Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

Бакалаврська дипломна робота на тему:

«АНАЛІЗ ТА ПРОГНОЗУВАННЯ ЦІНИ НА АКЦІЇ NETFLIX»

Виконав: студент 4 курсу, групи СА-20б
спеціальності 124 «Системний аналіз»
_____ Вадим РУДИК
Керівник: к.т.н., доц. каф. САІТ
_____ Ілона ВАРЧУК
«03» _____06_____2024 р.
Рецензент: к.т.н., доц. каф. КН
_____Володимир ОЗЕРАНСЬКИЙ
«11» _____06_____ 2024 р.

Допущено до захисту

Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«04» _____06_____ 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 124 – «Системний аналіз»
Освітньо-професійна програма – Системний аналіз

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
"05" 03 2024 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Рудику Вадиму Васильовичу

1. Тема роботи: «Аналіз та прогнозування ціни на акції Netflix»
керівник роботи: Варчук І.В., к.т.н., доц. каф. САІТ
затверджені наказом ВНТУ від «11» 03 2024 року № 80
2. Термін подання студентом роботи 31.05
3. Вихідні дані до роботи:
1) Дані ціни на акції компанії Netflix за 2014 – 2023 рр.
3. Зміст звіту (перелік питань, які потрібно розробити):
1) Загальна характеристика стрімінгових сервісів та ціноутворення на їх акції;
2) Розвідувальний аналіз даних та інженерія ознак;
3) Тренування моделей машинного навчання.
4) Результат застосування технології на реальних даних
5. Перелік ілюстративного матеріалу:
1) Графік ціни на акції Netflix за 2014 – 2023 рр ;
2) Графік важливості ознак;
3) Графік прогнозування моделлю Prophet;
3) Графік прогнозованих даних;
4) Таблиця порівняння оцінок точності прогнозування моделей.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 11.03

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	11.03	31.03	вик
2	Порівняльний аналіз методів прогнозування ціни акцій	01.04	15.04	вик
3	Розвідувальний аналіз даних та інженерія ознак	16.04	30.04	вик
4	Прогнозування ціни акцій Netflix	01.05	15.05	вик
5	Оформлення матеріалів до захисту БДР	16.05	31.05	вик

Студент _____ Вадим РУДИК

Керівник роботи _____ Ілона ВАРЧУК

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 74 сторінок формату А4, на яких є 45 рисунка, список використаних джерел містить 25 найменувань.

Метою роботи є аналіз та прогнозування ціни акцій Netflix.

В роботі проведено огляд стрімінгових компаній Netflix, Amazon Prime, Disney+, HBO Max. Проаналізовано ціноутворення на акції стрімінгової компанії Netflix. Розглянуто альтернативні методи прогнозування ціни на акції Netflix.

Проведено розвідувальний аналіз даних ціни акцій Netflix. Виконано інженерію ознак за допомогою TSFRESH. Сформовано набір ознак для використання їх у моделях машинного навчання для прогнозування ціни на акції Netflix.

Для прогнозування ціни акцій Netflix використано моделі машинного навчання Prophet, ARIMA, XGBoost. Проведено тренування обраних моделей на тренувальних даних. Проведено тестування натренованих моделей на тестовому наборі даних. Візуалізовано прогнозовані дані на графіках. Проведено оцінку використаних моделей та обрано оптимальну модель для прогнозування ціни на акції Netflix. Для реалізації роботи використано мову Python.

Ключові слова: системний аналіз, передбачення даних, машинне навчання, Python.

ABSTRACT

The bachelor thesis consists of 74 pages of A4 format, on which there are 45 figures, the list of used sources contains 25 names.

The purpose of the work is to analyze and forecast the price of Netflix shares.

The work includes an overview of the streaming companies Netflix, Amazon Prime, Disney+, and HBO Max. The pricing of shares of the streaming company Netflix has been analyzed. Considered alternative methods of forecasting the price of Netflix shares.

Exploratory analysis of Netflix share price data. Performed feature engineering using TSFRESH. A set of features was formed for use in machine learning models to predict the price of Netflix shares.

Prophet, ARIMA, XGBoost machine learning models were used to forecast the price of Netflix shares. The selected models were trained on the training data. The trained models were tested on the test data set. The forecasted data is visualized on the graphs. The used models were evaluated and the optimal model was selected for forecasting the price of Netflix shares. The Python language was used to implement the work.

Keywords: system analysis, data prediction, machine learning, Python.

ЗМІСТ

ВСТУП

Актуальність теми. Netflix є лідером у галузі стрімінгових медіа, і будь-які зміни в їхніх стратегіях, такі як ціноутворення, рекламні стратегії, або нові релізи контенту, можуть впливати на їхні фінансові показники та ціну акцій.

Збільшення конкуренції від таких компаній, як Amazon Prime, Disney+, HBO Max, та інших, може впливати на ринкову частку Netflix та їхню дохідність. Аналіз цих тенденцій допомагає інвесторам розуміти потенційні ризики та можливості.

Технологічний прогрес у галузі штучного інтелекту та машинного навчання відкриває нові можливості для персоналізації контенту та покращення користувацького досвіду, що також може впливати на акції Netflix.

Регуляторні зміни у різних країнах, де Netflix працює, можуть впливати на їхню діяльність і прибутки.

Загальноекономічні фактори, такі як ставки процентів, інфляція, або рецесії, також можуть впливати на інвестиційну привабливість акцій Netflix.

Аналіз цих аспектів дозволяє краще розуміти, як зміни в бізнесі, технологіях, ринковому середовищі та макроекономічних умовах можуть вплинути на вартість акцій Netflix. Прогнозування ціни акцій на основі цих даних є важливим інструментом для інвесторів та аналітиків.

Мета і задачі дослідження. Метою дослідження є аналіз та прогнозування ціни акцій Netflix .

Для досягнення поставленої мети необхідно розв’язати такі задачі:

- Провести аналіз ціноутворення на акції;
- Провести розвідувальний аналіз даних;
- Провести інженерію ознак;
- Натренувати та застосувати моделі машинного навчання для прогнгозування ціни на акції компанії Netflix.

Об’єктом дослідження є застосування методів машинного навчання для прогнозування ціни на акції Netflix.

Предметом дослідження є моделі машинного навчання для прогнозування ціни на акції Netflix.

1. ЗАГАЛЬНА ХАРАКТЕРИСТИКА ОБ'ЄКТУ ДОСЛІДЖЕНЬ

1.1 Аналіз стрімінгових сервісів

Netflix Inc. — американська транснаціональна медіа-компанія, яка надає послуги стрімінгового відео на вимогу, виробництво та розповсюдження оригінального контенту. Була заснована 29 серпня 1997 року у Скотте-Веллі, Каліфорнія, Рідом Гастінгсом і Марком Рендольфом. Ідея компанії виникла після того, як Гастінгс отримав великий штраф за пізнє повернення DVD, що спонукало його створити зручніший сервіс прокату фільмів. Спочатку Netflix пропонувала прокат DVD поштою через інтернет-сайт, де клієнти могли обирати фільми для доставки додому (рис. 1.1) [1].

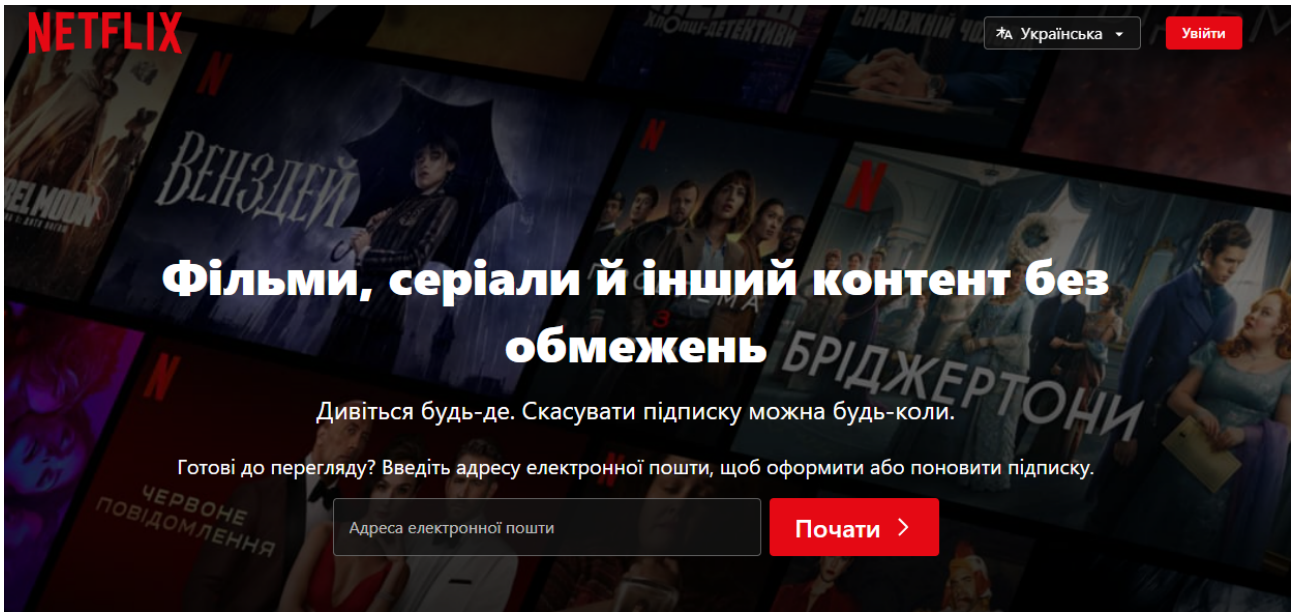

Рисунок 1.1 – Головна сторінка Netflix

У 1999 році Netflix запровадила модель підписки, яка дозволяла клієнтам платити фіксовану щомісячну плату за необмежену кількість прокатів DVD без термінів повернення, що зробило сервіс ще зручнішим і популярнішим. Ця модель допомогла компанії залучити більше клієнтів та значно збільшити свою базу підписників.

У січні 2007 року Netflix запустила свою послугу потокового відео, що дозволяло клієнтам миттєво дивитися телевізійні шоу та фільми онлайн. Це стало важливим кроком у розвитку компанії, оскільки технології потокового передавання почали швидко розвиватися, і споживачі все більше цікавилися можливістю перегляду відео в режимі реального часу.

У 2013 році Netflix розпочала виробництво власного оригінального контенту, випустивши серіал "House of Cards". Цей крок виявився дуже успішним і започаткував еру активного створення оригінальних фільмів, серіалів та документальних стрічок. Інші успішні проекти включають "Stranger Things", "The Crown", "Orange Is the New Black", "Narcos" та багато інших [1].

З 2010 року Netflix почала розширювати свою присутність за межами США. Спочатку компанія запустила свої послуги в Канаді, а потім поступово вийшла на ринки Латинської Америки, Європи, Азії та інших регіонів. Сьогодні Netflix доступна у понад 190 країнах світу, пропонуючи контент на різних мовах і враховуючи культурні особливості різних регіонів.

Netflix активно співпрацює з різними партнерами для покращення якості своїх послуг. Компанія використовує потужні алгоритми аналізу даних для персоналізації рекомендацій, покращує інтерфейс користувача, інвестує в технології адаптивного стрімінгу та розробляє інтерактивний контент, наприклад, "Black Mirror: Bandersnatch".

Фінансовий успіх Netflix значною мірою зумовлений її стратегією інвестування у власний контент та міжнародну експансію. Компанія продовжує залучати мільйони нових підписників щороку, що призводить до постійного зростання доходів. Водночас Netflix має значні боргові зобов'язання через великі інвестиції в контент, але компанія зберігає стабільне фінансове становище завдяки зростанню кількості передплатників та доходів.

Netflix реалізує різноманітні програми соціальної відповідальності, включаючи підтримку різноманітності та інклюзії, створення контенту, що відображає різні культури та досвіди, а також екологічні ініціативи, спрямовані на зменшення вуглецевого сліду.

Netflix продовжує інвестувати в технологічні інновації, щоб покращити якість стрімінгу та користувацький досвід:

– Використання машинного навчання для покращення рекомендацій та персоналізованого контенту.

– Технологія, яка автоматично налаштовує якість відео залежно від швидкості інтернет-з'єднання.

– Розробка інтерактивних фільмів та серіалів, таких як "Black Mirror: Bandersnatch", що дозволяють глядачам приймати рішення щодо розвитку сюжету.

Netflix продовжує активно розширювати свою бібліотеку оригінального контенту. Компанія інвестує мільярди доларів у виробництво нових фільмів, серіалів та документальних стрічок, щоб залучити та утримати передплатників. Одні з найбільш значущих нових проєктів включають [1]:

"The Queen's Gambit": Серіал, що став феноменом і значно збільшив інтерес до шахів.

"Bridgerton": Романтичний серіал, який швидко здобув велику популярність.

"The Witcher": Епічний фентезійний серіал, заснований на популярних книгах та відеоіграх.

Netflix активно інвестує в місцевий контент у різних країнах для залучення регіональних аудиторій. Наприклад:

Індія – виробництво індійських фільмів і серіалів, таких як "Sacred Games" та "Delhi Crime".

Південна Корея – випуск популярних корейських драм, таких як "Kingdom" та "Sweet Home".

Європа – виробництво місцевих серіалів, таких як німецький "Dark" та іспанський "Money Heist".

Netflix продовжує залишатися лідером на ринку стрімінгових послуг, активно розвиваючи свій оригінальний контент та технологічні можливості. Конкуренція з боку інших гравців, таких як Amazon Prime Video, Disney+, HBO

Max та інших, стимулює компанію до подальших інновацій та покращення користувацького досвіду. Враховуючи тенденції зростання споживання стрімінгового контенту, Netflix має добрі перспективи для подальшого розвитку та зміцнення своїх позицій на світовому ринку.

Amazon Prime Video — це відеострімінговий сервіс, який належить і управляється американською компанією Amazon. Він був офіційно запущений 7 вересня 2006 року як частина платформи Amazon Instant Video. Спочатку Amazon Instant Video пропонував своїм користувачам можливість оренди або покупки фільмів та телевізійних шоу (рис 1.2) [2].

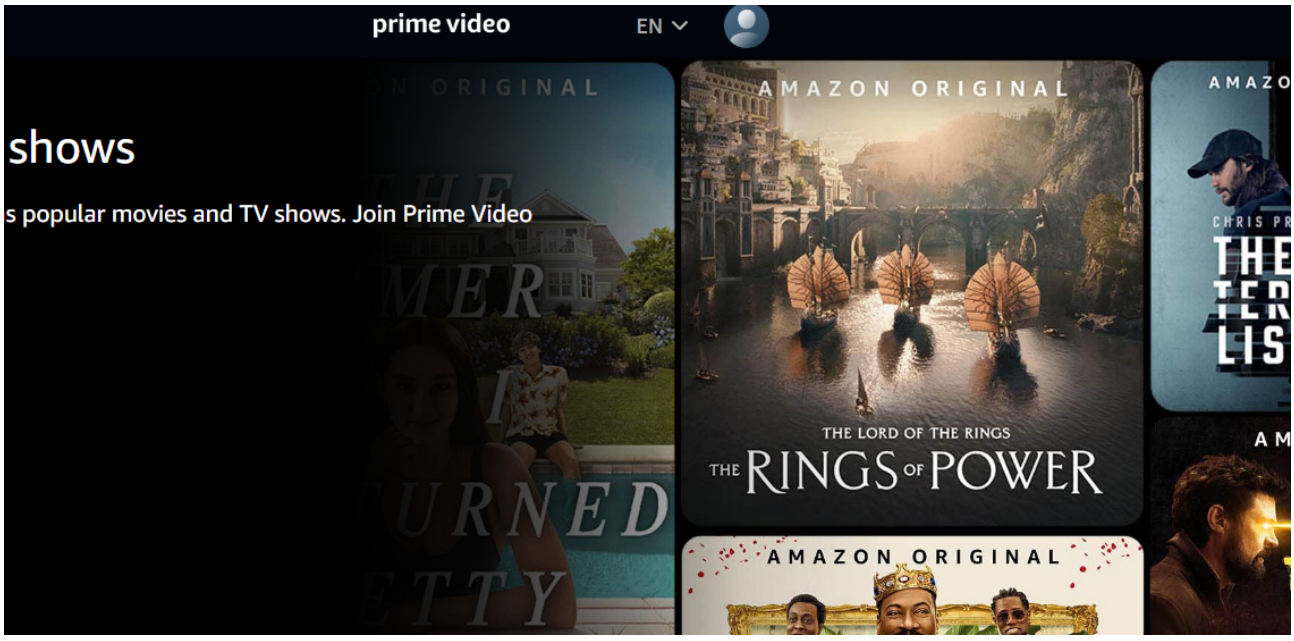

Рисунок 1.2 – Головна сторінка Amazon Prime Video

Однак сервіс розширився, коли Amazon запустив Amazon Prime, програму передплати, яка надавала доступ до безкоштовної доставки товарів та інших переваг. Пізніше Amazon почав включати в цю програму доступ до Amazon Prime Video, що дозволяло передплатникам переглядати тисячі фільмів та телевізійних шоу в межах однієї передплати.

Протягом наступних років Amazon Prime Video активно розвивався та розширював свій контентний каталог. Крім того, компанія почала виробляти

власний оригінальний контент, включаючи серіали та фільми, які стали відомими і популярними серед глядачів.

Ключові події та досягнення [2]:

2006–2011: Початок розвитку

2006 рік: Amazon запускає Amazon Unbox, відеосервіс для оренди та покупки фільмів та телевізійних шоу.

2011 рік: Amazon запускає Amazon Prime, програму з безкоштовною доставкою товарів та іншими перевагами для передплатників.

2011–2015: Початок відеострімінгу

2011 рік: Amazon запускає Amazon Instant Video, відеострімінговий сервіс, доступний для передплатників Amazon Prime.

2013 рік: Amazon починає виробництво оригінальних контенту, випустивши пілотні серії для серіалу "Alpha House" та "Betas".

З 2015 року: Зростання та розвиток

2015 рік: Amazon розширює доступність Prime Video за межі США, включаючи такі країни, як Об'єднане Королівство, Німеччина та Японія.

2016 рік: Amazon випускає великі оригінальні серіали, такі як "The Man in the High Castle" та "Transparent", які отримують високу оцінку критиків.

2017 рік: Amazon оголошує придбання прав на великі спортивні події, такі як "Thursday Night Football" у Національній футбольній лізі (NFL).

2018 рік: Amazon випускає серіал "The Marvelous Mrs. Maisel", який стає популярним та отримує безліч нагород, включаючи премію "Еммі".

З 2020 року: Міжнародний розширення та нові проекти

2020 рік: Amazon Prime Video активно розширює свою присутність у світі, надаючи доступ до великого каталогу в країнах Азії, Європи та Латинської Америки.

2021 рік: Amazon випускає серіал "The Lord of the Rings: The Rings of Power", який стає одним з найбільш очікуваних проектів в історії стрімінгового телебачення.

Amazon Prime Video відомий своїми великими інвестиціями в оригінальний контент, включаючи серіали, фільми та документальні фільми.

Amazon активно придбає права на спортивні події, такі як футбольні матчі NFL та тенісні турніри.

Сервіс постійно розширює свою глобальну присутність, надаючи доступ до контенту в різних країнах.

Amazon використовує передові технології для покращення якості стрімінгу та користувацького досвіду.

Amazon Prime Video продовжує залучати аудиторію своїм оригінальним контентом, розширює свою присутність у різних країнах світу і вдосконалює користувацький досвід для своїх передплатників [2].

Disney+ — це відеострімінговий сервіс, який належить і управляється американською медіакомпанією The Walt Disney Company. Офіційно запущений 12 листопада 2019 року, Disney+ став додатковим способом для Disney поширювати свій великий каталог відеоконтенту (рис 1.3) [3].

Рисунок 1.3 – Головна сторінка Disney+

У серпні 2017 року Disney оголосив про свій намір створити власний відеострімінговий сервіс. У квітні 2019 року компанія представила назву Disney+ та деякі деталі щодо контенту. Сам сервіс був запущений у листопаді 2019 року у США, Канаді та Нідерландах [3].

Після успішного запуску в США, Канаді та Нідерландах Disney+ почав розширювати свою присутність у різних країнах світу. У листопаді 2019 року сервіс був запущений у кількох країнах Європи, а згодом поширився на багато інших ринків.

Disney+ швидко став популярним завдяки своєму великому каталогу класичних анімаційних фільмів Disney, фільмів Marvel та серіалів Star Wars. Крім

того, Disney+ активно виробляв оригінальний контент, такий як серіали "The Mandalorian", "WandaVision" та "The Falcon and the Winter Soldier", які здобули велику популярність серед глядачів.

Disney+ укладав різноманітні партнерства та угоди з іншими медіа-компаніями, що допомагало розширити його каталог контенту та привернути нових передплатників. Також Disney+ приступив до виробництва нових серіалів та фільмів, які спрямовані на різні аудиторії та демографічні групи.

Disney+ продовжує залучати нових передплатників, розширюючи свій каталог контенту та розвиваючи нові серіали та фільми. Компанія також досліджує можливості впровадження технологічних інновацій для поліпшення користувацького досвіду.

HBO Max – це відеострімінговий сервіс, який належить і управляється американським медіаконгломератом WarnerMedia, дочірньою компанією AT&T. Він був запущений 27 травня 2020 року в Сполучених Штатах, згодом розширився на інші країни світу (рис.1.4) [4].

Рисунок 1.4 – Логотип HBO Max

Анонс HBO Max відбувся у липні 2019 року. Сервіс був офіційно запущений у травні 2020 року. Він поєднав в собі вміст каналу HBO, а також інші

відомі бренди та фірмовий контент WarnerMedia, такі як Warner Bros., DC, Cartoon Network, Turner Classic Movies та інші.

Запуск HBO Max супроводжувався прем'єрою оригінального контенту, включаючи серіали "Love Life", "The Not-Too-Late Show with Elmo", "Legendary", "Raised by Wolves" та інші. Крім того, HBO Max оголосив про багато нових оригінальних проектів, що мають вийти в майбутньому.

Після успішного запуску в США HBO Max почав розширюватися на міжнародні ринки. У 2021 році сервіс був запущений у кількох країнах Європи та Латинської Америки, і він планує подальше розширення.

WarnerMedia уклав різноманітні партнерства та угоди з іншими компаніями, щоб привернути нових передплатників та розширити свій контентний каталог. Наприклад, угода з HBO Max та Roku дозволила платформі отримати доступ до мільйонів нових передплатників [3].

HBO Max продовжує активно розвиватися, додаючи новий оригінальний контент, розширюючи свою глобальну присутність та підписуючи нові партнерські угоди.

Цілком можливо, що в майбутньому HBO Max продовжить свій розвиток та розширення, а також буде продовжувати виробляти оригінальний та високоякісний контент для своїх передплатників.

1.2 Огляд методів прогнозування ціни на акції Netflix

Формування ціни на акції стрімінгового сервісу Netflix. На рисунку 1.7 зображено зміни на ціну даної компанії за останні 3 роки [4].

Рисунок 1.5 – Зміна ціни на акції Netflix

1. Фінансові показники компанії:

– Публікація фінансових звітів, що показують доходи, прибутки, витрати та інші ключові фінансові показники, безпосередньо впливає на ціну акцій. Високі доходи та прибутки зазвичай підвищують вартість акцій [4].

– Прогнози майбутніх доходів та прибутків також впливають на ціну акцій. Якщо прогнози перевищують очікування інвесторів, ціна акцій може зрости.

2. Рівень підписників:

– Збільшення кількості підписників Netflix свідчить про зростаючу популярність сервісу, що позитивно впливає на ціну акцій.

– Втрата підписників або менше очікуваного зростання може негативно вплинути на ціну акцій.

3. Конкуренція:

– Поява нових конкурентів або агресивні дії існуючих конкурентів (наприклад, Disney+, Amazon Prime Video, HBO Max) можуть вплинути на ринкову позицію Netflix, що може відобразитися на ціні акцій.

– Інновації або успішні стратегії конкурентів можуть зменшити частку ринку Netflix.

4. Новини та події:

– Позитивні новини, такі як успішні випуски нових серіалів або фільмів, укладення вигідних угод, можуть підвищити ціну акцій.

– Зміни в економічній ситуації, новини про технологічний сектор і ринок стрімінгових послуг також впливають на акції Netflix.

5. Макроекономічні фактори [4]:

– Загальний стан економіки, включаючи показники ВВП, рівень безробіття, інфляцію, впливає на фондові ринки загалом і на акції Netflix зокрема.

– Політичні рішення, такі як регулювання інтернет-індустрії, міжнародна торгівля, можуть мати значний вплив.

6. Аналітичні оцінки та рекомендації:

– Рекомендації аналітиків з інвестиційних банків і дослідницьких фірм можуть впливати на попит на акції. Підвищення рейтингів або цільових цін зазвичай підвищує ціну акцій, і навпаки.

– Звіти та дослідження, що вказують на перспективи зростання або ризики, можуть також впливати на ціну акцій.

7. Інсайдерська інформація:

– Купівля або продаж акцій керівництвом компанії може подавати сигнали ринку про їхні очікування щодо майбутнього компанії, що впливає на ціну акцій.

– Зміни у складі керівництва або ключових менеджерів можуть впливати на ринкові очікування та ціну акцій.

Ноутбук «Time-Series-Analysis-Netflix-Stock-Price». Основні методи аналізу включають перевірку стаціонарності серій, використання ADF тесту, та сезонне декомпонування. Через низьку стаціонарність даних вдається до сезонного декомпонування та ресемплінгу даних для аналізу [5].

Для прогнозування майбутніх цін, автор застосував модель SARIMAX (рис. 1.6 , 1.7).

```
# Assuming your time series data is in a column named 'stock_price'
ts = df_timeseries_filtered_monthly['typical']

# Define and fit the SARIMA model
model = SARIMAX(ts, order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))
results = model.fit()

# Forecast future values
forecast = results.forecast(steps=12)  # Forecast for a year

# Plot the original data and the forecast
plt.plot(ts.index, ts, label='Original Data')
plt.plot(forecast.index, forecast, label='Forecast')
plt.xlabel('Date')
plt.ylabel('Typical Stock Price')
plt.title('Time Series Forecasting with SARIMA')
plt.legend()
plt.show()
```

Рисунок 1.6 – Налаштування моделі SARIMAX в ноутбуці «Time-Series-Analysis-Netflix-Stock-Price»

Модель SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous variables) є розширенням моделі ARIMA, що включає сезонні компоненти та зовнішні змінні. Ця модель часто використовується для аналізу часових рядів, де є як сезонність, так і зовнішні впливи на дані. SARIMAX дозволяє моделювати вплив незалежних змінних на часовий ряд, що робить її особливо корисною в економетриці, фінансах та інших галузях, де потрібно прогнозувати майбутні значення з урахуванням додаткових факторів [5].

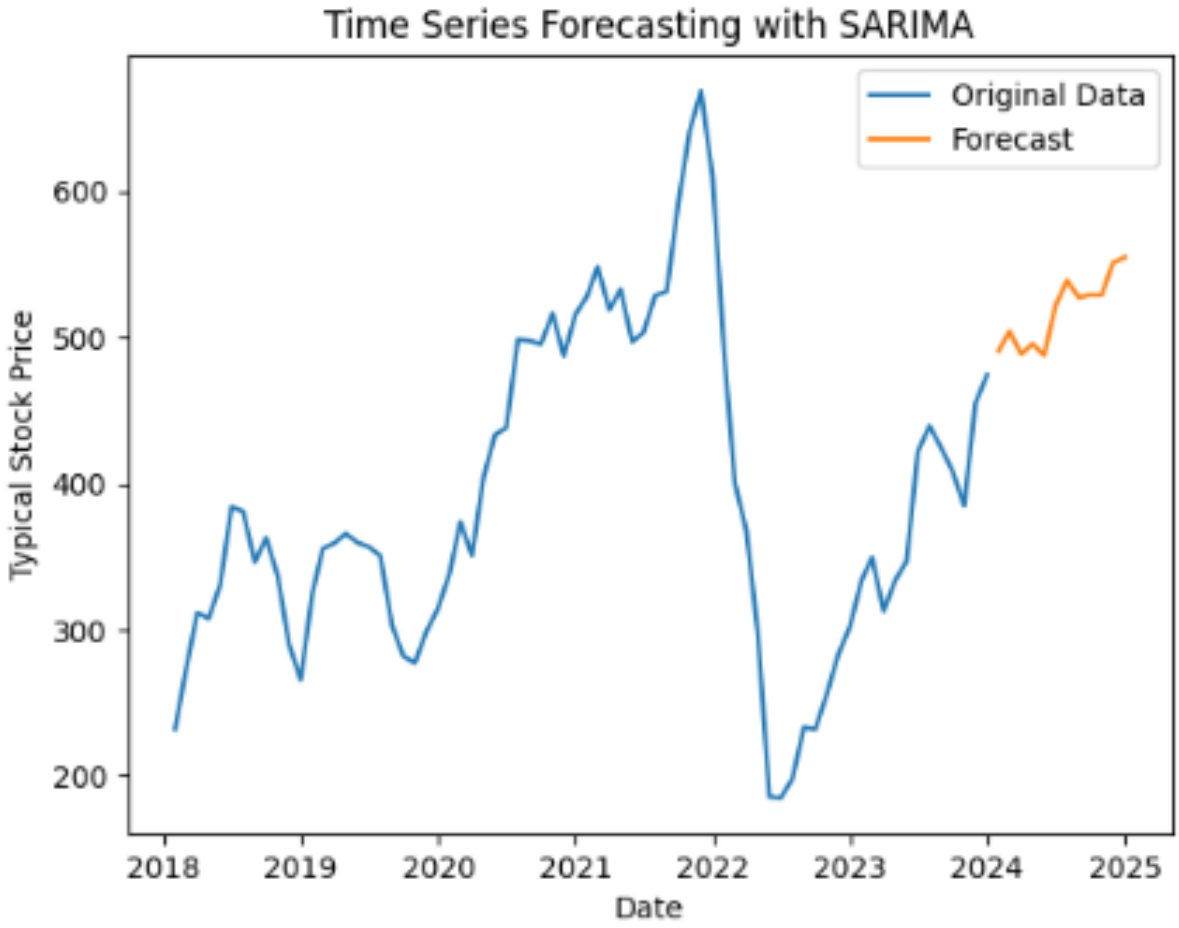

Рисунок 1.7 – Результат роботи SARIMAX в ноутбуці «Time-Series-Analysis-Netflix-Stock-Price»

Ноутбук «Garch 1». Автор використав модель GARCH (Generalized Autoregressive Conditional Heteroskedasticity) — це статистична модель, яка використовується для аналізу і прогнозування часових рядів, особливо тих, що показують волатильність, як у фінансових ринках. Модель GARCH дозволяє

моделювати і передбачати зміни волатильності часових рядів, залежно від часу, що робить її дуже корисною для фінансових даних, де волатильність не є постійною або однорідною. GARCH часто використовується для оцінювання ризику і управління інвестиційними портфелями (рис. 1.8 , 1.9) [6].

```
model = arch_model(returns, vol='GARCH', p=5, o=2, q=0)
res = model.fit(update_freq=5)
print(res.summary())
```

```
Iteration:      5,   Func. Count:     60,   Neg. LLF: 5626.111069952705
Iteration:     10,   Func. Count:    116,   Neg. LLF: 4270.291888851274
Iteration:     15,   Func. Count:    171,   Neg. LLF: 4248.749690160375
Optimization terminated successfully    (Exit mode 0)
            Current function value: 4248.7436034518805
            Iterations: 19
            Function evaluations: 211
            Gradient evaluations: 19
                  Constant Mean - GJR-GARCH Model Results
==============================================================================
Dep. Variable:                  Close   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                  GJR-GARCH   Log-Likelihood:               -4248.74
Distribution:                  Normal   AIC:                            8515.49
Method:            Maximum Likelihood   BIC:                            8564.74
                                        No. Observations:                 1759
Date:                Sun, Apr 14 2024   Df Residuals:                     1758
Time:                        10:54:28   Df Model:                            1
                            Mean Model
============================================================================
                 coef    std err          t      P>|t|  95.0% Conf. Int.
----------------------------------------------------------------------------
mu             0.0976      0.204      0.477      0.633 [ -0.303,  0.498]
                              Volatility Model
============================================================================
                 coef    std err          t      P>|t|     95.0% Conf. Int.
----------------------------------------------------------------------------
omega          4.4335      1.302      3.406  6.595e-04    [  1.882,  6.985]
alpha[1]       0.2194      0.475      0.462      0.644    [ -0.711,  1.150]
alpha[2]       0.0316  7.317e-02      0.432      0.666    [ -0.112,  0.175]
alpha[3]       0.0757      0.459      0.165      0.869    [ -0.823,  0.974]
alpha[4]       0.2141      0.145      1.480      0.139 [-6.945e-02,  0.498]
alpha[5]       0.0247  4.970e-02      0.496      0.620 [-7.274e-02,  0.122]
gamma[1]      -0.0267      0.390 -6.836e-02      0.946    [ -0.792,  0.738]
gamma[2]   4.5829e-03  9.570e-02  4.789e-02      0.962    [ -0.183,  0.192]
============================================================================

Covariance estimator: robust
```

Рисунок 1.8 – Використання моделі GARCH

Результати, представлені на зображенні 2., відносяться до аналізу моделі GJR-GARCH, використаної для вимірювання волатильності цін на закриття (змінної `Close`). Модель включає декілька параметрів [6]:

– mu (μ) – середнє значення моделі, коефіцієнт становить 0.0976 зі стандартною помилкою 0.204 і p-значенням 0.633, що свідчить про те, що воно не є статистично значущим.

– omega – коефіцієнт базової волатильності дорівнює 4.4335 з p-значенням, близьким до 0, що підтверджує його значимість.

– alpha[1] до alpha[5] та gamma[1] і gamma[2] – параметри, що відображають вплив минулих значень та асиметрії шоків на поточну волатильність. більшість з них не є статистично значущими, окрім omega.

Модель не має значущого R-квадрату, що вказує на низьку пояснювальну здатність моделі для змінної `Close`. Загалом, ця модель дозволяє зрозуміти динаміку волатильності, але має певні обмеження в прогнозуванні точних значень цін закриття.

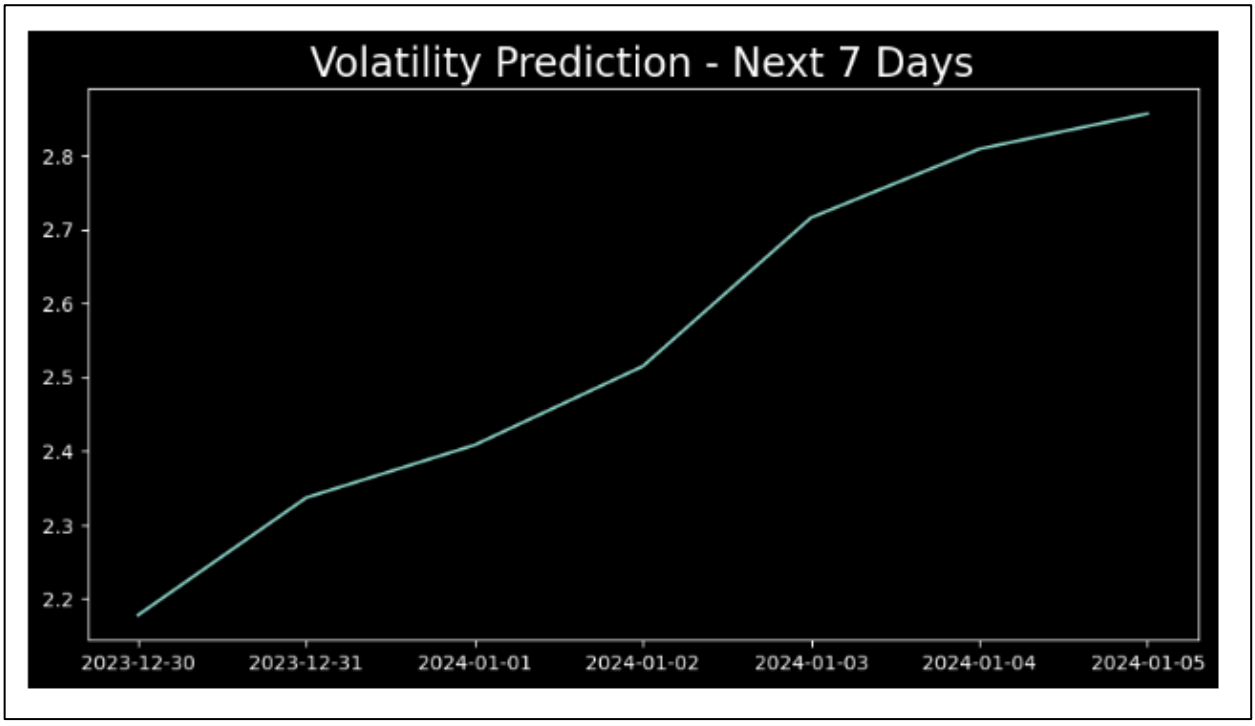

Рисунок 1.9 – Результат роботи моделі GARCH

Інші автори використовували моделі: лінійна регресія, градієнтний бустинг та Prophet. Лінійна регресія показала точність 0,97, але оскільки автор рішення не перевірив дані на автокореляцію, що викликає підозру до надійності моделі. Така ж ситуація і для моделі XGBoost Regressor. У автора рішення точність моделі 0,997. Такий результат говорить про те, що модель перенавчилась і є ненадійною [7,8].

Публікації використані в роботі охоплюють різні аспекти аналізу та прогнозування цін на акції за допомогою сучасних технічних підходів, включаючи нейронні мережі, алгоритми машинного навчання, і гібридні моделі. Вони досліджують ефективність таких методів, як GARCH, ARIMA у контексті фінансових ринків. Статті аналізують волатильність, вплив зовнішніх чинників на ціни акцій, а також практичне застосування передових моделей для оптимізації інвестиційних стратегій [9-15].

У деяких статтях описується прогнозування ціни на акції з використанням рекурентних нейронних мереж[15-26].

Найчастіше зустрічається мережа LSTM (Long Short-Term Memory) — це тип рекурентної нейронної мережі (RNN), оптимізований для вирішення проблеми зникнення градієнту, що часто зустрічається в традиційних RNN. LSTM дизайновано так, щоб краще запам'ятовувати інформацію на тривалі часові проміжки. Це робить його особливо корисним для завдань, де необхідно враховувати контекст і довгострокові залежності, таких як прогнозування часових рядів, обробка природної мови та розпізнавання рукописного тексту.

1.3 Висновки

В першому розділі проаналізовано загальну тематику стрімінгових сервісів. Проведено порівняльний аналіз популярних стрімінгів у світі, таких як Netflix, Amazon Prime Video, Disney+. За даними аналізу найбільш популярний стрімінг це Netflix.

Розглянуто процес ціноутворення акцій стрімінгових сервісів. Визначено основні характеристики за якими може утворюватися ціна акцій, а саме фінансові показники компанії, рівень підписників, конкуренція, новини та події, макроекономічні фактори, аналітичні оцінки та рекомендації інсайдерська інформація.

Проведено огляд основних підходів до аналізу та прогнозуванню ціни на акції Netflix в Kaggle Notebooks:

- «Time-Series-Analysis-Netflix-Stock-Price»,
- «Garch 1»,
- «XGboost: Predict "next_day_close"»,
- «Very Simple Prediction of Next Day Stock (99.76%)».

В даних підходах використано методи ARIMA, Garch, XGboost та лінійна регресія.

2. РОЗВІДУВАЛЬНИЙ АНАЛІЗ ДАНИХ ТА ІНЖЕНЕРІЯ ОЗНАК ЦІНИ НА NETFLIX

2.1 Розвідувальний аналіз даних ціни на Netflix

Для виконання роботи обрано датасет ціни на акції Netflix «Netflix Stock Price With Indicators». Цей набір даних містить інформацію про зміни ціни компанії з 2014 року по 2023 рік [27].

Першим кроком завантажимо дані в Kaggle notebook, що зображено на рисунку 2.1.

```
df = pd.read_csv('/kaggle/input/netflix-stock-price-with-indicators/nflx_2014_2023.csv')
df.head()
```

Рисунок 2.1 – Завантаження даних для аналізу

Переглянемо дані доступні для аналізу (рис. 2.2).

	date	open	high	low	close	volume	rsi_7	rsi_14	cci_7
0	2014-01-02	52.401428	52.511429	51.542858	51.831429	12325600	34.729664	49.183584	-89.573201
1	2014-01-03	52.000000	52.495712	51.842857	51.871429	10817100	35.587886	49.457208	-65.820581
2	2014-01-06	51.889999	52.044285	50.475716	51.367142	15501500	29.820674	46.087900	-121.472559
3	2014-01-07	49.684284	49.698570	48.152859	48.500000	36167600	14.371863	32.522091	-206.762171
4	2014-01-08	48.104286	49.425713	48.074287	48.712856	20001100	18.049045	34.073549	-117.836707

cci_7	cci_14	sma_50	ema_50	sma_100	ema_100	macd	bollinger	TrueRange	atr_7	atr_14	next_day_close
-89.573201	-131.288579	50.112828	50.235157	46.385428	46.650698	0.751929	52.607357	1.052857	1.161182	1.247748	51.871429
-65.820581	-103.026189	50.228771	50.299327	46.537571	46.754726	0.624259	52.656143	0.664283	1.090197	1.206072	51.367142
-121.472559	-139.640566	50.312571	50.341203	46.680971	46.846621	0.476890	52.666928	1.568569	1.158535	1.231965	48.500000
-206.762171	-238.029120	50.336228	50.268997	46.791957	46.879558	0.127277	52.560214	3.214283	1.452214	1.373559	48.712856
-117.836707	-180.766801	50.373257	50.207969	46.917071	46.916075	-0.131106	52.455357	1.351426	1.437815	1.371978	48.150002

Рисунок 2.2 – Дані доступні для аналізу

За допомогою команди info() переглянемо інформацію про дані, такі як назви ознак, кількість даних в кожній ознаці та тип даних (рис. 2.3).

```
RangeIndex: 2516 entries, 0 to 2515
Data columns (total 20 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   date            2516 non-null   object
 1   open            2516 non-null   float64
 2   high            2516 non-null   float64
 3   low             2516 non-null   float64
 4   close           2516 non-null   float64
 5   volume          2516 non-null   int64
 6   rsi_7           2516 non-null   float64
 7   rsi_14          2516 non-null   float64
 8   cci_7           2516 non-null   float64
 9   cci_14          2516 non-null   float64
 10  sma_50          2516 non-null   float64
 11  ema_50          2516 non-null   float64
 12  sma_100         2516 non-null   float64
 13  ema_100         2516 non-null   float64
 14  macd            2516 non-null   float64
 15  bollinger       2516 non-null   float64
 16  TrueRange       2516 non-null   float64
 17  atr_7           2516 non-null   float64
 18  atr_14          2516 non-null   float64
 19  next_day_close  2516 non-null   float64
dtypes: float64(18), int64(1), object(1)
memory usage: 393.2+ KB
```

Рисунок 2.3 – Інформація про дані

Зробимо опис кожної ознаки:

– Відкриття (Open): Ціна акції на момент відкриття торгів;

– Максимум (High): Найвища ціна акції протягом торгового дня;

– Мінімум (Low): Найнижча ціна акції протягом дня;

– Закриття (Close): Ціна акції на кінець торгового дня;

– Об'єм (Volume): Загальна кількість акцій, що були продані протягом дня;

– RSI_7 / RSI_14 (Індекс відносної сили): Цей індекс показує швидкість та зміни цінових рухів за 7 або 14 днів;

– CCI_7 / CCI_14 (Індекс каналу товару): Цей індикатор допомагає виявляти нові тренди або екстремальні умови на ринку за 7 або 14 днів;

– SMA_50 / SMA_100 (Проста ковзна середня): Розрахунок середньої ціни акції за останні 50 або 100 днів;

– EMA_50 / EMA_100 (Експоненційна ковзна середня): Дає більшу вагу недавнім цінам і швидше реагує на зміни цін за останні 50 або 100 днів;

– MACD (Індикатор конвергенції/дивергенції ковзних середніх): Показує відносини між двома ковзними середніми цінами акції;

– Болліндженрові стрічки (Bollinger Bands): Встановлені на відстані двох стандартних відхилень від простої ковзної середньої ціни акції, можуть вказувати на рівні підтримки або опору;

– Істинний діапазон (True Range): Найбільший із наступних: поточний максимум мінус поточний мінімум, абсолютна величина поточного максимуму мінус попереднє закриття, або абсолютна величина поточного мінімуму мінус попереднє закриття;

– ATR_7 / ATR_14 (Середній істинний діапазон): Міра волатильності, що показує, наскільки середньо змінюється ціна акції за 7 або 14 днів;

– Ціна закриття наступного дня (Next Day Close): Ціна закриття акції на наступний торговий день, використовується для прогнозування.

За допомогою функції describe() сформуємо звіт описової статистики по даним ціни акції Netflix (рис.2.4).

	open	high	low	close	volume	rsi_7	rsi_14	cci_7	cci_14	sma_50	ema_50	sma_100	ema_100
count	2516.000000	2516.000000	2516.000000	2516.000000	2.516000e+03	2516.000000	2516.000000	2516.000000	2516.000000	2516.000000	2516.000000	2516.000000	2516.000000
mean	274.455767	278.638946	270.124592	274.487023	1.049230e+07	53.399584	53.424079	11.920763	15.705041	270.416745	270.522273	266.760420	266.813442
std	166.005094	168.205188	163.612279	165.902954	9.173072e+06	17.763862	13.088270	100.831995	111.396602	164.404981	163.454752	163.189487	161.050027
min	44.605713	45.842857	42.785713	44.887142	1.144000e+06	4.374756	9.152344	-233.333333	-424.012878	49.679943	49.409771	46.385428	46.650698
25%	109.982502	111.887501	107.117498	110.064998	5.017050e+06	40.015924	43.731175	-74.565801	-71.870319	107.627900	105.923105	105.662982	102.490532
50%	288.000000	292.690002	282.660004	288.229995	7.795950e+06	53.953919	53.668026	25.632766	27.165946	286.034599	287.382611	287.634151	291.379066
75%	384.542511	391.317505	377.795006	384.560005	1.299060e+07	67.151119	63.378696	94.212764	100.321908	376.592201	381.284113	365.740875	369.343945
max	692.349976	700.989990	686.090027	691.690002	1.333875e+08	96.305710	91.547868	233.333333	356.795719	648.592997	642.765550	616.771798	610.750059

Рисунок 2.4 – Описова статистика ціни на акції Netflix

Загальна кількість записів становить 2516. Середня ціна відкриття становить 274,46 USD, з максимумом у 692,35 USD. Об'єм торгів в середньому дорівнює близько 10 мільйонів акцій за день. Індекс RSI_7 має середнє значення 53,40, а RSI_14 — 53,42, що вказує на не дуже високий рівень зміни ціни. Індикатор CCI для 7 днів середньо становить 11,92, а для 14 днів — 15,71, що відображає тренди та екстремальні умови ринку. Вказані також середні та експоненційні ковзні середні (SMA та EMA) за 50 і 100 днів, які допомагають аналізувати довгострокові тренди.

Побудуємо графік динаміки зміни ціни на акції (рис. 2.5)

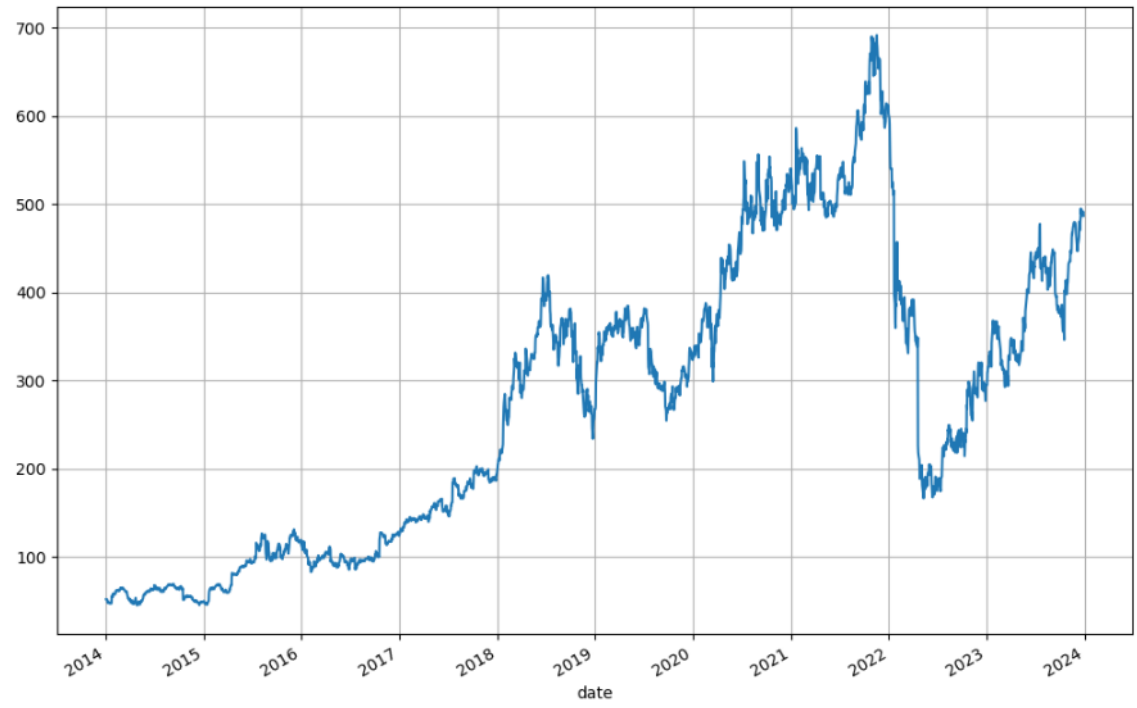

Рисунок 2.5 – Зміна ціни на акції з 2014 по 2023 рік

Створимо динамічний графік ціни на акції за допомогою бібліотек Python (рис. 2.6, 2.7) [28].

```
def c_chart(data,label):
    # Thanks to https://www.kaggle.com/code/fangya/cryptocurrency-data-visualization-arima
    candlestick = go.Figure(data = [go.Candlestick(x=data.index,
                                                   open = data['open'],
                                                   high = data['high'],
                                                   low = data['low'],
                                                   close = data['close'])])
    candlestick.update_xaxes(title_text = 'Time',
                             rangeslider_visible = True)

    candlestick.update_layout(
    title = {
            'text': '{:} Candelstick Chart'.format(label),
            "y":0.8,
            "x":0.5,
            'xanchor': 'center',
            'yanchor': 'top'})

    candlestick.update_yaxes(title_text = 'Price in USD', ticksuffix = '$')
    return candlestick
```

Рисунок 2.6 – Функція побудови динамічного графіку

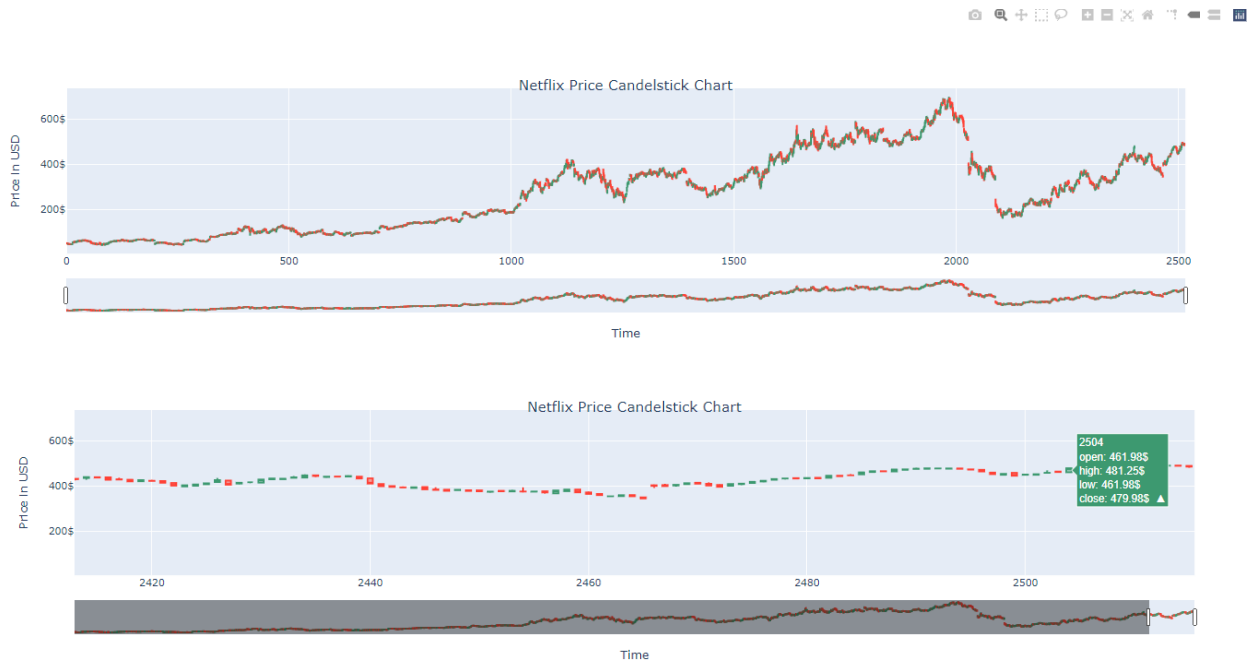

Рисунок 2.7 – Динамічний графік ціни на акції

Перевіримо дані на сезонність та побудуємо відповідні графіки, що зображено на рисунках 2.8 та 2.9.

```
result = seasonal_decompose(df['close'], model='additive', period=252)  # 252 торгові дні у році
result.plot()

plt.show()
```

Рисунок 2.8 – Код візуалізації даних для розбиття на сезони

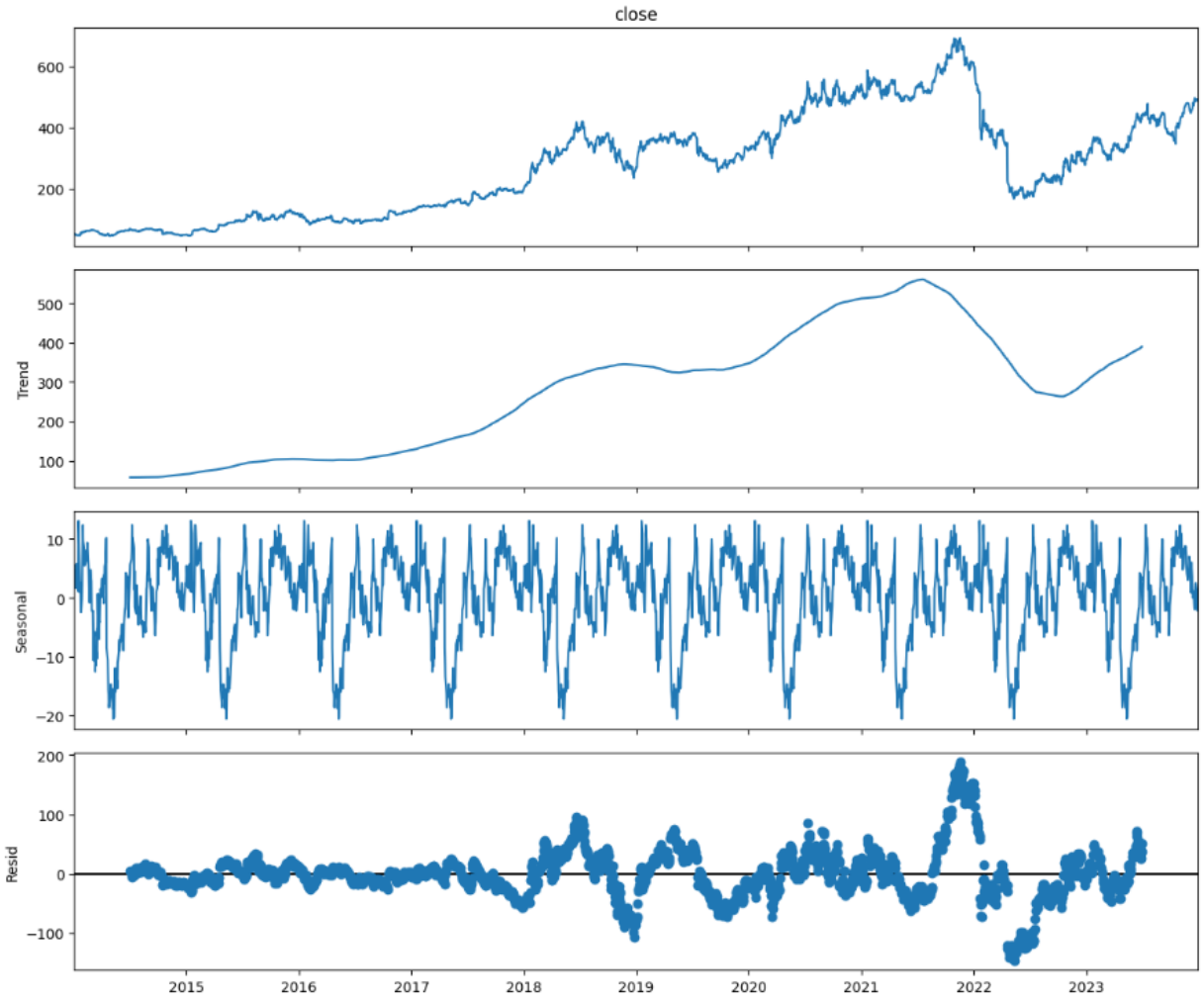

Рисунок 2.9 – Декомпозиція даних

Графіки декомпозиції часового ряду для цін закриття акцій:

1. Перший графік (вгорі) – відображає денні ціни закриття акцій протягом часу. Ви можете бачити загальний тренд зростання до середини періоду, потім зниження, і знову зростання ближче до кінця періоду.

2. Другий графік – показує тренд цін закриття, який було отримано за допомогою видалення сезонності та випадковостей із даних. Тренд зростає до 2018 року, після чого спостерігається зниження та стабілізація.

3. Третій графік – показує сезонні коливання в даних. Видно, що коливання мають досить регулярний характер, хоча і не виражені дуже сильно.

4. Четвертий графік (внизу) – відображає випадкові компоненти або "шум" даних, який залишається після вилучення тренду та сезонності. Ці залишки показують коливання, які не можна пояснити трендом чи сезонними факторами.

2.2 Відбір ознак для передбачення даних

Використаємо бібліотеку TSFRESH для створення нових ознак.

TSFRESH - це бібліотека для автоматичного виділення та екстракції ознак з часових рядів. Вона розроблена для вирішення задач машинного навчання, де вхідні дані є часовими рядами, наприклад, у сенсорних даних або в задачах моніторингу технічного стану.

Основні функції TSFRESH включають [28]:

1. TSFRESH може автоматично створювати великий набір статистичних ознак з часових рядів, таких як середнє значення, медіана, стандартне відхилення, автокореляція, та багато інших.

2. TSFRESH дозволяє використовувати додаткові знання про дані для налаштування процесу екстракції ознак.

3. Бібліотека підтримує роботу з різними типами часових рядів, включаючи часові ряди з однією змінною, мультизмінні часові ряди та декілька часових рядів.

4. Ознаки, що витягуються за допомогою TSFRESH, можуть бути використані як вхідні дані для класифікаційних та регресійних моделей машинного навчання.

	index__sum_values	index__abs_energy	index__median	index__mean	index__root_mean_square	index__maximum	index__absolute_maximum	index__minimum	index__benford_correlation	index__qu
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	NaN	
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.864123	
2	2.0	4.0	2.0	2.0	2.0	2.0	2.0	2.0	0.295657	
3	3.0	9.0	3.0	3.0	3.0	3.0	3.0	3.0	0.062915	
4	4.0	16.0	4.0	4.0	4.0	4.0	4.0	4.0	-0.064614	
...	...	...	...	...	...	...	...	...	...	
2511	2511.0	6305121.0	2511.0	2511.0	2511.0	2511.0	2511.0	2511.0	0.295657	
2512	2512.0	6310144.0	2512.0	2512.0	2512.0	2512.0	2512.0	2512.0	0.295657	
2513	2513.0	6315169.0	2513.0	2513.0	2513.0	2513.0	2513.0	2513.0	0.295657	
2514	2514.0	6320196.0	2514.0	2514.0	2514.0	2514.0	2514.0	2514.0	0.295657	
2515	2515.0	6325225.0	2515.0	2515.0	2515.0	2515.0	2515.0	2515.0	0.295657	

516 rows × 51 columns

Рисунок 2.10 – Результат роботи TSFRESH

Сформовано 51 нова ознака, які можна використати для тренування моделей прогнозування. Об’єднаємо основний та додатковий набори даних, результат на рисунку 2.11.

```
df = pd.merge(df, extracted_features_clean, how='left', on='date')
df
```

	date	open	high	low	close	volume	rsi_7	rsi_14	cci_7	
0	2014-01-02	52.401428	52.511429	51.542858	51.831429	12325600	34.729664	49.183584	-89.573201	-1.
1	2014-01-03	52.000000	52.495712	51.842857	51.871429	10817100	35.587886	49.457208	-65.820581	-1
2	2014-01-06	51.889999	52.044285	50.475716	51.367142	15501500	29.820674	46.087900	-121.472559	-1.
3	2014-01-07	49.684284	49.698570	48.152859	48.500000	36167600	14.371863	32.522091	-206.762171	-2
4	2014-01-08	48.104286	49.425713	48.074287	48.712856	20001100	18.049045	34.073549	-117.836707	-1
...	...	...	...	...	...	...	...	...	...	
2511	2023-12-22	494.000000	496.019989	485.450012	486.760010	2701100	61.286700	62.629263	42.406864	
2512	2023-12-26	489.390015	491.480011	486.380005	491.190002	2034500	65.324458	64.613544	31.476352	
2513	2023-12-27	491.239990	494.019989	489.250000	491.790009	2561300	65.886676	64.885498	54.412289	
2514	2023-12-28	492.000000	492.890015	489.070007	490.510010	1710500	63.331047	63.759744	-14.020755	
2515	2023-12-29	490.369995	492.230011	481.940002	486.880005	2739500	56.127883	60.551251	-157.815293	

2516 rows × 70 columns

Рисунок 2.11 – Результат об’єднання даних

Для перевірки точності моделей будемо прогнозувати ціну на акції в 2023 році. Для цього надані дані розділимо на для набори, перший з 2014 по 2022 рік та другий — 2023 рік (рис. 2.12).

```
split_date = pd.Timestamp('2022-12-31')
train = df[df['date'] < split_date]
test = df[df['date'] >= split_date]
```

Рисунок 2.12 – Розбиття на тренувальний та тестовий набір даних

2.3 Вибір моделей прогнозування ціни на акції

Вибір моделей машинного навчання для прогнозування цін на акції залежить від багатьох факторів, включаючи тип даних, масштаб задачі, доступні обчислювальні ресурси, та бажану точність. Ось деякі популярні моделі та техніки, які використовуються в аналізі фінансових ринків:

1. Лінійна регресія. Це одна з найпростіших і найбільш традиційних моделей для прогнозування. Вона добре працює, коли існує лінійний зв'язок між незалежними змінними та ціною акцій.

2. Лінійні моделі для часових рядів:

– ARIMA/SARIMA: Ефективні для даних, де важливі тренди та сезонність;

– Експоненціальне згладжування, добре підходить для даних з вираженою сезонністю;

– Prophet.

3. Моделі на основі дерев рішень:

– Random Forest та Gradient Boosting Machines (GBM) – потужні ансамблеві методи, які можуть враховувати нелінійність і взаємодії між змінними;

– XGBoost, LightGBM – ці моделі широко використовуються в індустрії завдяки своїй високій продуктивності та швидкості.

4. Моделі на основі нейронних мереж:

– Штучні нейронні мережі (ANN) – гнучкі та потужні, можуть моделювати складні взаємозв'язки між вхідними змінними.

– Рекурентні нейронні мережі (RNN), особливо LSTM та GRU – ідеальні для моделювання секвенційних даних, які часто зустрічаються в фінансових часових рядах.

Для прогнозування ціни на акції компанії Netflix використаємо моделі Prophet, ARIMA та XGBoost.

2.4 Висновки

В другому розділі проведено розвідувальний аналіз ціни на акції компанії Netflix. Дані про ціну на акції перевірено на аномалії та викиди.

Проведено інженерію ознак. Використано TSFRESH для генерації нових ознак. Перевірено дані на аномалії та викиди. Визначено вплив ознак Open, High, Low на ознаку Close, що є цільовою.

Проаналізовано стандартні моделі машинного навчання для прогнозування ціни та обрано 3 моделі для виконання роботи, а саме Prophet, ARIMA та XGBoost.

3. РЕЗУЛЬТАТИ ПРОГНОЗУВАННЯ ЦІНИ НА АКЦІЇ

3.1 Прогнозування моделлю Prophet

Для прогнозування ціни на акції використаємо бібліотеку Prophet. Для цього сформуємо новий набір даних який складається з дати та ціни закриття акції (рис. 3.1) [29].

```
df_p = df[['date', 'close']]
```

+ Code + Markdown

```
df_p.rename(columns={'date': 'ds', 'close': 'y'}, inplace=True)
```

Рисунок 3.1 – Результат роботи коду

Наступним кроком натренуємо модель на основі наших даних, що зображено на рисунку 3.2.

```
model = Prophet()
model.fit(df_p)
```

```
16:23:25 - cmdstanpy - INFO - Chain [1] start processing
16:23:26 - cmdstanpy - INFO - Chain [1] done processing
<prophet.forecaster.Prophet at 0x7abc122fb580>
```

Рисунок 3.2 – Прогнозування ціни на акції

Для перевірки роботи моделі порівняємо дані за 2023 рік та зроблений прогноз (рис. 3.3) та зробимо оцінку точності роботи моделі за допомогою метрики RMSE та MAPE (рис. 3.4).

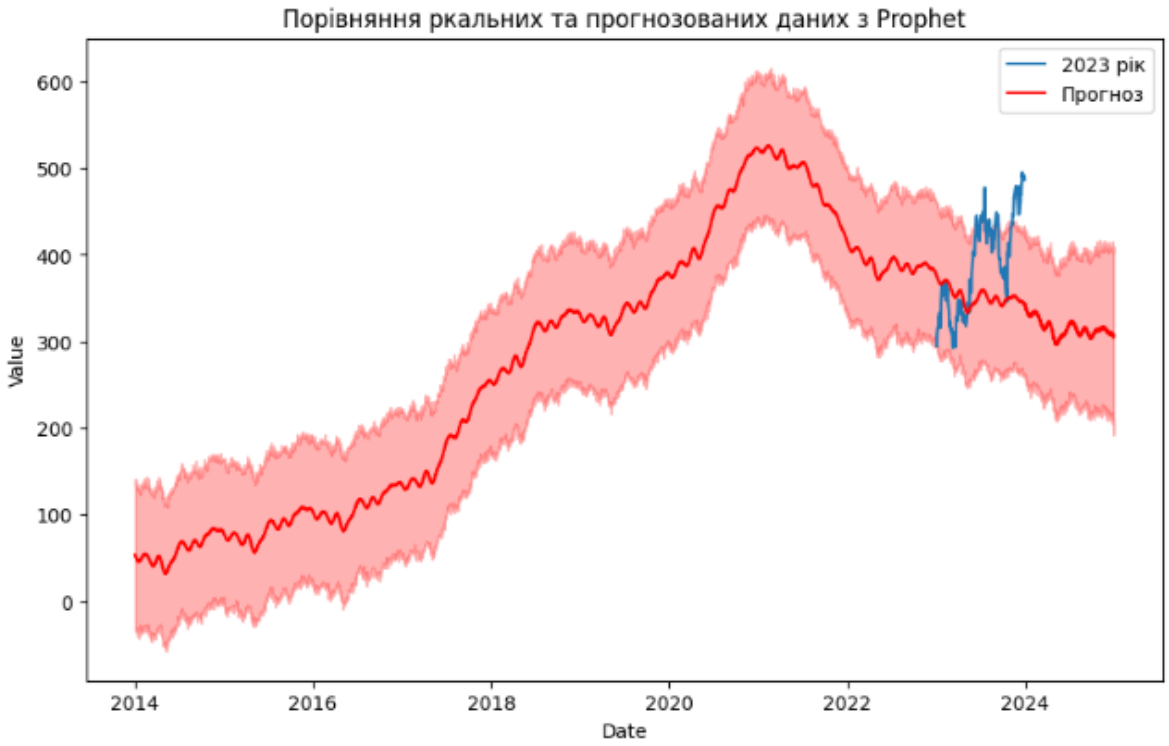

Рисунок 3.3 – Порівняння реальних та прогнозованих даних

```
rmse = np.sqrt(mean_squared_error(test_p['y'], forecast_t['yhat']))
mape = np.mean(np.abs((test_p['y'] -forecast_t['yhat']) / forecast_t['yhat'])) * 100
print("Root Mean Squared Error (RMSE):", rmse)
print(f"MAPE: {mape}%")
```

```
Root Mean Squared Error (RMSE): 69.36505993869258
MAPE: 16.556988081727265%
```

Рисунок 3.4 – Оцінка точності

Як видно з рисунку 3.4, точність моделі дуже низька і складає 16,6%. Отже потрібно провести тюнінг моделі Prophet та перевірити результат (рис 3.5).

Параметри, які можна тюнити:

– seasonality_mode – вибір між "additive" та "multiplicative" сезонністю залежно від характеру даних.

– changepoint_prior_scale – регулювання гнучкості моделі для виявлення змін у тренді; більше значення веде до більш чутливої моделі.

– seasonality_prior_scale – Налаштування гнучкості сезонних компонентів.

– holidays_prior_scale – Впливає на ступінь впливу святкових днів на модель.

```
model1 = Prophet(
    daily_seasonality=True,
    weekly_seasonality=True,
    yearly_seasonality=True,
    changepoint_range=1,
    changepoint_prior_scale = 0.01,
    seasonality_mode = 'multiplicative'
)

# Додавання зовнішніх регресорів, якщо вони є
# model.add_regressor('additional_regressor')
model1.add_country_holidays(country_name='US')

# Навчання моделі
model1.fit(train_p)

# Створення майбутніх дат для прогнозування
future1 = model1.make_future_dataframe(periods=10, freq='D')

# Прогнозування
forecast1 = model1.predict(future1)

# Виведення результатів
print(forecast1[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail())
```

```
20:09:10 - cmdstanpy - INFO - Chain [1] start processing
20:09:15 - cmdstanpy - INFO - Chain [1] done processing
            ds        yhat  yhat_lower  yhat_upper
2511 2023-12-20  432.493369  403.300154  461.194184
2512 2023-12-21  433.492956  404.649510  463.621173
2513 2023-12-22  432.811235  401.061531  465.793047
2514 2023-12-23  364.062085  334.722477  393.966730
2515 2023-12-24  364.941586  333.087175  394.386197
```

Рисунок 3.5 – Оновлені параметри моделі Prophet

Для покращення результату період прогнозування зменшили з 365 днів до 10. За заданими параметрами проведемо навчання та побудуємо графік (рис. 3.6)

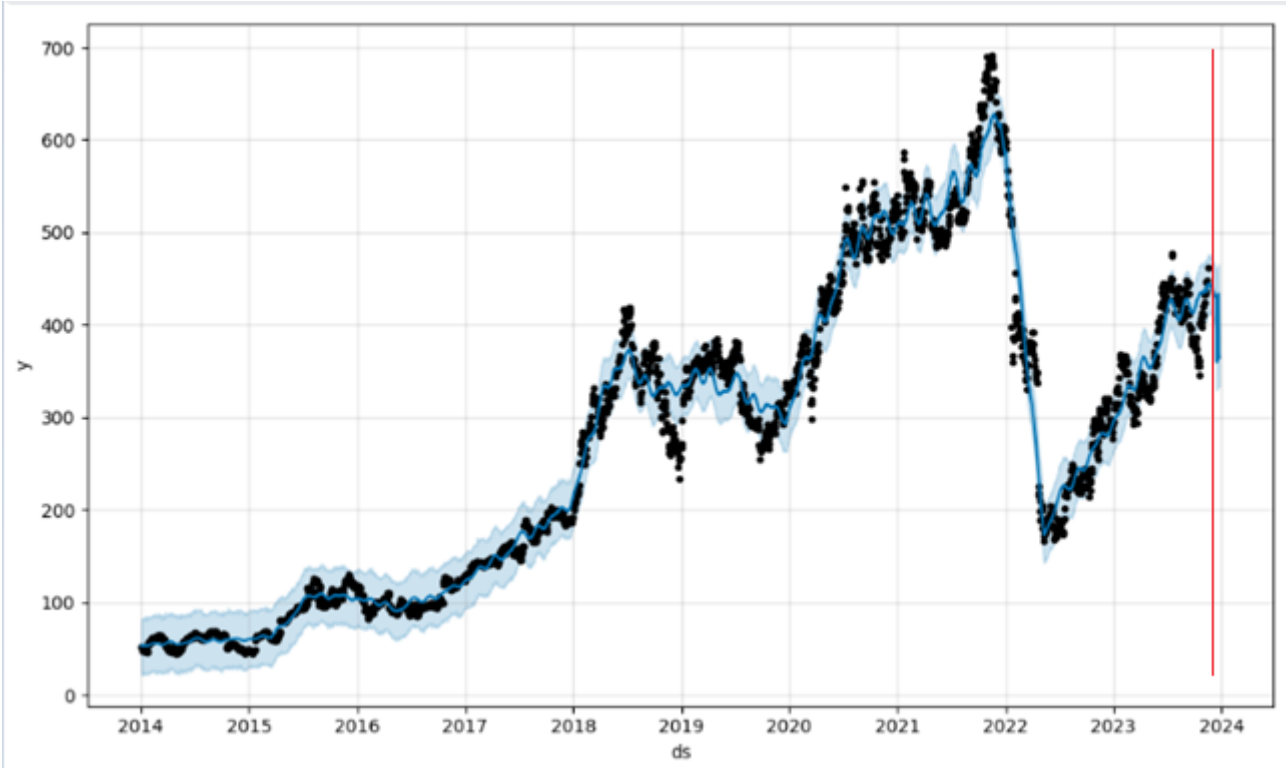

Рисунок 3.6 – Прогноз налаштованої моделі (розділити на тест та трейн)

Основні спостереження:

– Ціна акцій показує явний тренд на зростання протягом декількох років до 2020, після чого спостерігається волатильність із значними коливаннями;

– Прогноз на 2023-2024 роки показує тренд на відновлення після падіння, хоча з великою невизначеністю в довгостроковій перспективі.

Проведемо декомпозицію графіка та проаналізуємо його на рисунку 3.7.

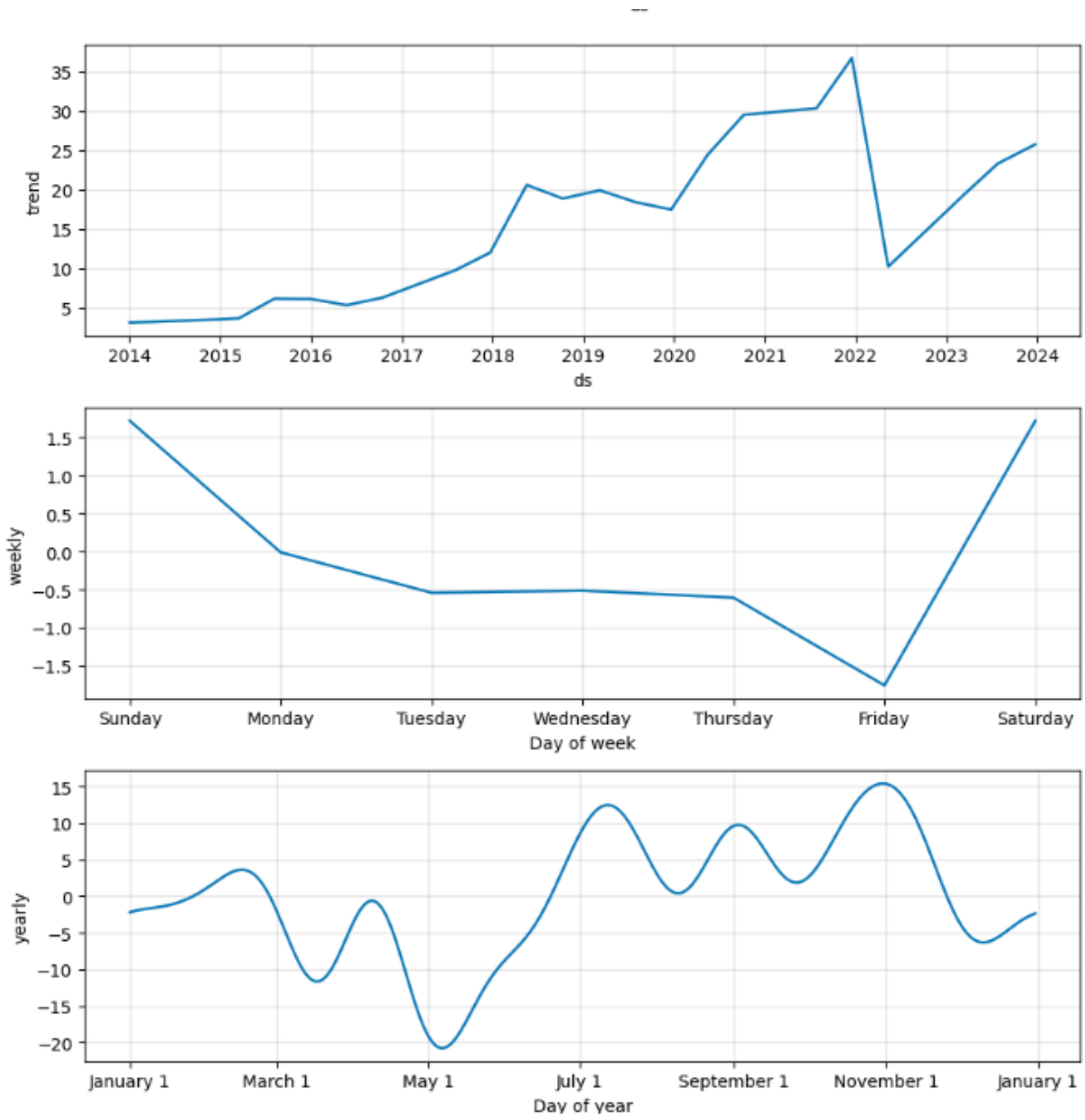

Рисунок 3.7 – Декомпозиція графіка прогнозу на 10 днів

Тренд (Trend):

– Видно стабільне зростання з 2014 по 2020 рік;

– Після 2020 року спостерігається зниження, за яким слід швидке відновлення.

Тижнева сезонність (Weekly):

– Найменша активність (негативні значення) спостерігається у вихідні дні, що може відображати знижену торгівельну активність в ці дні;

– Відносно вищі значення в середу та четвер, що може вказувати на підвищену торгівельну активність.

Річна сезонність (Yearly):

– Видно циклічні коливання, з піками влітку та восени;

– Найнижчі точки спостерігаються взимку, що може вказувати на сезонні коливання в інвестиційній активності або корпоративних фінансових циклах.

Проведемо оцінку точності налаштованої моделі, що зображено на рисунку 3.8.

```
 #Обрізання прогнозу до тестового періоду
forecast1 = forecast1[-10:]

# Розрахунок метрик
mse1 = mean_squared_error(test_p['y'], forecast1['yhat'])
rmse1 = np.sqrt(mse1)
mae1 = mean_absolute_error(test_p['y'], forecast1['yhat'])
mape1 = np.mean(np.abs((test_p['y'] - forecast1['yhat']) / test_p['y'])) * 100
#r2_score1 = round(r2_score(test_p['y'], forecast1['yhat']),2)
print(f'MSE: {mse1}')
print(f'RMSE: {rmse1}')
print(f'MAE: {mae1}')
print(f'MAPE: {mape1}%')
#print(f'r2_score: {r2_score1}')
```

```
MSE: 8343.503365145332
RMSE: 91.34277949102125
MAE: 84.08384661304396
MAPE: 17.19711349603453%
```

Рисунок 3.8 – Оцінку точності налаштованої моделі

– MSE (Mean Squared Error) = 8343.50. Це середньоквадратична помилка, яка вимірює середнє з квадратів відхилень прогнозованих значень від фактичних. Велике значення MSE вказує на потенційно великі помилки прогнозування.

– RMSE (Root Mean Squared Error) =91.34. Це квадратний корінь з MSE, що дає помилку в оригінальних одиницях даних. RMSE є більш інтуїтивно зрозумілою мірою помилок прогнозування, оскільки вона в тих же одиницях, що й дані. Високе значення RMSE вказує на великі відхилення прогнозованих значень від фактичних.

– MAE (Mean Absolute Error) = 84.08. Це середнє абсолютне відхилення прогнозованих значень від фактичних. MAE надає пряме уявлення про середню помилку в абсолютних термінах. Як і RMSE, велике значення MAE вказує на значні помилки в прогнозах.

– MAPE (Mean Absolute Percentage Error) = 17.2%. Це середнє абсолютне відсоткове відхилення прогнозованих значень від фактичних. MAPE є важливим показником, оскільки він вимірює помилку як відсоток від фактичних значень, що дозволяє легше оцінити масштаб помилки у контексті даних. Відсоток близько 18% може бути прийнятним у деяких контекстах, але зазвичай вказує на можливість покращення моделі.

Отже, налаштування не дали значного покращення. Застосуємо автоматизацію підбору параметрів моделі Prophet.

Автоматизація налаштувань параметрів для моделі Prophet може значно покращити процес моделювання, забезпечуючи більш точні та надійні прогнози. Для цього використаємо бібліотеку hyperopt, яка дозволяє автоматично підбирати гіперпараметри за допомогою різних методів оптимізації, таких як "дерева пошукових алгоритмів" (TPE) (рис. 3.9).

```python
# Функція для оптимізації
def objective(params):
    m = Prophet(
        changepoint_prior_scale=params['changepoint_prior_scale'],
        seasonality_prior_scale=params['seasonality_prior_scale'],
        holidays_prior_scale=params['holidays_prior_scale'],
        seasonality_mode=params['seasonality_mode']
    )
    m.fit(train_p)

    df_cv = cross_validation(m, initial='730 days', period='180 days', horizon = '365 days')
    df_p = performance_metrics(df_cv)
    rmse = df_p['rmse'].values[0]

    return {'loss': rmse, 'status': STATUS_OK}

# Простір параметрів
space = {
    'changepoint_prior_scale': hp.loguniform('changepoint_prior_scale', np.log(0.01), np.log(0.5)),
    'seasonality_prior_scale': hp.loguniform('seasonality_prior_scale', np.log(0.1), np.log(10)),
    'holidays_prior_scale': hp.loguniform('holidays_prior_scale', np.log(0.01), np.log(10)),
    'seasonality_mode': hp.choice('seasonality_mode', ['additive', 'multiplicative'])
}

# Запуск оптимізації
trials = Trials()
best = fmin(fn=objective, space=space, algo=tpe.suggest, max_evals=100, trials=trials)

print(best)
```

Рисунок 3.9 – Автоматизація оптимізації параметрів Prophet

Оцінимо точність налаштованої моделі на рисунку 3.10.

```
 #Обрізання прогнозу до тестового періоду
forecast2 = forecast2[-10:]

# Розрахунок метрик
mse2 = mean_squared_error(test_p['y'], forecast2['yhat'])
rmse2 = np.sqrt(mse1)
mae2 = mean_absolute_error(test_p['y'], forecast2['yhat'])
mape2 = np.mean(np.abs((test_p['y'] - forecast2['yhat']) / test_p['y'])) * 100
#r2_score1 = round(r2_score(test_p['y'], forecast1['yhat']),2)
print(f'MSE: {mse2}')
print(f'RMSE: {rmse2}')
print(f'MAE: {mae2}')
print(f'MAPE: {mape2}%')
#print(f'r2_score: {r2_score1}')
```

```
MSE: 6004.911648278265
RMSE: 91.34277949102125
MAE: 77.27146258124796
MAPE: 15.817967214952905%
```

Рисунок 3.10 – Оцінка точності оптимізованої моделі Prophet

Оскільки точність моделі не підвищилась, можна зробити висновок, що дану модель для прогнозування цього часового ряду використовувати недоцільно.

3.2 Прогнозування моделлю ARIMA

Для порівняння методів створимо та натренуємо модель ARIMA. У статистиці та економетриці, і зокрема в аналізі часових рядів, модель авторегресійної інтегрованої ковзної середньої, ARIMA є узагальненням моделі авторегресійної ковзної середньої (ARIMA). Обидві ці моделі адаптуються до даних часових рядів або для кращого розуміння даних, або для прогнозування. Моделі ARIMA застосовуються в деяких випадках, коли дані демонструють докази нестаціонарності. Коли сезонність відображається в часовому ряді, можна застосувати сезонну різницю, щоб усунути сезонний компонент.

Для початку визначимо параметри p, d, q для моделі ARIMA:

– p — порядок авторегресії: Визначає кількість лагових термінів, які включаються в модель. Для його вибору можна використовувати часткову автокореляційну функцію (PACF).

– d — ступінь диференціювання: Визначає, скільки разів потрібно диференціювати часовий ряд, щоб зробити його стаціонарним. Це можна визначити, використовуючи тест Дікі-Фуллера на стаціонарність або спостерігаючи за характером ряду.

– q — порядок рухомого середнього: Визначає кількість лагових термінів помилок прогнозу, які включаються в модель. Для його вибору можна використовувати автокореляційну функцію (ACF).

Зазвичай починають з моделі ARIMA(0,0,0) та послідовно збільшують p, d, і q, використовуючи критерії, такі як AIC (Akaike Information Criterion) чи BIC (Bayesian Information Criterion), для визначення найкращої моделі (рис.3.11).

```
from statsmodels.tsa.stattools import adfuller, acf, pacf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf


# Завантаження даних

series = train_p['y']

# Перевірка на стаціонарність
result = adfuller(series)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])

# Якщо ряд нестаціонарний, можна диференціювати
if result[1] > 0.05:
    series = series.diff().dropna()

# Аналіз ACF та PACF
plot_acf(series)
plot_pacf(series)
plt.show()

# Встановлення значень p, d, q
p = 1 # Виберіть на основі PACF
d = 1 # 1 якщо було виконано диференціювання, інакше 0
q = 1 # Виберіть на основі ACF
```

```
ADF Statistic: -1.348610
p-value: 0.606651
```

Рисунок 3.11 – Перевірка даних для налаштування параметрів моделі

При перевірці, отримали наступні дані ADF Statistic: -1.34, p-value: 0,6. Вони вказують на те, що ваш часовий ряд не є стаціонарним., що нульова гіпотеза про нестаціонарність часового ряду не відкидається. Це означає, що в часовому ряді ймовірно присутній тренд або сезонність, які потребують додаткової обробки перед використанням у прогнозних моделях. Графік автокореляційної функції (рис.3.12).

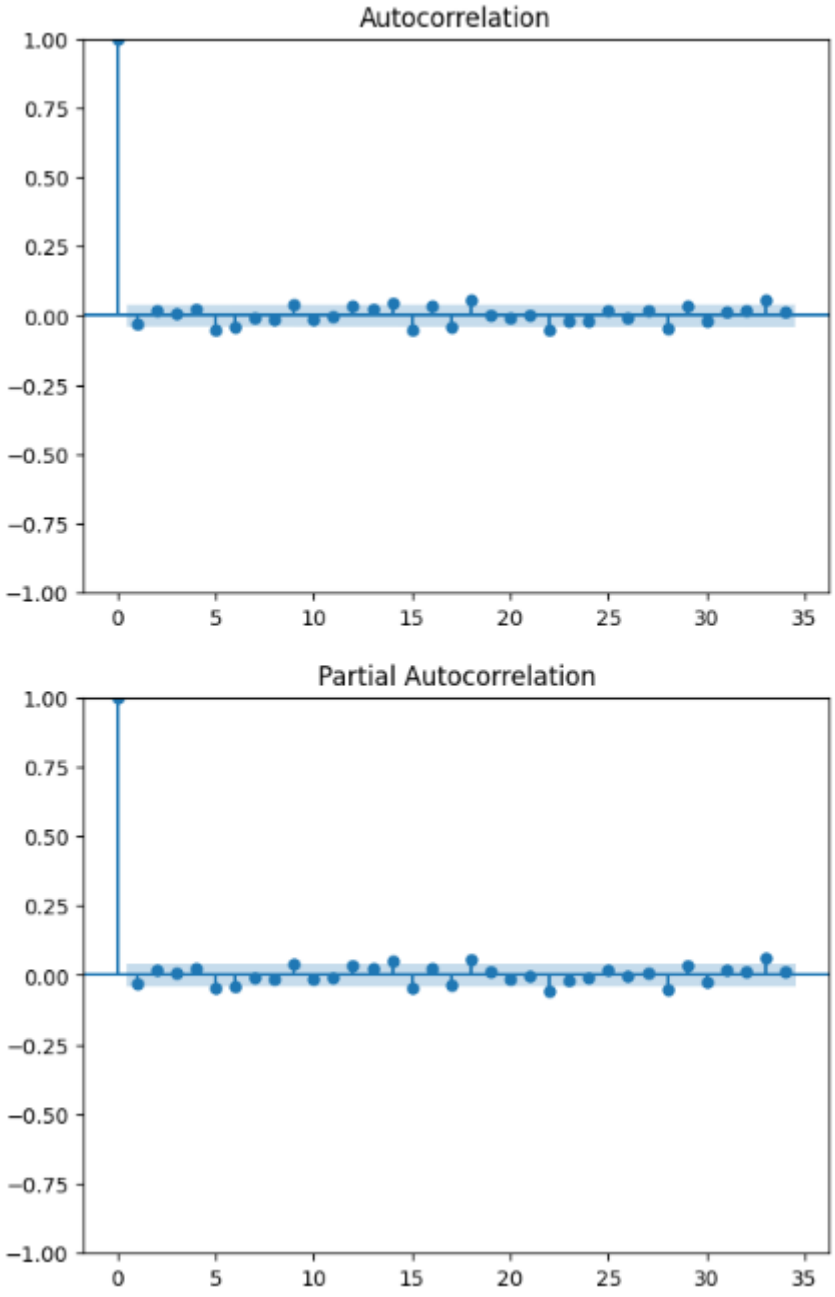

Рисунок 3.12 – Автокореляційна функція ряду

Графік автокореляції (ACF) показує, що відбувається затухання автокореляції з часом, що означає наявність можливого кореляційного зв'язку в часовому ряду на багатьох лагах. Однак, їх вплив зменшується зі збільшенням лагу.

Графік часткової автокореляції (PACF) показує стрибок на першому лагу, за яким слідує швидке затухання на наступних лагах. Це типова поведінка для процесу AR(1) – авторегресивної моделі першого порядку.

Вибір параметрів ARIMA:

Параметр p, значення 1 вказується як параметр p у моделі ARIMA, оскільки на PACF ми бачимо значущий пік на першому лагу;

Параметр d, для визначення параметра d (ступінь диференціювання), потрібно звернути увагу на результати тесту АДФ (Augmented Dickey-Fuller), які показують, що часовий ряд може бути нестаціонарним (p-значення 0.66651 > 0.05). Це може вказувати на необхідність виконання першого рівня диференціювання (d=1);

Параметр q, значення 0 для параметра q у моделі ARIMA, оскільки на ACF не бачимо значущих піків після лагу, що вказує на відсутність MA (Moving Average) компоненти.

Код реалізації APIMA з параметрами (1,1,0) на рисунку 3.13.

```python
from statsmodels.tsa.arima.model import ARIMA


#
# Вибір ряду даних
ts = train_p['y']

# Підгонка моделі ARIMA
model_ar = ARIMA(ts, order=(p, d, q))  # p - порядок авторегресії, d - ступінь інтеграції, q - порядок ру
results = model_ar.fit()

# Прогнозування
preds = results.get_forecast(steps=10)  # Прогноз на 10 кроків вперед
forecast_ar = preds.predicted_mean
conf_int = preds.conf_int()

# Візуалізація результатів
plt.figure(figsize=(10, 5))
plt.plot(ts[:: 200], label='Original')
plt.plot(forecast_ar, label='Forecast')
plt.fill_between(forecast_ar.index, conf_int.iloc[:, 0], conf_int.iloc[:, 1], color='pink', alpha=0.3)
plt.legend()
plt.show()
```

Рисунок 3.13 – Код APIMA

Результат роботи моделі APIMA на рисунку 3.14.

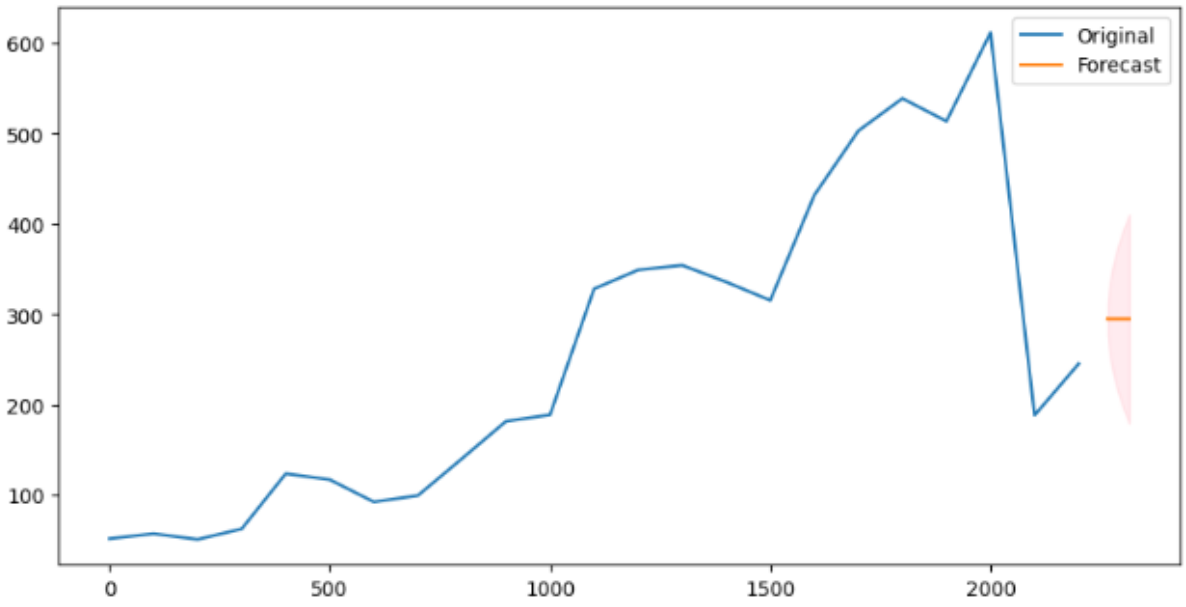

Рисунок 3.14 – Прогнозування APIMA

Як видно з рисунка 3.14, точність моделі надто низька. Потрібно змінити параметри на потрібні. Використаємо AutoARIMA для налаштування параметрів моделі (рис.3.15).

```
model = auto_arima(series, start_p=0, start_q=0, max_p=5, max_q=5, seasonal=False,
                   stepwise=True, suppress_warnings=True, error_action="ignore", trace=True)

# Найкращі параметри від auto_arima
print("Найкращі параметри:")
print(model.get_params())
```

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=17863.232, Time=0.06 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=17863.332, Time=0.15 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=17863.400, Time=0.25 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=17862.186, Time=0.04 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=17864.304, Time=0.80 sec

Best model:  ARIMA(0,1,0)(0,0,0)[0]
Total fit time: 1.310 seconds
Найкращі параметри:
{'maxiter': 50, 'method': 'lbfgs', 'order': (0, 1, 0), 'out_of_sampl
ept': False}
```

Рисунок 3.15 – Налаштування моделі за допомогою AutoARIMA

Застосуємо найкращі параметри до моделі ARIMA та побудуємо графік прогнозування на рисунку 3.16.

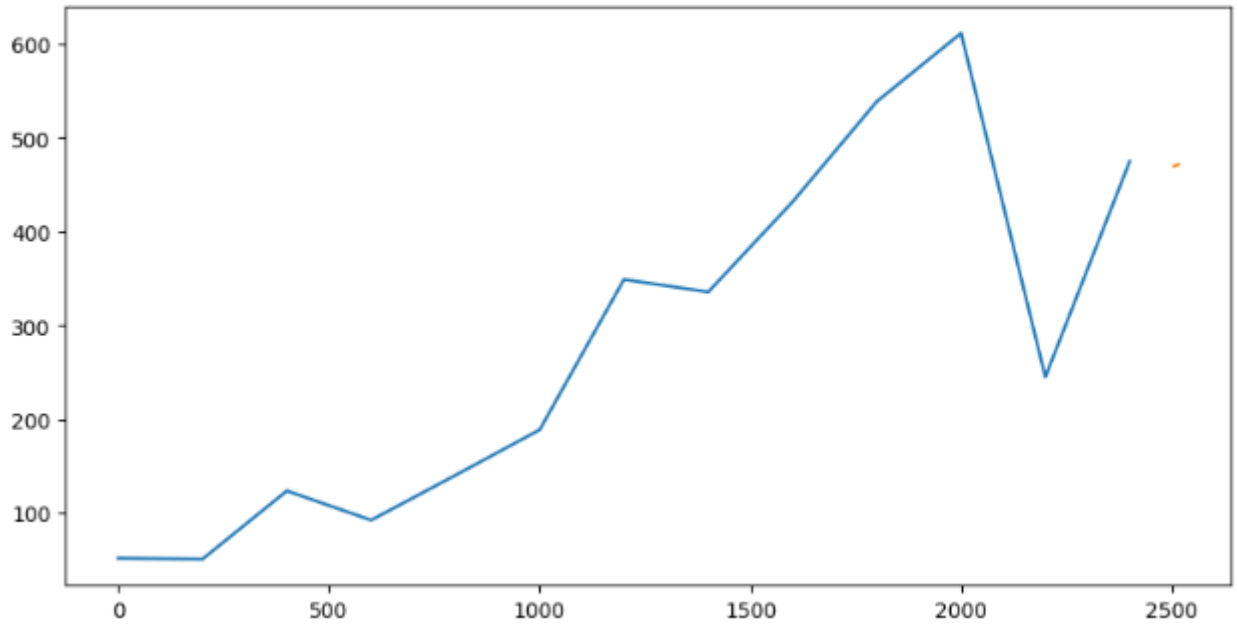

Рисунок 3.16 – Використання ARIMA з найкращими параметрами

Проаналізувавши графік, можна зробити висновок, що модель має велику похибку та не може використовуватись для прогнозування ціни на акціїї

3.3 Прогнозування моделлю XGBoost

Оскільки попередні моделі не дали позитивного результату використаємо модель градієнтного бустингу XGBoost.

Створимо тренувальний та тестовий набори даних за допомогою train_test_split (рис. 3.17).

```
from sklearn.model_selection import train_test_split
X = df.drop(['close','date'], axis=1)
y = df['close']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Рисунок 3.17 – Створення тренувального та тестового наборів даних

X – це ознаки, що характеризують цільову ознаку close. А y – це досліджувана функція. Розділимо дані на два набори тренувальний та тестовий з коефіцієнтом поділу 0,2 – це означає, що тренувальний набір даних містить 80% даних. Відповідно тренувальний містить 20% даних.

Переглянемо дані для моделі (рис. 3.18). Дані складаються зі 68 колонок. Всі ознаки мають числовий тип даних.

```
Data columns (total 68 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   open                                                    2516 non-null   float64
 1   high                                                    2516 non-null   float64
 2   low                                                     2516 non-null   float64
 3   volume                                                  2516 non-null   int64
 4   rsi_7                                                   2516 non-null   float64
 5   rsi_14                                                  2516 non-null   float64
 6   cci_7                                                   2516 non-null   float64
 7   cci_14                                                  2516 non-null   float64
 8   sma_50                                                  2516 non-null   float64
 9   ema_50                                                  2516 non-null   float64
 10  sma_100                                                 2516 non-null   float64
 11  ema_100                                                 2516 non-null   float64
 12  macd                                                    2516 non-null   float64
 13  bollinger                                               2516 non-null   float64
 14  TrueRange                                               2516 non-null   float64
 15  atr_7                                                   2516 non-null   float64
 16  atr_14                                                  2516 non-null   float64
 17  next_day_close                                          2516 non-null   float64
 18  index__sum_values                                       2516 non-null   float64
 19  index__abs_energy                                       2516 non-null   float64
 20  index__median                                           2516 non-null   float64
 21  index__mean                                             2516 non-null   float64
 22  index__root_mean_square                                 2516 non-null   float64
 23  index__maximum                                          2516 non-null   float64
 24  index__absolute_maximum                                 2516 non-null   float64
 25  index__minimum                                          2516 non-null   float64
 26  index__benford_correlation                              2515 non-null   float64
 27  index__quantile__q_0.1                                  2516 non-null   float64
 28  index__quantile__q_0.2                                  2516 non-null   float64
 29  index__quantile__q_0.3                                  2516 non-null   float64
 30  index__quantile__q_0.4                                  2516 non-null   float64
 31  index__quantile__q_0.6                                  2516 non-null   float64
 32  index__quantile__q_0.7                                  2516 non-null   float64
 33  index__quantile__q_0.8                                  2516 non-null   float64
```

Рисунок 3.18 – Інформація про дані

Для автоматизації підбору даних використаємо GridSearchCV. Використання GridSearchCV з бібліотеки scikit-learn для автоматизації підбору параметрів моделі є чудовим способом оптимізувати моделі машинного навчання. GridSearchCV систематично перевіряє комбінації параметрів, задані у вигляді сітки, і використовує перехресну валідацію для визначення найкращих значень параметрів для моделі.

На рисунку 3.19 зображена модель XGBoost з параметрами для GridSearchCV.

```
param_grid = {
    'n_estimators': [50, 100, 150 ],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01]
}
model = xgb.XGBRegressor(objective ='reg:squarederror')

grid_search = GridSearchCV(estimator=model, param_grid=param_grid,

# Запуск grid search
grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_
```

Рисунок 3.19 – модель XGBoost з параметрами для GridSearchCV

Застосуємо натреновану модель на тренувальних даних та оцінимо натреновану модель відповідними метриками (рис. 3.20).

```
y_pred_gb = best_model.predict(X_train)
mse = mean_squared_error(y_pred_gb, y_train)
mae = mean_absolute_error(y_pred_gb, y_train)
r2 = r2_score(y_pred_gb,  y_train)
# Виведення результатів
print(f"Mean Squared Error: {mse:.2f}")
print(f"Mean Absolute Error: {mae:.2f}")
print(f"R-squared Score: {r2:.2f}")
```

```
Mean Squared Error: 1395.26
Mean Absolute Error: 32.13
R-squared Score: 0.91
```

Рисунок 3.20 – Оцінка моделі XGBoost

Mean Squared Error (MSE) = 1395.26. Високе значення MSE може вказувати на великі відхилення прогнозів від фактичних даних, особливо при роботі з великими числами.

Mean Absolute Error (MAE) = 32.13. Значення 32.13 може бути хорошим або поганим в залежності від контексту даних і масштабу значень у датасеті.

R2 = 0.91. Значення 0.91 є дуже високим, що вказує на те, що модель дуже добре пояснює варіацію даних. Це вказує на високу якість моделі.

Застосуємо натреновану модель на тестових даних. На рисунку 3.21 зображено прогнозування на тестових даних та оцінка точності моделі.

```
y_pred = best_model.predict(X_test)
mse_t = mean_squared_error(y_pred, y_test)
mae_t = mean_absolute_error(y_pred, y_test)
r2_t = r2_score(y_pred,  y_test)
# Виведення результатів
print(f"Mean Squared Error: {mse_t:.2f}")
print(f"Mean Absolute Error: {mae_t:.2f}")
print(f"R-squared Score: {r2_t:.2f}")
```

```
Mean Squared Error: 1478.05
Mean Absolute Error: 33.40
R-squared Score: 0.91
```

Рисунка 3.21 – Оцінка точності на тестових даних

Оскільки точність моделі не має високої різниці між тренувальним та тестовим наборами даних, можна зробити висновок, що модель навчилась нормально та є адекватною.

Побудуємо графік важливості ознак на рисунку 3.22, 3.23.

```
fig =  plt.figure(figsize = (10,8))
axes = fig.add_subplot(111)
xgb.plot_importance(best_model,ax = axes,height = 0.5)
plt.show();
plt.close()
```

Рисунок 3.22 – Код графіка важливості ознак

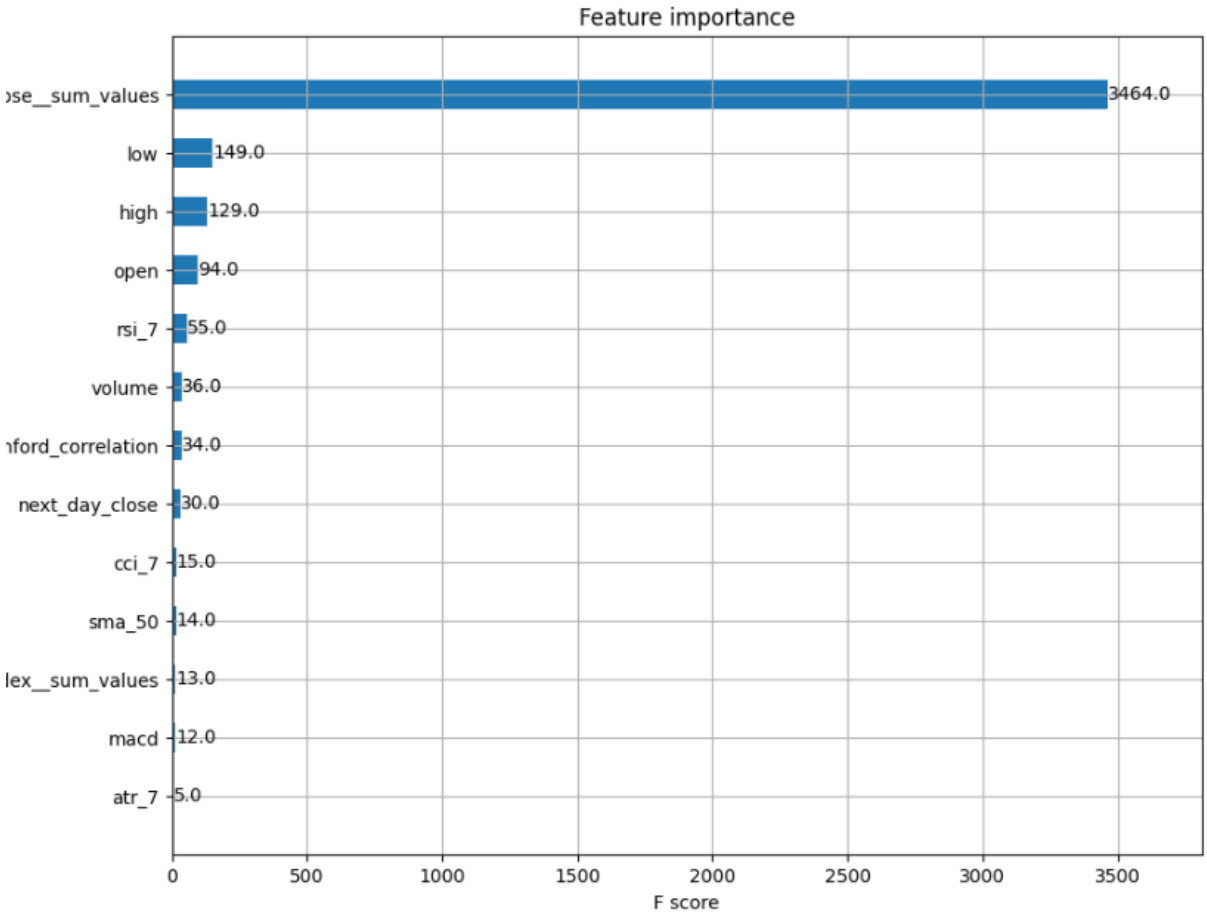

Рисунок 3.23 – Графік важливості ознак

Ознаки та їх F-бали:

– close_sum_values, значення 3464.0 — це найбільший внесок, що може свідчити про те, що сума значень закриття (можливо, за певний період) є дуже значущим індикатором для моделі;

– low, значення 149.0 — наступна за значенням ознака, що вказує на те, що мінімальні ціни в ряду також мають великий вплив на прогноз;

– high, значення 129.0 — високі ціни також важливі, але трохи менше, ніж низькі;

– open, значення 94.0 — ціни відкриття також відіграють роль, але вже менш важливу;

– rsi_7, значення 55.0 — індекс відносної сили за 7 днів вказує на важливість моментуму у моделі;

– volume, значення 36.0 — обсяги торгівлі мають деяку, але невелику значимість;

– close_benford_correlation, next_day_close, cci_7, sma_50, index_sum_values, macd, atr_7 – ці ознаки мають нижчі значення F-балів, що вказує на їх меншу важливість у моделі.

На рисунку 3.24 відобразимо графік прогнозованих та реальних даних.

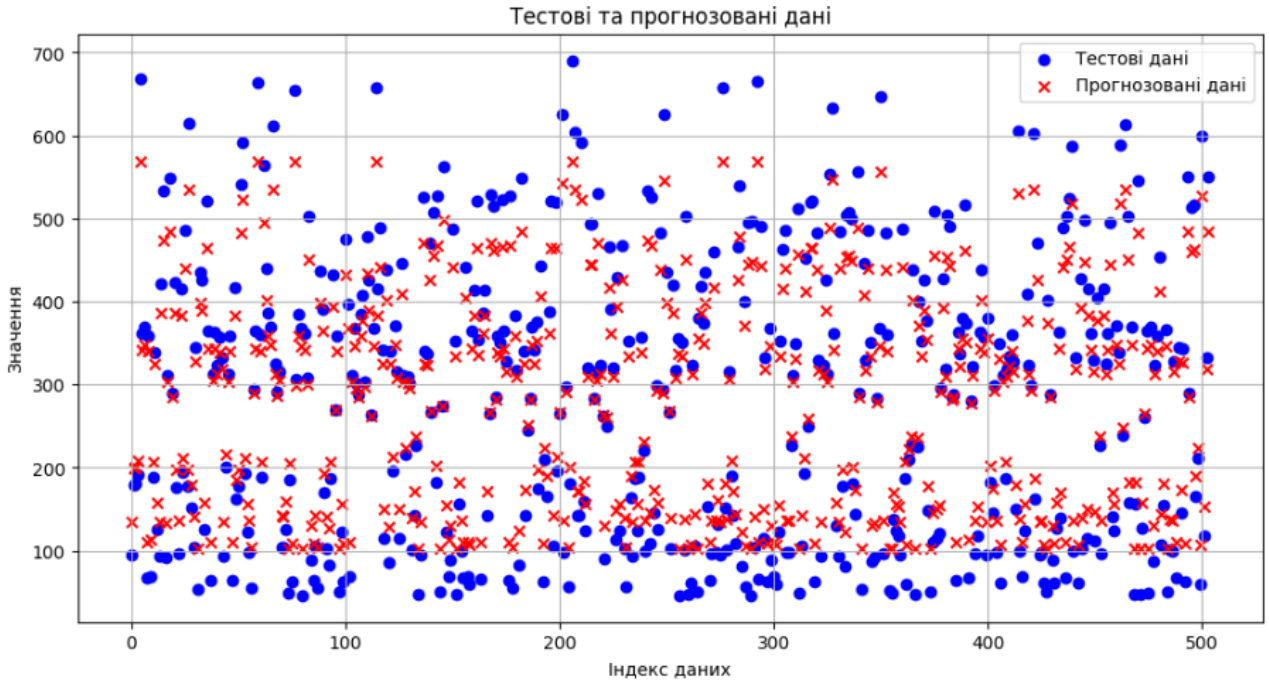

Рисунок 3.24 – Графік прогнозованих та реальних даних

На графіку ви відобразили реальні та прогнозовані дані, де реальні дані позначені синіми крапками, а прогнозовані — червоними хрестиками. Цей графік дозволяє оцінити точність моделі шляхом візуального порівняння між

реальними значеннями та їхніми прогнозами на одному графіку. Ось деякі спостереження та аналіз:

– Розподіл даних. Видно, що реальні дані (сині крапки) та прогнозовані дані (червоні хрестики) охоплюють подібні діапазони значень, проте є помітні відмінності в їх розміщенні по осі y;

– Згущення значень. Червоні хрестики, як правило, здаються менш розсіяними, ніж сині крапки, що може вказувати на меншу варіативність у прогнозах порівняно з реальними даними;

– Високі та низькі значення. Здається, що модель може недооцінювати піки та надооцінювати провали, особливо в крайніх значеннях.

На рисунку 3.25 зображена порівняльна таблиця точності результатів прогнозування ціни на акції Netflix.

	Model	Точність на тренувальних даних MSE	Точність на тестових даних MSE	Точність на тренувальних даних MAE	Точність на тестових даних MAE	Точність на тренувальних даних R2	Точність на тестових даних R2
0	Prophet	2698.335730	3646.411677	32.475711	58.689257	0.900000	-101.710000
1	ARIMA	74.150846	370.064045	5.042152	18.291012	0.997298	-9.423483
2	XGBoost	1395.255825	1478.054292	32.134727	33.397530	0.910000	0.910000

Рисунок 3.25 – Таблиця точності моделей прогнозування ціни акцій Netflix

3.4 Висновки

В третьому розділі використано три моделі прогнозування даних, а саме Prophet, ARIMA та XGBoost. Проведено оптимізацію та автоматизацію підбору параметрів відповідних моделей прогнозування ціни на акції.

При оцінюванні моделей прогнозування даних, модель Prophet та ARIMA показали низьку точність за метрикою R2_score.

На відміну від моделей Prophet та ARIMA, XGBoost має точність 0,91.

За результатами оцінки важливості ознак визначено, що найбільший вплив має ознака close_sum_values.

Отже модель XGBoost обрана, як найкраща модель для прогнозування ціни на акції компанії Netflix.

ВИСНОВКИ

В першому розділі проаналізовано загальну тематику стрімінгових сервісів. Проведено порівняльний аналіз популярних стрімінгів у світі, таких як Netflix, Amazon Prime Video, Disney+. За даними аналізу найбільш популярний стрімінг це Netflix.

Розглянуто процес ціноутворення акцій стрімінгових сервісів. Визначено основні характеристики за якими може утворюватися ціна акцій, а саме фінансові показники компанії, рівень підписників, конкуренція, новини та події, макроекономічні фактори, аналітичні оцінки та рекомендації інсайдерська інформація.

Проведено огляд основних підходів до аналізу та прогнозуванню ціни на акції Netflix в Kaggle Notebooks:

- «Time-Series-Analysis-Netflix-Stock-Price»,
- «Garch 1»,
- «XGboost: Predict "next_day_close"»,
- «Very Simple Prediction of Next Day Stock (99.76%)».

В даних підходах використано методи ARIMA, Garch, XGboost та лінійна регресія.

В другому розділі проведено розвідувальний аналіз ціни на акції компанії Netflix. Дані про ціну на акції перевірено на аномалії та викиди.

Проведено інженерію ознак. Використано TSFRESH для генерації нових ознак. Перевірено дані на аномалії та викиди. Визначено вплив ознак Open, High, Low на ознаку Close, що є цільовою.

Проаналізовано стандартні моделі машинного навчання для прогнозування ціни та обрано 3 моделі для виконання роботи, а саме Prophet, ARIMA та XGBoost.

В третьому розділі використано три моделі прогнозування даних, а саме Prophet, ARIMA та XGBoost. Проведено оптимізацію та автоматизацію підбору параметрів відповідних моделей прогнозування ціни на акції.

При оцінюванні моделей прогнозування даних, модель Prophet та ARIMA показали низьку точність за метрикою R2_score.

На відміну від моделей Prophet та ARIMA, XGBoost має точність 0,91.

За результатами оцінки важливості ознак визначено, що найбільший вплив має ознака close_sum_values.

Отже модель XGBoost обрано, як найкращу модель для прогнозування ціни на акції компанії Netflix.

Бакалаврська дипломна робота виконана в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Netflix. (н.д.). Офіційний сайт. URL: https://www.netflix.com

2. Amazon Prime Video. (н.д.). Офіційний сайт. URL: https://www.primevideo.com

3. Disney+. (н.д.). Офіційний сайт. URL: https://www.disneyplus.com

4. HBO Max. (н.д.). Офіційний сайт. URL: https://www.hbomax.com

5. Kaggle Notebook «Time-Series-Analysis-Netflix-Stock-Price». URL: https://www.kaggle.com/code/lydia70/time-series-analysis-netflix-stock-price

6. Kaggle Notebook «Garch 1». URL: https://www.kaggle.com/code/kkimiraik/garch-1

7. Kaggle Notebook «XGboost: Predict "next_day_close"». URL: https://www.kaggle.com/code/lko9911/xgboost-predict-next-day-close

8. Kaggle Notebook «Very Simple Prediction of Next Day Stock (99.76%)». URL: https://www.kaggle.com/code/queenlover/very-simple-prediction-of-next-day-stock-99-76

9. Yun, K.K., Yoon, S.W., & Won, D. (2021). Prediction of stock price direction using a hybrid GA-XGBoost algorithm with a three-stage feature engineering process. Expert Systems with Applications, 186, 115716. DOI : https://doi.org/10.1016/j.eswa.2021.115716

10. Lahmiri, S. (2016). A variational mode decomposition approach for analysis and forecasting of economic and financial time series. Expert Systems with Applications, 55, 268-273. DOI: https://doi.org/10.1016/j.eswa.2016.02.025

Siami-Namini, S. (2017). A Comparative Analysis of Forecasting Financial Time Series Using ARIMA, LSTM, and BiLSTM. International Journal of Economics and Financial Issues, 7(4), 603-607. DOI : https://doi.org/10.48550/arXiv.1911.09512

11. He, F.(Eric), Feng, Y., & Hao, J. (2022). Information disclosure source, investors' searching and stock price crash risk. Economics Letters, 210, 110202. DOI : https://doi.org/10.1016/j.econlet.2021.110202

12. Razavi Hajiagha, S.H., & Farahani, M.S. (Comparison Year Not Specified). Forecasting stock price using integrated artificial neural network and metaheuristic algorithms compared to time series models. DOI : https://doi.org/10.1007/s00500-021-05775-5

13. He, M., Zhang, Y., & Li, S. (2023). Geopolitical risk and stock market volatility: A global perspective. Finance Research Letters, 53, 103620. DOI: https://doi.org/10.1016/j.frl.2022.103620

14. Kim, H.Y., & Won, C.H. (2018). Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models. Expert Systems with Applications, 103, 25-37. DOI: https://doi.org/10.1016/j.eswa.2018.03.002

15. Kanniainen, J., Iosifidis, A., Tran, D.T., & Gabbouj, M. (2018). Temporal attention-augmented bilinear network for financial time-series data analysis. IEEE Transactions on Neural Networks and Learning Systems, 30(5), 1407-1418. DOI : https://doi.org/10.1109/TNNLS.2018.2869225

Taylor, S., Areal, N.M.P.C., & Taylor, S.J. (1994). The realized volatility of FTSE-100 futures prices. Journal of Financial and Quantitative Analysis, 29(01), 57-74. DOI : https://doi.org/10.1002/fut.10018

16. Degiannakis, S., & Filis, G. (2017). Forecasting oil price realized volatility using information channels from other asset classes. Journal of International Money and Finance, 76, 28-49. DOI : https://doi.org/10.1016/j.jimonfin.2017.05.006

17. Khashei, M., & Hajirahimi, Z. (2019). A comparative study of series ARIMA/MLP hybrid models for stock price forecasting. Communications in Statistics-Simulation and Computation, 48(9), 2625-2640. DOI: https://doi.org/10.1080/03610918.2018.1458138

18. Zheng, T., Weng, Q., & Liu, R. (2022). Forecasting Tesla's Stock Price Using the ARIMA Model. Proceedings of Business and Economic Studies, 5(5), 38-45. DOI: https://doi.org/10.26689/pbes.v5i5.4331

19. Wang, Y., Wei, Y., Zhang, J., & Bai, L. (2023). Time-and frequency-domain evidence based on TVP-VAR model. Renewable Energy. DOI: https://doi.org/10.1016/j.renene.2022.11.098

20. Esfahanipour, A., Khodaee, P., & Taheri, H.M. (2016). Forecasting turning points in stock price by applying a novel hybrid CNN-LSTM-ResNet model fed by 2D segmented images. Neural Computing and Applications, 27, 651-658. DOI: https://doi.org/10.1016/j.engappai.2022.105464

21. Iosifidis, A., Ntakaris, M., Magris, M., Kanniainen, J., & Gabbouj, M. (2018). Forecasting stock prices from the limit order book using convolutional neural networks. Journal of Forecasting, 37(8), 852-866. DOI: https://doi.org/10.1109/CBI.2017.23

22. Dutta, A., Sen, J., Mehtab, S., & Mondal, S. (2021). Precise Stock Price Prediction for Robust and Optimized Portfolio Design Using an LSTM Model. URL : https://www.researchgate.net/profile/Jaydip-Sen-3/publication/355339185_Precise_Stock_Price_Prediction_for_Robust_and_Optimized_Portfolio_Design_Using_an_LSTM_Model_-_Presentation/links/61726ae7435dab3b7594bcaa/Precise-Stock-Price-Prediction-for-Robust-and-Optimized-Portfolio-Design-Using-an-LSTM-Model-Presentation.pdf

23. Kaggle dataset «Netflix Stock Price With Indicators» URL: https://www.kaggle.com/datasets/aspillai/netflix-stock-price-with-indicators

24. Mokin V.B. Notebook «crypto–btc advanced analysis forecasting» URL: https://www.kaggle.com/code/vbmokin/crypto-btc-advanced-analysis-forecasting

25. В. Б. Мокін, А. В. Лосенко, і М. В. Дратований, «Інтелектуальна технологія аналізу та передбачення цін на вживані автомобілі», *Вісник ВПІ*, вип. 6, с. 62–72, Груд. 2019.

Додаток А

(обов’язковий)

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
“____”_____ 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на бакалаврську дипломну роботу
АНАЛІЗ ТА ПРОГНОЗУВАННЯ ЦІНИ НА АКЦІЇ NETFLIX
08-34.БДР.003.00.000 ТЗ

Керівник: к.т.н., доц.
_____ Ілона ВАРЧУК
«__» _____ 2024 р.
Розробив студент гр. СА-20б
_____ Вадим РУДИК
«__» _____ 2024 р.

Вінниця 2024

1. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2024р., та індивідуальне завдання на БДР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2024р.

2. Джерела розробки: (зі списку літератури)

1) Mokin V.B. Notebook «crypto–btc advanced analysis forecasting» URL: https://www.kaggle.com/code/vbmokin/crypto-btc-advanced-analysis-forecasting

2) В. Б. Мокін, А. В. Лосенко, і М. В. Дратований, «Інтелектуальна технологія аналізу та передбачення цін на вживані автомобілі», *Вісник ВПІ*, вип. 6, с. 62–72, Груд. 2019.

3) Kaggle Notebook «XGboost: Predict "next_day_close"». URL: https://www.kaggle.com/code/lko9911/xgboost-predict-next-day-close

3. Мета і призначення роботи.

Метою дослідження є аналіз та прогнозування ціни акцій Netflix

4. Вихідні дані для проведення робіт:

Kaggle Dataset «Netflix Stock Price With Indicators» https://www.kaggle.com/datasets/aspillai/netflix-stock-price-with-indicators.

5. Методи дослідження:

Моделі машинного навчання, мова програмування Python.

6. Етапи роботи і терміни їх виконання:

a) Аналіз предметної області _____ – _____
b) Порівняльний аналіз проблематики _____ – _____
c) Порівняльний аналіз методів прогнозування ціни акцій _____ – _____
d) Розвідувальний аналіз даних та інженерія ознак _____ – _____
e) Прогнозування ціни на акції Netflix _____ – _____
f) Оформлення матеріалів до захисту БДР _____ – _____

7. Очікувані результати та порядок реалізації

Отримання оптимальної моделі машинного навчання для прогнозування ціни на акції Netflix.

8. Вимоги до розробленої документації

Текстова та ілюстративна частини роботи оформлені у відповідності до вимог «Методичних вказівок до виконання бакалаврських кваліфікаційних робіт для студентів спеціальностей: 124 «Системний аналіз», 126 «Інформаційні системи та технології» (освітня програма «Прикладні інформаційні технології»)».

9. Порядок приймання роботи

Публічний захист «___» _____ 2024 р.
Початок розробки «___» _____ 2024 р.
Граничні терміни виконання БДР «___» _____ 2024 р.

Розробив студент гр. СА-20б _____ Вадим РУДИК

Додаток Б

(обов'язковий)

Протокол перевірки бакалаврської дипломної роботи на наявність текстових запозичень

Назва роботи: «Аналіз та прогнозування ціни на акції Netflix»
Тип роботи: бакалаврська дипломна робота
Підрозділ: кафедра САІТ

Показники звіту подібності Unicheck

Оригінальність 97,1 Схожість 2,9

Аналіз звіту подібності (відмітити потрібне)

● Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
○ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
○ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ (підпис) Сергій ЖУКОВ

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

Автор роботи _____ (підпис) Вадим РУДИК

Керівник роботи _____ (підпис) Ілона ВАРЧУК

Додаток В

(довідниковий)

Фрагмент лістингу програми

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory
from sklearn.metrics import mean_squared_error, mean_absolute_error
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session

# Import libraries
import random
import os
import numpy as np
import pandas as pd
import requests
import pandas_datareader as web

# Date
import datetime as dt
from datetime import date, timedelta, datetime

# EDA
import matplotlib.pyplot as plt
from matplotlib.pylab import rcParams
import plotly.express as px
import plotly.graph_objects as go
from plotly.offline import init_notebook_mode
init_notebook_mode(connected=True)

# FE
from tsfresh import extract_features, select_features, extract_relevant_features
from tsfresh.utilities.dataframe_functions import impute
from sklearn.inspection import permutation_importance
import eli5
from eli5.sklearn import PermutationImportance
import shap

# Time Series - EDA and Modelling
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima_model import ARIMA

# Metrics
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error
from sklearn.metrics import mean_absolute_error
```

```
# Modeling and preprocessing
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR, LinearSVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import BaggingRegressor, AdaBoostRegressor
from sklearn.neural_network import MLPRegressor
from prophet import Prophet
import xgboost as xgb
from xgboost import XGBRegressor
import lightgbm as lgb
from lightgbm import LGBMRegressor

import warnings
warnings.filterwarnings("ignore")

df = pd.read_csv('/kaggle/input/netflix-stock-price-with-indicators/nflx_2014_2023.csv')
df['date'] = pd.to_datetime(df['date'])
df.set_index('date', inplace=True)
df.head()

df.tail()

df.info()

df.describe()

df['close'].plot(grid=True, figsize=(12,8))

def c_chart(data,label):
    # Thanks to https://www.kaggle.com/code/fangya/cryptocurrency-data-visualization-arima
    candlestick = go.Figure(data = [go.Candlestick(x=data.index,
                                        open = data['open'],
                                        high = data['high'],
                                        low = data['low'],
                                        close = data['close'])])
    candlestick.update_xaxes(title_text = 'Time',
                         rangeslider_visible = True)

    candlestick.update_layout(
    title = {
          'text': '{:} Candelstick Chart'.format(label),
          "y":0.8,
          "x":0.5,
          'xanchor': 'center',
          'yanchor': 'top'})

    candlestick.update_yaxes(title_text = 'Price in USD', ticksuffix = '$')
    return candlestick

%matplotlib inline
netflix_price=c_chart(df, label="Netflix Price")
netflix_price.show()

result = seasonal_decompose(df['close'], model='additive', period=252)  # 252 торгові дні у році

fig = result.plot()
fig.set_size_inches((12, 10))
fig.tight_layout()
plt.show()
```

```
df1=df

df1['close'].plot()

def get_tsfresh_features(data):
    # Get statistic features using library TSFRESH
    # Thanks to https://www.kaggle.com/code/vbmokin/btc-growth-forecasting-with-advanced-fe-for-ohlc

    data = data.reset_index(drop=False).reset_index(drop=False)

    # Extract features
    extracted_features = extract_features(data, column_id="date", column_sort="date")

    # Drop features with NaN
    extracted_features_clean = extracted_features.dropna(axis=1, how='all').reset_index(drop=True)

    # Drop features with constants
    cols_std_zero  = []
    for col in extracted_features_clean.columns:
        if extracted_features_clean[col].std()==0:
            cols_std_zero.append(col)
    extracted_features_clean = extracted_features_clean.drop(columns = cols_std_zero)

    extracted_features_clean['date'] = data['date']   # For the merging

    return extracted_features_clean

extracted_features_clean = get_tsfresh_features(df['close'])
extracted_features_clean

df = pd.merge(df, extracted_features_clean, how='left', on='date')
df

def check_stationarity(series):
    # Thanks to https://machinelearningmastery.com/time-series-data-stationary-python/

    result = adfuller(series.values)

    print('ADF Statistic: %f' % result[0])
    print('p-value: %f' % result[1])
    print('Critical Values:')
    for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))

    if (result[1] <= 0.05) & (result[4]['5%'] > result[0]):
        print("\u001b[32mStationary\u001b[0m")
    else:
        print("\x1b[31mNon-stationary\x1b[0m")

check_stationarity(df1['close'])

check_stationarity(df['close'].diff().dropna())

df1 = df[['date','close']]

df1.rename(columns={'date': 'ds', 'close': 'y'}, inplace=True)

train_p = df1.iloc[:-10]  # Всі дані крім останніх 10 днів
test_p = df1.iloc[-10:]   # Останні 10 днів

model = Prophet()

model.fit(train_p)
```

```
futue = model.make_future_dataframe(periods=10)  # For example, predict the next 365 days
forecast = model.predict(future)

# Plot the forecast
fig = model.plot(forecast)
fig2 = model.plot_components(forecast)

#Обрізання прогнозу до тестового періоду
forecast = forecast[-10:]

# Розрахунок метрик
mse = mean_squared_error(test_p['y'], forecast['yhat'])
rmse = np.sqrt(mse)
mae = mean_absolute_error(test_p['y'], forecast['yhat'])
mape = np.mean(np.abs((test_p['y'] - forecast['yhat']) / test_p['y'])) * 100
r2_score_p = r2_score(test_p['y'], forecast['yhat'])
print(f'MSE: {mse}')
print(f'RMSE: {rmse}')
print(f'MAE: {mae}')
print(f'MAPE: {mape}%')
print(f'r2_score_p: {r2_score_p}')

model1 = Prophet(
    daily_seasonality=True,
    weekly_seasonality=True,
    yearly_seasonality=True,
    changepoint_range=1,
    changepoint_prior_scale = 0.01,
    seasonality_mode = 'multiplicative'
)

# Додавання зовнішніх регресорів, якщо вони є
# model.add_regressor('additional_regressor')
model1.add_country_holidays(country_name='US')

# Навчання моделі
model1.fit(train_p)

# Створення майбутніх дат для прогнозування
future1 = model1.make_future_dataframe(periods=10, freq='D')

# Прогнозування
forecast1 = model1.predict(future1)

# Виведення результатів
print(forecast1[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail())

# Plot the forecast
fig = model.plot(forecast1)
fig2 = model.plot_components(forecast1)

# %% [code]

#Обрізання прогнозу до тестового періоду
forecast1 = forecast1[-10:]
# Розрахунок метрик
mse1 = mean_squared_error(test_p['y'], forecast1['yhat'])
rmse1 = np.sqrt(mse1)
mae1 = mean_absolute_error(test_p['y'], forecast1['yhat'])
mape1 = np.mean(np.abs((test_p['y'] - forecast1['yhat']) / test_p['y'])) * 100
r2_score1 = round(r2_score(test_p['y'], forecast1['yhat']),2)
print(f'MSE: {mse1}')
print(f'RMSE: {rmse1}')
```

```
print(f'MAE: {mae1}')
print(f'MAPE: {mape1}%')
print(f'r2_score: {r2_score1}')

from prophet.diagnostics import cross_validation, performance_metrics
from hyperopt import hp, fmin, tpe, Trials, STATUS_OK

# Функція для оптимізації
def objective(params):
    m = Prophet(
        changepoint_prior_scale=params['changepoint_prior_scale'],
        seasonality_prior_scale=params['seasonality_prior_scale'],
        holidays_prior_scale=params['holidays_prior_scale'],
        seasonality_mode=params['seasonality_mode']
    )
    m.fit(train_p)

    df_cv = cross_validation(m, initial='730 days', period='180 days', horizon = '365 days')
    df_p = performance_metrics(df_cv)
    rmse = df_p['rmse'].values[0]

    return {'loss': rmse, 'status': STATUS_OK}

# Простір параметрів
space = {
    'changepoint_prior_scale': hp.loguniform('changepoint_prior_scale', np.log(0.01), np.log(0.5)),
    'seasonality_prior_scale': hp.loguniform('seasonality_prior_scale', np.log(0.1), np.log(10)),
    'holidays_prior_scale': hp.loguniform('holidays_prior_scale', np.log(0.01), np.log(10)),
    'seasonality_mode': hp.choice('seasonality_mode', ['additive', 'multiplicative'])
}

# Запуск оптимізації
trials = Trials()
best = fmin(fn=objective, space=space, algo=tpe.suggest, max_evals=10, trials=trials)

print(best)

best

# Використання найкращих гіперпараметрів від оптимізації
best_params = {
    'changepoint_prior_scale': np.exp(best['changepoint_prior_scale']),
    'seasonality_prior_scale': np.exp(best['seasonality_prior_scale']),
    'holidays_prior_scale': np.exp(best['holidays_prior_scale']),
    'seasonality_mode': ['additive', 'multiplicative'][best['seasonality_mode']]
}

# Ініціалізація та навчання моделі Prophet з оптимальними параметрами
model2 = Prophet(
    changepoint_prior_scale=best_params['changepoint_prior_scale'],
    seasonality_prior_scale=best_params['seasonality_prior_scale'],
    holidays_prior_scale=best_params['holidays_prior_scale'],
    seasonality_mode=best_params['seasonality_mode']
)
model2.fit(train_p)

# Створення датасету для майбутніх прогнозів
future2 = model2.make_future_dataframe(periods=10)  # Прогноз на наступний рік

# Прогнозування
forecast2 = model2.predict(future2)

# Візуалізація прогнозу
```

```
fig1 = model.plot(forecast2)
plt.title("Прогнозовані ціни акцій на наступний рік")

# Візуалізація компонентів прогнозу
fig2 = model.plot_components(forecast2)

mseT = mean_squared_error(train_p['y'], forecast2['yhat'][:-10])
rmseT = np.sqrt(mse1)
maeT = mean_absolute_error(train_p['y'], forecast2['yhat'][:-10])
mapeT = np.mean(np.abs((train_p['y'] - forecast2['yhat'][:-10]) / train_p['y'])) * 100
r2_score2_tren = round(r2_score(train_p['y'], forecast2['yhat'][:-10]),2)
print(f'MSE: {mseT}')
print(f'RMSE: {rmseT}')
print(f'MAE: {maeT}')
print(f'MAPE: {mapeT}%')
print(f'r2_score: {r2_score2_tren}')

#Обрізання прогнозу до тестового періоду
forecast3 = forecast2[-10:]

# Розрахунок метрик
mse2 = mean_squared_error(test_p['y'], forecast3['yhat'])
rmse2 = np.sqrt(mse1)
mae2 = mean_absolute_error(test_p['y'], forecast3['yhat'])
mape2 = np.mean(np.abs((test_p['y'] - forecast3['yhat']) / test_p['y'])) * 100
r2_score2_test = round(r2_score(test_p['y'], forecast3['yhat']),2)
print(f'MSE: {mse2}')
print(f'RMSE: {rmse2}')
print(f'MAE: {mae2}')
print(f'MAPE: {mape2}%')
print(f'r2_score: {r2_score2_test}')

train_p
test_p


from statsmodels.tsa.stattools import adfuller, acf, pacf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Завантаження даних

series = train_p['y']

# Перевірка на стаціонарність
result = adfuller(series)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])

# Якщо ряд нестаціонарний, можна диференціювати
if result[1] > 0.05:
    series = series.diff().dropna()

# Аналіз ACF та PACF
plot_acf(series)
plot_pacf(series)
plt.show()

# Встановлення значень p, d, q
p = 1 # Виберіть на основі PACF
d = 1 # 1 якщо було виконано диференціювання, інакше 0
q = 1 # Виберіть на основі ACF
```

```
# Встановлення значень p, d, q
p = 1 # Виберіть на основі PACF
d = 1 # 1 якщо було виконано диференціювання, інакше 0
q = 0# Виберіть на основі ACF


from statsmodels.tsa.arima.model import ARIMA


#
# Вибір ряду даних
ts = train_p['y']

# Підгонка моделі ARIMA
model_ar = ARIMA(ts, order=(p, d, q))  # p - порядок авторегресії, d - ступінь інтеграції, q - порядок рухомого середнього
results = model_ar.fit()

# Прогнозування
preds = results.get_forecast(steps=10)  # Прогноз на 10 кроків вперед
forecast_ar = preds.predicted_mean
conf_int = preds.conf_int()

# Візуалізація результатів
plt.figure(figsize=(10, 5))
plt.plot(ts[:: 200], label='Original')
plt.plot(forecast_ar, label='Forecast')
plt.fill_between(forecast_ar.index, conf_int.iloc[:, 0], conf_int.iloc[:, 1], color='pink', alpha=0.3)
plt.legend()
plt.show()

!pip install pmdarima
from pmdarima import auto_arima
from pmdarima import ARIMA

model_ara = auto_arima(ts, start_p=0, start_q=0, max_p=5, max_q=5, d=None,
                       seasonal=False, trace=True, error_action='ignore', suppress_warnings=True,
                       stepwise=True, information_criterion='aic')

print(model_ara.summary())

series =train_p['y']

model = auto_arima(series, start_p=0, start_q=0, max_p=5, max_q=5, seasonal=False,
                   stepwise=True, suppress_warnings=True, error_action="ignore", trace=True)

# Найкращі параметри від auto_arima
print("Найкращі параметри:")
print(model.get_params())

# Побудова моделі ARIMA з найкращими параметрами
final_model = ARIMA(order=model.order, seasonal_order=model.seasonal_order)
final_model.fit(series)

# Прогнозування
n_periods = 10  # Кількість періодів для прогнозування
forecast, conf_int = final_model.predict(n_periods=n_periods, return_conf_int=True)

# Створення DataFrame для збереження прогнозів
future_dates = pd.date_range(start=train_p['ds'].iloc[-1], periods=n_periods + 1, freq='D')[1:]
forecast_df = pd.DataFrame(data=forecast, index=future_dates, columns=['Forecast'])

# Візуалізація
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))
plt.plot(ts[:: 200], label='Original')
plt.plot(forecast, label='Forecast')
plt.fill_between(forecast.index, conf_int[:, 0], conf_int[:, 1], color='pink', alpha=0.3)
plt.legend()
plt.show()

import pandas as pd
import matplotlib.pyplot as plt
from pmdarima import auto_arima
from statsmodels.tsa.arima.model import ARIMA
series = train_p['y']  # Дані для тренування
# Використання auto_arima для визначення найкращих параметрів
model = auto_arima(series, start_p=0, start_q=0, max_p=5, max_q=5, seasonal=False,
                   stepwise=True, suppress_warnings=True, error_action="ignore", trace=True)
# Виведення найкращих параметрів
print("Найкращі параметри:")
print(model.get_params())

# Побудова моделі ARIMA з найкращими параметрами
final_model = ARIMA(series, order=model.order)
fitted_model = final_model.fit()

# Прогнозування
n_periods = 10
forecast_object = fitted_model.get_forecast(steps=n_periods)
forecast = forecast_object.predicted_mean
conf_int = forecast_object.conf_int(alpha=0.05)

# Створення DataFrame для збереження прогнозів
future_dates = pd.date_range(start=series.index[-1], periods=n_periods + 1, freq='D')[1:]
forecast_df = pd.DataFrame(forecast, index=future_dates, columns=['Forecast'])

# Візуалізація
plt.figure(figsize=(10, 5))
plt.plot(series, label='Original')
plt.plot(forecast_df['Forecast'], label='Forecast')
plt.fill_between(future_dates, conf_int.iloc[:, 0], conf_int.iloc[:, 1], color='pink', alpha=0.3)
plt.legend()
plt.show()

from sklearn.metrics import mean_squared_error, mean_absolute_error

# Прогнозування на тренувальних даних для оцінки
train_forecast = fitted_model.predict(start=series.index[0], end=series.index[-1])

# Оцінка на тренувальних даних
train_mse = mean_squared_error(series, train_forecast)
train_mae = mean_absolute_error(series, train_forecast)
r2_ar = r2_score(series, train_forecast)
# Прогнозування на тестових даних
test_series = test_p['y']  # Припустимо, тестові дані
test_forecast = fitted_model.predict(start=test_series.index[0], end=test_series.index[-1])

# Оцінка на тестових даних
test_mse = mean_squared_error(test_series, test_forecast)
test_mae = mean_absolute_error(test_series, test_forecast)
r2_ar_t = r2_score(test_series, test_forecast)
print(f'Training MSE: {train_mse}, Training MAE: {train_mae}, Training r2: {r2_ar}')
print(f'Test MSE: {test_mse}, Test MAE: {test_mae}, Test r2: {r2_ar_t}')
```

```
from sklearn.model_selection import train_test_split
X = df.drop(['close','date'], axis=1)
y = df['close']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X.info()
from sklearn.model_selection import GridSearchCV
import xgboost as xgb
param_grid = {
   'n_estimators': [50, 100, 150 ],
   'max_depth': [3, 5, 7],
   'learning_rate': [0.01]
}
model = xgb.XGBRegressor(objective ='reg:squarederror')
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=3, scoring='neg_mean_squared_error',
verbose=1)
# Запуск grid search
grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_
# Зробіть прогнози
y_pred = best_model.predict(X_test)
y_pred_gb = best_model.predict(X_train)
mse = mean_squared_error(y_pred_gb, y_train)
mae = mean_absolute_error(y_pred_gb, y_train)
r2 = r2_score(y_pred_gb,  y_train)
# Виведення результатів
print(f"Mean Squared Error: {mse:.2f}")
print(f"Mean Absolute Error: {mae:.2f}")
print(f"R-squared Score: {r2:.2f}")
mse_t = mean_squared_error(y_pred, y_test)
mae_t = mean_absolute_error(y_pred, y_test)
r2_t = r2_score(y_pred,  y_test)
# Виведення результатів
print(f"Mean Squared Error: {mse_t:.2f}")
print(f"Mean Absolute Error: {mae_t:.2f}")
print(f"R-squared Score: {r2_t:.2f}")
fig =  plt.figure(figsize = (10,8))
axes = fig.add_subplot(111)
xgb.plot_importance(best_model,ax = axes,height = 0.5)
plt.show();
plt.close()
plt.figure(figsize=(12, 6))
plt.scatter(range(len(y_test)),y_test, label='Тестові дані', color='blue', marker='o')
plt.scatter(range(len(y_test)),y_pred, label='Прогнозовані дані', color='red',  marker='x')
plt.title('Тестові та прогнозовані дані')
plt.xlabel('Індекс даних')
plt.ylabel('Значення')
plt.legend()
plt.grid(True)
plt.show()
results = pd.DataFrame({
   'Model': ['Prophet', 'ARIMA', 'XGBoost'],
   'Точність на тренувальних даних MSE': [mseT , train_mse, mse],
   'Точність на тестових даних MSE': [mse2, test_mse, mse_t],
   'Точність на тренувальних даних MAE': [maeT , train_mae, mae],
   'Точність на тестових даних MAE': [mae2, test_mae, mae_t],
   'Точність на тренувальних даних R2': [r2_score2_tren , r2_ar, round(r2,2)],
   'Точність на тестових даних R2': [r2_score2_test, r2_ar_t, round(r2_t,2)]
})

# Виводимо результати
results
```

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

АНАЛІЗ ТА ПРОГНОЗУВАННЯ ЦІНИ НА АКЦІЇ NETFLIX

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2024 р.

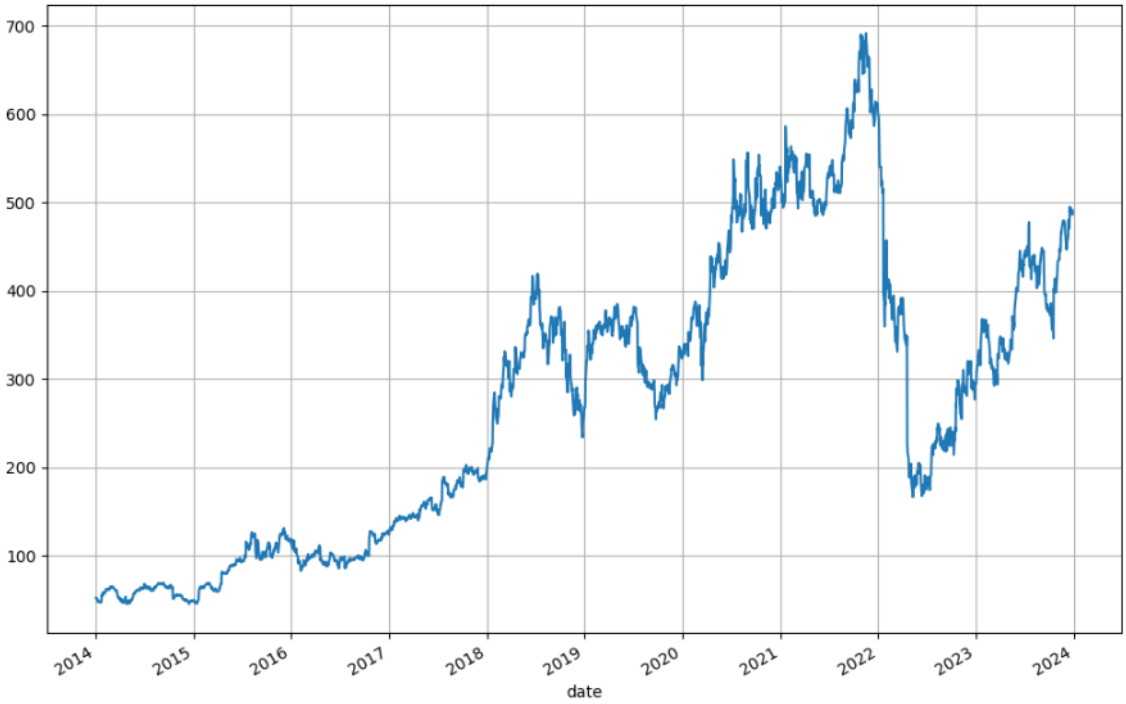

Рисунок Г.1 – Зміна ціни на акції з 2014 по 2023 рік

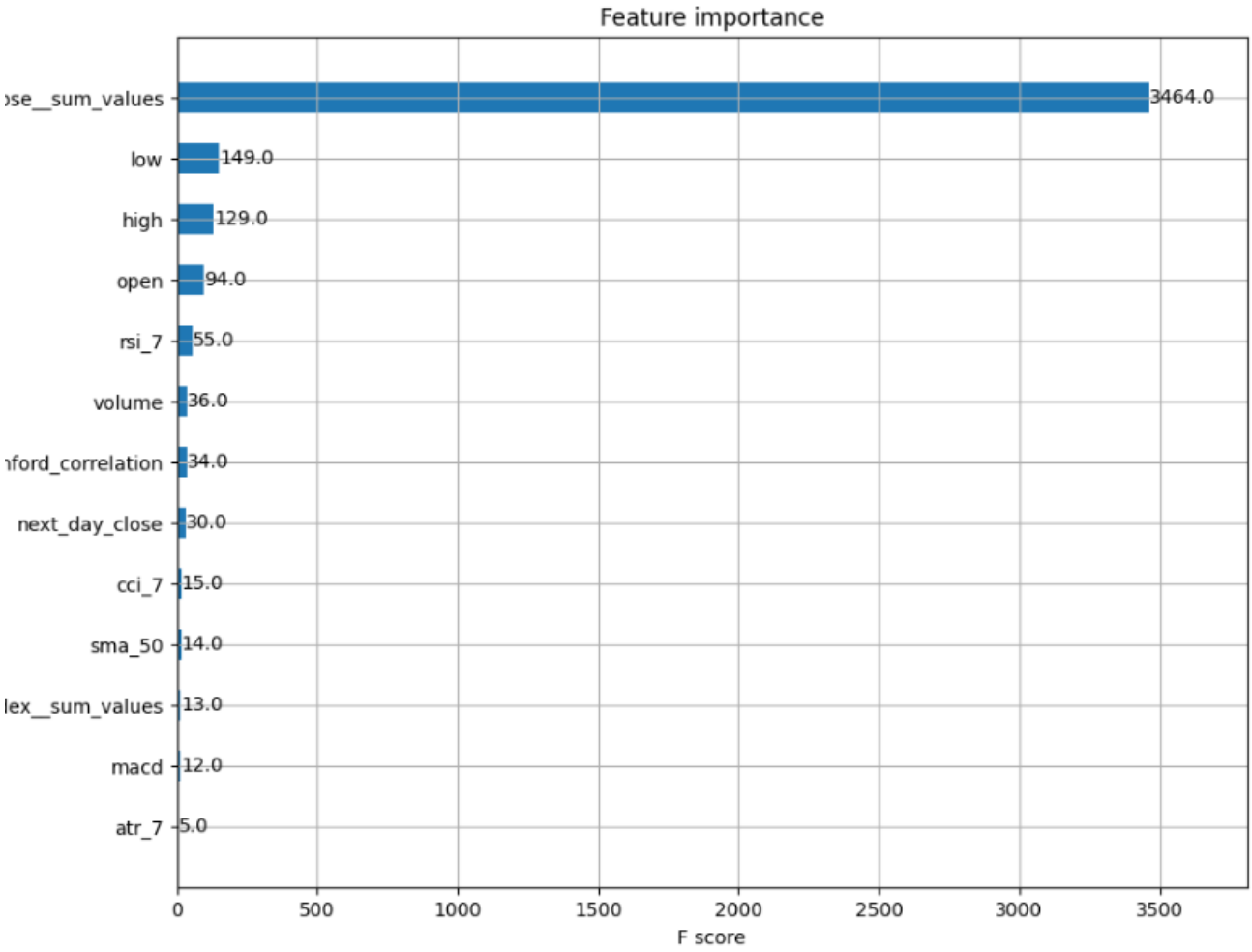

Рисунок Г.2 – Графік важливості ознак

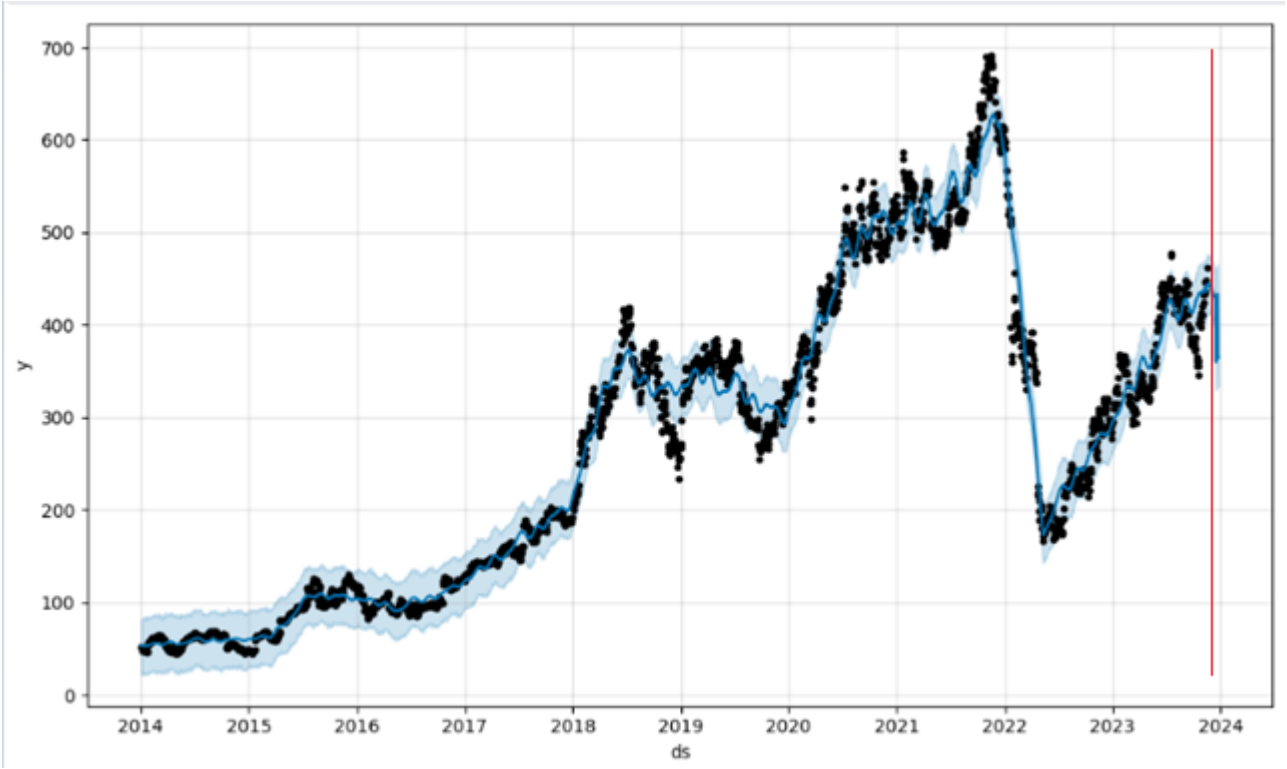

Рисунок Г.3 – Графік прогнозування моделлю Prophet

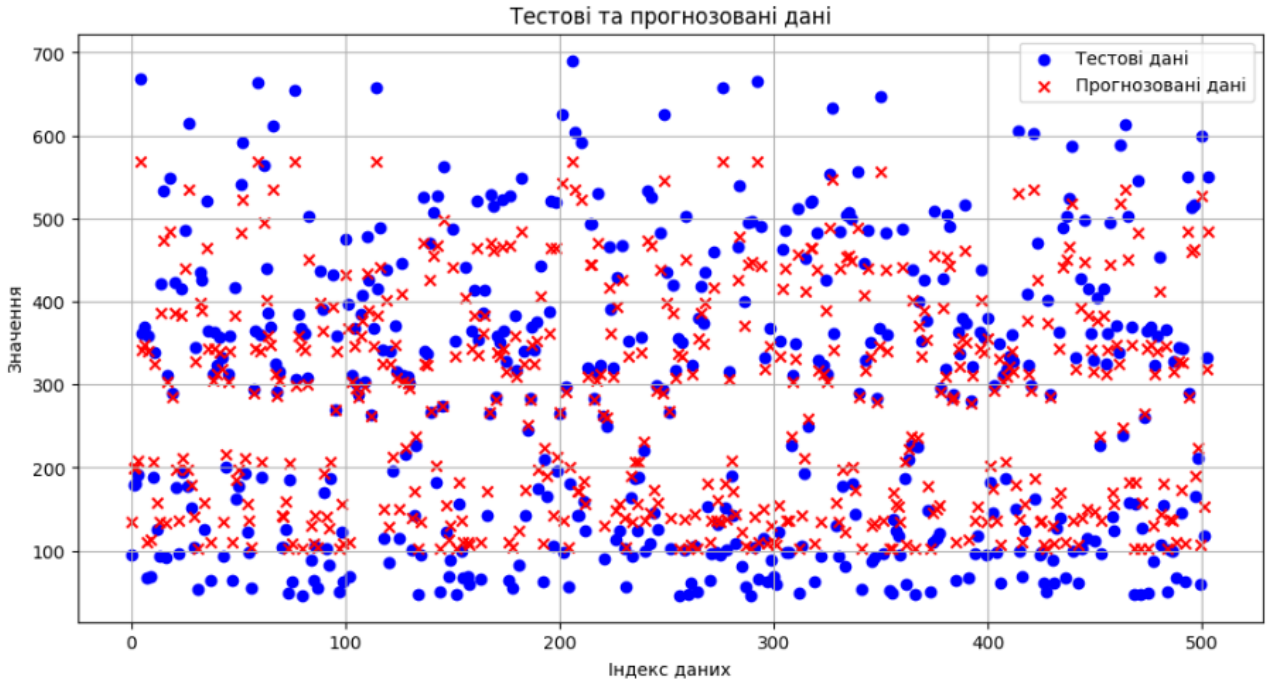

Рисунок Г.4 – Графік прогнозованих даних

	Model	Точність на тренувальних даних MSE	Точність на тестових даних MSE	Точність на тренувальних даних MAE	Точність на тестових даних MAE	Точність на тренувальних даних R2	Точність на тестових даних R2
0	Prophet	2698.335730	3646.411677	32.475711	58.689257	0.900000	-101.710000
1	ARIMA	74.150846	370.064045	5.042152	18.291012	0.997298	-9.423483
2	XGBoost	1395.255825	1478.054292	32.134727	33.397530	0.910000	0.910000

Рисунок Г.5 – Таблиця порівняння оцінок точності прогнозування моделей