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

На правах рукопису
УДК 519.216.3

До захисту допущено
Завідувач кафедри ММСА
_____ Оксана ТИМОЩУК
«___» _____ 2024 р.

Магістерська дисертація
на здобуття ступеня магістра
за освітньо-професійною програмою «Системний аналіз фінансового ринку»
зі спеціальності 124 «Системний аналіз»
на тему: «Порівняльний аналіз моделей для методів прогнозування»

Виконав:
Студент 2 курсу, групи КА-22мп
Макухін Євген Ігорович _____

Науковий керівник:
Професор кафедри ММСА, д.ф.-м.н., проф.
Макаренко Олександр Сергійович _____

Рецензент:
Професор кафедри ММСА, д.т.н., професор
Данилов Валерій Якович _____

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів
без відповідних посилань
Студент (*підпис*): _____

Київ - 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти — другий (магістерський)
Спеціальність — 124 «Системний аналіз»
Освітньо-професійною програмою «Системний аналіз фінансового ринку»

ЗАТВЕРДЖУЮ
Завідувач кафедри ММСА
_____ Оксана ТИМОЩУК
«___» _____ 2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Макухіну Євгену Ігоровичу

1. Тема дисертації: «Порівняльний аналіз моделей для методів прогнозування»,
науковий керівник дисертації Макаренко Олександр Сергійович, д.ф.-м.н., професор, затверджені наказом по університету від «08» листопада 2023 р. № 5200-с

2. Строк подання студентом дисертації: 12 січня 2024 р.

3. Об’єкт дослідження: методи прогнозування часових рядів.

4. Предмет дослідження: моделі для методів прогнозування.

5. Перелік завдань, які потрібно розробити:
- здійснити огляд технічної літератури за темою дисертації;
- дослідити актуальність обраної теми;
- дослідити теорію прогнозування часових рядів;
- визначитись із даними для моделювання;
- дослідити існуючі математичні методи та моделі для прогнозування даних;
- розробити програмний продукт, для проведення дослідження;
- провести експериментальне моделювання для дослідження ефективності обраних моделей прогнозування, провести порівняльну роботу;
- розробити стартап-проект виведення результатів дослідження на ринок;
- розробити концептуальні висновки за результатами дослідження.

6. Перелік графічного (ілюстративного) матеріалу: аналіз обраного набору даних, результати роботи програмного продукту на прикладі обраного набору даних, таблиці у розділі стартап-проекту.

7. Орієнтовний перелік публікацій:
- Макухін Є.І., Макаренко О.С., Бідюк П.І. Порівняльний аналіз моделей для методів прогнозування, II Всеукраїнська науково-практична конференція «Системні науки та інформатика», 4–8 грудня 2023 р., Київ : КПІ ім. Ігоря Сікорського, 2023. С. 172-180.

8. Консультанти розділів дисертації

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

9. **Дата видачі завдання**: 01.09.2023

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Затвердження теми магістерської дисертації та огляд технічної літератури.	01.09.2023-15.09.2023	Виконано.
2	Концептуальний вступ дисертації. Формулювання об’єкта, предмета, цілі, завдань, актуальності, практичної значущості результатів.	15.09.2023-22.09.2023	Виконано.
3	Визначення датасету для моделювання. Його аналіз та упорядкування.	22.09.2023-29.09.2023	Виконано.
4	Перший розділ. Дослідження загальної проблематики прогнозування фінансових ринків	29.09.2023-06.10.2023	Виконано.
5	Другий розділ. Огляд існуючих методів та підходів для прогнозування фінансових ринків.	06.10.2023-20.10.2023	Виконано.
6	Розробка початкового варіанту програмного забезпечення.	20.10.2023-27.10.2023	Виконано.
7	Третій розділ. Експериментальне моделювання та аналіз отриманих результатів.	27.10.2023-03.11.2023	Виконано.
8	Четвертий розділ. Розробка стартап-проекту	03.11.2023-10.11.2023	Виконано.
9	Висновки по роботі і перспективи подальших досліджень	11.11.2023-20.11.2023	Виконано.
10	Оформлення магістерської дисертації	20.11.2023-31.12.2023	Виконано.

Студент _____ Євген МАКУХІН

Науковий керівник дисертації _____ Олександр МАКАРЕНКО

РЕФЕРАТ

Магістерська дисертація: 91 с., 18 рис., 22 табл., 1 додаток, 16 джерел.

Тема магістерської дисертації: «Порівняльний аналіз моделей для методів прогнозування».

Мета роботи – порівняння різних моделей для методів прогнозування.

Об’єкт дослідження: методи прогнозування часових рядів.

Предмет дослідження: моделі для методів прогнозування.

Отримані результати – побудоване спеціалізоване програмне забезпечення мовою Python, що надає змогу досліджувати результати роботи моделей для методів прогнозування, а також дає можливість порівняти результати їх прогнозів. Для порівняння прогнозів, були розглянуті різні математичні методи та моделі машинного навчання, які використовуються для вирішення практичних завдань аналізу та прогнозування нестаціонарних процесів. Такими є модель авторегресії інтегрованого ковзного середнього ARIMA, LSTM, N-BEATS та ансамблева модель.

МОДЕЛЬ ПРОГНОЗУВАННЯ, МЕТОД ПРОГНОЗУВАННЯ, ARIMA, LSTM, N-BEATS, АНАЛІЗ ДАНИХ.

ABSTRACT

Master’s thesis: 91 p., 18 fig., 22 tabl., 1 appendix, 16 sources.

Research topic: comparative analysis of models for forecasting methods.

Research goal: comparison of different models for forecasting methods.

Object of the study: time series forecasting methods.

Subject of the study: models for forecasting methods.

Obtained results: we developed specialized software in Python, which allows to study the results of models for forecasting methods and allows to compare the results of their forecasts. To compare the forecasts, various mathematical methods and machine learning models used to solve practical problems of analyzing and forecasting non-stationary processes were considered. These include the ARIMA autoregressive integrated moving average model, LSTM, N-BEATS, and the ensemble model.

FORECASTING MODEL, FORECASTING METHOD, ARIMA, LSTM, N-BEATS, DATA ANALYSIS.

ЗМІСТ

ВСТУП

На сьогодні фондовий ринок є необхідним інструментом, який дозволяє фінансовим аналітикам, інвесторам та трейдерам вкладати кошти у цінні папери і оцінювати вартість активів. Водночас, для досягнення більш ефективних інвестицій користувачам необхідно мати надійні прогнози. З цією метою виникає завдання знаходження та використання методів та моделей для аналізу та створення точних прогнозів динаміки ціноутворення на фондовому ринку.

Знаходження відповідних моделей дозволяє зробити точний прогноз і допомогти в прийнятті рішення щодо інвестицій. Використання різних моделей дозволить знайти метод який найкраще підходить для відповідних даних. При цьому важливо враховувати усі особливості та властивості прогнозованих даних. Знаходження відповідних моделей дозволить покращити якість прогнозування.

Для порівняння прогнозів, були розглянуті різні математичні методи та моделі машинного навчання, які використовуються для вирішення практичних завдань аналізу та прогнозування нестаціонарних процесів. Такими є модель авторегресії інтегрованого ковзного середнього ARIMA, LSTM, N-BEATS та ансамблева модель.

У першому розділі проводиться огляд загальної проблематики прогнозування фондових ринків.

У другому розділі розглядаються методи прогнозування часових рядів.

У третьому відділі описується розроблений програмний продукту, а також представлені результати експериментів, пов'язаних із використанням розглянутих моделей, які порівнюються між собою.

Четвертий розділ присвячений розробці стартап-проекту на основі створеного програмного забезпечення.

РОЗДІЛ 1 АНАЛІЗ ТА ОСОБЛИВОСТІ МОДЕЛЕЙ ПРОГНОЗУВАННЯ

1.1 Актуальність дослідження

На сьогоднішній день важливість завдань прогнозування за допомогою алгоритмів машинного навчання надзвичайно висока. Зі збільшенням обсягу інформації та прогресом у сфері технологій аналізу, здатність прогнозувати майбутні події та тенденції є важливим фактором у різних галузях. У галузі бізнесу прогнозування сприяє прийняттю обґрунтованих рішень щодо запасів, виробництва та маркетингу для підтримки конкурентоспроможності та оптимізації витрат. У сфері фінансів прогнозування відіграє важливу роль у процесі прийняття рішень щодо інвестицій та управління ризиками. У медицині використання прогнозів може бути спрямоване на діагностику, передбачення розповсюдження захворювань і розробку нових методів лікування. У науці й технологіях прогнозування дозволяє передбачити кліматичні зміни, сприяє розвитку нових матеріалів і технологій і т.д. У сфері транспорту прогнози допомагають оптимізувати маршрути та планувати обслуговування. При вирішенні задач великого масштабу прогнозування може використовуватися для довгострокового планування. Машинне навчання у цьому контексті виявляється невід'ємним інструментом, що дозволяє аналізувати великі обсяги даних та створювати моделі для прогнозування різних явищ. Застосовуючи методи штучного інтелекту, машинне навчання допомагає підвищити точність та якість прогнозів.

На фондовому ринку обертаються цінні папери різних категорій, їхні ціни і строки випуску можуть значно відрізнятися. Поміж цих цінних паперів виділяють класичні часткові та боргові цінні папери, похідні (опціони, ф'ючерси, варранти), а також фінансові інструменти (векселі, облігації, депозитні сертифікати).

Усі операції з цінними паперами на фондовому ринку, такі як купівля, продаж, дарування, обмін і інші, можна відстежити в режимі онлайн. Це дозволяє дізнатися, хто і за якою ціною придбав або продав їх. Інвестування стало надзвичайно доступним - достатньо мати авторизований акаунт на торговій платформі (наприклад, Yahoo! Finance) і необхідну суму для придбання акцій. У сучасних економічних умовах фондовий ринок постійно розвивається, привертаючи десятки мільйонів жителів планети як акціонерів.

Також важливу роль на ринку цінних паперів відіграють інвестиційні компанії. Для них особливо важливо передбачити цінові зміни акцій, особливо в періоди криз, коли збільшується волатильність і різко підвищуються ризики та ступінь невизначеності.

Прогнозування в сфері фондових ринків є критично важливим елементом для учасників цього складного економічного сегменту. Інвестори та трейдери покладаються на прогнози для прийняття обґрунтованих рішень про торгівлю акціями та іншими фінансовими інструментами. Точні прогнози полегшують управління ризиками та дозволяють розробляти ефективні стратегії торгівлі. Аналітичні моделі також служать інструментом для оцінки впливу різних факторів, таких як економічні зміни та геополітичні події, на фондові ринки. Врахування глобальних подій та їхніх можливих наслідків допомагає адаптувати стратегії до змінливих умов. В цілому, прогнозування в цій сфері визначає успіх інвесторів та допомагає уникнути можливих ризиків у волатильному середовищі фондового ринку.

1.2 Використання машинного навчання для прогнозування

Машинне навчання відіграє ключову роль у прогнозуванні часових рядів. Такі методи, як рекурентні нейронні мережі (RNN), мережі з довгою

короткочасною пам'яттю (LSTM) та моделі авторегресійного інтегрованого ковзного середнього (ARIMA), довели свою ефективність у виявленні часових залежностей та складних закономірностей у даних часових рядів. Ці алгоритми чудово виявляють тенденції, сезонність та аномалії, що дозволяє робити точні прогнози в різних сферах, таких як фінанси, прогнозування погоди та енергоспоживання. Однією з ключових переваг машинного навчання у прогнозуванні часових рядів є його адаптивність до динамічних середовищ. Моделі можуть навчатися і пристосовуватися до змін, забезпечуючи надійні прогнози, навіть коли стикаються з нелінійними і непередбачуваними тенденціями. Однак проблеми залишаються, зокрема, потреба в достатній кількості високоякісних даних і потенціал для надмірного пристосування. Постійний прогрес у вдосконаленні алгоритмів та інтерпретації моделей має вирішальне значення для підвищення надійності та зручності використання машинного навчання в додатках для прогнозування часових рядів.

Крім того, важлива здатність машинного навчання обробляти великі обсяги даних і автоматично виокремлювати релевантні ознаки підвищує його застосовність у прогнозуванні часових рядів. Моделі глибокого навчання, такі як згорткові нейронні мережі (CNN) та моделі, що базуються на увазі, додають додаткові рівні складності, дозволяючи виокремлювати складні часові патерни.

Ансамблеві методи, що поєднують декілька моделей для підвищення точності, пропонують надійний підхід до зменшення невизначеності, притаманної прогнозуванню часових рядів. Крім того, включення знань про предметну область за допомогою функціональної інженерії або гібридних моделей може ще більше підвищити продуктивність і інтерпретованість. Додатки, що працюють в реальному часі, отримують значну користь від машинного навчання при прогнозуванні часових рядів, надаючи своєчасну інформацію для прийняття рішень. Оскільки технології продовжують

розвиватися, інтеграція машинного навчання з іншими новими технологіями, такими як периферійні обчислення і датчики Інтернету речей, сприяє створенню децентралізованих і ефективних систем прогнозування.

Таким чином, використання машинного навчання для прогнозування часових рядів дає галузям промисловості практичні знання, що сприяють кращому управлінню ресурсами, вдосконаленню планування та глибшому розумінню динамічних процесів. Постійне вдосконалення алгоритмів і методології продовжує стимулювати розвиток цієї галузі, сприяючи інноваціям та задоволенню зростаючих потреб різних секторів.

1.3 Огляд моделей прогнозування

Регресійний аналіз представляє собою метод прогнозного моделювання, який вивчає зв'язок між цільовою або залежною змінною та незалежною змінною в наборі даних. Різні підходи до регресійного аналізу використовуються в тих випадках, коли цільова та незалежна змінні демонструють лінійну або нелінійну взаємодію, а цільова змінна представлена неперервними значеннями. Основні застосування регресійних моделей включають визначення сили предиктора, прогнозні тенденції, роботу з часовими рядами та аналіз причинно-наслідкових зв’язків.

В машинному навчанні регресійний аналіз є ключовим методом для вирішення задач регресії через моделювання даних. Основною метою регресійного аналізу є встановлення зв'язку між вихідною змінною та різноманітними зовнішніми факторами (регресорами).

Існує ряд методів регресійного аналізу, і вибір конкретного залежить від різноманітних чинників, таких як тип цільової змінної, форма кривої

регресії та кількість незалежних змінних. Найпростішим варіантом регресійної моделі є лінійна регресія.

За основу моделі покладемо твердження, що існує дискретний зовнішній фактор $X(t)$, що виявляє вплив на шуканий процес $Z(t)$, де зв'язок між процесом і зовнішнім фактором є лінійним. Модель на основі лінійної регресії описується рівнянням:

$$Z(t)=a_0+a_1X(t)+\varepsilon_t, \qquad (1.1)$$

де a_0, a_1 – коєфіцієнти регресії,

ε_t – помилка моделі.

Щоб отримати прогнозні значення в момент часу t потрібно мати значення $X(t)$ в той же самий момент часу t, але ймовірність цього на практиці невелика. Багатозначна модель прогнозування має вигляд

$$Z(t)=a_0+a_1X_1(t)+a_2X_2\ (t)+\ldots+a_sX_s\ (\mathrm{t})+\varepsilon_t. \qquad (1.2)$$

В основі нелінійної регресійної моделі передбачається, що існує визначена функція, яка визначає зв'язок між вихідним процесом. $Z(t)$ і зовнішнім фактором $X(t)$

$$Z(t)=F(X(t),A). \qquad (1.3)$$

Для побудови моделі визначаємо параметри функції $A=[a_1, a_0]$. Процеси для яких вид функціональної залежності між процесом $Z(t)$ і зовнішнім фактором $X(t)$ є відомим, на практиці зустрічаються не часто. Через це нелінійні регресійні моделі застосовуються рідко.

В основі авторегресійних моделей закладаємо припущення, що значення процесу $Z(t)$ лінійно залежить від деякої кількості попередніх значень того ж процесу $Z(t-1),\ldots,Z(t-p)$.

В аналізічасових рядів однією з широко використовуваних є модель авторегресії (AR) і модель ковзного середнього (MA).

Модель авторегресії виявляється дуже корисною для пояснення характеристик часових рядів. У цій моделі поточне значення процесу представлено як лінійна комбінація попередніх значень процесу та впливу, який визначається як білий шум.

$$Z(t)=C+k_1Z(t-1)+ k_2Z(t-2)+\cdots+ k_pZ(t-p)+ \varepsilon_t, \qquad (1.4)$$

де C - константа,

$k_1,...,k_p$- коефіцієнти,

ε_t- помилка моделі.

Формула (1.4) описує процес авторегресії порядку p, який в літературі зазвичай позначається AR(p). Щоб визначити k_p і C використовується метод найменших квадратів або метод максимальної правдоподібності.

Інший вид моделі є важливим для характеризації часових рядів і часто використовується в поєднанні з авторегресією. Цей вид називається моделлю ковзного середнього порядку q і описується відповідним рівнянням.

$$Z(t)=\frac{1}{q}(Z(t-1)+Z(t-2)+\cdots+Z(t-q))+ \varepsilon_t, \qquad (1.5)$$

де q – порядок ковзного середнього,

ε_t- помилка прогнозування.

У літературі процес (1.5) зазвичай позначається MA (q). Ця модель є фільтром низьких частот. Варто додати, що існують прості, зважені, кумулятивні, експоненціальні моделі ковзного середнього.

Для забезпечення більшої гнучкості у налаштуванні моделі часто розумно об'єднати авторегресію і ковзне середнє в єдиній моделі. Загальна

модель називається ARMA(p, q), і в ній об'єднуються фільтр у вигляді ковзного середнього порядку q та авторегресія фільтрованих значень процесу порядку p.

Якщо для вхідних даних використовуються не просто значення часового ряду, а їх різниця порядку d (зазвичай d визначається емпірично, але, в більшості випадків, d ≤ 2), то така модель отримує назву авторегресії та інтегрованого ковзного середнього ARIMA(p, d, q).

Покращенням моделі ARIMA (p, d, q) є модель ARIMAX (p, d, q), яка описується рівнянням

$$Z(t)=AR(p)+a_1X_1(t)+a_2X_2(t)+\dots+a_sX_s(t), \qquad (1.6)$$

де $a_0,\dots, a_1$– коєфіцієнти коефіцієнти зовнішніх факторів $X_1(t),.., X_s(t)$.

У даній моделі результатом моделі MA (q) є процес $Z(t)$, що позначає відфільтровані значення вихідного процесу. Щоб прогнозувати $Z(t)$ використовується модель авторегресії, в якій введені додаткові регресори зовнішніх факторів $X_1(t),.., X_s(t)$.

Дерева класифікації та регресії, або CART, є терміном, введеним Лео Брейманом для опису алгоритмів дерева рішень, призначених для вирішення завдань класифікації або регресії у прогностичному моделюванні. Цей алгоритм служить основою для таких важливих методів, як випадкові ліси та підсилених дерев рішень. Структура моделі CART представляє собою бінарне дерево, де кожен кореневий вузол відображає одну вхідну змінну (x) та точку розбиття на цю змінну, припускаючи, що змінна є числовою.

Листові вузли дерева містять вихідну змінну (y), що використовується для прогнозування. Дерево може бути представлено як графік або набір правил.

Вибір вхідної змінної та конкретної точки розбиття визначається за допомогою жадібного алгоритму для мінімізації функції витрат. Побудова дерева завершується використанням заздалегідь визначеного критерію зупинки, такого як мінімальна кількість навчальних екземплярів, призначених кожному листовому вузлу дерева.

Створення бінарного дерева рішень фактично представляє собою процес розподілу вхідного простору. Жадібний підхід використовується для рекурсивного бінарного розщеплення простору. Це числова процедура, в якій всі значення знаходять експериментально, а різні точки розбиття перевіряються за допомогою функції витрат. Обирається розбиття з найкращою вартістю (найнижчою вартістю, оскільки мінімізуються витрати).

Процедура рекурсивного бінарного розщеплення повинна знати, коли припинити розщеплення, пересуваючись вниз по дереву з навчальними даними. Найбільш розповсюдженою процедурою зупинки є використання мінімального підрахунку кількості навчальних екземплярів, призначених кожному листовому вузлу. Якщо підрахунок менший за певний мінімум, то розбиття відхиляється, і вузол стає кінцевим.

Кількість екземплярів для навчання може бути налаштована на набір даних, наприклад, 5 або 10. Це визначає, наскільки конкретним буде дерево для навчальних даних. Занадто конкретне дерево може перевищити навчання та показати низьку ефективність на тестовому наборі.

Критерій зупинки є важливим, оскільки він значно впливає на ефективність дерева. Складність дерева рішень визначається кількістю розділень у дереві, і віддається перевага простим деревам, які легко зрозуміти та менше схильні до перевизначення наявних даних.

Швидший та простіший метод вирізання полягає в оцінці кожного листового вузла на дереві та визначенні впливу його видалення за допомогою тестів витримки. Листові вузли видаляються лише у випадку, якщо це призводить до зменшення загальних витрат на всьому наборі тестів.

Процедура видалення припиняється, якщо подальше удосконалення неможливе.

Можуть використовуватися більш складні методи вирізання, такі як обрізка складності витрат, де навчальний параметр (альфа) використовується для зважування можливості видалення вузлів на основі розміру піддерева.

1.4 Оцінка точності прогнозованої моделі

Точність прогнозу важлива для бізнесу та постійного вдосконалення процесу прогнозування. Тому прогнози слід розглядати як процес постійного удосконалення. Часто формуються базові прогнози за допомогою статистичних підходів, а потім вносяться оціночні корективи, щоб врахувати свої знання про майбутні події. Відстеження точності статистичних та скоригованих прогнозів, допоможе визначити, де коригування покращують прогнози, а де вони погіршують їх. Ці дані дозволяють зосередити увагу на тих елементах, де коригування додають цінності.

Для ефективного використання прогнозу потрібно розуміти очікувану точність. Статистика в межах вибірки та обмеження довіри можуть дати уявлення про очікувану точність; хоча вони майже завжди недооцінюють фактичну (поза вибіркою) помилку прогнозування. Так відбувається через те, що параметри статистичної моделі вибираються, щоб мінімізувати помилку на історичних даних. Отже, такі параметри адаптовані до минулого, а не майбутнього.

Загалом, статистика поза вибіркою (історичні помилки прогнозу) надає кращий показник очікуваної точності прогнозу, ніж статистика у вибірці. Якщо в галузі доступна статистика щодо точності прогнозу, порівняння отриманої точності з цими тестами дозволяє оцінити ефективність прогнозування. Якщо такі показники недоступні, періодичне порівняння

поточної точності прогнозу з попередньою допомагає вимірювати отримане покращення. Різкі несподівані зміни точності прогнозу часто є результатом основної події. Постійне спостереження за помилками прогнозу дозволить виявити, дослідити та відреагувати на ці зміни на ранніх стадіях, до того як вони вплинуть на загальні результати.

Середня абсолютна відсоткова помилка (MAPE) вимірює розмір помилки у відсотках. Вона знаходиться як середнє значення відсоткової помилки.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i}, \quad (1.7)$$

Зазвичай під час прогнозування зосереджуються на MAPE при оцінюванні точності. Якщо виразити результат у відсотках, то MAPE легше розглядати. Ця оцінка може бути корисною і в тому випадку, коли користувач не володіє інформацією щодо обсягу попиту на товар. Проте слід пам'ятати, що MAPE є чутливим до масштабу і не слід використовувати його для аналізу обмежених обсягів даних. Також, MAPE стає невизначеним, коли фактичний попит дорівнює нулю. Крім того, при невеликих значеннях фактичного попиту MAPE часто набуває екстремальних значень. Ця чутливість до масштабу робить MAPE менш ефективним показником похибки для даних з низьким обсягом.

Середня абсолютна помилка (MAE) - це просто середнє значення абсолютної різниці між цільовим значенням та значенням, передбаченим моделлю:

$$MAE = \frac{1}{n}\Sigma_{i=1}^{n}|\hat{y}_i - y_i|. \quad (1.8)$$

Спостереження за точністю прогнозу є ключовою складовою частиною процесу прогнозування. Якщо неможливо оцінити точність поточного процесу, його вдосконалення стає дуже складним завданням. Крім того, відстеження точності прогнозу дозволяє розуміти очікувану ефективність, порівнювати прогнози та вчасно виявляти, досліджувати та реагувати на можливі проблеми. Щоб забезпечити відстеження точності, необхідно зберігати прогнози з часом, щоб у подальшому мати можливість порівняти їх із реальними подіями.

Також RMSE (Root Mean Squared Error або корінь середньої квадратичної помилки) є популярною метрикою для вимірювання точності моделі в прогнозуванні. Формула RMSE виглядає наступним чином:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad . \qquad (1.9)$$

RMSE вимірює середню квадратичну відхиленість прогнозованих значень від фактичних. Ця метрика особливо корисна, оскільки вона покриває великі помилки, що дозволяє зосередитися на точності моделі. Чим менше значення RMSE, тим точніше прогнози моделі.

1.5 Висновки до розділу

В цьому розділі було розглянуто актуальність дослідження, визначено що задача прогнозування має високу актуальність і є невід’ємною частиною роботи багатьох компаній. Було описано використання машинного навчання для прогнозування і розглянуті інші основні моделі. Основною ціллю даної роботи буде порівняльний аналіз моделей для прогнозування. Також у розділі

були розглянуті основні критерії для оцінки прогнозу та виявлення найкращої моделі.

РОЗДІЛ 2 ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ І МОДЕЛЕЙ ДЛЯ ВИРІШЕННЯ ЗАВДАНЬ ПРОГНОЗУВАННЯ ПРОЦЕСІВ НА ФОНДОВОМУ РИНКУ

2.1 Інтегровані моделі авторегресії з ковзним середнім

Моделі ARIMA (авторегресії ковзного середнього з інтеграцією) є розширенням стандартних моделей ARMA для нестаціонарних часових рядів. Для забезпечення стаціонарності часового ряду, які можуть бути досягнуті за допомогою операції взяття різниць порядку d, ці моделі були вперше введені Боксом та Дженкінсом у 1970 році.

Метод ARIMA є популярним і ефективним інструментом для прогнозування часових рядів. ARIMA включає в себе три основні складові: авторегресію (AR), інтегрування (I) та ковзне середнє (MA). ARIMA розшифровується як Авторегресійна інтегрована модель ковзного середнього. Вона належить до класу моделей, які пояснюють заданий часовий ряд на основі його власних минулих значень, тобто власних лагів і помилок прогнозування. Використовуючи рівняння, можна прогнозувати майбутні значення. Будь-який часовий ряд без сезонності, що виявляє закономірності і не є випадковим білим шумом, можна змоделювати за допомогою ARIMA. ARIMA моделі задаються трьома параметрами порядку: (p, d, q), де

- p - порядок AR-члена;
- q - порядок члена MA;
- d - кількість диференціювань, необхідних для того, щоб зробити часовий ряд стаціонарним.

Авторегресія AR(p) - регресійна модель, яка використовує залежний зв'язок між поточним спостереженням і спостереженнями за попередній період. Авторегресійний компонент (AR(p)) означає використання минулих

значень у рівнянні регресії для часового ряду. I(d) Інтегрування - використовує диференціювання спостережень (віднімання спостереження від спостереження на попередньому часовому кроці) для того, щоб зробити часовий ряд стаціонарним. Диференціювання передбачає віднімання поточних значень ряду від його попередніх значень d разів. MA(q) Moving Average - модель, яка використовує залежність між спостереженням і залишковою похибкою від моделі ковзного середнього, застосованої до запізнілих спостережень. Компонент ковзного середнього відображає похибку моделі як комбінацію попередніх членів похибки. Порядок q показує кількість членів, які включаються в модель.

У рамках моделі ARIMA(p,d,q), прогноз майбутнього значення процесу представляє собою обмежену лінійну комбінацію його попередніх значень та помилок, що можна виразити так:

$$y_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \cdots + \alpha_p y_{t-p} + \varepsilon_t - \beta \varepsilon_{t-1} - \cdots - \beta_q \varepsilon_{t-q}, \quad (2.1)$$

де y_t – поточне значення процесу,

ε_t – випадкова похибка у момент часу t,

α_i , β_j – коефіцієнти,

p, q - цілі числа, що відповідають порядкам авторегресії та ковзної середньої відповідно.

Використовуючи лаговий оператор L, загальний вигляд моделі можна записати так:

$$\beta(L)\, y_t = \beta(L)\, \nabla^d y_t = \alpha_0 + \alpha(L)\, x_t, \quad (2.2)$$

$$\beta(L) = 1 - \beta_{j1} L - \beta_2 L^2 - \cdots - \beta_p L^p \ , \quad (2.3)$$

$$\alpha(L) = 1 - \alpha_1\, L - \alpha_2 L^2 - \cdots - \alpha_{\mathrm{q}} L^{\mathrm{q}}, \quad (2.4)$$

де $\alpha(L) = \nabla^d A(L)$ − оператор авторегресії, який є нестаціонарним оператором, для якого d коренів рівняння $\alpha(L) = 0$ дорівнюють одиниці, а $\beta(L)$ − оператор ковзного середнього, тобто корені рівняння $\beta(L) = 0$ розташовані зовні одиничного кола.

Алгоритм побудови моделі ARIMA(p,d,q) можна провести у такі основні кроки.

1. Спочатку визначається пробна модель (тобто значення параметрів p,d,q) для проведення експериментів: на цьому етапі потрбіно отримати стаціонарний часовий ряд.

Для здійснення переходу від нестаціонарного до стаціонарного ряду зазвичай користуються операцією послідовного взяття різниць, що визначає значення параметра d. Таким чином, було встановлено значення одного з параметрів ARIMA-моделі (p, d, q).

Далі зазвичай потрібно провести аналіз автокореляційної функції (АКФ) та часткової автокореляційної функції (ЧАКФ).

Автокореляційна функція (АКФ) описує кореляцію між змінною y_t та її лаговим значенням y_{t-k} як функцію від k, характеризує розвиток y_t з плином часу. З її допомогою можна прийти до висновку про ступінь корельованості величин процесу із попередніми величинами, і відповідно, тривалість та силу пам'яті процесу. Функція показує, наскільки довго збурення ε_t впливає на значення y_t.

Для стаціонарного процесу авторегресії $y_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \cdots + \alpha_p y_{t-p} + \varepsilon_t$ рекурентне співвідношення для АКФ записується за загальною формулою:

$$\mathrm{p}_k = \alpha_1 \mathrm{p}_{k-1} + \alpha_2 \mathrm{p}_{k-2} + \cdots + \alpha_p \mathrm{p}_{k-p} + \varepsilon_t. \qquad (2.5)$$

Тоді, якщо відомі автокореляції, дисперсію y_t знаходять за формулою:

$$\gamma_0 = \sigma^2 = \frac{\sigma_\varepsilon^2}{1-\alpha_1 p_{k-1}+\alpha_2 p_{k-2}+\cdots+\alpha_p p_{k-p}} \ . \qquad (2.6)$$

Більш виразними характеристиками автокореляції є часткові автокореляції, які визначають ступінь кореляції між значеннями часового ряду y_t та y_{t-k}, не враховуючи усередненого впливу проміжних значень ряду.

Частковий коефіцієнт автокореляції α_{kk} k-го порядку як оцінку α_k у моделі AP(k) і знаходиться:

$$\alpha_{11} = \rho_1, \qquad (2.7)$$

$$\alpha_{22} = \frac{\rho_2-\rho_1^2}{1-\rho_1^2}. \qquad (2.8)$$

Часткову автореляційну функцію ЧАКФ розглядають як функцію часткової автокореляції від затримки k, k=1,2…

Для процесу авторегресії порядку p функція буде ненульовою для $k \leq p$ та дорівнює нулю при $k > p$.

Під час побудови моделі ARIMA(p,d,q) часового ряду y_t, потрібно досягти мінімізації кількості її параметрів, що називається принципом економії, який передбачає вибір більш простої моделі. Параметри моделі оцінюють за допомогою коефіцієнтів автокореляції процесу y_t. Зі збільшенням кількості параметрів у моделі необхідно використовувати більше автокореляційних коефіцієнтів (з великим інтервалом). Зі збільшенням інтервалу точність оцінювання зменшується, що призводить до менш надійних оцінок коефіцієнтів у моделях часових рядів, а також до зниження їхньої якості.

Більшість інформації про порядок p авторегресійної моделі AP можна взяти із її ковзної ЧАКФ. Для моделі AP(p), що описує процес, значення коефіцієнтів авторегресії порядків, менших або дорівнюють p, виділяються крайніми значеннями ЧАКФ. У випадку, коли значення ЧАКФ рівні нулю,

починаючи з зсуву p+1, це вказує на відповідність властивостям спостережуваного процесу авторегресії порядку p.

Для моделі КС(q), так само як і для моделей АР, можна побудувати ковзну ЧАКФ для будь-якого порядку. Для пов’язання параметрів моделі та поведінки її АКФ та ЧАКФ використовуються наступні закономірності:

1) при p=1, q=0 АКФ демонструє експоненційне затухання. У випадку, коли значення авторегресійного параметра є від'ємним, АКФ змінює свій знак, а при додатньому значенні не зазнає змін. ЧАКФ має викид на першому лагу, а для всіх інших лагів автокореляція відсутня.

2) при p=2, q=0 в залежності від знаку параметра АКФ затухання може мати експоненційну або синусоїдальну форму . У випадку від'ємного значення параметра авторегресії АКФ змінює свій знак, в той час як при додатньому значенні не відбувається зміни. ЧАКФ проявляє викиди на лагах 1 і 2, і відсутня автокореляція для інших лагів.

3) при p=0, q=1 на першому лазі АКФ має викид, а для інших лагів автокореляції відсутні. ЧАКФ затухає експоненційно, монотонно або змінюючи знак.

4) при p=0, q=2 на перших двох лагах спостерігається викид у АКФ, а для інших лагів кореляції відсутні. ЧАКФ затухає експоненційно або синусоїдально.

5) при p=1, q=1 АКФ експоненційно, починаючи з першого лагу, і воно має монотонний або коливальний характер так само затухає і ЧАКФ.

Цей принцип використовується для підтвердження вибору моделей АРКС. Цей етап є початковим у побудові оптимальної моделі спостережуваного процесу, і він базується на використанні більш точних методів оцінки параметрів моделі.

2. Оцінка параметрів моделі (α_i, β_j коефіцієнтів) та перевірка адекватності моделі.

Для отримання оцінок параметрів моделей часто користуються методом максимальної правдоподібності (ММП). Щоб перевірити тестові моделі на адекватність проводять аналіз ряду залишків цієї моделі. Якщо модель є адекватною, залишки повинні мати властивості білого шуму, тобто їхні значення автокореляційної функції змінюються навколо нуля. Також використовуються різні критерії, такі як коефіцієнт детермінації, статистика Дарбіна-Уотсона, інформаційні критерії Акаіке та Байєса-Шварца, Q-статистика Льюнга-Бокса і так далі.3. Використання моделі для побудови прогнозу.

Щоб оцінити точність прогнозу використовують ряд стандартних критеріїв якості оцінок прогнозів.

Схема алгоритму побудови моделі ARIMA(p,d,q) зображена на рисунку 2.1.

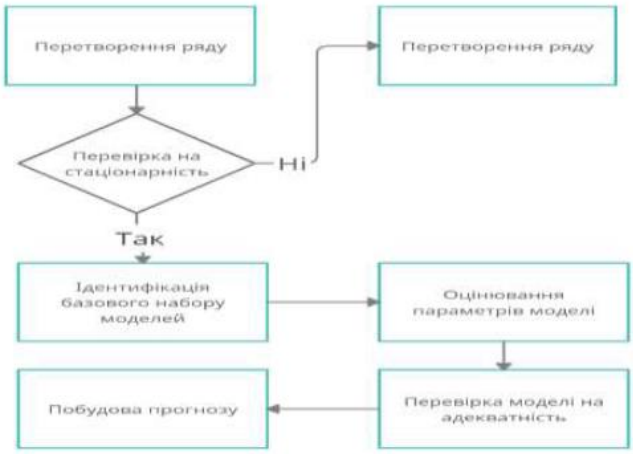


Рисунок 2.1 Схема вибору моделі ARIMA

В даній роботі буде використаний метод auto_arima з бібліотеки pmdarima мови програмування Python для автоматизації процесу прогнозування ARIMA. Метод auto_arima() використовує покроковий підхід для перебору декількох комбінацій параметрів p,d,q і вибирає найкращу модель, яка має найменший AIC(інформаційний критерій Акаіке). Він працює шляхом проведення тестів на диференціювання (наприклад, Квятковського-Філліпса-Шмідта-Шина, розширеного Дікі-Фуллера або Філліпса-Перрона)

для визначення порядку диференціювання, d, а потім підбирає моделі в межах визначених діапазонів start_p, max_p, start_q, max_q. Якщо увімкнено опцію сезонності, auto_arima також намагається визначити оптимальні гіперпараметри P та Q після проведення тесту Канови-Хансена для визначення оптимального порядку сезонного диференціювання, D.

2.2 Мережі з довготривалою короткочасною пам'яттю

Мережі з довготривалою короткочасною пам'яттю (ДКЧП, англ. long short-term memory, LSTM) представляють собою спеціальний тип рекурентних нейронних мереж, який може ефективно вивчати довгострокові залежності. Цей підхід був запропонований Хохрайтером і Шмідгубером у 1997 році і був пізніше удосконалений та поширений в інших роботах. Мережі LSTM демонструють високу ефективність у вирішенні різних завдань та сьогодні широко використовуються. Як і більшість рекурентних нейронних мереж, LSTM є універсальними у тому сенсі, що при достатній кількості вузлів мережі вони можуть виконувати будь-які обчислення, які може виконувати звичайний комп'ютер, за умови, що у них є належна матриця вагових коефіцієнтів, яку можна розглядати як їх програму. Однак, на відміну від традиційних рекурентних нейронних мереж, мережі LSTM особливо ефективні для навчання на досвіді з метою класифікації, обробки або прогнозування часових рядів, особливо в умовах, коли між важливими подіями існують часові затримки невідомої тривалості.. LSTM явно розроблені для того, щоб уникнути проблеми довготривалої залежності.

Мережа з довготривалою короткочасною пам'яттю (ДКЧП) представляє собою штучну нейронну мережу, що включає в себе вузли ДКЧП,

які можуть бути використані замість або додатково до інших вузлів у мережі. Вузол ДКЧП, який входить до складу рекурентної нейронної мережі, оснащений здатністю запам'ятовування значень на тривалі періоди часу, будь то довгі чи короткі. Ключовою особливістю є те, що він не використовує функції активації в межах своїх рекурентних компонент. Таким чином, збережені значення не розпливаються з плином часу, і член градієнту або вини (англ. blame) не має тенденції розмиватися під час тренування за допомогою зворотного поширення в часі.

Вузли часто втілені у «блоках» (англ. blocks), що містять декілька вузлів мережі. Цей тип конструкції є характерним для "глибоких" багатошарових нейронних мереж і сприяє реалізаціям на паралельному апаратному забезпеченні. У представлених нижче рівняннях кожна змінна, записана курсивом у нижньому регістрі, визначає вектор, розмір якого відповідає кількості вузлів ДКЧП в блоку.

Блоки ДКЧП включають три або чотири "вентилі" (англ. gates), які використовуються для управління потоком інформації до або з їхньої пам'яті. Ці вентилі реалізовані за допомогою логістичної функції для обчислення значень в межах від 0 до 1. Для часткового дозволу або блокування потоку інформації до або з цієї пам'яті використовується множення на ці значення. Наприклад, "вхідний вентиль" (англ. input gate) контролює міру, до якої нове значення вводиться в пам'ять. "Забувальний вентиль" (англ. forget gate) керує мірою, до якої значення залишається в пам'яті. "Вихідний вентиль" (англ. output gate) керує мірою, до якої значення в пам'яті використовується для обчислення активації виходу блоку. (В деяких реалізаціях вхідний та вентиль забуття об'єднують в один. Ідея їх об'єднання полягає в тому, що час забути настає тоді, коли з'являється нове значення, яке варте запам'ятовування.) Схема блоку зображена на рисунку 2.2.

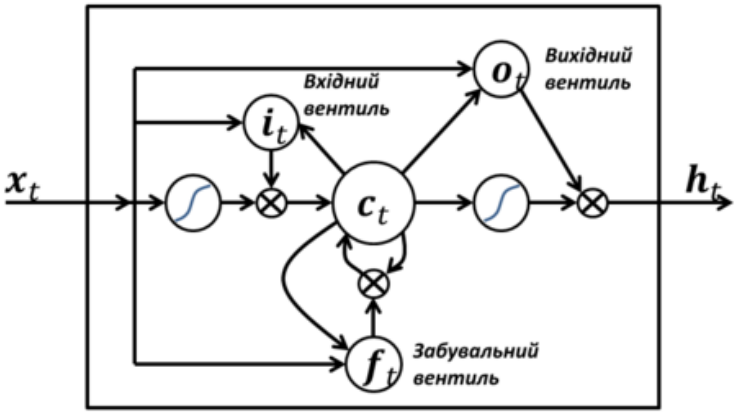


Рисунок 2.2 – Схема блоку мережі

Єдині наявні в блоці ваги ДКЧП (W та U), будуть використані для спрямовування дії вентилів. Ці ваги застосовуються між значеннями, що надходять до блоку (включно з входовим вектором x_t та виходом з попереднього моменту часу h_{t-1}) та кожним із вентилів. З цього слідує, що блок ДКЧП визначає, як забезпечити свою пам'ять як функцію від цих значень, і тренування ваг блока ДКЧП призводить до навчання такої функції, що мінімізує втрати. Тренування блоків ДКЧП зазвичай здійснюється за допомогою зворотного поширення в часі.

Основна ідея застосування LSTM полягає в здатності мережі "вирішувати", яку інформацію тримати та яку забути, що дозволяє їй ефективно моделювати довгострокові залежності. Це здатність особливо корисна в задачах прогнозування часових рядів, обробці природної мови та інших завданнях, де важлива контекстуальна інформація. У Python для реалізації LSTM можна використовувати бібліотеки, такі як TensorFlow чи PyTorch.

2.3 Нейронні мережі N-BEATS

N-BEATS (Neural Basis Expansion Analysis and Transformation System) - це архітектура нейронних мереж для задач прогнозування часових рядів, яка була представлена в статті "N-BEATS: Neural Basis Expansion Analysis and Transformation System" в 2020 році. Ця архітектура відрізняється своєю гнучкістю та здатністю адаптуватися до різноманітних завдань прогнозування. Ключові особливості N-BEATS:

- підтримка декількох часових рядів: N-BEATS можна навчати на декількох часових рядах, кожен з яких представляє різний розподіл;
- швидке навчання: Модель не містить жодних рекурентних шарів або шарів самоуваги - таким чином, швидше навчання та стабільний градієнтний потік;
- багатогоризонтне прогнозування: Модель виробляє багатокрокові прогнози;
- інтерпретованість: Автори розробили 2 версії моделі: загальну та інтерпретовану. Інтерпретована версія може виводити інтерпретовані прогнози щодо тренду та сезонності.

Запропонована у статті архітектура мережі зображена на рисунку 2.3.

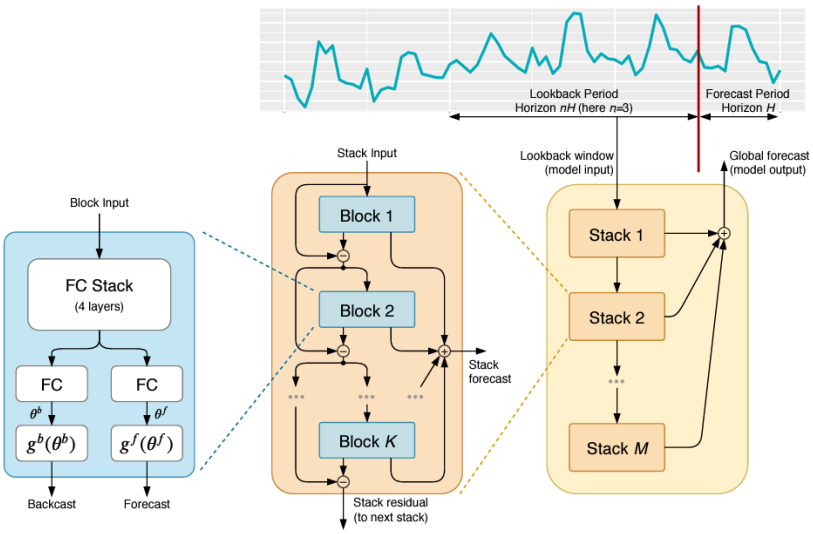


Рисунок 2.3 Запропонована архітектура моделі

Основні елементи:

- блок (синій колір) - основна одиниця обробки;
- стек (помаранчевий колір) - колекція блоків;
- кінцева модель (жовтий колір) - сукупність стеків.

1) Вхідний часовий ряд

На рисунку 2.4 зображено приклад часового ряду що розглядається.

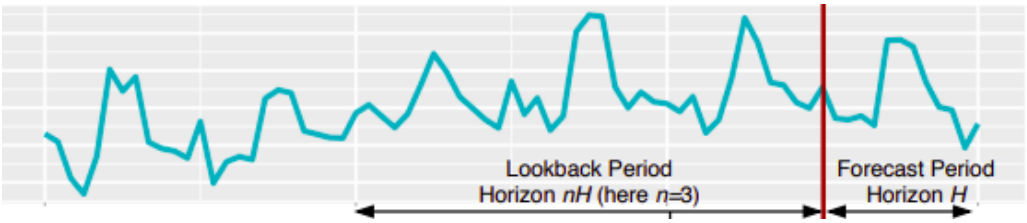


Рисунок 2.4 Часовий ряд що розглядається

Модель приймає дані часового ряду до x_t (t = точки даних до моменту часу t) як вхідні дані і прогнозує майбутнє x_{t+h} (h = довжина вікна прогнозу). Розмір вхідних даних становить n*H (n зазвичай коливається від 2 до 7), який також називають періодом ретроспективи, модель вивчає поведінку часового ряду протягом періоду ретроспективи і намагається передбачити поведінку даних до "H майбутніх точок", які також називають періодом прогнозування.

2) Блоки архітектури:

Дані часових рядів за минулий період слугують вхідними даними для стеку 1, який, у свою чергу, складається з декількох блоків, що розташовані за принципом подвійного залишкового стекування. Для того, щоб зрозуміти роботу архітектури, потрібно спочатку зрозуміти Базовий блок що зображений на рисунку 2.5. Його внутрішню структуру можна побачити на рисунку 2.6

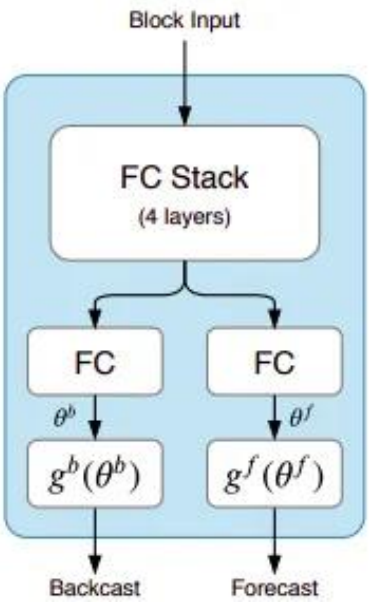


Рисунок 2.5 Загальний базовий блок

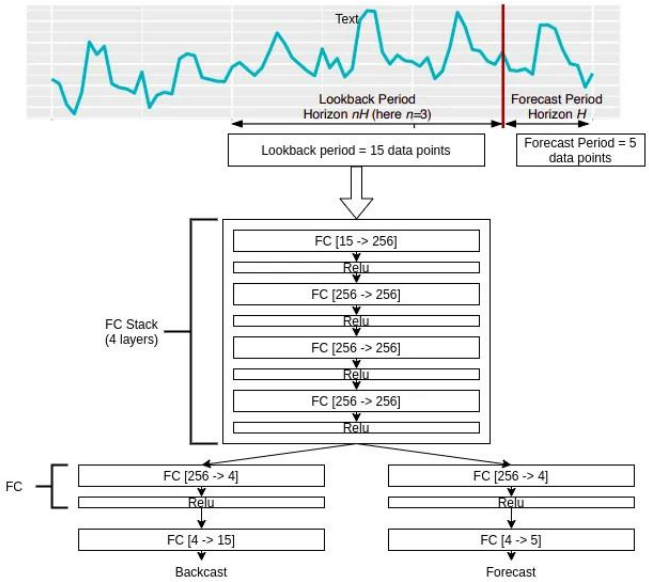


Рисунок 2.6 Внутрішня структура загального базового блоку

Вхідні дані до стеку 1 будуть проходити через блок, який показано на рисунку 2.5, на рисунку 2.6 показано конфігурацію внутрішніх шарів базового

блоку для кращого розуміння. Встановлюємо довжину періоду прогнозування 5 точок даних, а періоду ретроспективного аналізу - 15 точок даних. 15-мірні вхідні дані з Lookback Period спочатку пропускаються через 4-шаровий стек [FC+Relu], а потім розділяються на дві частини. Кожна з яких далі пропускається через інший FC і, нарешті, отримуємо два виходи, 15-мірний вектор у формі Backcast і 5-мірний вектор у формі Forecast. Цей базовий блок навчається прогнозувати не тільки майбутні точки даних у формі прогнозу, але й прогнозує вхідні дані у формі ретроспективного прогнозу. Структуру стеку зображено на рисунку 2.7.

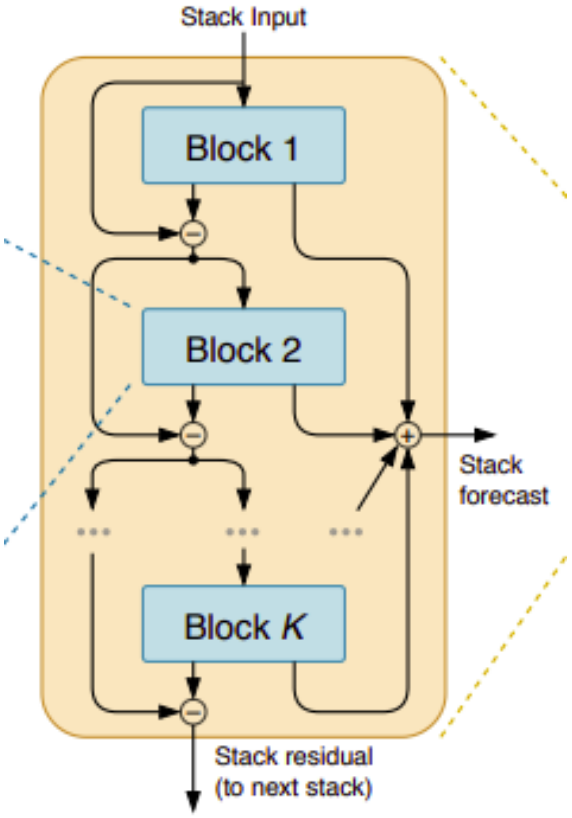


Рисунок 2.7 Внутрішня структура стека

Один стек складається з декількох базових блоків, розташованих за принципом подвійного залишкового стекування. З виходом базового блоку виконується дві арифметичні операції (тобто прогноз і ретроспективний прогноз), звідси і термін "подвійне залишкове підсумовування".

Вхідний 15-мірний сигнал за минулий період (Lookback_inp) проходить через Блок 1, який дає нам два виходи Backcast_1 і Forecast_1.

Вхідні дані для Блоку 2 будуть поелементним відніманням Backcast_1 від Lookback_inp(Backcast_1-Lookback_inp). Віднімаючи Lookback_inp від Backcast_1, на вхід Блоку 2 буде передано вектор, який включає лише ті дані, які не були достатньо вивчені Блоком 1.

Дотримуючись цієї логіки, на вхід кожного Блоку буде подаватися 15-мірний вектор, який складається з поелементного віднімання виходу та входу Backcast попереднього Блоку. Вихідний результат прогнозу останнього Блоку в стеку називається вихідним результатом прогнозу стеку. Тоді як прогнозні виходи всіх блоків у стеку будуть додаватися поелементно, щоб отримати вектор. Цей вектор буде слугувати вихідним прогнозом стека. Далі об’єднаємо стеки, структуру результату можна побачити на рисунку 2.8.

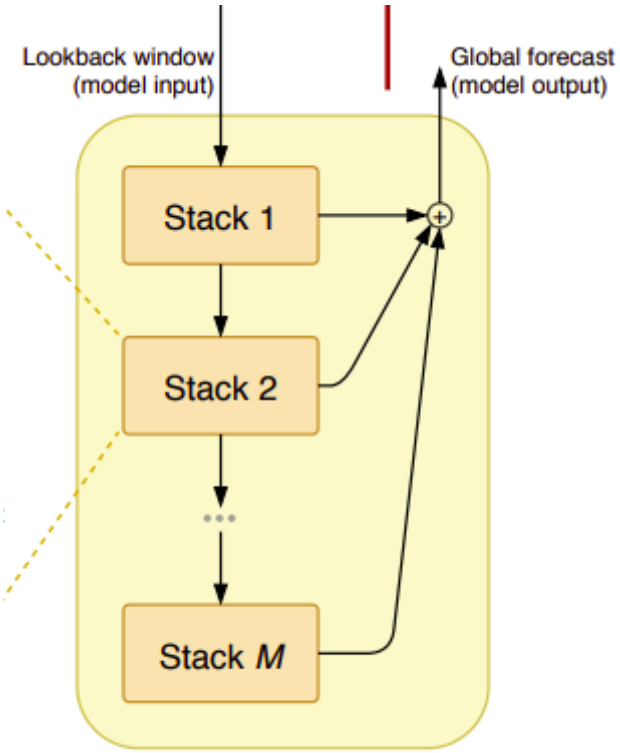


Рисунок 2.8 Структура стеків

З наведеного вище зображення видно, що вектор lookback_inp при проходженні через стек 1 дасть два виходи: вихід зворотного прогнозу стека і вихід прогнозу стека з роздільною здатністю. Вихідні дані стека слугуватимуть вхідними даними для стека 2 (цей вектор представляє досвід, не засвоєний стеком 1), і так само вихідні дані стека з окремого стека

слугуватимуть вхідними даними для наступного стека в ланцюжку. Вихідні дані стека прогнозів з усіх стеків будуть підсумовуватися поелементно, щоб отримати остаточний 5-вимірний вектор глобального прогнозу.

Втрати з використанням MSE будуть розраховані на основі цього прогнозованого вектора глобального прогнозу і даних наземної істини. Всі градієнти в архітектурі будуть оновлені на основі цієї втрати.

2.4 Ансамблевий метод

Ансамблеві методи використовують кілька алгоритмів навчання, щоб отримати кращу прогностичну ефективність, ніж можна було б отримати від будь-якого з складових алгоритмів навчання окремо. Для створення ансамблевих моделей буде використано комбінацію:

- різних функцій втрат (MAE, MSE, MAPE);
- випадково ініціалізованих моделей.

Буде створено набір різних моделей, які намагатимуться моделювати одні й ті самі дані та функції для списку різних моделей, навчених з різними функціями втрат. Кожен шар в ансамблі моделей буде ініціалізовано випадковим нормальним (гаусівським) розподілом за допомогою Не нормальної ініціалізації, це допоможе оцінити інтервали прогнозування пізніше.

2.5 Висновки до розділу

В цьому розділі було розглянуто основні методи, що будуть використовуватися для порівняння результатів прогнозування, їх специфіку, особливості використання та ефективність. Розкрито їх переваги та недоліки, принципи роботи методів машинного навчання.

Використання нейронних мереж для прогнозування допомагає вирішувати завдання прогнозування часових рядів, а також дозволить виділяти слабкі сигнали та виявляти складні закономірності з великих обсягів даних. Ефективні системи програмування сприяють полегшенню численних труднощів, з якими часто зіштовхуються розробники при використанні інших методів, таких як ручне проектування функцій або необхідність виведення градієнтів.

РОЗДІЛ 3 РОЗРОБКА МОДЕЛЕЙ ДЛЯ ПРОГНОЗУВАННЯ АКЦІЙ КОМПАНІЙ

3.1 Обґрунтування вибору платформи та мови програмування

Моделі, які використовуються для прогнозування часових рядів, відіграють важливу роль у багатьох галузях, включаючи економіку, фінанси, кліматологію, логістику та медицину. Однією з найпоширеніших підходів є використання моделей часових рядів ARIMA (авторегресійне інтегроване ковзне середнє). ARIMA враховує попередні значення часового ряду та його структуру для прогнозування майбутніх значень.

Однак з розвитком машинного навчання нейронні мережі стали популярним інструментом для прогнозування часових рядів. Особливо розповсюджені рекурентні нейронні мережі (RNN), які здатні враховувати довгострокові залежності в даних. Машинне навчання стає дуже популярним упродовж останніх років. Основна мета цієї сфери полягає в трансформації важливих даних у стратегії маркетингу та бізнесу, що сприяє розвитку компаній. Дані зберігаються та аналізуються для отримання обґрунтованих рішень. Перші зацікавленість в цій галузі виявляли лише провідні IT-компанії, але на сьогодні підприємства з різних секторів, таких як електронна комерція, охорона здоров'я, фінанси та інші, використовують аналіз даних. Для цього вони використовують різноманітні інструменти, такі як Hadoop, R-програмування, SAS, SQL та інші. Однак Python став найбільш популярним та простим у використанні інструментом для аналізу даних. Цей мовний інструмент визнаний як універсальний, оскільки підтримує структуроване програмування, об'єктно-орієнтоване програмування та інші парадигми.

Python - це найважливіша та найбільш популярна мова програмування у світі, яка відома своєю відмінною придатністю для інструментів та застосунків у галузі науки про дані. Ця мова також вважається однією з найпоширеніших

для створення моделей, призначених для прогнозування часових рядів. Велика кількість бібліотек, таких як Pandas, NumPy, і Matplotlib, надають дослідникам і аналітикам широкий спектр інструментів для обробки, візуалізації та підготовки даних перед моделюванням. Однак основною силою Python у цій області є бібліотека Machine Learning - scikit-learn, а також спеціалізовані бібліотеки, такі як Statsmodels та Tensorflow.

За допомогою Python і цих бібліотек можна розробити різноманітні моделі для аналізу та прогнозування часових рядів, включаючи ARIMA, GARCH, експоненційні згладжування та навіть нейронні мережі. Ці моделі дозволяють враховувати важливі характеристики часових рядів, такі як сезонність, тренди та циклічність, що робить їх придатними для прогнозування фінансових ринків, погоди, виробництва та інших сфер.

Крім того, велика спільнота розробників та дослідників активно працює над розширенням можливостей Python для аналізу часових рядів, що сприяє постійному вдосконаленню та впровадженню нових методів і підходів. Python є важливим інструментом у світі аналітики та прогнозування часових рядів, допомагаючи фахівцям у багатьох галузях приймати інформовані рішення на основі аналізу і передбачення даних.

Через незалежність платформи, Python також легко інтегрується з будь-якою існуючою інфраструктурою, що розв'язує найскладніші проблеми. Давайте розглянемо чому Python вважається вибором для багатьох у галузі науки про дані.

Перш за все, Python вражає своєю потужністю і простотою використання. Вважаючись мовою для початківців, вона доступна для будь-якого студента чи дослідника з базовими знаннями, дозволяючи їм легко розпочати роботу. Час, витрачений на налагодження коду та різні обмеження програмної інженерії, мінімізований. У порівнянні з іншими мовами програмування, такими як C, Java та C#, Python дозволяє розробникам і інженерам програмного забезпечення швидше реалізувати свої ідеї та алгоритми.

Інтернет наповнений навчальними посібниками та ресурсами з інформатики та машинного навчання, що легко доступні. У порівнянні з іншими мовами програмування, такими як Java та R, Python відзначається своєю високою масштабованістю та швидкістю.

Це надає гнучкість для вирішення проблем, які інші мови програмування можуть виявитися неефективними. Багато підприємств використовують Python для розробки швидких додатків та інструментів різного призначення.

Python відомий своєю простотою та легкістю читання коду, що робить його ідеальним вибором для спільної роботи в команді. Це також допомагає швидше вирішувати проблеми та адаптувати моделі.

Усі ці переваги роблять Python ідеальним інструментом для створення, навчання та впровадження моделей прогнозування в різних галузях, надаючи можливість використовувати потужні інструменти аналізу даних і машинного навчання для досягнення цілей і розв'язання складних завдань.

3.2 Аналіз алгоритму роботи системи

В даній роботі запропонований наступний алгоритм реалізації та порівняння моделей для прогнозування. У системі реалізовано 4 моделі:

- ARIMA,
- LSTM,
- N-BEATS,
- Ensemble model.

Для побудови першої моделі проводимо наступні дії:

- здійснення первинної обробки даних,
- здійснення аналізу часового ряду, його перевірка на стаціонарність, декомпозиція, побудова АКФ та ЧАКФ,

- побудова моделі,
- оцінка моделі використовуючи валідаційний набір даних,
- візуалізація прогнозу.

Для кожної з трьох моделей нейромереж проводяться такі кроки:

- попередня обробка даних,
- визначення архітектури та компіляція нейронної мережі,
- встановлення оптимальних гіперпараметрів, функції оптимізації та функції втрат для кожної моделі,
- оцінка ефективності моделі за допомогою валідаційного набору даних,
- візуалізація результатів прогнозу.

3.3 Побудова моделей

Порівняльний аналіз моделей прогнозування буде здійснюватися на прикладі моделювання вартості акцій компанії Apple. Apple Inc. - це одна з найбільших та найвідоміших технологічних компаній у світі. Вона спеціалізується на розробці та виробництві продуктів, які включають смартфони iPhone, комп'ютери Mac, планшети iPad, годинники Apple Watch, аудіо- та відеотехніку, програмне забезпечення та інші інноваційні пристрої. Компанія має величезну глобальну аудиторію та значний вплив на ринок технологій.

Графік цін на акції Apple станом з 1-го січня 2012 року по жовтень 2023 року зображено на рисунку 3.1.

Рисунок 3.1 - Графік цін на акції Apple

Далі проводимо аналіз часового ряду. Даний часовий ряд складається з трьох систематичних компонентів, включаючи рівень, тренд, сезонність та один несистематичний компонент, який називається шумом.

Ці компоненти визначаються наступним чином:

Рівень: Середнє значення в ряді.

Тренд: Зростання або зменшення значення в ряді.

Сезонність: Короткостроковий цикл, що повторюється в ряді.

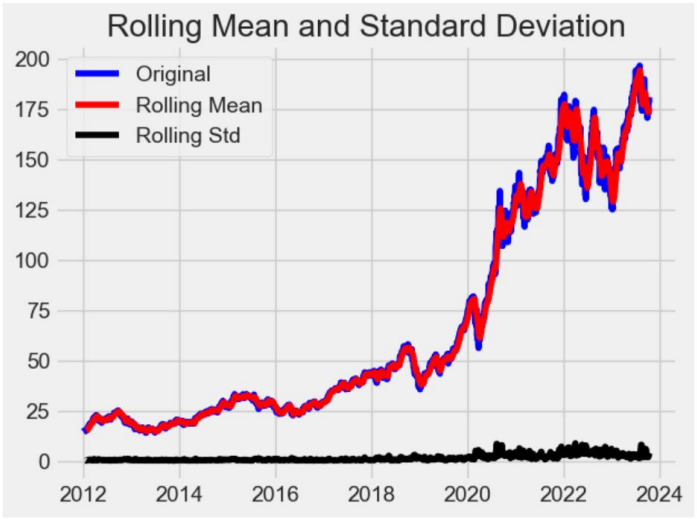
Шум: Випадкові коливання в ряді.

Спочатку нам потрібно перевірити, чи є ряд стаціонарним, оскільки аналіз часових рядів працює тільки зі стаціонарними даними за допомогою ADF (розширений тест Дікі-Фуллера). Тест Дікі-Фуллера є одним з найпопулярніших статистичних тестів. З його допомогою можна визначити наявність одиничного кореня в ряді, а отже, зрозуміти, чи є ряд стаціонарним, чи ні. Розрізняють нульову та альтернативну гіпотези цього тесту:

- Нульова гіпотеза: Ряд має одиничний корінь (значення a = 1)
- Альтернативна гіпотеза: Ряд не має одиничного кореня.

Якщо не можна відкинути нульову гіпотезу, то вважаємо, що ряд є нестаціонарним. Це означає, що ряд може бути лінійним або різницево стаціонарним. Якщо і середнє, і середньоквадратичне відхилення є плоскими

лініями (постійне середнє і постійна дисперсія), то ряд стає стаціонарним. Тож перевіримо його на стаціонарність, результат чого зображений на рисунку 3.2:




```
Results of dickey fuller test
Test Statistics                  0.415162
p-value                          0.982053
No. of lags used                18.000000
Number of observations used    2945.000000
critical value (1%)              -3.432572
critical value (5%)              -2.862522
critical value (10%)             -2.567293
dtype: float64
```

Рисунок 3.2 Графік ковзного середнього, стандартного відхилення та результати тесту

На наведеному вище графіку можна побачити, що середнє значення та стандартне відхилення зростають, а отже, ряд не є стаціонарним.

Також можна побачити, що p-значення більше 0,05, тому не можна відкинути нульову гіпотезу. Крім того, тестова статистика є більшою за критичні значення, отже, дані є нестаціонарними. Тому буде проведено диференціювання ряду.

Для наступного кроку аналізу часового ряду, буде потрібно провести декомпозицію часового ряду. Декомпозиція часових рядів передбачає розгляд ряду як комбінації рівня, тренду, сезонності та шумових компонентів. Декомпозиція забезпечує корисну абстрактну модель для мислення про часові ряди загалом і для кращого розуміння проблем під час аналізу та прогнозування часових рядів. Результати декомпозиції та графіки АКФ та ЧАКФ зображені на рисунках 3.3 та рисунку 3.4.

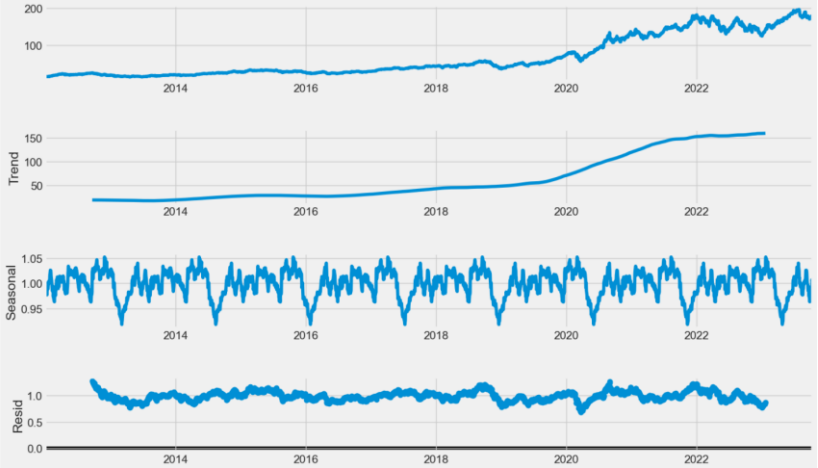


Рисунок 3.3 Результат декомпозиції ряду

Наступним кроком побудуємо графіки АКФ та ЧАКФ.

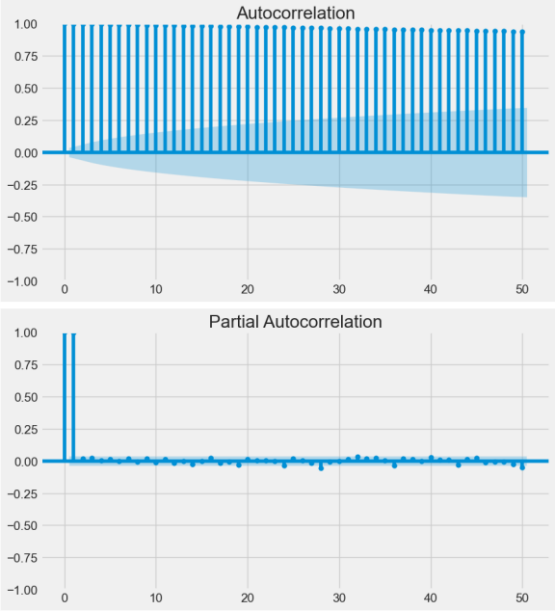


Рисунок 3.4 Графіки АКФ та ЧАКФ

При побудові ARIMA моделі потрібно вибрати параметри p,q,d. Для цього використаємо Auto ARIMA функцію з бібліотеки pmdarima, щоб дозволяє отримати найкращі параметри, навіть не будуючи графіки АКФ та ЧАКФ.

Функція здійснює автоматичний пошук оптимального порядку для ARIMA моделі. Функція auto_arima шукає найоптимальніші параметри для ARIMA моделі і повертає підібрану ARIMA модель. Вона працює шляхом проведення тестів на диференціювання (наприклад, Квятковського-Філліпса-Шмідта-Шина, розширеного Дікі-Фуллера або Філліпса-Перрона) для

визначення порядку диференціювання, d, а потім підбирає моделі в межах визначених діапазонів start_p, max_p, start_q, max_q. Якщо увімкнено опцію сезонності, auto_arima також намагається визначити оптимальні гіперпараметри P та Q після проведення тесту Канови-Хансена для визначення оптимального порядку сезонного диференціювання, D.

Після розділення вибірки на тестову та навчальну у співвідношенні 90 на 10 знайдемо оптимальну модель. Результати пошуку зображені на рисунку 3.5.

```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=9370.149, Time=0.52 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=9376.414, Time=0.03 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=9369.051, Time=0.08 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=9368.795, Time=0.10 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=9378.707, Time=0.03 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=9370.375, Time=0.16 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=9372.148, Time=0.45 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=9371.671, Time=0.05 sec

Best model:  ARIMA(0,1,1)(0,0,0)[0] intercept
Total fit time: 1.429 seconds
```

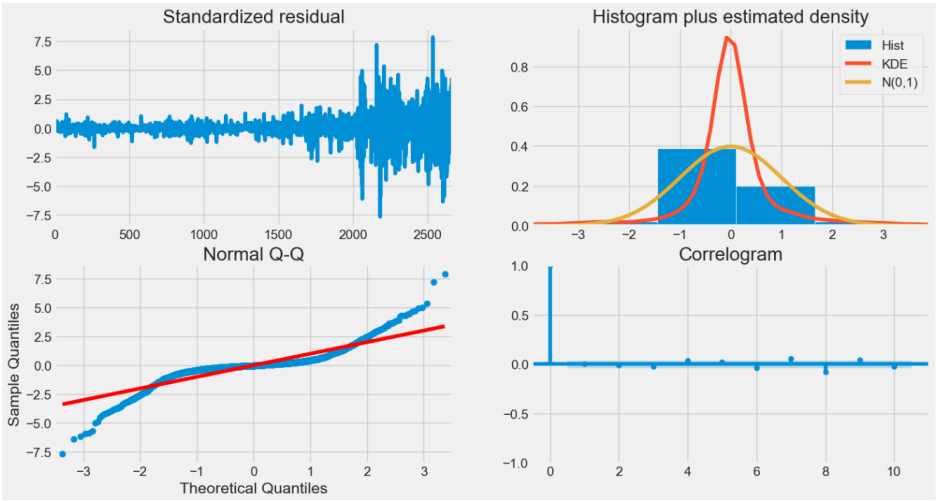


Рисунок 3.5 Результати пошуку оптимальної моделі

Інтерпретація модельної діагностики:

1) Стандартизований залишок: Це похибка прогнозування ціни та фактичної ціни акцій

2) Гістограма плюс оцінена щільність: Гістограма відображає нормальний розподіл помилок, графіки KDE та N(0,1) - це позначення індикативного середнього, що дорівнює нулю, та дисперсії розподілу, що дорівнює одиниці.

3) Нормальний Q-Q: Нормальний Q-Q графік означає нормальність розподілу, оскільки вибіркові величини здебільшого відповідають

теоретичним значенням. будь-яке відхилення в такому вирівнюванні вказує на те, що розподіл є асиметричним, або, кажучи неспеціалізованою мовою, помилка є або позитивною, або негативною.

4) Корелограма: Вона вказує на часткову автокореляцію часових рядів і показує, який з відкладених часових рядів є важливим для прогнозування фактичних часових рядів.

Оптимальною моделлю виявилася модель ARIMA(0,1,1).

Далі використаємо модель для прогнозування значень і оцінимо її. Для оцінки знайдемо значення MAE (середня абсолютна похибка), RMSE (середньоквадратична похибка), MAPE (середня абсолютна відсоткова похибка) і MASE (середня абсолютна масштабована похибка). Результати прогнозу можна побачити на рисунку 3.6.

Рисунок 3.6 Результати прогнозу значень

Було отримано критерії якості прогнозу, що дорівнюють:

MAE = 2.0305376, RMSE = 2.7141287, MAPE = 1.2882643, MASE = 1.000115.

Як можна помітити за результатами, модель має досить точний прогноз.

Далі переходимо до моделей нейромереж. Показники для оцінки моделей нейромереж є дуже важливими. Вибір метрик впливає на те, як вимірюється та порівнюється ефективність. В алгоритмі використовується MAE.

Спочатку потрібно розділити часовий ряд за допомогою вікон. Вікно - це метод перетворення набору даних часового ряду в керовану навчальну задачу. Іншими словами, будуть використовуватися вікна минулого для прогнозування майбутнього. Будемо використовувати розмір горизонту 1 і розмір вікна 7. Потім перетворимо наші вікна на навчальні та тестові розбиття. Замість того щоб розбити на вікна наші існуючі навчальні та тестові вибірки, враховуючи природу розбиття на вікна (розбиття на вікна часто вимагає зсуву в певній точці даних), зазвичай краще спочатку розбити дані на вікна, а потім розділити їх на навчальні та тестові вибірки. В результаті для кожної з моделей маємо розділення вибірки на навчальну та тестову у співвідношенні 80 на 20.

Модель LSTM має функцію активації ReLu, 100 епох та розмір партії 128. За допомогою порівняння середньої абсолютної похибки знаходимо найкращу модель і прогнозуємо значення. Результати для моделі LSTM зображені на рисунку 3.7.

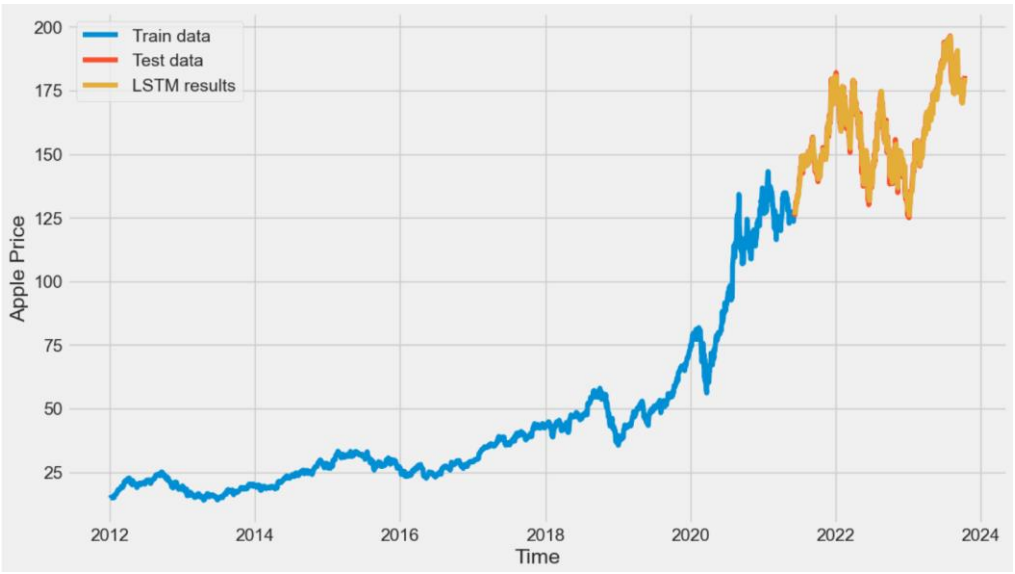


Рисунок 3.7 Результати прогнозу значень для моделі LSTM

Було отримано критерії якості прогнозу, що дорівнюють:

MAE = 2.3084176, RMSE = 3.0007837, MAPE = 1.4737139, MASE = 1.1020792

За результатами можна помітити що модель досить добре прогнозує значення ціни акцій.

Далі будемо оцінювати модель N-BEATS в якій для покращення продуктивності моделі збільшена кількість шарів у порівнянні з іншими аналогічними моделями. Побудуємо модель за запропонованою архітектурою.

Почнемо з побудови блочного шару N-BEATS. Створимо основний будівельний блок для архітектури N-BEATS. Цього разу, оскільки буде використана більшу архітектуру моделі, щоб забезпечити максимально швидке навчання моделі, то потрібно налаштувати набори даних за допомогою API tf.data. Налаштування гіперпараметрів для алгоритму N-BEATS ввізьмемо зі схеми на рисунку 3.8, що описує гіперпараметри, які використовуються для різних варіантів N-BEATS. Використовується N-BEATS-G, що означає загальну версію N-BEATS.

Table 18: Settings of hyperparameters across subsets of M4, M3, TOURISM datasets.

	M4						M3				TOURISM		
	Yly	Qly	Mly	Wly	Dly	Hly	Yly	Qly	Mly	Other	Yly	Qly	Mly
Parameter	N-BEATS-I												
L_H	1.5	1.5	1.5	10	10	10	20	5	5	20	20	10	20
Iterations	15K	15K	15K	5K	5K	5K	50	6K	6K	250	30	500	300
Losses	sMAPE/MAPE/MASE						sMAPE/MAPE/MASE				MAPE		
S-width	2048												
S-blocks	3												
S-block-layers	4												
T-width	256												
T-degree	2												
T-blocks	3												
T-block-layers	4												
Sharing	STACK LEVEL												
Lookback period	$2H, 3H, 4H, 5H, 6H, 7H$												
Batch	1024												
Parameter	N-BEATS-G												
L_H	1.5	1.5	1.5	10	10	10	20	20	20	10	5	10	20
Iterations	15K	15K	15K	5K	5K	5K	20	250	10K	250	30	100	100
Losses	sMAPE/MAPE/MASE						sMAPE/MAPE/MASE				MAPE		
Width	512												
Blocks	1												
Block-layers	4												
Stacks	30												
Sharing	NO												
Lookback period	$2H, 3H, 4H, 5H, 6H, 7H$												
Batch	1024												

Рисунок 3.8 Рекомендовані налаштування гіперпараматерів моделі

Після налаштування гіперпараметрів, тепер, перш ніж буде створена модель N-BEATS, нам потрібно пройти через два шари, які відіграють велику роль в архітектурі. Саме вони роблять можливим подвійне залишкове укладання N-BEATS:

- tf.keras.layers.subtract(inputs) - віднімає список вхідних тензорів один від одного
- tf.keras.layers.add(inputs) - додає список вхідних тензорів один до одного

Залишковий зв'язок (також званий "пропускним зв'язком") передбачає, що більш глибокий шар нейронної мережі отримує виходи, а також входи більш поверхневого шару нейронної мережі. У випадку N-BEATS архітектури використовуються залишкові зв'язки, які:

- віднімають зворотні виходи попереднього блоку від зворотних входів поточного блоку
- додають прогнозні виходи з усіх блоків разом у стек

В результаті для створення і навчання моделі робимо наступне:

1. Створимо екземпляр шару блоків N-BEATS за допомогою NBeatsBlock (це буде початковий блок для мережі, решта будуть створені як частина стеків)
2. Створимо вхідний шар для стека N-BEATS (для цього буде використано Keras Functional API)
3. Зробимо початкові беккаст та прогноз для моделі з шаром, створеним у (1)
4. Створимо за допомогою циклу стеки блокових шарів
5. Використаємо клас NBeatsBlock у циклі, створеному в пункті (4), для створення блоків, які повертають беккасти та прогнози на рівні блоків
6. Створюємо подвійний залишковий стек, використовуючи віднімання та додавання шарів
7. Об'єднаємо входи та виходи моделі за допомогою tf.keras.Model()

8. Скомпілюємо модель з оцінкою MAE та оптимізатором Adam

9. Підготуємо модель N-BEATS для 5000 епох, і оскільки вона має таку кількості епох, то використаємо декілька зворотних викликів:

- tf.keras.callbacks.EarlyStopping() - зупиняє навчання моделі, якщо вона не покращує валідаційні втрати за 200 епох, і відновлює найкращі ваги, використовуючи restore_best_weights=True

- tf.keras.callbacks.ReduceLROnPlateau() - якщо втрата валідації моделі не покращується протягом 100 епох, зменшити швидкість навчання у 10 разів, щоб спробувати допомогти їй зробити поступові покращення (чим менша швидкість навчання, тим менші оновлення намагається зробити модель)

Таким чином, для цієї моделі маємо такі результати прогнозування на рисунку 3.9.

Рисунок 3.9 Результати прогнозу значень для моделі N-BEATS

Було отримано критерії якості прогнозу, що дорівнюють:

MAE = 2.1842735, RMSE = 2.8518758, MAPE = 1.3949207, MASE = 1.0428106

За результатами можна помітити що якість прогнозу трохи покращилась, але не суттєво.

Наш наступний експеримент - створення ансамблю моделей. Ансамбль передбачає навчання та об'єднання декількох різних моделей.

Для створення наших ансамблевих моделей будемо використовувати комбінацію:

- Різних функцій втрат (MAE, MSE, MAPE)
- Випадково ініціалізованих моделей

По суті, буде створено набір різних моделей, які намагатимуться моделювати одні й ті самі дані. Створимо функції для створення списку різних моделей, навчених з різними функціями втрат. Кожен шар в ансамблі моделей буде ініціалізовано випадковим нормальним (гаусівським) розподілом за допомогою Не нормальної ініціалізації, це допоможе оцінити інтервали прогнозування пізніше. Перевіримо для 5 ітерацій та 1000 епох. Це дасть 15 моделей (по 5 для кожної функції втрат). Однак, оскільки навчається 15 моделей, то отримаємо 15 наборів прогнозів. Замість того, щоб порівнювати кожен набір прогнозів з істиною в останній інстанції, візьмемо медіану.

Отже, маємо такі результати прогнозування:

MAE = 2.1178198, RMSE = 2.7985322, MAPE = 1.3546352, MASE = 1.0110844

Також маємо графік прогнозу з довірчим інтервалом на рисунку 3.10.

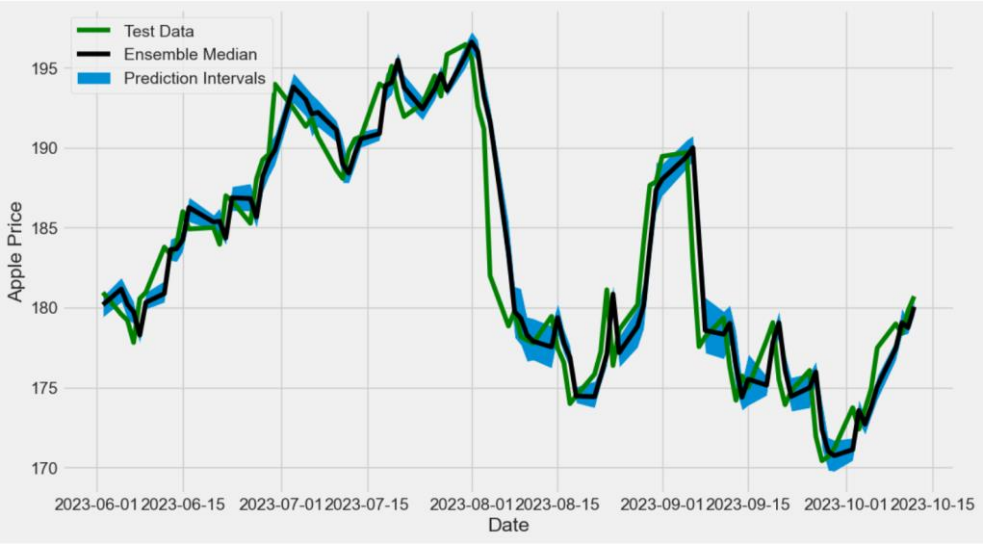


Рисунок 3.10 Результати прогнозу для ансамблю моделей

Тепер порівняємо результати кожної з моделей на рисунку 3.11.

	mae	mse	rmse	mape	mase
model_1_arima	2.030538	7.366495	2.714129	1.288264	1.000115
model_2_LSTM	2.308418	9.004703	3.000784	1.473714	1.102079
model_3_NBEATs	2.184273	8.133196	2.851876	1.394921	1.042811
model_4_ensemble	2.117820	7.831783	2.798532	1.354635	1.011084

Рисунок 3.11. Критерії якості прогнозу для кожної з моделей

Можна побачити, що найкращі результати має arima модель, хоча результати LSTM, N-BEATS та ансамблю майже аналогічні і усі моделі мають достатньо високу якість прогнозування.

3.4 Висновки до розділу

В даному розділі розглянуто та обґрунтовано вибір програмної мови та архітектури, на яких базується реалізація алгоритму. Було описано методи побудови моделей. Було проведено якісний аналіз чотирьох моделей на датасеті та обрано найкращий, а також побудовано прогнози для кожної з них. Найкращою за результатами виявилася модель авторегресійної інтегрованої ковзної середньої.

РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї проекту

"StockPrice AI Forecast" - це інноваційний стартап, який використовує різні моделі штучного інтелекту для прогнозування ціни акцій на фінансовому ринку. Це надасть інвесторам, трейдерам та іншим учасникам ринку надійний та точний інструмент для прийняття рішень щодо інвестування та торгівлі акціями. В таблиці 4.1 описано ідею стартап-проекту.

Таблиця 4.1 Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Створення системи призначеної для кращого розуміння ринкових механізмів, що дозволятиме автоматично завантажувати дані з фондового ринку, зберігати їх та будувати якісні моделі та точні прогнози.	1. Брокерськими компаніями та біржевими аналітиками з метою аналізу фондового ринку	Точні та передбачувані прогнози: Сервіс використовує передові алгоритми машинного навчання та нейронні мережі для створення точних та передбачуваних прогнозів цін акцій
	2.Індивідуальними інвесторами, які прагнуть отримувати більш точні та передбачувані прогнози щодо цін акцій	Можливість отримувати точний прогноз без особливих навичок в математичному моделюванні
	3. Інвестиційними компаніями що мають на меті прийняття рішень щодо вкладання інвестицій у певні компанії	Зменшення ризиків: Надійні прогнози можуть допомогти користувачам уникнути невдалих інвестицій та мінімізувати можливі збитки.

Застосування зрозумілої математичної моделі спрощує реалізацію, а знаходження кращих показників прогнозування може сприяти розповсюдженню системи. Безумовно, існують аналоги такої системи. Для

порівняння розробленої системи проведемо аналіз потенційних техніко-економічних переваг ідеї.

Метою аналізу техніко-економіних переваг є чітке виокремлення технічних і маркетингових особливостей розробленого продукту:

1) дослідження характеристик і властивостей розробленої системи;

2) дослідження конкурентів, товарів-аналогів, товарів-замінників та загальної ситуації на ринку де буде комерціалізувати стартам;

3) проведення порівняльного аналізу слабких, нейтральних та сильних характеристик розробленої системи.

Визначимо сильні, слабкі та нейтральних характеристик ідеї проекту (табл. 4.2). З таблиці можна зробити висновок, що розроблена система є конкурентоспроможною.

Таблиця 4.2. Визначення сильних, слабких та нейтральних характеристик проекту.

№ п/п	Техніко-Економічні характеристики ідеї	Потенційні товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	StocksNeural	Neural Trader			
1	Доступ до бази історичних даних	+	+	+			+
2	Точність прогнозування	Застосування моделей в залежності від вхідних даних	Власний алгоритм	Власний алгоритм			+
3	Якісна візуалізація даних	+	+	+		+	
4	Доступність та простота у використанні	Простий алгоритм, що дає результати на основі вхідних даних	Власний алгоритм, що вимагає попередньої підготовки користувача	Простий алгоритм, що дає результати на основі вхідних даних			+
5	Ціна	$10/місяць	$50/місяць	$25/місяць			+

4.2 Технологічний аудит проекту

Метою технічного аудиту є визначення переліку технологій, за допомогою який реалізована система і їх аналіз. Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (таблиця 4.3):

1) за допомогою якої технологією розроблена система згідно ідеї проекту;

2) чи існують у відкритому доступі ці технології, чи їх потрібно додатково розробляти або купувати;

3) чи має доступ розробник до описаних технологій.

Таблиця 4.3. Технологічна здійсненність ідеї проекту

№п/п	Ідея проекту	Технології реалізації	Наявність технологій	Достпуність технологій
1	Створення системи для найкращого вибору метода прогнозування	Авторегресійні моделі та рекурентні або згорткові нейронні мережі	Наявна	Доступна
Обраною мовою програмування є python, моделі створюються за допомогою наявних у мові засобів				

За результатами аналізу можна зробити висновок щодо можливості технологічної реалізації проекту. Технологічним шляхом реалізації проекту було обрано мову програмування python через її доступність та безкоштовність.

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Спочатку було проведено аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 4.4).

Таблиця 4.4 - Попередня характеристика потенційного ринку стартап проекту

№ п/п	Показники стану ринку(найменування)	Характеристика
1	Кількість систем-конкурентів на ринку, од	
2	Загальний обсяг продаж, грн/ум.од	
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає

Характеристика потенційних груп клієнтів та орієнтовний перелік вимог до товару для кожної групи наведені у таблиці 4.5.

Таблиця 4.5 - Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Наявність програмного забезпечення для короткострокового прогнозування	Трейдери-аналітики, брокерські компанії, інвестиційні компанії, індивідуальні трейдери та власники акцій	Цільові групи клієнтів мають різні доходи потенційний прибуток від застосування системи, різні очікування та мотивація	Висока точність побудованих ко-роткострокових прогнозів, простота та доступність у використанні, низька ціна системи.

Після дослідження потенційних категорій клієнтів було проведено дослідження ринкового середовища: складено таблиці факторів, що сприяють реалізації розробленої системи як стартап-проекту, та факторів, що йому заважають (табл. 4.6, 4.7).

Таблиця 4.6 - Фактори загроз

№п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Висока конкурентність на ринку	Поява на ринку більш зручних і кращих засобів для прогнозування	Покращення якісних характеристик системи прогнозування. Підвищення зручності та доступності
2	Недостатність даних	Точність прогнозів цін акцій залежить від доступу до якісних та актуальних даних. Проблеми з джерелами даних або їхньою якістю можуть призвести до неточностей в прогнозах.	Пошук та інтеграція нових джерел даних може допомогти збільшити обсяг доступних інформації. Встановлення партнерств і співпраця з іншими компаніями в галузі фінансів або аналітики можуть допомогти отримувати додатковий доступ до даних.

Таблиця 4.7 - Фактори можливостей

№п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Розвиток технологій	Швидкий розвиток штучного інтелекту та обробки природної мови створює нові можливості для покращення точності інструментів прогнозування	Впровадження нових ідей та розробок для покращення сервісу
2	Ринковий попит	Зростання інтересу до інвестицій в акції та фінансовий аналіз створює великий попит на інструменти, які надають точні та передбачувані прогнози цін акцій	Реклама сервісу для залучення більшої кількості клієнтів. Покращення зручності і доступності користування
3	Застосування гібридних моделей	Застосування гібридних моделей для підвищення точності отриманих прогнозів.	Витрачати ресурси на дослідження нових моделей для прогонзування.

Надалі було проведено дослідження пропозиції: визначили загальні характеристик конкуренції на ринку (таблиця 4.8).

Таблиця 4.8 — Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції - досконала	Багато незалежних компаній на рівних умовах конкурують одна з одною.	Потрібно створити конкурентноспроможний продукт, що задовольняє потреби користувача
2. За рівнем конкурентної боротьби - міжнародний	Компанії-конкуренти існують в різних країнах	Співпраця з іноземними спеціалістами, збільшення кількості мов, що підтримує система.
3.За галузевою ознакою - внутрішньогалузева	Багато компаній пропонують послуги в одній галузі.	Постійне вдосконалення системи прогнозування та функціоналу
4. Конкуренція за видами товарів: - товарно-видова	Конкуренція між товарами одного виду.	Створення продукту, враховуючи позитивні сторони та недоліки конкурентів.
5. За характером конкурентних переваг -нецінова	Якість моделей для прогнозування	Створення нових моделей для прогонзування, їх тестування з різними даними та на різних платформах. Вдосконалення вже наявних моделей
6. За інтенсивністю - не марочна	Торгова марка майже не впливає на попит	-

Проведено більш детальний аналіз умов конкуренції в галузі за М. Портером у таблиці 4.9.

Таблиця 4.9 - Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Навести перелік прямих конкурентів	Визначити бар'єри входження на ринок	Визначити фактори сили постачальників	Визначити фактори сили споживачів	Фактори загроз з боку замінників
	StocksNeural, Neural Trader	На ринку вже наявні схожі рішення	Відсутні фактори сили постачальників	Якість сервісу	Кращий функціонал, більша зручність, кращі результати
Висновок	Достатньо інтенсивна конкуренція на ринку із компаніями, що пропонують продукти для аналізу фондового ринку	Є можливість виходу на ринок, але наявна конкуренція	Відсутні фактори сили постачальників	Клієнти диктують умови на ринку, такі як: точність роботи, зручність інтерфейсу, доступність	Необхідно постійно вдосконалювати роботу сервісу і його елементи для взаємодії з користувачами

На основі аналізу конкуренції, проведеного в таблиці 4.9, а також із урахуванням характеристик ідеї проекту (таблиця 4.2), вимог споживачів до товару (таблиця 4.5) та факторів маркетингового середовища (таблиці 4.6, 4.7) визначається та обґрунтовується перелік факторів конкурентоспроможності, та наводиться у таблиці 4.10

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Функціонал сервісу	Сервіс має задовольняти потреби користувачів і давати їм можливість отримати бажаний результат
2	Інноваційність	Сервіс має постійно покращуватися, щоб мати перевагу у якості наданих послуг
3	Ціна	Ціна має бути доступною і привабливою у порівнянні з конкурентами

За визначеними факторами конкурентоспроможності (таблиця 4.10) проводиться аналіз сильних та слабких сторін стартап-проекту, що наведені у таблиці 4.11.

Таблиця 4.11 - Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з розробленою системою прогнозування						
			-3	-2	-1	0	1	2	3
1	Функціонал сервісу	18			+				
2	Інноваційність	14					+		
3	Ціна	16		+					

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання матриці SWOT-аналізу (сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities)). Матриця

SWOT-аналізу наведена у таблиці 4.12.

Таблиця 4.12 - SWOT- аналіз стартап-проекту

Сильні сторони: висока якість отриманих прогнозів, можливість навчання моделей, доступна ціна	Слабкі сторони: високий рівень конкуренції, відсутність кросплатформенності
Можливості: розширення функціоналу шляхом додавання моделей для прогнозування, вдосконалення вже наявних моделей, прогнозування інших фінансових даних	Загрози: швидке зростання конкуренції, зміна потреб користувачів, зміни в ринкових умовах

Визначені на основі SWOT-аналізу альтернативи ринкової поведінки для виведення проекту на ринок аналізуються з точки зору строків та ймовірності отримання ресурсів у таблиці 4.13.

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
Вихід на ринок із програмним продуктом з додатковим функціоналом	80%	6 місяців
Створення продукту для співпраці з фінансовими установами	85%	9 місяців
Розширення функціоналу продукту на інші платформи	70%	8 місяців

4.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим етапом передбачає визначення стратегії охоплення ринку: дослідження цільових груп потенційних клієнтів, які приведенні в таблиці 4.14.

Таблиця 4.14. Вибір цільових груп потенційних споживачів

Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтований попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу в сегмент
Брокерські компанії та біржеві аналітики	Середній, бо більшість брокерських компаній використовують програмні продукти відомих компаній	Високий	Висока	Висока складність
Інвестиційні компанії	Середній	Високий	Висока	Висока складність
Індивідуальні трейдери	Високий	Високий	Середня	Середня

У результаті будемо взаємодіяти з третьою цільовою групою. Для роботи в обраному сегменті ринку необхідно сформувати базову стратегію розвитку.

Особливості вибори стратегії відображено у таблиці 4.15.

Таблиця 4.15 - Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Вихід на ринок із програмним продуктом з додатковим функціоналом для моделювання, прогнозування та взаємодії з сервісом	Визначення та задоволення потреб цільового сегмента	Постійне оновлення то доопрацювання сервісу згідно з потребами	Стратегія диференціації

Обгрунтування вибору стратегії конкурентної поведінки наводиться у таблиці 4.16.

Таблиця 4.16 - Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Ні	Компанія буде шукати нових споживачів і конкурувати за вже існуючих	Компанія буде копіювати базову концепцію проекту	Стратегія зайняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до стартап-проекту та до продукту та в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки у таблиці 4.17 описуються шляхи розроблення стратегії позиціонування.

Таблиця 4.17 - Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати позицію власного проекту
Точність побудови прогнозів, Зручність сервісу, доступна ціна	Стратегія диференціації	Висока точність прогнозу, доступна ціна	Точний прогноз, зручність, доступність

4.5 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, що має одержати користувач. У таблиці 4.18 наведені підсумки аналізу конкурентоспроможності товару.

Таблиця 4.18 - Визначення ключових переваг концепції потенційного товару

Потреба	Вигода, яку пропонує товар	Ключові переваги над конкурентами (існуючі або такі, що потрібно створити)
Точний прогноз ціни акцій компанії	Висока точність прогнозування	Використання різних моделей для знаходження кращого прогнозу
Візуалізація даних	Якісна візуалізація даних для кращого розуміння прогнозу	Реалізовано виведення необхідних графіків

У таблиці 4.19 описано трирівневу маркетингову модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання.

Таблиця 4.19 - Опис трьох рівнів моделі товару

Рівні товару	Сутність і складові
I. Товар за задумом	Побудова прогнозів на фондовому ринку. Продукт має давати якісні прогнози, бути простим та швидким
II. Товар у реальному виконанні	Програмне забезпечення або хмарна платформа яке буде доступним для користування і буде надавати точний прогноз
III. Товар з підкріпленням	Постійне оновлення функціоналу, покращення роботи і результатів

У таблиці 4.20 проведено визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар, а також аналіз рівня доходів цільової групи користувачів продукту.

Таблиця 4.20 - Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
-	25-50$/місяць	1000$/місяць	10-25$/місяць

У таблиці 4.21 обґрунтовано визначення оптимальної системи збуту, в межах якого приймається рішення.

Таблиця 4.21 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Інвестиційні компанії, брокерські компанії: використання річної підписки з додатковими послугами	Продаж	Нульовий рівень (напряму) та перший (через посередника)	Багатоканальні системи
Індивідуальні трейдери: більш доступна місячна підписка	Продаж	Нульовий та перший	Багатоканальні системи

Розроблення концепції маркетингових комунікацій проводиться у таблиці 4.22.

Таблиця 4.22 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуютьс я цільові клієнти	Ключові позиції, обрані для позиціонуванн я	Завдання рекламного повідомлення	Концепція рекламног о звернення
Інвестиційні компанії, брокерські компанії: покупка за допомогою веб-сервісів	Веб-сайти, торгові майданчики	Точність прогнозу, доступна ціна	Переконання Оптимальност і продукту для бізнесу клієнта	Якісні прогнози для отримання доходу на фондовому ринку
Індивідуальн і трейдери: покупка за допомогою веб-сервісів	Соціальні мережі, веб-сайти, торгові майданчики	Точність прогнозу, доступність для користування, ціна	Наголошення на перевагах продукту перед конкурентами	Якісні прогнози для отримання доходу на фондовому ринку

4.6 Висновки до розділу

Було проведено аналіз створеної системи для порівняння моделей прогнозування як стартап-проекту. Зроблено висновок, що проект має можливість ринкової комерціалізації, через те, що ринок систем прогнозування потребує якісного та інноваційного продукту для вибору найкращого методу прогнозування в різних областях.

Результатом роботи є розроблений стартап-проект, план виходу на ринки програмного забезпечення та маркетингова стратегія. Розроблений стартап-проект доцільно застосувати при комерціалізації розробки.

Існує декілька перспектив входження на ринок та орієнтування на декілька цільових груп, ключовими конкурентоспроможними позиціями проекту є висока точність побудови прогнозу, якісна візуалізація, доступність сервісу.

Для реалізації проекту доцільно обрати альтернативу виходу на ринок із програмним продуктом з додатковим функціоналом для моделювання, прогнозування та взаємодії з сервісом. Подальший розвиток проекту є доцільним, оскільки є висока можливість виходу на ринок.

ВИСНОВКИ

У магістерській дисертації було проведено порівняльний аналіз методів прогнозування для нестаціонарних і нелінійних процесів фондових ринків. Було зроблено огляд моделей прогнозування, досліджено їх результати.

Для порівняння прогнозів, були розглянуті різні математичні методи та моделі машинного навчання, які використовуються для вирішення практичних завдань аналізу та прогнозування нестаціонарних процесів. Такими є модель авторегресії інтегрованого ковзного середнього ARIMA, LSTM, N-BEATS та ансамблева модель. Користувач має змогу подивитись на параметри моделі, порівняти результати і вибрати найкращу модель.

Для порівняння моделей була реалізована програма що дозволяє побудову структури даних моделей, знаходження прогнозу та його результатів. Як об’єкт експериментальних досліджень було обрано акції компанії Apple, та найкращі результати показала модель авторегресійної інтегрованої ковзної середньої, яка дозволила отримати дуже якісні прогнози. Інші моделі показали схожі результати, тому прогнози нейромереж можна також вважати достатньо точними.

У четвертому розділі був представлений стартап-проект, спрямований на впровадження програми, що ґрунтується на виборі найкращої моделі прогнозування.

Реалізація та порівняння прогнозуючих моделей, представлених у магістерській дисертації, призначені для вирішення складних, але вельми важливих бізнес-проблем. Це дозволяє ефективно вести інвестиційну політику та впроваджувати штучний інтелект у різних галузях.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бідюк П. І., Романенко В. Д., Тимощук О. Л. Аналіз часових рядів: навч. посіб. ННК «Інститут прикладного системного аналізу» Національний технічний університет України «Київський політехнічний інститут». 2010. 317 с.

2. Бідюк П. І. Економетричний аналіз часових рядів. Київ: Політехніка, 2007. 250 с.

3. Канторович Г.Г. Анализ временных рядов: лекционные и методические материалы. М.: Экономический журнал ВШЭ, 2002. 129 с.

4. Бідюк П. І. Часові ряди: моделювання і прогнозування: монографія. Київ: ЕКМО, 2003. 144 с.

5. Магнус Я.Р., Катышев П.К., Пересецкий А.А., Магнус Я.Р.Эконометрика: Начальный курс: учеб. 6-е изд., перед. и доп. М.: Дело, 2004. 576 с.

6. Вербик М. Путеводитель по современной эконометрике / пер. С англ. Банникова В.А.; научн. ред. и пред. Айвазяна С.А. М.: Научная книга, 2008. 616 с.

7. Бокс Дж., Дженкинс Г. Анализ временных рядов, прогноз и управление / пер.с англ. М.: Мир, 1974. 406 с.

8. Simonyan K. Very Deep Convolutional Networks for Large-Scal Image Recognition URL: http://arxiv.org/abs/1409.1556

9. Oreshkin B. N., Chapados N., Carpov D., Bengio Y. N-BEATS:Neural basis expansion analysis for interpretable time series forecasting URL: https://arxiv.org/pdf/1905.10437

10. Luke B. Godfrey, Michael S. Gashler Neural Decomposition of Time-Series Data for Effective Generalization URL: https://arxiv.org/pdf/1705.09137

11. Daniel L. Marino, Kasun Amarasinghe, Milos Manic Building Energy Load Forecasting using Deep Neural Networks URL: https://arxiv.org/ftp/arxiv/papers/1610/1610.09460

12. Soofi A.S., Liangyue C. Modelling and Forecasting Financial Data. Techniques of Nonlinear Dynamics. Boston: Springer US. 2002. 488 p.

13. Mujtaba S. M., Nadeem M. Analyzing Stock Markets using Data Warehousing. Journal of Independent Studies and Research. Jan. 2006. Vol. 4. P. 8.

14. Mondal D. A., Maji G., Goto T., Debnath N.C., Sen S. Data Warehouse Based Modelling Technique for Stock Market Analysis. International Journal of Engineering & Technology. 2018. Vol. 7 (3.13). P. 165-170.

15. Selva Prabhakaran ARIMA Model – Complete Guide to Time Series Forecasting in Python URL: https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/

16. Макухін Є.І., Макаренко О.С., Бідюк П.І. Порівняльний аналіз моделей для методів прогнозування. II Всеукраїнська науково-практична конференція «Системні науки та інформатика», 4–8 грудня 2023 р., Київ: КПІ ім. Ігоря Сікорського, 2023. С. 172-180.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
plt.style.use("fivethirtyeight")
%matplotlib inline

# For reading stock data from yahoo
from pandas_datareader.data import DataReader
import yfinance as yf
from pandas_datareader import data as pdr

yf.pdr_override()

# For time stamps
from datetime import datetime
import math
from scipy import stats
import matplotlib.dates as mdates
from plotly import tools
import plotly.tools as tls
import plotly.figure_factory as ff
import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, plot, iplot
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
```

```
from sklearn.metrics import mean_squared_error
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import pmdarima as pm
from pmdarima.arima.utils import ndiffs



# Get the stock quote
df = pdr.get_data_yahoo('AAPL', start='2012-01-01', end='2023-10-13')
# Save the data to a CSV file
csv_filename = "apple_stock_data.csv"
df.to_csv(csv_filename)
df = pd.read_csv("apple_stock_data.csv",
                 parse_dates=["Date"],
                 index_col=["Date"])
df.describe()
df.head()
#plot close price
plt.figure(figsize=(10,6))
plt.grid(True)
plt.xlabel('Date')
plt.ylabel('Close Prices')
plt.plot(df['Close'])
plt.title('Price of Apple stock from 3 Jan 2012')
plt.show()
#Test for staionarity
def test_stationarity(timeseries):
    #Determing rolling statistics
    rolmean = timeseries.rolling(12).mean()
    rolstd = timeseries.rolling(12).std()
    #Plot rolling statistics:
    plt.plot(timeseries, color='blue',label='Original')
    plt.plot(rolmean, color='red', label='Rolling Mean')
```

```
    plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean and Standard Deviation')
    plt.show(block=False)

    print("Results of dickey fuller test")
    adft = adfuller(timeseries,autolag='AIC')
    # output for dft will give us without defining what the values are.
    #hence we manually write what values does it explains using a for loop
    output = pd.Series(adft[0:4],index=['Test Statistics','p-value','No. of lags used','Number of observations used'])
    for key,values in adft[4].items():
        output['critical value (%s)'%key] =  values
    print(output)

test_stationarity(df['Close'])
# define function to return copies of stock dataframe with moving averages
def mav_function(df):

    # calclulate moving averages of 10,50 and 30 days
    df['10_d_avg'] = df['Close'].rolling(window=10).mean()
    df['20_d_avg'] = df['Close'].rolling(window=20).mean()
    df['30_d_avg'] = df['Close'].rolling(window=30).mean()

    return df

# let's analyse stocks using moving averages methods
df = mav_function(df)
# plot moving avearges chart
dfn = df.copy()

def mav_chart(df):
    fig = tls.make_subplots(rows=2, cols=1, shared_xaxes=True)
```

```
    # set colors and cols names to be plotted
    colors = ['#ff4500', '#92a1cf', '#6E6E6E']
    avgs = ['10_d_avg','20_d_avg','30_d_avg']


    for col, c in zip(avgs, colors):
        fig.append_trace({'x': df.index, 'y': df[col], 'type': 'scatter', 'name': col, 'line': {'color': c}},1,1)
    for col in ['Close']:
        fig.append_trace({'x': df.index, 'y': df[col], 'type': 'scatter', 'name': 'closing price', 'line': {'color':
'#393f5e'}},2,1)


    fig['layout'].update(height=800,title=f"Relationship between Moving averages <br> and Closing
Price",
                paper_bgcolor='#4bd659', plot_bgcolor='#F2DFCE')
    fig.show()
mav_chart(dfn)
# create function to return dataframe for forecating
def df_formatting(df):
    df = df.loc[:, ['Close']]
    df.rename(columns={'Close':'y'}, inplace=True)


    return df


ph_df = df_formatting(df)
#To separate the trend and the seasonality from a time series,
# we can decompose the series using the following code.
result = seasonal_decompose(ph_df, model='multiplicative', period = 365)
fig = plt.figure()
fig = result.plot()
fig.set_size_inches(16, 9)
# function to return acf and pacf plots
def acf_pacf(df, lags):
    var = df['Close']
    # plot the acf plot
    fig = plot_acf(var, lags=lags)
```

```
    fig.set_size_inches((9, 5))
    fig.tight_layout()
    plt.show()

    # plot the pacf plots
    fig = plot_pacf(var, lags=lags)
    fig.set_size_inches((9,5))
    fig.tight_layout()
    plt.show()
# acf and pacf of aapl stock
acf_pacf(df, 50)
# Split the data into training and testing sets
def arima_split(df):
    size = int(len(df)*0.9)
    train_df = (df['y'])[:size]
    test_df = (df['y'])[size:]

    print(f"Train Size: {len(train_df)}, Test Size: {len(test_df)}")
    print("-------------------------------")

    return train_df, test_df

train_data, test_data = arima_split(ph_df)
plt.figure(figsize=(10,6))
plt.grid(True)
plt.xlabel('Dates')
plt.ylabel('Closing Prices')
plt.plot(train_data, 'blue', label='Train data')
plt.plot(test_data, 'green', label='Test data')
plt.legend()
# Fit an ARIMA model to the training data
model = pm.auto_arima(train_data, test = 'adf',
                      start_p = 1, start_q = 1,
```

```
                max_p = 3, max_q = 3,
                d = None, seasonal = False,
                start_P = 0, m = 1,
                trace = True, error_action = 'ignore',
                suppress_warnings = True, stepwise = True,
                D = 1, information_criterion = 'aic')
model.plot_diagnostics(figsize=(15,8))
plt.show()
model.summary()
# Make predictions for the test set with a 95% confidence interval
forecast, conf_int = model.predict(n_periods=len(test_data), return_conf_int=True, alpha=0.05)
# Plot the training data, test data, and forecast data with the 95% confidence interval
plt.figure(figsize=(12, 6))
plt.plot(train_data, label='Train Data')
plt.plot(test_data, label='Test Data')
plt.plot(test_data.index, forecast, label='Forecast', color='green')
plt.fill_between(test_data.index, conf_int[:, 0], conf_int[:, 1], color='pink', alpha=0.3, label='95%
Confidence Interval')
plt.title('Stock Price Forecast with 95% Confidence Interval')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
# MASE implemented courtesy of sktime - https://github.com/alan-turing-
institute/sktime/blob/ee7a06843a44f4aaec7582d847e36073a9ab0566/sktime/performance_metrics/for
ecasting/_functions.py#L16
def mean_absolute_scaled_error(y_true, y_pred):
  """
  Implement MASE (assuming no seasonality of data).
  """
  mae = tf.reduce_mean(tf.abs(y_true - y_pred))


  # Find MAE of naive forecast (no seasonality)
```

```
  mae_naive_no_season = tf.reduce_mean(tf.abs(y_true[1:] - y_true[:-1])) # our seasonality is 1 day (hence the shifting of 1 day)


  return mae / mae_naive_no_season
def evaluate_preds(y_true, y_pred):
  # Make sure float32 (for metric calculations)
  y_true = tf.cast(y_true, dtype=tf.float32)
  y_pred = tf.cast(y_pred, dtype=tf.float32)


  # Calculate various metrics
  mae = tf.keras.metrics.mean_absolute_error(y_true, y_pred)
  mse = tf.keras.metrics.mean_squared_error(y_true, y_pred) # puts and emphasis on outliers (all errors get squared)
  rmse = tf.sqrt(mse)
  mape = tf.keras.metrics.mean_absolute_percentage_error(y_true, y_pred)
  mase = mean_absolute_scaled_error(y_true, y_pred)


  return {"mae": mae.numpy(),
          "mse": mse.numpy(),
          "rmse": rmse.numpy(),
          "mape": mape.numpy(),
          "mase": mase.numpy()}
model_1_results = evaluate_preds(y_true=test_data,
                                 y_pred=forecast)
model_1_results
# Get the stock quote
df = pdr.get_data_yahoo('AAPL', start='2012-01-01', end='2023-10-13')
# Save the data to a CSV file
csv_filename = "apple_stock_data.csv"
df.to_csv(csv_filename)
df = pd.read_csv("apple_stock_data.csv",
                 parse_dates=["Date"],
                 index_col=["Date"])
# Only want closing price for each day
```

```
apple_prices = pd.DataFrame(df["Close"]).rename(columns={"Close": "Price"})
apple_prices.head()
# Get apple date array
timesteps = apple_prices.index.to_numpy()
prices = apple_prices["Price"].to_numpy()

timesteps[:10], prices[:10]
# Create train and test splits the right way for time series data
split_size = int(0.8 * len(prices)) # 80% train, 20% test

# Create train data splits (everything before the split)
X_train, y_train = timesteps[:split_size], prices[:split_size]

# Create test data splits (everything after the split)
X_test, y_test = timesteps[split_size:], prices[split_size:]

len(X_train), len(X_test), len(y_train), len(y_test)
# Create a function to plot time series data
def plot_time_series(timesteps, values, format='.', start=0, end=None, label=None):
  """
  Plots a timesteps (a series of points in time) against values (a series of values across timesteps).

  Parameters
  ---------
  timesteps : array of timesteps
  values : array of values across time
  format : style of plot, default "."
  start : where to start the plot (setting a value will index from start of timesteps & values)
  end : where to end the plot (setting a value will index from end of timesteps & values)
  label : label to show on plot of values
  """
  # Plot the series
  plt.plot(timesteps[start:end], values[start:end], format, label=label)
```

```
  plt.xlabel("Time")
  plt.ylabel("Apple Price")
  if label:
    plt.legend(fontsize=14) # make label bigger
  plt.grid(True)
HORIZON = 1 # predict 1 step at a time
WINDOW_SIZE = 7 # use a week worth of timesteps to predict the horizon
# Create function to label windowed data
def get_labelled_windows(x, horizon=1):
  """
  Creates labels for windowed dataset.

  E.g. if horizon=1 (default)
  Input: [1, 2, 3, 4, 5, 6] -> Output: ([1, 2, 3, 4, 5], [6])
  """
  return x[:, :-horizon], x[:, -horizon:]
# Create function to view NumPy arrays as windows
def make_windows(x, window_size=7, horizon=1):
  """
  Turns a 1D array into a 2D array of sequential windows of window_size.
  """
  # 1. Create a window of specific window_size (add the horizon on the end for later labelling)
  window_step = np.expand_dims(np.arange(window_size+horizon), axis=0)
  # print(f"Window step:\n {window_step}")

  # 2. Create a 2D array of multiple window steps (minus 1 to account for 0 indexing)
  window_indexes = window_step + np.expand_dims(np.arange(len(x)-(window_size+horizon-1)),
axis=0).T # create 2D array of windows of size window_size
  # print(f"Window indexes:\n {window_indexes[:3], window_indexes[-3:],
window_indexes.shape}")

  # 3. Index on the target array (time series) with 2D array of multiple window steps
  windowed_array = x[window_indexes]
```

```
  # 4. Get the labelled windows
  windows, labels = get_labelled_windows(windowed_array, horizon=horizon)


  return windows, labels
full_windows, full_labels = make_windows(prices, window_size=WINDOW_SIZE,
horizon=HORIZON)
len(full_windows), len(full_labels)
# Make the train/test splits
def make_train_test_splits(windows, labels, test_split=0.2):
  """
  Splits matching pairs of windows and labels into train and test splits.
  """
  split_size = int(len(windows) * (1-test_split)) # this will default to 80% train/20% test
  train_windows = windows[:split_size]
  train_labels = labels[:split_size]
  test_windows = windows[split_size:]
  test_labels = labels[split_size:]
  return train_windows, test_windows, train_labels, test_labels
train_windows, test_windows, train_labels, test_labels = make_train_test_splits(full_windows,
full_labels)
len(train_windows), len(test_windows), len(train_labels), len(test_labels)
import os


# Create a function to implement a ModelCheckpoint callback with a specific filename
def create_model_checkpoint(model_name, save_path="model_experiments"):
  return tf.keras.callbacks.ModelCheckpoint(filepath=os.path.join(save_path, model_name), # create
filepath to save model
                                            verbose=0, # only output a limited amount of text
                                            save_best_only=True) # save only the best model to file
def make_preds(model, input_data):
  """
  Uses model to make predictions on input_data.


  Parameters
```

```
  ----------
  model: trained model
  input_data: windowed input data (same kind of data model was trained on)

  Returns model predictions on input_data.
  """
  forecast = model.predict(input_data)
  return tf.squeeze(forecast) # return 1D array of predictions
from tensorflow.keras import layers
tf.random.set_seed(42)

# Let's build an LSTM model with the Functional API
inputs = layers.Input(shape=(WINDOW_SIZE))
x = layers.Lambda(lambda x: tf.expand_dims(x, axis=1))(inputs) # expand input dimension to be compatible with LSTM
x = layers.LSTM(128, activation="relu")(x) # using the tanh loss function results in a massive error
# x = layers.Dense(32, activation="relu")(x)
output = layers.Dense(HORIZON)(x)
model_2 = tf.keras.Model(inputs=inputs, outputs=output, name="model_2_lstm")

# Compile model
model_2.compile(loss="mae",
                optimizer=tf.keras.optimizers.Adam())

# Seems when saving the model several warnings are appearing: https://github.com/tensorflow/tensorflow/issues/47554
model_2.fit(train_windows,
            train_labels,
            epochs=100,
            verbose=0,
            batch_size=128,
            validation_data=(test_windows, test_labels),
            callbacks=[create_model_checkpoint(model_name=model_2.name)])
# Load in best version of model 2 and evaluate on the test data
```

```
model_2 = tf.keras.models.load_model("model_experiments/model_2_lstm/")
model_2.evaluate(test_windows, test_labels)
# Make predictions with our LSTM model
model_2_preds = make_preds(model_2, test_windows)
model_2_preds[:10]
# Evaluate model 2 preds
model_2_results = evaluate_preds(y_true=tf.squeeze(test_labels),
                                 y_pred=model_2_preds)
model_2_results
# Evaluate model 2 preds
model_2_results = evaluate_preds(y_true=tf.squeeze(test_labels),
                                 y_pred=model_2_preds)
model_2_results
# Plot Lstm forecast
plt.figure(figsize=(12, 7))
plot_time_series(timesteps=X_train, values=y_train, format="-", label="Train data")
plot_time_series(timesteps=X_test, values=y_test, format="-", label="Test data")
plot_time_series(timesteps=X_test[1:], values=model_2_preds, format="-", label="LSTM results");
# Create NBeatsBlock custom layer
class NBeatsBlock(tf.keras.layers.Layer):
  def __init__(self, # the constructor takes all the hyperparameters for the layer
               input_size: int,
               theta_size: int,
               horizon: int,
               n_neurons: int,
               n_layers: int,
               **kwargs): # the **kwargs argument takes care of all of the arguments for the parent class (input_shape, trainable, name)
    super().__init__(**kwargs)
    self.input_size = input_size
    self.theta_size = theta_size
    self.horizon = horizon
    self.n_neurons = n_neurons
```

```
    self.n_layers = n_layers

    # Block contains stack of 4 fully connected layers each has ReLU activation
    self.hidden = [tf.keras.layers.Dense(n_neurons, activation="relu") for _ in range(n_layers)]
    # Output of block is a theta layer with linear activation
    self.theta_layer = tf.keras.layers.Dense(theta_size, activation="linear", name="theta")

  def call(self, inputs): # the call method is what runs when the layer is called
    x = inputs
    for layer in self.hidden: # pass inputs through each hidden layer
      x = layer(x)
    theta = self.theta_layer(x)
    # Output the backcast and forecast from theta
    backcast, forecast = theta[:, :self.input_size], theta[:, -self.horizon:]
    return backcast, forecast
HORIZON = 1 # how far to predict forward
WINDOW_SIZE = 7 # how far to lookback
# Create NBEATS data inputs (NBEATS works with univariate time series)
apple_prices.head()
# Add windowed columns
apple_prices_nbeats = apple_prices.copy()
for i in range(WINDOW_SIZE):
  apple_prices_nbeats[f"Price+{i+1}"] = apple_prices_nbeats["Price"].shift(periods=i+1)
apple_prices_nbeats.dropna().head()
# Make features and labels
X = apple_prices_nbeats.dropna().drop("Price", axis=1)
y = apple_prices_nbeats.dropna()["Price"]

# Make train and test sets
split_size = int(len(X) * 0.8)
X_train, y_train = X[:split_size], y[:split_size]
X_test, y_test = X[split_size:], y[split_size:]
len(X_train), len(y_train), len(X_test), len(y_test)
```

```
# 1. Turn train and test arrays into tensor Datasets
train_features_dataset = tf.data.Dataset.from_tensor_slices(X_train)
train_labels_dataset = tf.data.Dataset.from_tensor_slices(y_train)

test_features_dataset = tf.data.Dataset.from_tensor_slices(X_test)
test_labels_dataset = tf.data.Dataset.from_tensor_slices(y_test)

# 2. Combine features & labels
train_dataset = tf.data.Dataset.zip((train_features_dataset, train_labels_dataset))
test_dataset = tf.data.Dataset.zip((test_features_dataset, test_labels_dataset))

# 3. Batch and prefetch for optimal performance
BATCH_SIZE = 1024 # taken from Appendix D in N-BEATS paper
train_dataset = train_dataset.batch(BATCH_SIZE).prefetch(tf.data.AUTOTUNE)
test_dataset = test_dataset.batch(BATCH_SIZE).prefetch(tf.data.AUTOTUNE)

train_dataset, test_dataset
# Values from N-BEATS paper Figure 1 and Table 18/Appendix D
N_EPOCHS = 5000 # called "Iterations" in Table 18
N_NEURONS = 512 # called "Width" in Table 18
N_LAYERS = 4
N_STACKS = 30

INPUT_SIZE = WINDOW_SIZE * HORIZON # called "Lookback" in Table 18
THETA_SIZE = INPUT_SIZE + HORIZON

INPUT_SIZE, THETA_SIZE
# Make tensors
tensor_1 = tf.range(10) + 10
tensor_2 = tf.range(10)

# Subtract
subtracted = layers.subtract([tensor_1, tensor_2])
```

```
# Add
added = layers.add([tensor_1, tensor_2])
%%time

tf.random.set_seed(42)

# 1. Setup N-BEATS Block layer
nbeats_block_layer = NBeatsBlock(input_size=INPUT_SIZE,
                                 theta_size=THETA_SIZE,
                                 horizon=HORIZON,
                                 n_neurons=N_NEURONS,
                                 n_layers=N_LAYERS,
                                 name="InitialBlock")

# 2. Create input to stacks
stack_input = layers.Input(shape=(INPUT_SIZE), name="stack_input")

# 3. Create initial backcast and forecast input (backwards predictions are referred to as residuals in the paper)
backcast, forecast = nbeats_block_layer(stack_input)
# Add in subtraction residual link, thank you to: https://github.com/mrdbourke/tensorflow-deep-learning/discussions/174
residuals = layers.subtract([stack_input, backcast], name=f"subtract_00")

# 4. Create stacks of blocks
for i, _ in enumerate(range(N_STACKS-1)): # first stack is already creted in (3)

  # 5. Use the NBeatsBlock to calculate the backcast as well as block forecast
  backcast, block_forecast = NBeatsBlock(
      input_size=INPUT_SIZE,
      theta_size=THETA_SIZE,
      horizon=HORIZON,
      n_neurons=N_NEURONS,
```

```
    n_layers=N_LAYERS,
    name=f"NBeatsBlock_{i}"
  )(residuals) # pass it in residuals (the backcast)

  # 6. Create the double residual stacking
  residuals = layers.subtract([residuals, backcast], name=f"subtract_{i}")
  forecast = layers.add([forecast, block_forecast], name=f"add_{i}")

# 7. Put the stack model together
model_3 = tf.keras.Model(inputs=stack_input,
                         outputs=forecast,
                         name="model_3_N-BEATS")

# 8. Compile with MAE loss and Adam optimizer
model_3.compile(loss="mae",
                optimizer=tf.keras.optimizers.Adam(0.001),
                metrics=["mae", "mse"])

# 9. Fit the model with EarlyStopping and ReduceLROnPlateau callbacks
model_3.fit(train_dataset,
            epochs=N_EPOCHS,
            validation_data=test_dataset,
            verbose=0, # prevent large amounts of training outputs
            # callbacks=[create_model_checkpoint(model_name=stack_model.name)] # saving model every epoch consumes far too much time
            callbacks=[tf.keras.callbacks.EarlyStopping(monitor="val_loss", patience=200, restore_best_weights=True),
                       tf.keras.callbacks.ReduceLROnPlateau(monitor="val_loss", patience=100, verbose=1)])
# Evaluate N-BEATS model on the test dataset
model_3.evaluate(test_dataset)
# Make predictions with N-BEATS model
model_3_preds = make_preds(model_3, test_dataset)
model_3_preds[:10]
# Evaluate N-BEATS model predictions
```

```
model_3_results = evaluate_preds(y_true=y_test,
                                 y_pred=model_3_preds)
model_3_results
# Extract the training, test, and predicted data
train_data = y_train.values  # Assuming y_train is a Pandas Series
test_data = y_test.values  # Assuming y_test is a Pandas Series
predicted_data = model_3_preds  # Assuming model_3_preds is a TensorFlow tensor

# Extract the dates from your dataset (assuming you have a 'Date' column)
dates = apple_prices_nbeats.index

# Create the plot
plt.figure(figsize=(12, 6))
plt.plot(dates[:len(train_data)], train_data, label="Train Data", linewidth=2)
plt.plot(dates[len(train_data):len(train_data) + len(test_data)], test_data, label="Test Data", linewidth=2)
plt.plot(dates[len(train_data):len(train_data) + len(test_data)], predicted_data, label="N-Beats results", linewidth=2)
plt.xlabel("Dates")
plt.ylabel("Apple Price")
plt.legend()
plt.grid(True)
plt.show()
def get_ensemble_models(horizon=HORIZON,
                        train_data=train_dataset,
                        test_data=test_dataset,
                        num_iter=10,
                        num_epochs=100,
                        loss_fns=["mae", "mse", "mape"]):
  """
  Returns a list of num_iter models each trained on MAE, MSE and MAPE loss.

  For example, if num_iter=10, a list of 30 trained models will be returned:
  10 * len(["mae", "mse", "mape"]).
```

```
  """
  # Make empty list for trained ensemble models
  ensemble_models = []

  # Create num_iter number of models per loss function
  for i in range(num_iter):
    # Build and fit a new model with a different loss function
    for loss_function in loss_fns:
      print(f"Optimizing model by reducing: {loss_function} for {num_epochs} epochs, model number: {i}")

      # Construct a simple model (similar to model_1)
      model = tf.keras.Sequential([
        # Initialize layers with normal (Gaussian) distribution so we can use the models for prediction
        # interval estimation later: https://www.tensorflow.org/api_docs/python/tf/keras/initializers/HeNormal
        layers.Dense(128, kernel_initializer="he_normal", activation="relu"),
        layers.Dense(128, kernel_initializer="he_normal", activation="relu"),
        layers.Dense(HORIZON)
      ])

      # Compile simple model with current loss function
      model.compile(loss=loss_function,
                    optimizer=tf.keras.optimizers.Adam(),
                    metrics=["mae", "mse"])

      # Fit model
      model.fit(train_data,
                epochs=num_epochs,
                verbose=0,
                validation_data=test_data,
                # Add callbacks to prevent training from going/stalling for too long
                callbacks=[tf.keras.callbacks.EarlyStopping(monitor="val_loss",
                                                            patience=200,
```

```
                                        restore_best_weights=True),
                                tf.keras.callbacks.ReduceLROnPlateau(monitor="val_loss",
                                        patience=100,
                                        verbose=1)])

    # Append fitted model to list of ensemble models
    ensemble_models.append(model)

  return ensemble_models # return list of trained models
%%time
# Get list of trained ensemble models
ensemble_models = get_ensemble_models(num_iter=5,
                                num_epochs=1000)
# Create a function which uses a list of trained models to make and return a list of predictions
def make_ensemble_preds(ensemble_models, data):
  ensemble_preds = []
  for model in ensemble_models:
    preds = model.predict(data) # make predictions with current ensemble model
    ensemble_preds.append(preds)
  return tf.constant(tf.squeeze(ensemble_preds))
# Create a list of ensemble predictions
ensemble_preds = make_ensemble_preds(ensemble_models=ensemble_models,
                                data=test_dataset)
ensemble_preds
# Evaluate ensemble model(s) predictions
ensemble_results = evaluate_preds(y_true=y_test,
                                y_pred=np.median(ensemble_preds, axis=0)) # take the median across all
ensemble predictions
ensemble_results
# Find upper and lower bounds of ensemble predictions
def get_upper_lower(preds): # 1. Take the predictions of multiple randomly initialized deep learning
neural networks

  # 2. Measure the standard deviation of the predictions
```

```
  std = tf.math.reduce_std(preds, axis=0)


  # 3. Multiply the standard deviation by 1.96
  interval = 1.96 * std # https://en.wikipedia.org/wiki/1.96
  # 4. Get the prediction interval upper and lower bounds
  preds_mean = tf.reduce_mean(preds, axis=0)
  lower, upper = preds_mean - interval, preds_mean + interval
  return lower, upper
# Get the upper and lower bounds of the 95%
lower, upper = get_upper_lower(preds=ensemble_preds)
# Get the median values of our ensemble preds
ensemble_median = np.median(ensemble_preds, axis=0)
# Plot the median of our ensemble preds along with the prediction intervals (where the predictions fall between)
offset=500
plt.figure(figsize=(12, 7))
plt.plot(X_test.index[offset:], y_test[offset:], "g", label="Test Data")
plt.plot(X_test.index[offset:], ensemble_median[offset:], "k-", label="Ensemble Median")
plt.xlabel("Date")
plt.ylabel("Apple Price")
plt.fill_between(X_test.index[offset:],
                 (lower)[offset:],
                 (upper)[offset:], label="Prediction Intervals")
plt.legend(loc="upper left", fontsize=14);
# Compare different model results (w = window, h = horizon, e.g. w=7 means a window size of 7)
model_results = pd.DataFrame({"model_1_arima": model_1_results,
                              "model_2_LSTM": model_2_results,
                              "model_3_NBEATs": model_3_results,
                              "model_4_ensemble": ensemble_results}).T
model_results.head(4)
# Sort model results by MAE and plot them
model_results[["mae"]].sort_values(by="mae").plot(figsize=(10, 7), kind="bar");
```