential()
    # Input layer
    model.add(GRU(units = 128, activation='relu', input_shape =
[n_steps, n_features], return_sequences=True))
    model.add(Dropout(0.1))
    # Hidden layer
    model.add(GRU(units = 64, activation='relu'))
    model.add(Dropout(0.1))
    model.add(Dense(units = 1))
    #Compile model
    #model.compile(optimizer='adam',loss='mse')
    model.compile(loss=tf.keras.losses.Huber(),
                  optimizer='adam',
                  metrics=["mse"])
    model.fit(train_X,    train_y,    epochs=100,    verbose=1,
batch_size=32, validation_data=[test_X, test_y])
```

```
      #history = fit_model(model, epochs=500)

      '''def plot_loss (history, model_name):
          plt.figure(figsize = (10, 6))
          plt.plot(history.history['loss'])
          plt.plot(history.history['val_loss'])
          plt.title('Model Train vs Validation Loss for ' +
model_name)
          plt.ylabel('Loss')
          plt.xlabel('epoch')
          plt.legend(['Train loss', 'Validation loss'], loc='upper
right')

      plot_loss (history_gru, 'GRU')'''

      pred = model.predict(train_X)
      mean_squared_error(scaler_y.inverse_transform(pred),
scaler_y.inverse_transform(train_y.reshape(pred.shape[0],
pred.shape[1])))

      plt.figure(figsize=(12,6))
      plt.plot(train_y, label='Actual Price')
      plt.plot(pred, label='Train Predict')

      plt.title('Price Prediction')
      plt.xlabel('Time')
      plt.ylabel('Price')
      plt.legend()
      plt.show()

      pred = model.predict(test_X)
      forecast_accuracy(scaler_y.inverse_transform(pred),
scaler_y.inverse_transform(test_y.reshape(pred.shape[0],
pred.shape[1])))
```

```
plt.figure(figsize=(12,6))
plt.plot(test_y, label='Actual Price')
plt.plot(pred, label='Test Predict')

plt.title('Price Prediction')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt.show()

model.save('gru_model.h5')

"""#WaveNet"""

train_x = x_scaled[:TRAIN_SIZE]
train_y = y_scaled[:TRAIN_SIZE]
test_x = x_scaled[TRAIN_SIZE:]
test_y = y_scaled[TRAIN_SIZE:]

seq_length = 112
tf.random.set_seed(42)  # ensures reproducibility
train_ds = tf.keras.utils.timeseries_dataset_from_array(
    train_x,
    targets=train_y[seq_length:],
    sequence_length=seq_length,
    batch_size=32,
    shuffle=True,
    seed=42
)
test_ds = tf.keras.utils.timeseries_dataset_from_array(
    test_x,
    targets=test_y[seq_length:],
    sequence_length=seq_length,
    batch_size=32,
    shuffle=True,
```

```
        seed=42
    )

    class GatedActivationUnit(tf.keras.layers.Layer):
        def __init__(self, activation="tanh", **kwargs):
            super().__init__(**kwargs)
            self.activation                                    =
tf.keras.activations.get(activation)

        def call(self, inputs):
            n_filters = inputs.shape[-1] // 2
            linear_output     =     self.activation(inputs[...,
:n_filters])
            gate   =   tf.keras.activations.sigmoid(inputs[...,
n_filters:])
            return self.activation(linear_output) * gate

    def        wavenet_residual_block(inputs,        n_filters,
dilation_rate):
        z = tf.keras.layers.Conv1D(2 * n_filters, kernel_size=2,
padding="causal",

dilation_rate=dilation_rate)(inputs)
        z = GatedActivationUnit()(z)
        z = tf.keras.layers.Conv1D(n_filters, kernel_size=1)(z)
        return tf.keras.layers.Add()([z, inputs]), z

    tf.random.set_seed(42)

    n_layers_per_block = 3  # 10 in the paper
    n_blocks = 1  # 3 in the paper
    n_filters = 32  # 128 in the paper
    n_outputs = 1  # 256 in the paper

    inputs = tf.keras.layers.Input(shape=[n_steps, 5])
```

```
    z = tf.keras.layers.Conv1D(n_filters, kernel_size=2,
padding="causal")(inputs)
    skip_to_last = []
    for dilation_rate in [2**i for i in
range(n_layers_per_block)] * n_blocks:
        z, skip = wavenet_residual_block(z, n_filters,
dilation_rate)
        skip_to_last.append(skip)

    z =
tf.keras.activations.relu(tf.keras.layers.Add()(skip_to_last))
    z = tf.keras.layers.Conv1D(n_filters, kernel_size=1,
activation="relu")(z)
    Y_preds = tf.keras.layers.Conv1D(n_outputs,
kernel_size=1)(z)

    wavenet_model = tf.keras.Model(inputs=[inputs],
outputs=[Y_preds])
    wavenet_model.summary()

    opt = tf.keras.optimizers.SGD(learning_rate=0.1,
momentum=0.9)
    wavenet_model.compile(loss='mse', optimizer=opt,
metrics=["mse"])
    wavenet_model.fit(train_ds, epochs=100, verbose=1,
batch_size=32, validation_data=test_ds)

    pred = wavenet_model.predict(train_X[:1])
    train_y

    train_y

    from keras.models import Model
    from keras.layers import Input, Conv1D, Dense, Dropout,
Lambda, Concatenate
```

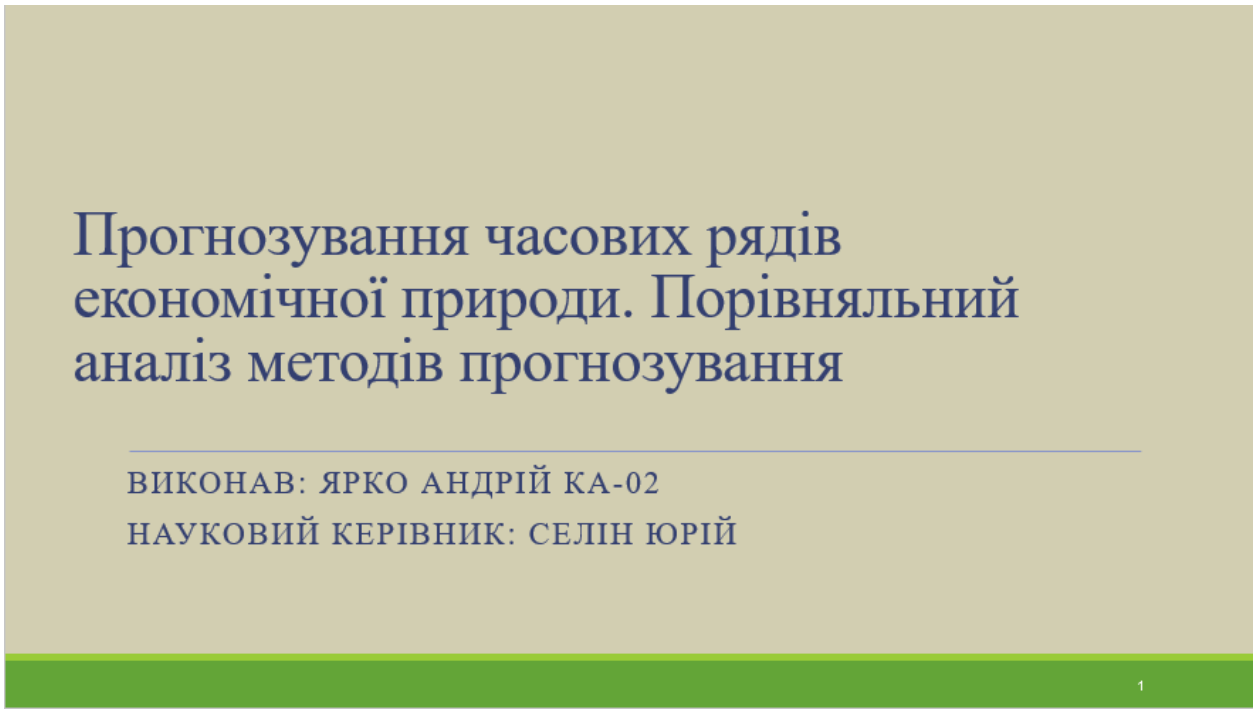
```
n_filters = 128
filter_width = 5
dilation_rates = [2**i for i in range(5)]

    # define an input history series and pass it through a
stack of dilated causal convolutions
history_seq = Input(shape=(None, 5))
x = history_seq

for dilation_rate in dilation_rates:
    x = Conv1D(filters = n_filters,
               kernel_size=filter_width,
               padding='causal',
               dilation_rate=dilation_rate)(x)
    # for Dense Layer:
    # Input shape
    # nD tensor with shape: (batch_size, ..., input_dim).
The most common situation would be a 2D input with shape
(batch_size, input_dim).
    #
    # Output shape
    # nD tensor with shape: (batch_size, ..., units).
    # For instance, for a 2D input with shape  (batch_size,
input_dim), the output would have shape (batch_size, units).
x = Dense(32, activation='relu')(x)
x = Dropout(.5)(x)
x = Dense(16)(x)
x = Dense(1)(x)

    # extract the last 14 time steps as the training target
def slice(x, seq_length):
    return x[:, -seq_length:, :]
```

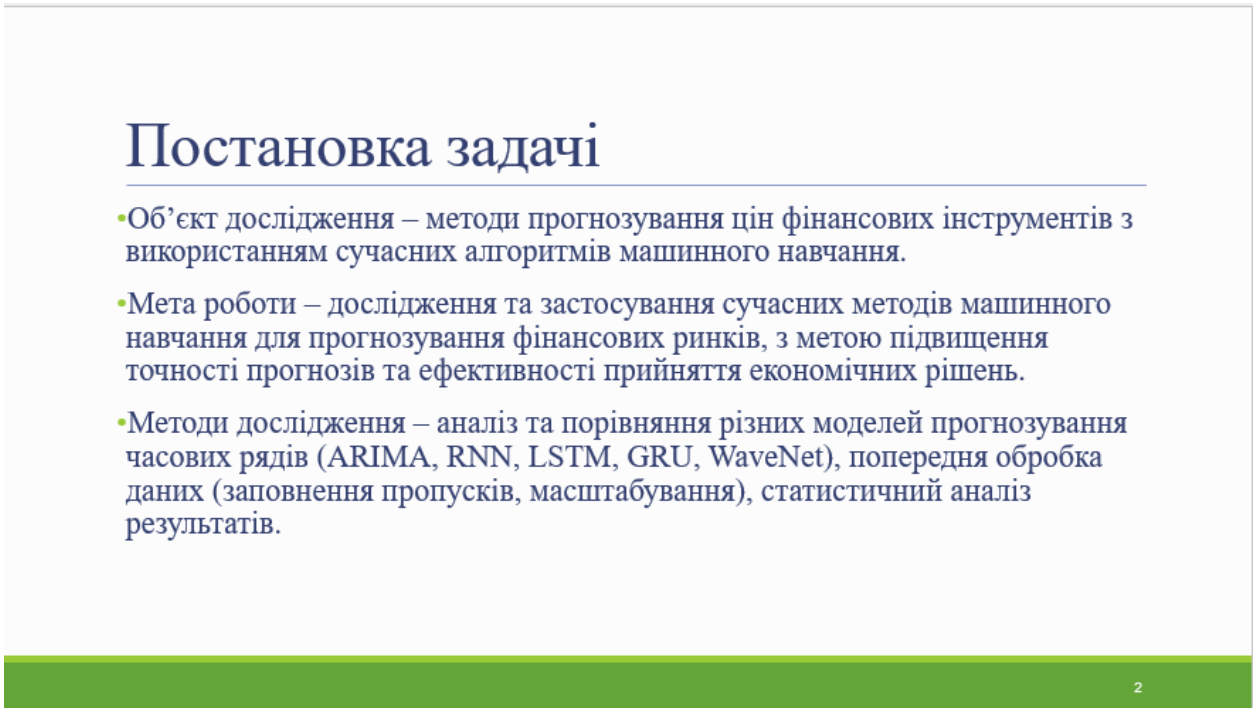
```
      pred_seq_train                =                Lambda(slice,
arguments={'seq_length':24})(x)

      model = Model(history_seq, pred_seq_train)

      model.summary()
      #opt     =     tf.keras.optimizers.SGD(learning_rate=0.05,
momentum=0.9)
      #wavenet_model.compile(loss='mse',           optimizer=opt,
metrics=["mse"])
      #wavenet_model.fit(train_X, train_y, epochs=100, verbose=1,
batch_size=32, validation_data=[test_X, test_y])

      batch_size = 32

      model.compile(Adam(), loss='mse')
      history = model.fit(x_scaled[:-1], y_scaled[1:],
                          batch_size=batch_size,
                          epochs=20,
                          validation_split=0.2)

      train_y

      pred = wavenet_model.predict(train_X)
      train_X.shape

      pred = wavenet_model.predict(train_X)
      mean_squared_error(scaler_y.inverse_transform(pred[0]),
scaler_y.inverse_transform(train_y.reshape(pred.shape[0],
pred.shape[1])))

      plt.figure(figsize=(12,6))
      plt.plot(train_y, label='Actual Price')
      plt.plot(pred, label='Train Predict')
```

```
      plt.title('Price Prediction')
      plt.xlabel('Time')
      plt.ylabel('Price')
      plt.legend()
      plt.show()

      pred = model.predict(test_X)
      mean_squared_error(scaler_y.inverse_transform(pred),
scaler_y.inverse_transform(test_y.reshape(pred.shape[0],
pred.shape[1])))

      test_y.shape

      plt.figure(figsize=(12,6))
      plt.plot(test_y, label='Actual Price')
      plt.plot(pred, label='Train Predict')

      plt.title('Price Prediction')
      plt.xlabel('Time')
      plt.ylabel('Price')
      plt.legend()
      plt.show()

      def forecast_accuracy(forecast, actual):
          mape = np.mean(np.abs(forecast - actual)/np.abs(actual))
# MAPE
          me = np.mean(forecast - actual)             # ME
          mae = np.mean(np.abs(forecast - actual))    # MAE
          mpe = np.mean((forecast - actual)/actual)   # MPE
          rmse = np.mean((forecast - actual)**2)**.5  # RMSE


          return({'mape':mape, 'me':me, 'mae': mae,
                  'mpe': mpe, 'rmse':rmse})
```

```
        forecast_accuracy(scaler_y.inverse_transform(pred),
scaler_y.inverse_transform(test_y.reshape(pred.shape[0],
pred.shape[1])))
```

ДОДАТОК Б ГРАФІЧНІ МАТЕРІАЛИ





Актуальність

Економічне зростання

Прогнозування ВВП, інфляції, безробіття.

Інвестиційні рішення

Оцінка ризиків та прибутковості інвестицій.

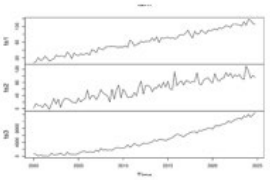
Економічна політика

Розробка ефективної бюджетної, податкової та грошової політики.

Методи економічного прогнозування

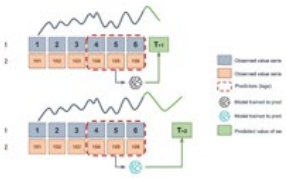
Прогнозування на основі суджень

Ґрунтується на суб'єктивних оцінках, інтуїції та знаннях експертів.



Прогнозування на основі часових рядів

Використовує авторегресію, АРКС та інші методи аналізу даних.



Прогнозування на основі багатьох змінних

Враховує вплив різних факторів на прогнозовану змінну.

Реалізація програмного продукту

1 **Мова програмування Python**

Гнучка, проста у використанні, з багатьма бібліотеками та постійно розвивається.

2 **Середовище Colaboratory**

Середовище для програмування, що використовує потужності апаратного забезпечення Google, зокрема графічних і тензорних процесорів, що є дуже корисним при навчанні глибоких нейронних мереж.

5

Підготовка та аналіз даних

1 **Вибір даних**

В якості даних для прогнозування було взято денну ціну на акції компанії Microsoft Corporation за період з 2019-05-08 по 2024-05-07. Для прогнозування було обрано змінну High

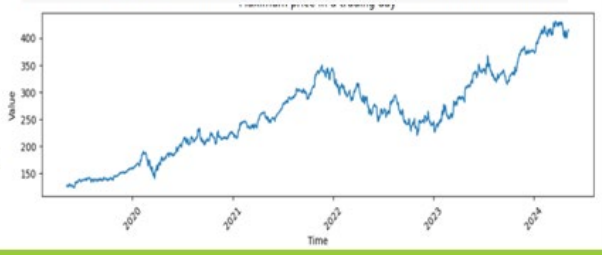
2 **Аналіз даних**

Перевірка на стаціонарність, побудова АКФ та ЧАКФ

3 **Підготовка даних**

Для підготовки даних до прогнозування було проведено масштабування за допомогою z-нормалізації та перевірка цілісності даних.

	Date	Open	High	Low	Close	Adj Close	Volume
0	2019-05-08	125.440002	126.370003	124.750000	125.510002	119.471992	28419000
1	2019-05-09	124.290001	125.790001	123.570000	125.500000	119.462502	27235800
2	2019-05-10	124.910004	127.930000	123.820000	127.129997	121.014053	30915100
3	2019-05-13	124.110001	125.550003	123.040001	123.349998	117.415916	33944900
4	2019-05-14	123.870003	125.879997	123.699997	124.730003	118.729538	25266300
5	2019-05-15	124.260002	126.709999	123.699997	126.019997	120.401512	24722700
6	2019-05-16	126.750000	129.380005	126.459999	128.929993	123.181755	30112200



6

Критерії оцінювання моделей

MAE

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y} - y|$$

RMSE

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y} - y)^2}$$

MAPE

$$MAPE = \left(\frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{|y_i|}\right) * 100\%$$

7

Авторегресійні моделі

ARIMA

$$y_t' = c + \phi_1 y_{t-1}' + \cdots + \phi_p y_{t-p}' + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t,$$

де

y_t' - різниця d-го порядку,

p - порядок авторегресійної частини,

d - порядок інтегрованої частини,

q - порядок КС.

SARIMA

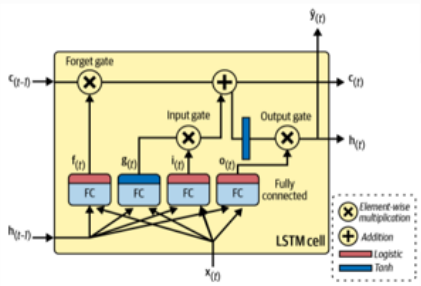
ARIMA $\underbrace{(p, d, q)}_{\text{Non-seasonal part of the model}}$ $\underbrace{(P, D, Q)_m}_{\text{Seasonal part of of the model}}$

$$(1 - \phi_1 B)(1 - \Phi_1 B^4)(1 - B)(1 - B^4) y_t = (1 + \theta_1 B)(1 + \Theta_1 B^4)\varepsilon_t,$$

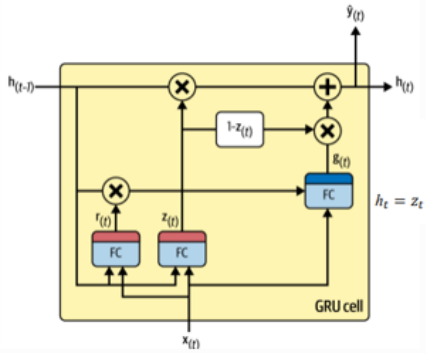
8

LSTM та GRU

- LSTM: Ключова ідея полягає в тому, що мережа може дізнатися, що зберігати в довгостроковому стані, що викидати, а що читати з нього. Коли довгостроковий стан c(t–1) перетинає мережу зліва направо, можна побачити, що він спочатку проходить через забувальний вентиль, відкидаючи деякі дані з пам'яті, а потім додає кілька нових даних за допомогою операції додавання
- GRU: GRU є спрощеною версією LSTM, обидва вектори стану об'єднані в один вектор h(t), один контролер z(t) керує як вентилем забуття, так і вентилем введення, відсутній вихідний вентиль; повний вектор стану виводиться на кожному кроці.



$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f),$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i),$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o),$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c),$$
$$h_t = o_t \circ \sigma_h(c_t),$$

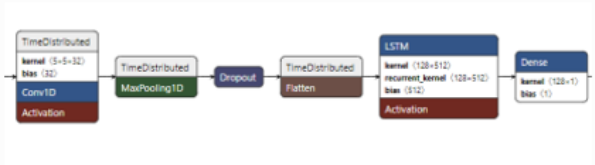


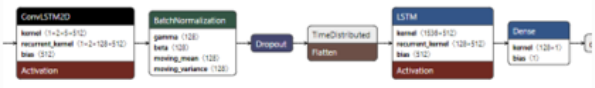
$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z),$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r),$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h),$$

9

LSTM+CNN

Поєднання CNN та LSTM дозволяє моделі враховувати як локальні, так і довгострокові закономірності в часових рядах. Було побудовано дві моделі. Перша модель використовує 1D згортку для виявлення локальних закономірностей, а потім LSTM для довгострокових залежностей. Друга модель використовує потужніший ConvLSTM2D шар, який одночасно враховує локальні закономірності в часі та ознаках.





10

Висновки

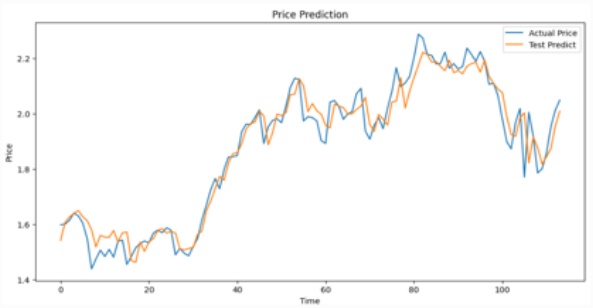
В результаті дослідження отримали наступне.

- Розроблено програмний продукт для прогнозування цін на акції за часовими рядами.

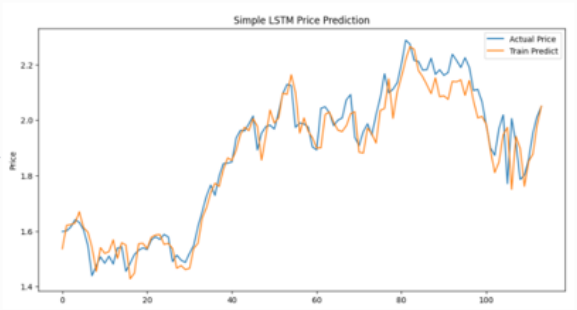
- Здійснено порівняльний аналіз різних моделей прогнозування та визначено найбільш ефективну модель.

- Запропоновані моделі можуть бути використані для прогнозування цін на акції, що є актуальним для інвесторів, компаній та державних органів. Це дозволить більш ефективно планувати діяльність та приймати обґрунтовані економічні рішення.

GRU



Simple LSTM



12

Аналіз результатів

Модель	MAPE	MAE	MSE	RMSE
ARIMA	0.1088	-	2393.64	-
SARIMA	0.0087	-	20.5589	-
Simple LSTM	0.0091	3.7097	24.2945	4.9289
Stacked LSTM	0.0112	4.523	37.8378	6.1512
BiLSTM	0.0095	3.8211	24.5267	4.9524
Conv1D+LSTM	0.0124	5.0163	37.2705	6.1049
Conv2D+LSTM	0.0098	3.9551	26.0453	5.1034
GRU	**0.008**	**3.2456**	**19.8908**	**4.4599**

11

Дякую за увагу