НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

До захисту допущено:
завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз управління»
спеціальності 124 «Системний аналіз»
на тему: «Розробка комплексної моделі інвестування на ринку криптовалют з використанням часових рядів та нейронних мереж**»**

Виконав:
студент IV курсу, групи КА-92
Маринич Антон Юрійович _____

Керівник: професор кафедри ММСА,
д.т.н., проф. Кузнєцова Н.В. _____

Консультант з нормоконтролю:
Доцент, к.т.н. Статкевич В.М. _____

Консультант з економічного розділу:
Доцент, к.е.н. Рощина Н.В. _____

Рецензент:
Безносик О.Ю.

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

Київ – 2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.Л. Тимощук
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Мариничу Антону Юрійовичу

1. Тема роботи «Розробка комплексної моделі інвестування на ринку криптовалют з використанням часових рядів та нейронних мереж», керівник роботи Кузнєцова Н.В., затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи__.06.2023.

3. Вихідні дані до роботи:

Історичні дані про ціни криптовалют

4. Зміст роботи:

Аналіз та прогнозування цін криптовалют за допомогою фільтрів, нейронних мереж, часових рядів

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н.В., доцент		

7. Дата видачі завдання:_____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Затвердження теми ДР	17.04.2023 – 23.04.2023	виконано
2	Ознайомлення зі структурою БДР згідно з Положенням про державну атестацію студентів НТУУ «КПІ ім. І. Сікорського»	24.04.2023 – 25.04.2023	виконано
3	Ознайомлення з ДСТУ 3008-95 та стандарти ЄСПД	26.04.2023 – 30.04.2023	виконано
4	Проведення дослідження за темою БДР під керівництвом керівника	01.05.2023 – 03.05.2023	виконано
5	Завершення роботи над першим варіантом частини БДР	04.05.2023 – 10.05.2023	виконано
6	Проведення роботи над експериментальною частиною БДР	11.05.2023 – 18.05.2023	виконано
7	Проведення роботи над програмним продуктом	19.05.2023 – 26.05.2023	виконано
8	Оформлення БДР та аналіз отриманих результатів	26.05.2023 – 31.05.2023	виконано

Студент Антон МАРИНИЧ
Керівник Наталія КУЗНЄЦОВА

РЕФЕРАТ

Дипломна робота: 88 с., 19 рис., 1 дод., 17 джерел.

ФІЛЬТРАЦІЯ ДАНИХ, ЧАСОВІ РЯДИ, НЕЙРОННІ МЕРЕЖІ, КОМПЛЕКСНА МОДЕЛЬ ІНВЕСТУВАННЯ

Тема: Розробка комплексної моделі інвестування на ринку криптовалют з використанням часових рядів та нейронних мереж.

Об’єкт дослідження: інвестиційна діяльність на ринку криптовалют з використанням аналізу часових рядів та методів нейронних мереж.

Предмет дослідження: розробка комплексної моделі інвестування, яка базується на аналізі часових рядів цін криптовалют та використанні нейронних мереж для прогнозування майбутніх цін.

Мета роботи: дослідити можливості використання часових рядів та нейронних мереж для розробки ефективної моделі інвестування на ринку криптовалют, здатної прогнозувати майбутні цінові зміни з високою точністю.

Методи дослідження: аналіз часових рядів, використання нейронних мереж для прогнозування, статистичний аналіз даних.

Актуальність: зростаюча популярність ринку криптовалют та його висока волатильність створюють нові можливості для інвесторів. Розробка ефективної моделі інвестування на основі аналізу часових рядів та нейронних мереж може допомогти інвесторам зробити кращі рішення щодо торгівлі.

Результати роботи: був розроблений програмний продукт на мові програмування Python, який використовує аналіз часових рядів та нейронних мереж для прогнозування майбутніх цін. Проведені експерименти та тестування показали, що модель має високу точність в короткостроковому прогнозуванні цін та може бути ефективно використана в інвестиційній діяльності.

ABSTRACT

Diploma thesis: 88 p., 19 figures, 1 appendix, 17 sources.

DATA FILTERING, TIME SERIES, NEURAL NETWORKS, COMPLEX INVESTMENT MODEL

Theme: Development of a comprehensive model of investment in the cryptocurrency market using time series and neural networks.

Object of research: investment activity in the cryptocurrency market using time series analysis and neural network methods.

Subject of research: development of a comprehensive investment model based on the analysis of time series of cryptocurrency prices and the use of neural networks to predict future price trends.

Purpose: explore the possibilities of using time series and neural networks to develop an effective investment model in the cryptocurrency market capable of predicting future price changes with high accuracy.

Research methods: time series analysis, use of neural networks for forecasting, statistical data analysis.

Relevance: The growing popularity of the cryptocurrency market and its high volatility create new opportunities for investors. Developing an effective investment model based on time series analysis and neural networks can help investors make better trading decisions.

Results: A Python software product was developed that uses time series analysis and neural networks to predict future prices. Experiments and testing have shown that the model is highly accurate in short-term price forecasting and can be effectively used in investment activities.

ЗМІСТ

ВСТУП

В наш час почав набирати своєї актуальності ринок криптовалют та інструмент за допомогою якого відбуваються транзакції на ньому – криптовалюти. З кожним днем все ширше і ширше починають використовувати нефіатні валюти для інвгестування та заробітку грошей. Проте, ринок криптовалют є хитким та непередбачуваним, що створює складнощі для інвесторів та трейдерів, які на ньому працюють. Тому розробка комплексної моделі інвестування на ринку криптовалют є дуже актуальною проблемою, яка потребує дослідження та пошуку нових підходів.

Об’єктом дослідження на разі буде ринок криптовалют, який зазнає постійних змін та потребую постійної адаптації інструментів та підходів, що використовуються задля роботи з ним.

Предметом дослідження стає розробка комплексної моделі інвестування, яка базується на аналізі часових рядів та використанні нейронних мереж, а також супутніх методів та підходів.

Метою дослідження є розробка моделі, що дозволить зменшити ризики та збільшити дохід для інвесторів, що працюють на цьому ринку. По своїй суті це буде застосунок який, базуючись на вхідних даних про кожну з криптовалют буде давати поради, щодо того як варто вчинити, куди вкласти гроші.

Актуальність даної дипломної роботи полягає в тому, що розробка комплексної моделі інвестування є вкрай необхідною в сучасному світі, що потребує нового бачення та підходів до вирішення. Застосування часових рядів та нейронних мереж для аналізу та прогнозування руху курсів криптовалют може збільшити точність прогнозування та зменшити ризики для інвесторів. Результати дослідження можуть бути корисними для інвесторів та трейдерів, які займаються інвестуванням на ринку криптовалют.

Структурно, робота складається з чотирьох розділів. У першому розділі буде детально розглянуто предмет дослідження, проаналізовано існуючі підходи та методи вирішення поставленої задачі, а також проведено їх порівняльну характеристику. У другому розділі буде сформульовано основні принципи часових рядів, нейронних мереж та супутніх методів, що будуть розглянуті. Також будуть описані математичні основи та принципи практичної реалізації даних методів. У третьому розділі описано роботу програмного продукту та розглянуто його архітектуру. Також буде проведений опис інтерфейсу користувача та здійснено аналіз основних висновків роботи програмного продукту. У четвертому розділі буде проведено функціонально-вартісний аналіз роботи.

ПОСТАНОВКА ЗАДАЧІ

1. Дослідити ринок на предмет аналогічних чи схожих програмних продуктів, проаналізувати потенціал розробки продукту.

2. Зібрати та обробити відповідні дані про курси криптовалют за останні роки.

3. Визначити основні характеристики цих даних та проаналізувати їх для розуміння тенденцій та змін курсів криптовалют.

4. Дослідити вимоги та обмеження різних методів.

5. Обрати найбільш доцільні із існуючих методи для реалізації кожного із модулів.

6. Реалізувати програмний продукт на основі обраних методів.

7. Протестувати продукт на відповідність розглянутій проблемі та ступінь її вирішення. Зробити висновки про переваги та недоліки створеної системи, перспективи її подальшого використання та удосконалення.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

Дослідження предметної області дозволяє розглянути вже існуючи підходи, виявити слабкі місця, тим самим покращити підхід до дослідження. Загалом цей розділ дає можливість зрозуміти навіщо нам потрібне це дослідження та яка його цінність.

1. 1 Актуальність моделі інвестування на ринку криптовалют

Криптовалюти - це децентралізовані цифрові валюти, які використовують криптографічні методи для забезпечення безпеки транзакцій і контролю створення нових одиниць валюти. Ідея створення криптовалют виникла 2008 року, коли було опубліковано документ, що описує концепцію Bitcoin, створену Сатоші Накамото. Bitcoin був першою криптовалютою, і відтоді було створено безліч інших криптовалют.

Вони виникли як відповідь на проблеми, пов'язані з традиційними фінансовими системами, як-от високі комісії, тривалі терміни оброблення транзакцій і відсутність конфіденційності. Криптовалюти дають змогу користувачам надсилати й отримувати гроші в усьому світі без необхідності використовувати посередників, як-от банки, що зменшує комісії та час опрацювання транзакцій.

Криптовалюти також являють собою нову форму цифрового активу, яку можна використовувати для інвестування. Їх можна купувати, продавати і використовувати для оплати товарів і послуг. Водночас криптовалюти схильні до високої волатильності, що означає, що їхня ціна може значно коливатися на коротких проміжках часу.

Загалом, створення криптовалют було спробою отримати альтернативну фінансову систему, яка була б прозорішою, децентралізованою та безпечнішою. Однак, як і з будь-якою новою технологією, криптовалюти

продовжують розвиватися та змінюватися, тому їхнє майбутнє залишається невизначеним.

Всі переваги цифрової валюти викликали стрімкий розвиток ринку криптовалют. Це привернуло увагу не тільки індивідуальних інвесторів, але й великих корпоративних компаній. Технологія блокчейн посприяла і взагалі уможливила безпеку та надійність операцій з криптовалютами. Проте, у зв'язку зі значним ризиком та, як вже було сказано, високою волатильністю криптовалют, інвестування на цьому ринку може бути небезпечним для тих, хто не має достатнього досвіду та знань про цей ринок.

Вдосконалення підходів та розроблення нових інструментів є задачею, що сприятиме зменшенню ризиків. У зв’язку з цим, моделі інвестування на ринку криптовалют повинні бути добре обумовленні аналізом ринку та спиратися на дані про конкретні криптовалюти.

Існує досить велика кількість моделей інвестування, такі як торгівля на ринку криптовалют, інвестування в довгострокові проекти, інвестування в ICO, торгівля на біржі, тощо. Криптовалюти відомі своєю волатильністю, тому необачливе ставлення до інвестування може призвести до значних втрат капіталу інвестора. Однак перед вибором конкретної моделі інвестування необхідно ретельно проаналізувати ринок. Незважаючи на ризики ринок криптовалют відкритий для інвестицій та може принести великі прибутки людям.

Розробка комплексної моделі допоможе інвесторам не тільки обґрунтувати свої подальші рішення, але й обрати стратегію, яку використати при інвестуванні, який приблизний рівень ризику вони братимуть на себе, на який термін краще інвестувати та яким може бути можливий дохід.

1.2 Аналіз існуючих підходів до інвестування на ринку криптовалют.

Важливим етапом перед тим, як вкладати свої гроші є саме аналіз існуючих підходів інвестування на ринку. Їх, з моменту створення першої

криптовалюти, було запропоновано досить велику кількість, що відрізняються багатьма факторами один від одного, такими як ризикованість, прибутковість, тощо.

Враховуючи їх стрімкий ріст, не тільки у ціні, а й в розповсюдженості на ринку інвестування, пройшло вже досить багато часу. Тому варто розглядати сучасні підходи, що значно покращують досвід інвестування та підвищують дохід інвесторів. Також важливо зазначити, що нестабільна економічна ситуація у світі та ріст цін спонукають людей все більше шукати альтернативні способи збереження та збільшення капіталу. Така ситуація збільшує попит на дослідження та покращення існуючих та розробки нових підходів.

Загалом давайте розглянемо найбільш популярні та розповсюджені, які можуть бути використані інвесторами з різними рівнями ризиків та ставками доходу:

1. Торгівля на ринку криптовалют;
2. Інвестування в довгострокові проекти;
3. Інвестування в ICO;
4. Торгівля на біржі;
5. Створення портфеля криптовалют;

Тепер давайте розглянемо детальніше кожен з підходів:

1. Торгівля на ринку криптовалют.

Даний підхід є досить очевидним та простим – він полягає у купівлі окремо взятої криптовалюти за поточною ціною з метою продати її у майбутньому, за умови підвищення її вартості. З переваг цього підходу можна виділити можливість отримання великого прибутку, але він є досить ризикованим, оскільки ціна нефіатних валют можуть різко падати в будь-який момент, тому це є вагомим недоліком.

2. Інвестування в довгострокові проекти

Уявімо ситуацію, коли інвестор чітко розуміє майбутні перспективи окремої валюти. Він точно знає, що через, умовно, рік ціна окремої монети

буде у два рази більше, тоді він обирає саме такий підхід – він купує криптовалюту з метою утримання її протягом тривалого періоду. У цьому підході значно менші ризики ніж у розглянутому вище, оскільки він є незалежним від короткострокових коливань цін на ринку, а ми пам’ятаємо, що криптовалюти мають високу волатильності. З мінусів у цього це те, що можливий дохід звісно є меншим відносно більш ризикованого підходу.

3. Інвестування в ICO

Почнемо з того, що являє собою інвестування в ICO – це спосіб за допомогою якого компанії можуть залучати кошти на розвиток своїх проектів, продавши токени своєї криптовалюти інвесторам. Тобто інвестор купує токени на початковій стадії розвитку проекту, розраховуючи на його подальше зростання. Цей підхід має перевагу в тому, що інвестор може отримати дуже значний прибуток у разі, якщо проект достатньою мірою розвинувся, але оскільки це інвестування відбувається на самому початковому етапі, тому він є також досить ризикованим.

4. Торгівля на біржі

Принципи торгівля на біржі не сильно відрізняються від тих, що були наведені при торгівлі на ринку криптовалют, а також часто люди сприймають «ринок» та «біржа», як синоніми, але у торгівлі на біржі є деякі переваги. Біржа є надійною і безпечною для купівлі та продажу криптовалют. Також людей на біржі частіше за все більше, тому цей фактор забезпечує більшу ліквідність. З недоліків можна виділити високі комісійні збори, а також обмеження на вивід коштів. Тому при виборі між ринком та біржою треба враховувати ці фактори та обирати, що більше відповідає запити інвестора.

5. Створення портфеля криптовалют

Створення портфеля полягає у тому, що інвестор купує кілька різних криптовалют з різними на його думку ризиками та потенційними прибутками.

Тим самим він диверсифікує свій портфель та зменшує ризик втрати всіх інвестицій у разі падіння одної або декількох з криптовалют. Переваги цього підходу є досить очевидними, оскільки за допомогою нього ми зменшуємо ризики втрати капіталу, тим самим гарантуємо прибуток, але треба зазначити, що прибуток є досить усередненим при виборі такої стратегії дій.

1.3 Огляд існуючих засобів прогнозування

Нині криптовалюти настільки набрали популярності, що майже кожен чув про них. Тому простий засіб (рис. 1.1), який скаже тобі ціну на завтра, або яка ціна буде через тиждень виглядає дуже привабливим.

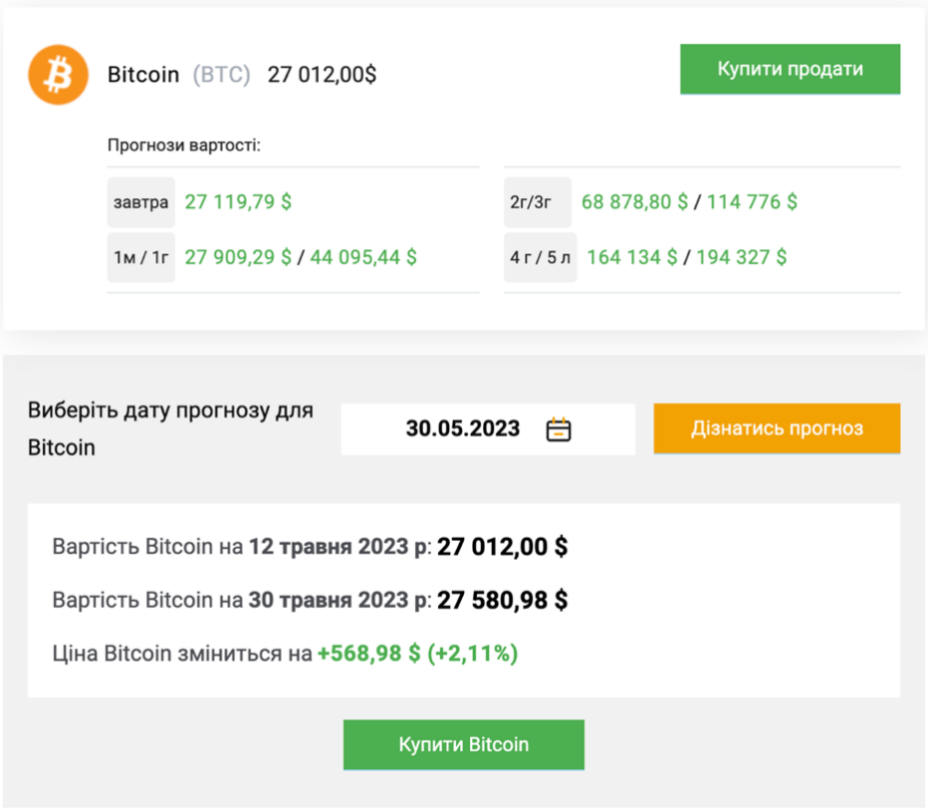

Рис 1.1 Приклад застосунку для прогнозування біткоіну

На рисунку видно, що користувачеві досить вибрати тільки дату прогнозу, щоб отримати однозначний прогноз на обрану дату. Звісно одразу пропонується придбати собі таку монету.

1.4 Формалізація постановки задачі дослідження

Предметна область та вже існуючі підходи та концепції в даній області були розглянуті та проаналізовані, тому ми можемо провести формалізацію постановки задачі:

— провести дослідження підходів та принципів побудови моделі інвестування на ринку криптовалют;
— спираюсь на результати дослідження запропонувати та розробити власний алгоритм вирішення даної задачі;
— провалідувати отримані результати;
— реалізувати сервіс, що буду являти собою застосунок, який буде використовувати побудований алгоритм дій.

1.5 Висновки до розділу 1

У даному розділі було розглянуто різні існуючі підходи до побудови моделі інвестування на криптовалютному ринку які широко використовуються у цій сфері. Були проаналізовані актуальність та проблема яку буде вирішувати наша модель. У результаті був сформований основний напрямок подальшої роботи. Моделі інвестування допоможуть зменшити ризики та покращити прибутки на ще такому досить нестабільному та новому ринку, як ринок криптовалют.

Окрім подальшого напрямку роботи, також було з’ясовано скільки напрямів побудов моделей існує та в чому полягає переваги на недоліки одної над іншою. Також окремим розділом можна бачити формалізацію постановки задачі предметної області задля чіткішого розуміння подальших дій та цілей.

2 МАТЕМАТИЧНІ ОСНОВИ РОЗВ'ЯЗАННЯ ЗАДАЧІ РОЗРОБКИ КОМПЛЕКСНОЇ МОДЕЛІ ІНВЕСТУВАННЯ НА РИНКУ КРИПТОВАЛЮТ ЗА ДОПОМОГОЮ ЧАСОВИХ РЯДІВ ТА НЕЙРОННИХ МЕРЕЖ

Огляд математичних основ розв’язання поставленої задачі є важливим, оскільки дозволяє оглянути теоретичну базу і методологію дослідження інвестиційних рішень. Основною метою наявності цього розділу є обґрунтування і пояснення математичних підходів, моделей та методів, які будуть використані при розробці програмного продукту.

2.1 Аналіз алгоритмів фільтрації даних

Загалом розвиток ринку криптовалют та інформаційних технологій вже створив та кожну секунду створює велику кількість даних. Ці дані можуть бути зашумлені, в такому випадку для покращення моделей широко використовується підхід з фільтрацією даних. Яскравим представником та фільтром, що широко використовується для виявлення та видалення аномалій в даних - є фільтр Калмана.

При прогнозуванні цін криптовалют ми говоримо про динамічні системи, тобто стан системи змінюється з часом. Маючи модель системи фільтри можуть передбачати яким буде стан системи в наступний момент часу. Саме це дає змогу різним типам фільтрів даних так ефективно усувати шуми і оцінювати параметри, які не вимірюються безпосередньо.

Фільтри можуть бути використані в багатьох різних сферах, де необхідно прогнозувати як можна точніше, зменшити шуми та відслідковувати динаміку процесів. У фільтрів багато областей, де вони можуть бути використані, наприклад:

— Авіаційна промисловість;

— Фінансовий сектор;

— Автомобільна промисловість;

— Медична техніка;

— Робототехніка.

Нас цікавить саме фінансовий сектор, оскільки майбутня модель буде зосереджена на прогнозуванні цін криптовалют. Існує багато різних підходів до фільтрації даних, розглянемо деякі з них:

1. фільтр Калмана(Kalman filter);
2. фільтр парзеновського вікна(Parzen window).
3. Сигма-точковий фільтр Калмана(Unscented Kalman Filter)
4. Розширений фільтра Калмана (Extended Kalman Filter)

2.1.1 Фільтр Калмана(Kalman filter)

Варто почати з того, що саме собою являю фільтр Калмана — це алгоритм, який використовує послідовності вимірювань протягом часу, які містять шум (випадкові відхилення) та інші неточності, й видає оцінки невідомих змінних, що є потенційно точнішими за базовані на самих лише вимірюваннях. Формальніше, фільтр Калмана працює рекурсивно на потоках зашумлених вхідних даних, і видає статистично оптимальну оцінку базового стану системи. Фільтр названо на честь Рудольфа Калмана, одного з головних розробників його теорії.

Робота фільтра Калмана (рис. 2.1) полягає у наступних кроках:

1. Визначення початкового стану системи: це оцінка початкового значення стану системи на основі доступних даних.
2. Визначення матриці початкової коваріації: це матриця, яка відображає не визначеність початкового стану системи на основі доступних даних.
3. Прогнозування наступного стану системи: цей крок полягає в використанні математичних моделей, щоб спрогнозувати наступний стан системи.

4. Оцінювання стану системи: цей крок полягає в оцінюванні стану системи на основі доступних даних та прогнозів.
5. Оцінювання похибки прогнозування: це оцінка того, наскільки точним був прогнозований стан системи.
6. Оновлення матриці коваріації: це оновлення матриці, яка відображає не визначеність стану системи.
7. Оновлення стану системи: це оновлення оцінки стану системи на основі нових даних та матриці коваріації.
8. Повторення кроків 3-7 для наступного проміжку часу: ця послідовність кроків повторюється для кожного наступного проміжку часу, щоб оцінити стан системи на основі нових даних та прогнозів.

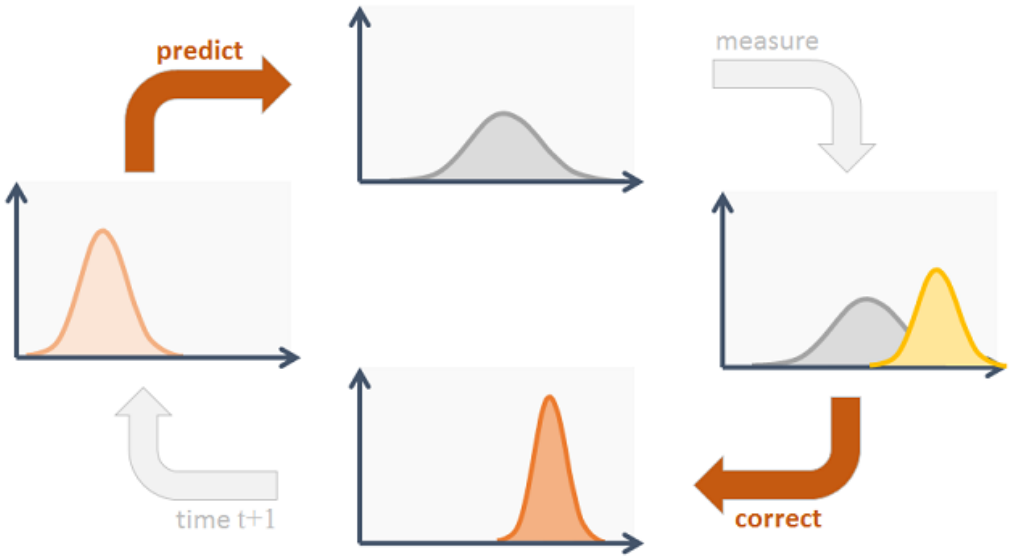

Рис 2.1: Процес роботи фільтра Калмана

Також варто зазначити, що цей фільтр накладає обмеження на моделі, що використовуються — це повинні бути дискретні моделі в просторі стану, а також вони мають бути лінійними. Використання звичайного фільтра Калмана у випадку фільтрації даних про ціни криптовалют є не дуже доречним з точки зору результативності, але для порівняльної характеристики з іншими підходами до фільтрації він є прийнятним, оскільки є одним з класичних.

Модель системи передбачає, що істинний стан системи у момент k отримується з істинного стану системи в момент $k-1$, це може бути представлено у вигляді наступного різницевого рівняння (2.1).

$$x_k = F_k x_{k-1} + B_k u_k + w_k, \quad (2.1)$$

де x_k – стан системи у поточний момент часу; F_k – матриця процесу, що задає систему лінійних рівнянь, що описують як отримати наступний стан систему; u_k – вектор керуючих впливів; B_k – матриця керування, які прикладається до вектора u_k. w_k – нормально розподілений випадковий процес з нульовим математичним очікуванням і коваріаційною матрицею Q_k, яка описує випадковий характер еволюції системи.

Для налаштування фільтра нам потрібно заповнити кілька коваріаційних матриць: P, R та Q. На кожному кроці фільтр Калмана будує припущення про стан системи, виходячи з попередньої оцінки стану і даних вимірювань. Якщо невизначеності вектора стану вищі, ніж помилка вимірювання, то фільтр обиратиме значення, ближчі до даних вимірювань. Якщо помилка вимірювання більша за оцінку невизначеності стану, то фільтр буде більше "довіряти" даним моделювання. Саме тому важливо правильно підібрати значення коваріаційних матриць - основного інструменту налаштування фільтра.

P – Квадратна матриця, порядок матриці дорівнює розміру вектора стану. Як вже було сказано вище, ця матриця визначає "впевненість" фільтра в оцінці змінних стану. Алгоритм самостійно оновлює цю матрицю в процесі роботи. Однак нам потрібно встановити початковий стан, разом із вихідним припущенням про вектор стану. У багатьох випадках нам невідомі значення коваріації між змінними для початкового стану (елементи матриці, розташовані поза головною діагоналлю). Тому можна проігнорувати їх, встановивши рівними 0. Фільтр самостійно оновить значення в процесі роботи. Якщо ж значення коваріації відомі, то, звичайно ж, варто використовувати їх. Дисперсію ж проігнорувати не вийде. Необхідно

встановити значення дисперсії залежно від нашої впевненості у вихідному векторі стану.

R − коваріаційна матриця шуму вимірювань. Квадратна матриця, порядок матриці дорівнює розміру вектора спостереження (кількості вимірюваних параметрів). У багатьох випадках можна вважати, що вимірювання не корелюють одне з одним. У цьому разі матриця R буде діагональною матрицею, де всі елементи поза головною діагоналлю дорівнюють нулю. Досить буде встановити значення дисперсії для кожного вимірюваного параметра.

Q − коваріаційна матриця помилки моделі. Квадратна матриця, порядок матриці дорівнює розміру вектора стану. Щоразу, коли фільтр передбачає стан системи, використовуючи модель процесу, він збільшує невпевненість в оцінці вектора стану. Для одновимірного випадку формула (2.2) має приблизно такий вигляд

$$P_k = P_{k-1} + Q \tag{2.2}$$

Якщо встановити дуже маленьке значення Q, то етап передбачення буде слабо збільшувати невизначеність оцінки. Це означає, що ми вважаємо, що наша модель точно описує процес. Якщо ж встановити велике значення Q, то етап передбачення буде сильно збільшувати невизначеність оцінки. Таким чином, ми показуємо, що модель може містити неточності або невраховані фактори.

2.1.2 Фільтр парзеновського вікна(Parzen window filter)

В статистиці, ядрова оцінка густини розподілу — це непараметричний метод оцінки функції густини випадкової величини за вибіркою. Ядрова оцінка густини є важливою задачею згладжування даних; при застосуванні методу судження щодо статистичних властивостей популяції здійснюється на базі скінченної вибірки. В деяких галузях (таких як обробка сигналів,

економетрика) поряд з ядровою оцінкою густини використовують назву вікно Парцель-Розенблата.

У алгоритму є наступні кроки:

1. Визначити ядро, яке буде використовуватися для оцінки щільності ймовірності розподілу даних. Ядро - це функція, яка визначає вагові коефіцієнти для кожного елемента даних у вікні.
2. Обрати ширину вікна, яка буде використовуватися для оцінки щільності ймовірності розподілу даних. Ширина вікна - це розмір вікна, у межах якого виконується оцінка щільності ймовірності розподілу даних.
3. Провести оцінку густини ймовірності розподілу даних з використанням обраного ядра і ширини вікна. Кожен елемент даних отримує ваговий коефіцієнт залежно від відстані від нього до центру вікна.
4. Провести фільтрацію даних, використовуючи оцінку щільності ймовірності розподілу. Для цього необхідно обчислити зважену суму елементів даних, де вага кожного елемента даних визначається оцінкою щільності ймовірності розподілу.
5. Повторити процес для кожного елемента даних у послідовності.

Перевагою фільтра Парзена є те, що він дає змогу гнучко налаштовувати параметри ядра і ширини вікна, що робить його придатним для різних типів даних і завдань фільтрації. Крім того, цей метод не вимагає знання параметрів розподілу даних, що дає змогу використовувати його у випадках, коли ці параметри невідомі або важко визначити. Однак, як і будь-який метод фільтрації даних, фільтр Парзена має свої обмеження.

Наприклад:

1. Чутливість до параметра вікна: Параметр вікна є ключовим фактором у роботі цього фільтра. Недостатньо широке вікно може призвести до недооцінки згладжування, в той час як занадто

широке вікно може призвести до надмірного згладжування і втрати деталей.

2. Обмеження на кількість даних: Щоб фільтр працював коректно, потрібна велика кількість даних, ніж в інших методах фільтрації. Це може обмежити застосування фільтра у випадках, коли дані обмежені або коли немає можливості отримати велику кількість даних.
3. Залежність від розподілу даних: Парзенівський фільтр має властивість, що його точність і ефективність залежать від розподілу даних. Фільтр може працювати неефективно, якщо дані мають сильне зміщення або нерівномірність розподілу.
4. Вплив шуму: Шум у даних може суттєво вплинути на роботу фільтра. Якщо шум не видаляється, то згладжування може призвести до втрати інформації.
5. Складність обчислень: Парзенівський фільтр вимагає великої кількості обчислень, що може бути витратним і часозатратним процесом, особливо за великої кількості даних.

Математично цей фільтр являє собою алгоритм, який оцінює щільність ймовірності розподілу випадкової величини на основі спостережуваних даних. Він ґрунтується на ядерній оцінці щільності ймовірності, яка використовує функцію для згладжування даних. Тобто нехай дано вибірку випадкової величини(2.3):

$$X = \{x_1, x_2, \dots, x_n\} \tag{2.3}$$

з щільністю $f(x)$. Щоб оцінити цю щільність імовірності фільтр парзеновського вікна використовує наступну формулу (2.4):

$$f(x) = \left(\frac{1}{nh}\right) \cdot \sum_{i=1}^{n} K\left(\frac{(x-x_i)}{h}\right), \tag{2.4}$$

де $h -$ ширина вікна; $K(x) -$ ядерна функція; $x_i -$ елемент вибірки.

$K(x)$ визначає як дані будуть згладжуватися. Вона має задовольняти дяким умовам(2.5):

$$K(x) \geq 0, \text{для } \forall x \in X,$$

$$\int K(x)dx = 1.$$

(2.5)

Прикладами ядерних функцій можуть бути гаусове ядро (функція Гауса) або рівномірне ядро. Ширина вікна h визначає, наскільки сильно дані будуть згладжуватися. Якщо вона занадто мала, то вийде сильно галаслива оцінка щільності ймовірності, а якщо занадто велика, то дані будуть сильно розмиті. Вибір оптимального значення h може бути непростим, і може знадобитися деяке налаштування для кожного конкретного завдання.

2.1.3 Сигма-точковий фільтр Калмана(Unscented Kalman Filter)

Чому звичаний фільтр Калмана погано працює для нелінійних моделей і як з цим впоратися. Вся справа в нормальному розподілі. Під час застосування лінійних перетворень до нормально розподіленої випадкової величини, результуючий розподіл буде являти собою нормальний розподіл, або буде пропорційним нормальному розподілу. Саме на цьому принципі і будується математика фільтра Калмана. Тут до нас приходить його модифікація Unscented Kalman Filter (UKF) - це варіант фільтра Калмана, який використовується для оцінки стану системи з неоднорідними або нелінійними процесами та помилками вимірювання. UKF розширює базовий фільтр Калмана, будуючи наближення розподілу, що виходить після нелінійного перетворення за допомогою сигма-точок. Перевагою цього методу є те, що він не потребує обчислення похідних.

Цей метод використовується для лінеаризації нелінійної функції випадкової величини за допомогою лінійної регресії між n точками, взятими з попереднього розподілу випадкової величини. Оскільки ми розглядаємо розкид випадкової величини, цей метод має тенденцію бути більш точним, ніж лінеаризація за допомогою ряду Тейлора. UKF(Unscented Kalman Filter) використовує техніку детермінованої вибірки для вибору мінімального набору

точок вибірки (так званих сигма-точок) навколо середнього значення. Сигма-точки потім поширюються через нелінійні функції, з яких потім формується нове середнє значення та оцінка коваріації. Результуючий фільтр залежить від того, як обчислюється перетворена статистика і який набір сигма-точок використовується. Крім того, цей метод усуває вимогу явно обчислювати якобіани, що для складних функцій може бути складним завданням саме по собі (тобто вимагати складних похідних, якщо це робиться аналітично, або бути обчислювально дорогим, якщо це робиться чисельно), якщо не неможливим (якщо ці функції не є диференційованими).

Основними етапами роботи є:

1. Ініціалізація: визначення початкового стану та матриць коваріації помилок вимірювань та системи.
2. Передбачення стану: використання динамічної моделі для передбачення стану системи наступного часового кроку. Для цього використовується функція передбачення та матриця коваріації помилок передбачення.
3. Генерація точок Сігма: для оцінки передбаченого стану генеруються точки Сігма з використанням формул трансформації Гауссівських випадкових величин.
4. Оновлення фільтра: застосування отриманих точок Сігма до оновлення стану та матриці коваріації вимірювань. Потім оцінюється спостережуваний стан, використовуючи функцію спостереження та матрицю коваріації помилок спостережень.
5. Корекція: оновлення оцінок стану та матриці коваріації на основі отриманих даних вимірювань та передбачення.

Базовою відмінністю від звичайного фільтра Калмана є те, що оцінюється стан дискретної динамічної системи стану саме нелінійної динамічної системи(2.6):

$$x_{k+1} = F(x_k, v_k);$$
$$y_k = H(x_k, n_k),$$

(2.6)

де x_k – неспостережуваний стан системи; y_k – єдиний спостережуваний сигнал; v_k – шум процесу керування; n_k – шум спостереження; F, H – динамічні моделі системи.

2.1.4 Розширений фільтра Калмана (Extended Kalman Filter)

EKF (Extended Kalman Filter) є розширенням звичайного фільтра Калмана, який призначений для роботи з нелінійними системами. На відміну від звичайного фільтра Калмана, який використовує лінійні моделі для прогнозування станів системи та оцінювання помилки, EKF використовує нелінійні моделі, що можуть бути лінеаризовані в околиці поточного стану. EKF працює у два етапи: прогнозування та коригування. На етапі прогнозування EKF використовує нелінійні моделі для передбачення наступного стану системи та оцінює помилку прогнозу. На етапі коригування EKF використовує лінеаризовані моделі для коригування стану та оцінки помилки.

Ми вже встигли розглянути UKF(Extended Kalman Filter), тому варто зазначити в чому саме полягає відмінність двох розширень класичного фільтра Калмана – EKF, UKF.

Відмінність між EKF і UKF полягає в тому, як вони наближають нелінійні функції. EKF наближає нелінійні функції за допомогою лінеаризації, а UKF - за допомогою апроксимації за допомогою безлічі точок – сигма-точок.

Конкретніше, EKF використовує лінійну апроксимацію нелінійних функцій шляхом розкладання їх у ряд Тейлора до першого порядку в околиці поточної точки. Це може бути не дуже точним наближенням, особливо для систем із високою нелінійністю або з великою дисперсією.

UKF, з іншого боку, використовує апроксимацію нелінійних функцій за допомогою безлічі точок, званих сигма-точками. Вони генеруються з поточного стану і коваріаційної матриці системи. Потім ці точки передаються через нелінійну функцію для апроксимації вихідних значень. Це дає змогу у

випадку високої нелінійності, або великої дисперсії UKF отримувати точніші наближення нелінійних функцій і, як наслідок, точніші оцінки станів.

Крім того, UKF може бути ефективнішим, ніж EKF, у тому сенсі, що його можна використовувати для ширшого діапазону систем і він може бути більш стійким до шумів. Однак, UKF може бути складнішим у реалізації та вимагати більшої кількості обчислювальних ресурсів, ніж EKF.

2.2 Аналіз алгоритмів прогнозування даних

Цей підрозділ є важливою складовою дипломної роботи, оскільки він дозволяє визначити та порівняти різні алгоритми, які можуть бути використані для прогнозування цін криптовалют. В цьому розділі проводиться аналіз різних підходів і методів прогнозування, зокрема моделей часових рядів та нейронних мереж.

2.2.1 Моделі часових рядів

Наші данні про криптовалюти можна сказати представляють собою пару – ключ та значення. Ключем у цьому випадку слугує дата, а значенням – ціна на окремо взяту монету в цей день. Це наштовхує на думку, що варто звернути увагу на часові ряди та розглянути різні моделі, оскільки ми оперуємо цінами на досить велику кількість різних криптовалют, що мають свої властивості та характеристики.

2.2.1.1 Модель авторегресії і ковзної середньої(Autoregressive moving average)

ARMA(Autoregressive moving average) – це модель часового ряду, котра використовує для опису стаціонарних випадкових процесів. Стаціонарним процесом називається процес, чия статистична властивість не залежить від

часу. Формально стаціонарний процес має постійне середнє значення та дисперсію, а також ступінь кореляції між двома будь якими моментами часу залежить тільки від різниці між ними. Модель авторегресії і ковзної середньої об’єднує в собі дві моделі – модель авторегресії та ковзної середньої. Перша передбачає, що значення ряду в поточний момент часу залежить від його попередніх значень. Тобто, для прогнозування значення ряду в майбутньому, необхідно знати його минулі значення. З іншого боку, друга – модель ковзної середньої передбачає, що поточне значення часового ряду залежить від минулих помилок прогнозу.

ARMA модель може бути записана у наступному вигляді (2.7):

$$y(t) = c + a(1) \cdot y(t-1) + a(2) \cdot y(t-2) + \cdots + a(p) \cdot y(t-p) + \\ + e(t) + b(1) \cdot e(t-1) + b(2) \cdot e(t-2) + \cdots + b(q) \cdot e(t-q), \tag{2.7}$$

де $y(t)$ – значення часового ряду в момент t; c – константа; $a(1)$ до $a(p)$ – коефіцієнт авторегресії; $\mathrm{e}(t)$ – випадкова похибка в момент часу t; $b(1)$ до $b(q)$ – коефіцієнт ковзного середнього; q – порядок ковзної середньої.

Визначення оптимальних значень коефіцієнтів в ARMA моделі може бути складним завданням. Для цього використовують методи, такі як метод максимальної правдоподібності (MLE), метод найменших квадратів (OLS) та інші.

2.2.1.2 Авторегресійна інтегрована ковзна середня(ARIMA)

Як зрозуміло по назві моделі, вона являє собою розширення ARMA моделі. Головна відмінність ARIMA від ARMA полягає в тому, що ARIMA модель може опрацьовувати нестаціонарні часові ряди, тобто ряди, у яких середнє значення та/або дисперсія змінюються з часом.

Ця модель модель використовує процес диференціювання для перетворення нестаціонарного ряду на стаціонарний, який може бути

апроксимований за допомогою ARMA моделі. Диференціювання дає змогу видалити тренди та сезонні компоненти з часового ряду, що робить його стаціонарним. Модифікована модель має низку переваг порівняно зі звичайною. По-перше, вона може працювати з нестаціонарними часовими рядами, що розширює її застосовність для аналізу та прогнозування різних часових рядів. По-друге, ARIMA модель має більшу гнучкість у налаштуванні, оскільки вона дає змогу вибрати порядок моделі як для авторегресійної, так і для ковзного середнього компонента, а також порядок диференціювання. Однак, необхідність диференціювання може ускладнити моделювання часових рядів, особливо для недосвідчених користувачів. Крім того, вибір порядку моделі може бути складним завданням, що вимагає експертного досвіду і використання різних статистичних критеріїв для вибору оптимальної моделі.

2.2.1.3 Сезонна авторегресійна інтегрована модель ковзна середня(SARIMA)

SARIMA – це модель, що є розширенням моделі ARIMA, яка включає всі переваги моделі, для якої вона є розширенням. Окрім цього, ця модель враховує сезонність у даних. Сезонність проявляється в періодичних повторюваних коливаннях у даних, які пов’язані часовими періодами. Ця модель містить у собі параметри, які, як ми вже зазначали, враховують сезонність у даних, такі як сезонний період і сезонний порядок інтегрування. Перший визначає через скільки часових кроків відбувається повторення сезонних коливань у даних. В свою чергу другий - визначає, скільки разів потрібно застосувати операцію різницевої обробки, щоб перетворити сезонний часовий ряд на стаціонарний. SARIMA модель дає змогу врахувати як сезонну, так і загальну часову залежність у даних, що робить її гнучкішою за модель ARMA або ARIMA, особливо для прогнозування даних із явною сезонністю. Однак, оскільки SARIMA модель містить більше параметрів, вона

також може бути складнішою в налаштуванні та вимогливішою до кількості даних для навчання. ARMA модель являє собою предка, тому очевидно є простішою, ніж ARIMA і SARIMA, оскільки вона не враховує ні сезонність, ні можливу автокореляцію. Її використовують для моделювання несезонних часових рядів із невипадковими трендами та циклічністю. Перевага SARIMA моделі в тому, що вона дає змогу моделювати складніші часові ряди, які мають як несезонні, так і сезонні компоненти. Це робить її гнучкішою та потужнішою для аналізу та прогнозування часових рядів у різних галузях, таких як економіка, фінанси, метеорологія та інші.

2.2.2 Моделі нейронних мереж

В наш час розвиток нейронних мереж має великі успіхи. Вони вміють генерувати картинки по тексту, який ти вводиш, генерувати код по запиту, що ти написав, тощо. Це все має такий стрімкий ріст, оскільки різного спектру нейронні мережі дуже давно розробляються. Тому варто розглянути хоча б якусь кількість з них, що зануритися у вир нейронних мереж. У цьому підрозділі пропоную розглянути наступні типи нейромереж:

1. Звичайна рекурентна нейронна мережа
2. Згорткова нейронна мережа
3. Нейронна мережа прямого розповсюдження
4. Довга короткочасна пам'ять
5. Вентильний рекурентний вузол
6. Трансформер

В нашому питанні розглядається велика кількість різних монет, з власними характеристиками та властивостями, тому розглянути досить велику кількість моделей та у подальшому обрати ту, яка буде найбільш якіснішою є на мою думку гарною ідеєю.

2.2.2.1 Звичайна рекурентна нейронна мережа(RNN)

RNN (Recurrent Neural Network) - це клас нейронних мереж, які здатні опрацьовувати послідовності даних, наприклад - часові ряди, тексти, мова тощо. Основний момент, що відрізняє їх від звичайних нейронних мереж це те, що вони мають зв'язки між прихованими шарами (рис. 2.2), даючи змогу інформації повертатися на попередні кроки обробки. Проста RNN-мережа має один прихований шар, який обробляє послідовність вхідних даних. Кожен елемент вхідної послідовності обробляється шаром разом зі станом прихованого шару, який зберігається і передається на наступний крок обробки. У RNN є різні варіації, як-от LSTM (Long Short-Term Memory) і GRU (Gated Recurrent Unit), які призначені для розв'язання проблеми градієнта, що зникає, під час навчання RNN на довгих послідовностях даних, їх ми розглянемо пізніше. Просту рекурентну нейронну мережу можна використовувати для різних завдань, як-от прогнозування часових рядів, генерація тексту, розпізнавання мови, машинний переклад тощо.

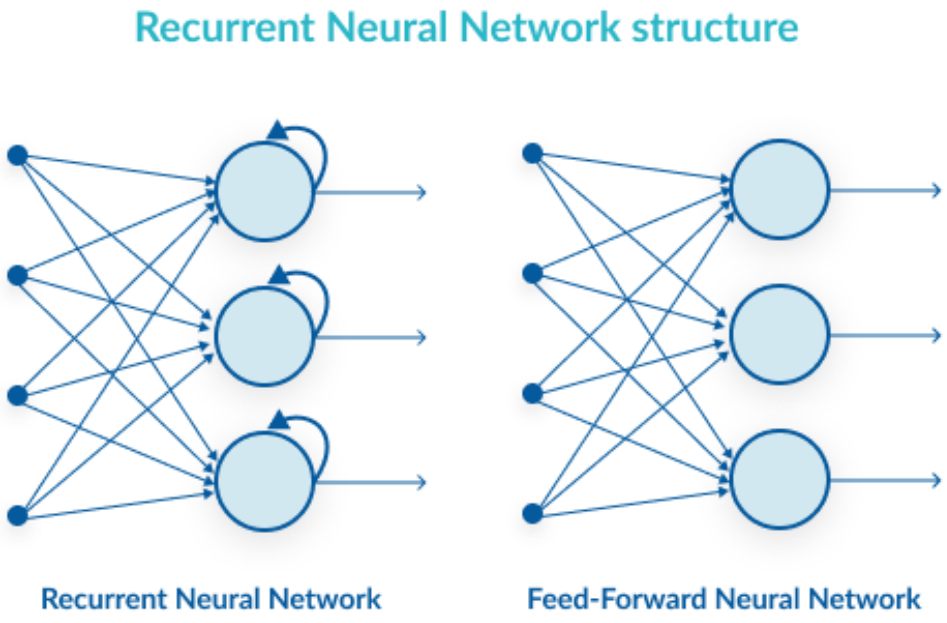

Рис. 2.2 Принцип роботи рекурентної нейронної мережі

2.2.2.2 Згорткова нейронна мережа(CNN)

CNN(Convolutional Neural Network) – це тип нейронних мереж, який використовується для обробки зображень, відео та інших двовимірних даних. Він застосовується для класифікації, сегментації та розпізнавання об'єктів у зображеннях. Нас цікавить чи може ця мережа взаємодіяти з часовими рядами та допомогти нам прогнозувати ціни на криптовалюти. Загалом вона може це зробити - одним із підходів до застосування CNN (рис. 2.3) для часових рядів є використання згортувальних шарів для вилучення ознак із часових даних, а потім передача цих ознак на повнозв'язні шари для прогнозування. Коли згорткові шари застосовують до тимчасових даних, кожен фрагмент часового ряду розглядають як зображення, де часова вісь еквівалентна горизонтальній осі зображення, а вісь ознак - вертикальній осі. Таким чином, фрагменти часового ряду можна розглядати як двовимірні зображення й обробляти згортковими шарами як зображення. Однак, необхідно враховувати, що використання згорткових нейронних мереж для часових рядів може бути менш ефективним, ніж використання рекурентних нейронних мереж.

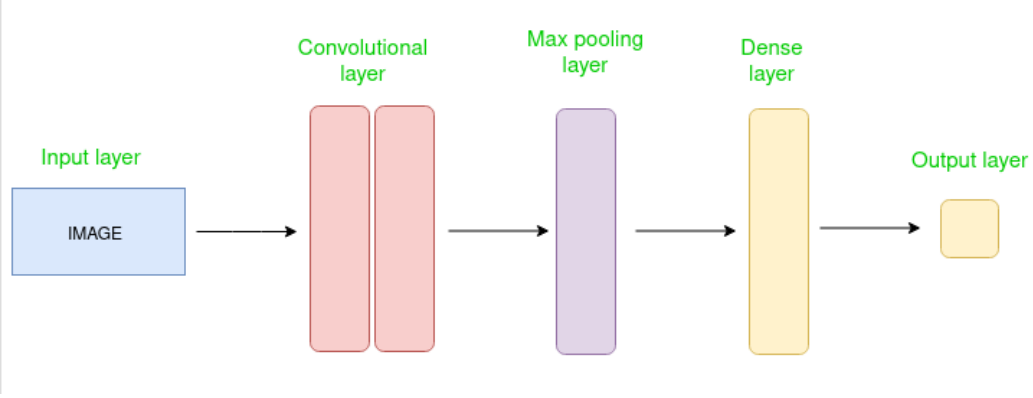

Рис. 2.3 Принцип роботи згорткової нейронної мережі

2.2.2.3 Нейронна мережа прямого розповсюдження(DFN)

DFN(Deep Feedforward Networks) (рис. 2.4) є одним із найпоширеніших типів нейронних мереж, які використовують для розв'язання задач

класифікації та регресії. Ці нейронні мережі складаються з декількох шарів, кожен з яких містить безліч нейронів. Вхідні дані подаються на вхідний шар, який передає вихідні дані в наступний прихований шар. Кожен прихований шар містить кілька нейронів, кожен з яких пов'язаний з усіма нейронами попереднього шару. Нейрони в останньому (вихідному) шарі зазвичай являють собою класи (у задачах класифікації) або безперервні значення (у задачах регресії). Також їх можна використовувати для прогнозування часових рядів. Одним із поширених підходів є використання DFN для прогнозування наступного значення ряду на основі попередніх значень. Для цього кожному значенню часового ряду відповідає вхідний вектор, що складається з кількох попередніх значень, а вихідний вектор містить наступне значення ряду. Вхідний вектор може мати різні розмірності залежно від кількості попередніх значень, що використовуються для прогнозування. Наприклад, якщо ми хочемо використовувати 5 попередніх значень часового ряду для прогнозування наступного значення, то розмірність вхідного вектора дорівнюватиме 5. Вихідний вектор завжди має розмірність 1, оскільки ми прогнозуємо тільки одне наступне значення. Навчання DFN для прогнозування часових рядів відбувається шляхом мінімізації функції втрат між прогнозованими і фактичними значеннями часового ряду. Як функцію втрат часто використовують середньоквадратичну помилку. Після навчання модель може бути використана для прогнозування значень часового ряду на нових даних, використовуючи попередні значення ряду як вхідні дані.

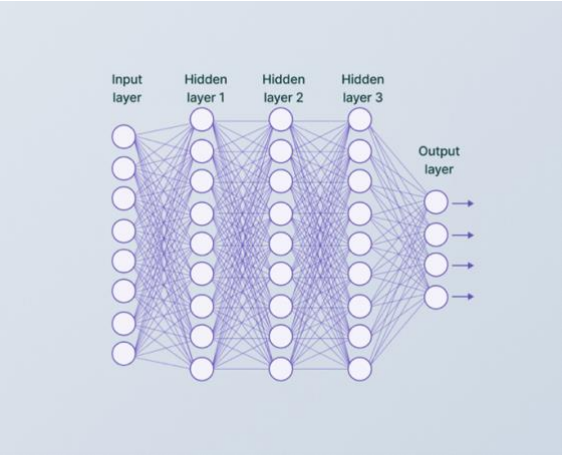

Рис. 2.4 Принцип роботи нейронної мережі прямого розповсюдження

2.2.2.4 Довга короткочасна пам'ять(LSTM)

Як ми згадували вище існує досить широкий спектр типів рекурентних нейронних мереж. LSTM(Long Short-Term Memory) – це один з них, він являє собою тип рекурентної нейронної мережі (RNN), який дає змогу працювати з послідовностями довільної довжини і зберігати довгострокові залежності між елементами послідовності. Він був представлений 1997 року Хохрайтером і Шмідхубером.

LSTM складається з так званих блоків пам'яті, кожен з яких має три взаємодіючі компоненти:

1. Осередок пам'яті (memory cell) - запам'ятовує довгострокову інформацію і контролює її потік через блок.
2. Ворота (gates) - визначають, яка інформація має зберігатися в комірці пам'яті, яка інформація має вилучатися з комірки пам'яті та яка інформація має бути забута.
3. Прихований стан (hidden state) - визначає, яка інформація буде передана на наступний крок і використовується для обчислення виходу на кожному кроці.

LSTM-мережі мають низку переваг порівняно зі звичайними RNN, адже вони можуть уникнути проблеми зникаючих градієнтів і здатні зберігати інформацію на довгий час. LSTM використовують у різних завданнях, включно з обробленням природної мови, комп'ютерним зором, аналізом часових рядів тощо.

2.2.2.5 Вентильний рекурентний вузол(GRU)

GRU(Gated Recurrent Unit) - це рекурентна нейронна мережа (RNN), яка допомагає розв'язувати проблему загасання градієнта і дає змогу передавати інформацію на триваліші часові інтервали, ніж проста RNN. GRU було представлено у 2014 році К'єном Тхеісеном і його колегами. Вона була

створена як альтернатива Long Short-Term Memory (LSTM) - іншій популярній формі RNN. GRU використовує механізми подібні до LSTM, зокрема фільтрацію вхідних даних і управління потоком інформації, що проходить через нейрони. Однак, GRU має менше параметрів, що полегшує її навчання та прискорює обчислення. Основна ідея GRU полягає у використанні "вентилів" (gates) для управління інформацією, яка буде збережена і передана в наступний момент часу. У GRU є два види вентилів: "вентиль забування" (reset gate) і "вентиль оновлення" (update gate). "Вентиль забування" вирішує, яку інформацію буде збережено, а яку - забуто. Він ґрунтується на поточних входах і попередніх прихованих станах. "Вентиль оновлення" визначає, яку інформацію буде передано в наступний момент часу. Він також ґрунтується на поточних входах і попередніх прихованих станах. Використання цих вентилів дає змогу GRU керувати потоком інформації та уникати проблеми загасання градієнта, що робить її дуже ефективною для роботи з послідовними даними, як-от текст, звук і часові ряди.

2.2.2.6 Трансформер(Transformer)

Трансформер (рис. 2.5) - це архітектура нейронної мережі, яка була вперше представлена в статті "Attention Is All You Need" компанії Google у 2017 році. Вона була розроблена для вирішення завдань обробки природної мови, але може використовуватися і для інших завдань, включно з прогнозуванням часових рядів. Основна ідея Transformer полягає в тому, що замість використання рекурентних нейронних мереж, таких як LSTM або GRU, для моделювання послідовностей, використовується механізм уваги (attention mechanism). Це дає змогу моделювати залежності між елементами послідовності паралельно, що прискорює навчання та підвищує якість моделі. Архітектура Transformer складається з декількох шарів, кожен з яких складається з декількох блоків. Кожен блок містить багатоканальний механізм уваги (Multi-Head Attention) і два шари нейронної мережі: повнозв'язний шар

із функцією активації ReLU (Feed-Forward Network) і шар нормалізації (Layer Normalization). Механізм уваги (Attention) у блоці Transformer дає змогу моделювати взаємодії між елементами послідовності. Він працює так: для кожного елемента послідовності обчислюється вектор запиту, який використовується для знаходження ваг для всіх елементів послідовності (векторів ключів) на основі їхньої схожості. Зважена сума значень елементів послідовності, взятих з урахуванням знайдених ваг, і є виходом механізму уваги. Трансформер може бути застосовано до задачі прогнозування часових рядів шляхом подачі послідовності минулих значень часового ряду як вхідних даних моделі. Навчання моделі здійснюється шляхом мінімізації помилки прогнозування на тренувальних даних, а потім модель може використовуватися для прогнозування майбутніх значень часового ряду.

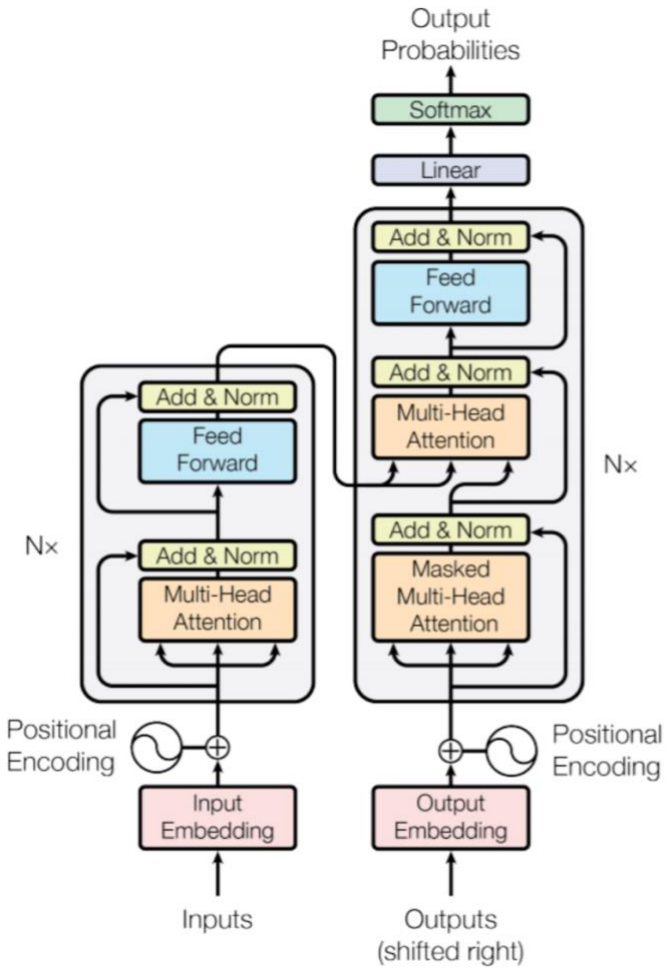

Рис. 2.5 Принцип роботи нейронної мережі трансформер

2.3 Аналіз існуючих метрик та підходів до оцінки якості моделі

В цьому підрозділі розглянутий аналіз метрик. Метрики потрібні для чисельного аналізу якості моделі. Вони слугують показниками, що дають змогу оцінити яка з моделей та підходів є більш якісною таким чином дозволяють отримати кращий результат.

2.3.1 Корінь з середньо-квадратичної помилки(RMSE)

RMSE (Root Mean Squared Error) є досить класичною та розповсюдженою для оцінки розбіжності між прогнозованими значеннями моделі і фактичними значеннями даних. Вона обчислюється шляхом обчислення середньоквадратичного значення квадратів відхилень між прогнозованими та фактичними значеннями, після чого береться квадратний корінь з отриманого значення, щоб повернути його до вихідного діапазону одиниць вимірювання. Математично його можна представити у наступному вигляді (2.8):

$$RMSE = \sqrt{\frac{1}{n} \cdot \sum_i^n (\hat{y}_i - y_i)^2}, \tag{2.8}$$

де $\hat{y}_i -$ прогнозоване значення для і-го спостереження, $y_i -$ спостережуване значення для і-го спостереження, $n -$ розмір вибірки.

2.3.2 Середня симетрична абсолютна похибка у відсотках(SMAPE)

SMAPE(Symmetric Mean Absolute Percentage Error) – метрика, що вимірює відсоткову розбіжність між прогнозованими значеннями моделі і фактичними значеннями даних. Вона виражає розбіжність між прогнозованими та фактичними значеннями як відсоток від суми двох абсолютних значень. Ця метрика враховує симетрію між недооцінкою та переоцінкою прогнозів, оскільки обчислюється для кожного спостереження

окремо. У результаті повертається значення від 0 до 200, де 0 відповідає ідеальній точності, а вищі значення вказують на більшу розбіжність між прогнозами і фактичними значеннями. Математичної її можна представити у наступному вигляді (2.9):

$$SMAPE = \frac{100}{n} \cdot \sum_i^n \left(2 \cdot \frac{|\hat{y}_i - y_i|}{|\hat{y}_i| + |y_i|}\right), \quad (2.9)$$

де $\hat{y}_i -$ прогнозоване значення для і-го спостереження, $y_i -$ спостережуване значення для і-го спостереження, $n -$ розмір вибірки.

2.3.3 Направлена симетрія (DS)

DS(Directional Symmetry) визначає, наскільки часто прогнози відповідають зростанню або спаду значень порівняно з фактичними даними. Ця метрика використовується в контексті часових рядів або прогнозування, де важливо враховувати не тільки абсолютні значення прогнозу, але й правильну орієнтацію чи зміну напрямку. Для розрахунку Directional Symmetry часто використовується така процедура:

1. Обчислити різницю між прогнозованими та фактичними значеннями.
2. Визначити знак різниці. Наприклад, позитивний знак вказує на те, що прогнози більші за фактичні значення, а негативний знак вказує на те, що прогнози менші за фактичні значення.
3. Порівняти зміну знаку між сусідніми значеннями. Якщо знак різниці міняється від позитивного до негативного або навпаки, вважається, що є наявна напрямкова симетрія.

Математично це можна представити наступним чином:

$$DS = \frac{1}{n-1} \sum_{i=2}^{n} d_i ,$$

$$d_i = \begin{cases} 1, & \text{якщо } (y_i - y_{i-1})(\hat{y}_i - \hat{y}_{i-1}) > 0 \\ 0, & \text{інакше} \end{cases},$$

(2.10)

де $\hat{y}_i$ – прогнозоване значення для і-го спостереження, y_i – спостережуване значення для і-го спостереження, n – розмір вибірки.

2.4 Висновки по розділу 2

Хороші вхідні дані є запорукою побудови якісної та корисної моделі, після чого слідує вибір моделі та методи оцінки їх якості.

У цьому розділі було розглянуті методи фільтрації даних, що допоможуть усунути шуми з вхідних даних, адже вони можуть зіграти важливу роль у прогнозуванні. Також був розглянутий ряд методів та підходів які стануть фундаментом для побудови комплексної моделі інвестування на ринку криптовалют. Наприкінці розділу ми подивилися на три метрики, що дадуть нам змогу оцінити якість та обрати найкращу модель серед запропонованих. Загалом цей розділ слугує екскурсом у базові етапи та математичні концепції, що використовуються при побудові комплексної моделі інвестування.

3 АНАЛІЗ АРХІТЕКТУРИ ПРОГРАМНОГО ПРОДУКТУ

Даний розділ являє собою опис обраних засобів розробки та реалізації комплексної моделі інвестування на ринку криптовалют. Він дозволяє пройтися по кожному етапу розробки програмного продукту та побачити результати роботи.

3.1 Обґрунтування вибору мови програмування та засобів реалізації

Обрана задача розробки комплексної моделі інвестування на ринку криптовалют потребує методів фільтрації даних, а також методів прогнозування за допомогою нейронних мереж та часових рядів. Для розробки функціональної частини програмного продукту, а також роботи з даними було обрано використовувати мову програмування Python.

Вибір такої мови програмування, як Python має вагоме обґрунтування. Ця мова програмування має низку переваг таких як висока продуктивність розробки, логічність та лаконічність. Python інтерпретована мова, тому дає можливість запускати програмний продукт на будь-яких операційних системах. Також великою перевагою є те, що дана мова має велику якісну базу бібліотек, що полегшує процес розробки та дає змогу сконцентруватися не на написанні логіки роботи, а на аналізі та покращенні моделей та підходів.

При розробці програмного продукту був задіяний широкий спектр пакетів програмного. Дані зберігаються та зчитуються за допомогою пакету Pandas. Цей засіб був обраний, оскільки має досить зручний програмний інтерфейс (API) для взаємодії з файлами, розширення яких – csv. Крім того, дані у такому вигляді займають менший об’єм пам’яті та мають більшу швидкодію, що дозволяє оптимізувати роботу програмного продукту.

Також було використану такі пакети програмного забезпечення, як filterpy, pykalman для фільтрації даних для усунення шум в даних, задля

оптимізації передбачення та прогнозування. Саме такий вибір був обумовлений тим, що мають якісну документацію, що полегшує взаємодію та подальшу підтримку вже написаного коду. Також вони мають велику перевагу над аналогами, оскільки дають зручний API, що дозволяє налаштовувати та розширювати фільтри задля гнучкого налаштування фільтрів.

Очевидним є те, що якою б модель не була геніальною без зручного та зрозумілого користувацького інтерфейсу нею важко оперувати. Тому на етапі розробки саме такого інтерфейсу була задіяна бібліотека Streamlit. Вона дозволяє реалізувати інтерактивний користувацький інтерфейс, де користувачі можуть взаємодіяти з даними, візуалізаціями та елементами керування. Також основною перевагою є - швидкість розробки, широкий набір компонентів, підтримка для наукових бібліотек, розгортання на сервері, всі ці переваги дають змогу зосередити увагу на розробці якісної моделі, а бізнес-логіку, аспекти якісної візуалізації, деполой на віддалений сервер, інтегрування з такими бібліотеками, як Pandas, Matplotlib, NumPy вона бере на себе.

3.2. Збір даних та навчання моделі.

Збір інформації про історичні значення цін криптовалют був проведений шляхом пошуку даних на різних ресурсах. Для отримання даних про ціни криптовалют були використані наступні ресурси:

1. CoinMarketCap.com: Цей сайт надає широкий обсяг даних про ціни, ринкову капіталізацію, обсяг торгів та інші характеристики криптовалют. Використовуючи веб-скрейпінг, були отримані дані зі сторінок криптовалют, таких як Bitcoin, Ethereum, Ripple та інших.
2. CryptoCompare.com: Цей ресурс також надає інформацію про ціни криптовалют та інші статистичні дані. Було проведено пошук та витягнуто дані про ціни криптовалют з різних ринків та бірж.

3. Kaggle.com: Для навчання моделі були використані дані, доступні на платформі Kaggle. Ці дані містять історичні ціни криптовалют та інші зв'язані з ними показники. Вони були завантажені у форматі CSV та використані для навчання та аналізу моделі.

В результаті були сформовані результати пошуку у вигляді окремих csv файлі, кожен з яких містить інформацію про ціни криптовалют та пов’язані з цим данні.

3.3 Опис алгоритму роботи програми

Загалом вище було зазначено досить велику кількість методів та підходів, тому цей підрозділ покроково описує алгоритм роботи розробленного програмного продукту. Розглядаються приклади реалізацій зазначених у другому розділі способів фільтрації даних, прогнозування, оцінки та передбачення. Також задля візуалізації кожного з кроків наведені відповідні графіки, що демонструють коректність їх роботи.

Основний інтерфейс дає змогу користувачеві обрати підхід серед короткострокового, довгострокового та пакетного інвестування. Після чого у випадку вибору інвестування одного з перших двох методів йому надається змога обрати в яку саме криптовалютну монету він має намір інвестувати, наряду з кількістю часу на який він хоче інвестувати.

Після того, як користувач натискає відповідну кнопку, починається відбір найкращої моделі серед моделей часових рядів та нейронних мереж. Для кожного з підходів до інвестування розмір тренувальної та тестової вибірок різний. Пошук моделі яка виявиться найкращою зводиться до пошуку такої, в якої сума нормованих метрик, що приведені до діапазону від 0 до 1, де 0 – ідеальний результат, 1 – найгірша. Звісно кожна метрика йде разом з вагами, що також різні для кожного підходу. Це зумовлено тим, що кожен підхід вимагає індивідуальності, як наприклад для короткострокового інвестування більше має значення RMSE, а при довгостроковому – DS та SMAPE оскільки

враховують спадання та зростання. Як тільки модель знайдена відбувається прогнозування (рис. 3.1) на задану кількість днів, обраховується відсоток можливого прибутку/збитку та виводиться на екран користувачеві.

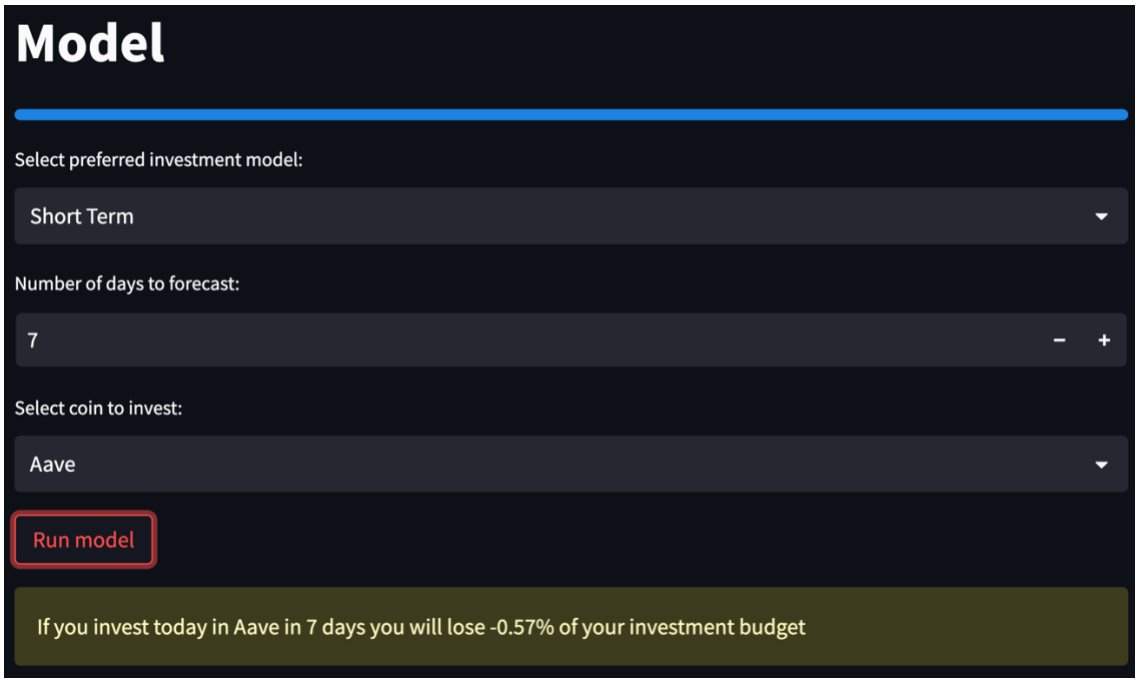

Рис. 3.1 Приклад роботи основної моделі

У розробленій комплексній моделі реалізується покроковий підхід, тобто кожен етап передує наступний. Модель являє собою комбінування відомих методів фільтрації – фільтр парзеновського вікна, фільтр Калмана та його розширення, методів прогнозування за допомогою нейронних мереж та часових рядів. Новизна та покращення результатів досягається за допомогою перехресного комбінування цих методів задля досягнення найбільш вдалої пари фільтр, метод прогнозування.

Процес роботи розробленої моделі полягає у тому, що у нас є перелік методів фільтрації та предбачення даних, крім того, у нас є метрики, що підходять для даної конкретної задачі якнайкраще. Обирається фільтр, дані проходять фільтрацію та розбиваються на тренувальні та тестові, після чого на цих даних тренуються наші моделі передбачення. Відбувається тестування моделей на заданих тестових даних. Після чого процес повторюється до поки в нас не вичерпаються моделі фільтрації. На отриманих результатах

використовуються метрики задля виокремлення найліпшої комбінації фільтрації та прогнозування. Пам’ятаємо, що користувач задає період для прогнозування, як тільки ми знайшли комбінацію, яка є найліпшою для заданої криптовалюти, відбувається прогнозування на заданий користувачем період.

3.3.1 Фільтрація даних

На вхід нами було отримано досить велику кількість монет та їх історичних значень. Задля покращення результатів роботи з цими даними було прийнято рішення скористатися алгоритмами фільтрації даних. Задля наочності та легкості сприйняття, окрім користувацького інтерфейсу задля взаємодії з основною моделюю, також було розроблено інтерфейс (рис. 3.2) для візуалізації кожного з кроків.

Рис. 3.2 Вигляд користувацького інтерфейсу для візуалізації фільтрації даних

Користувач має змогу обрати криптовалюту, методи фільтрації та історичний показник який його цікавить. Натиснувши кнопку “Visualize data filtration” отримаємо графік (рис. 3.3) наступного вигляду:

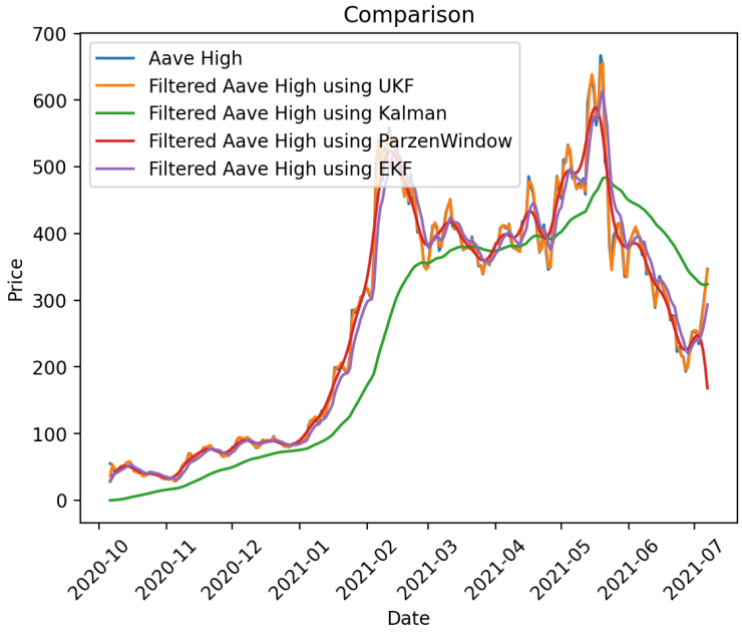

Рис. 3.3 Приклад роботи візуалізації фільтрації даних

3.3.2 Прогнозування

При роботі з даним вони діляться на тренувальні та тестові. Ми навчаємо обрану модель та використовуємо результат для прогнозування на тестових даних. Задля візуального представлення роботи зазначених у другому розділі також було реалізовано подібний до зазначеного вище інтерфейс (рис. 3.4):

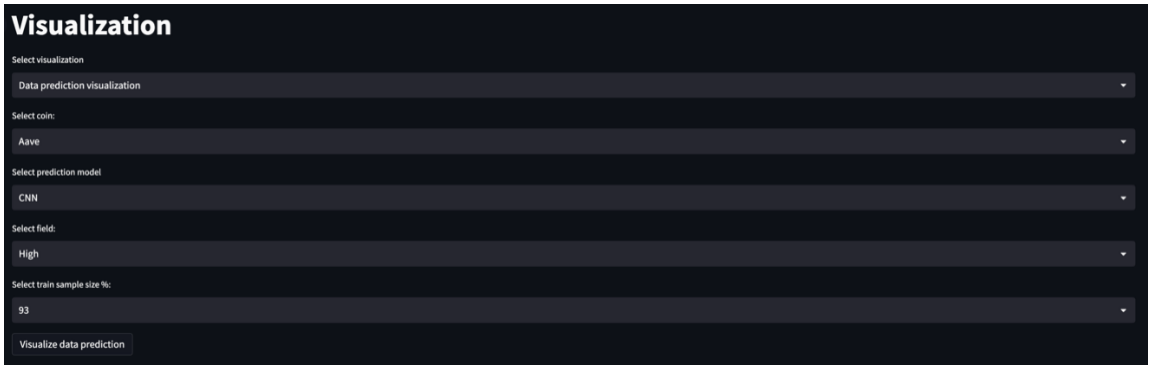

Рис. 3.4 Вигляд користувацького інтерфейсу для візуалізації роботи прогнозування

Користувач так само має змогу обрати криптовалюту, історичний показник, але крім того може обрати модель та розмір тренувальної вибірки та отримати візуальне представлення даних у вигляді графіку (рис. 3.5).

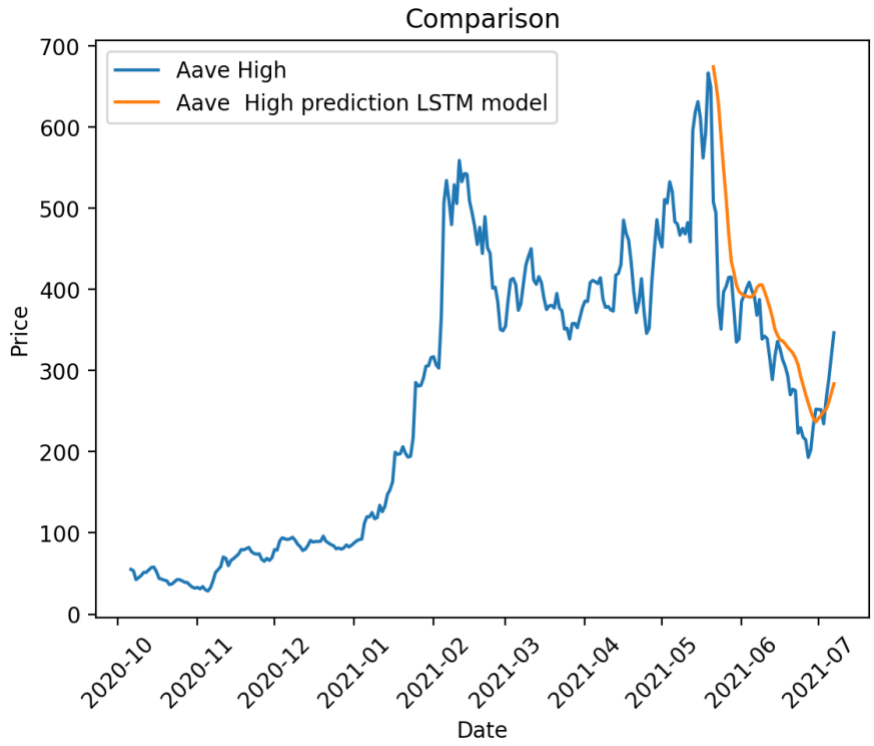

Рис. 3.5 Приклад роботи візуалізації прогнозування даних

Також можна подивитися на можливості (рис. 3.6) прогнозування даних, що вже пройшли фільтрацію:

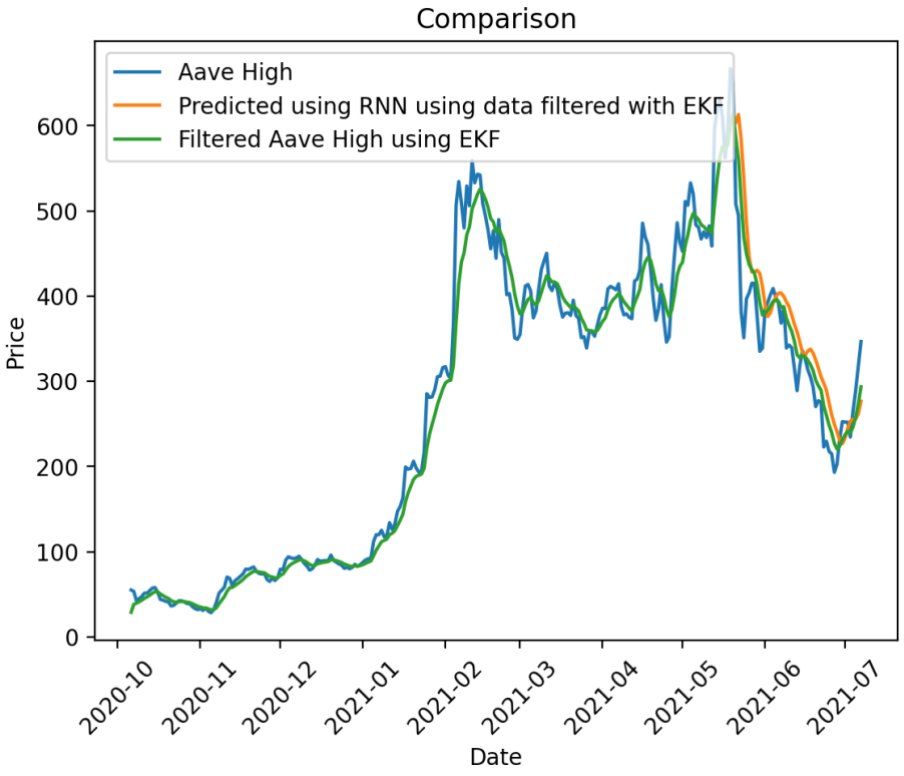

Рис. 3.6 Приклад роботи візуалізації прогнозування даних на відфільтрованих даних

3.3.3 Оцінка якості моделі

Стосовно аналізу якості моделі було розглянуто три метрики RMSE, SMAPE та DS. Ці метрики були обрані, оскільки в нашому випадку важливо враховувати співпадіння спадань і зростань між оригінальним набором даних та отриманим при прогнозуванні. Як вже було описано вище, метрики використовуються для підбору найкращої моделі. Значення метрик можна побачити в розділі для візуалізації (рис. 3.7) передбачення.

	NONE	UKF
RMSE	75.2018	63.7148
SMAPE	0.0766	0.0653
DS	0.4468	0.4255

Рис. 3.7 Приклад результатів оцінки за допомогою метрик

В даному конкретному прикладі розглянуто метрики, що були отримані при прогнозуванні за допомогою моделі CNN з фільтрацією даних та без неї, розмір тренувальної вибірки – 80%. Легко бачити, що кращі результати має модель, що використовувала відфільтровані дані.

3.3.4 Передбачення

Як ми пам’ятаємо наші дані являють собою історичні дані окремих криптовалютних монет, тому ми маємо змогу задати дату в якості індексу, та в основній моделі отримати передбачувану ціну монету через задану користувачем днів. Приклад роботи передбачення в основній моделі ми могли бачити вище, але крім того було реалізовано засіб для візуалізації (рис. 3.8).

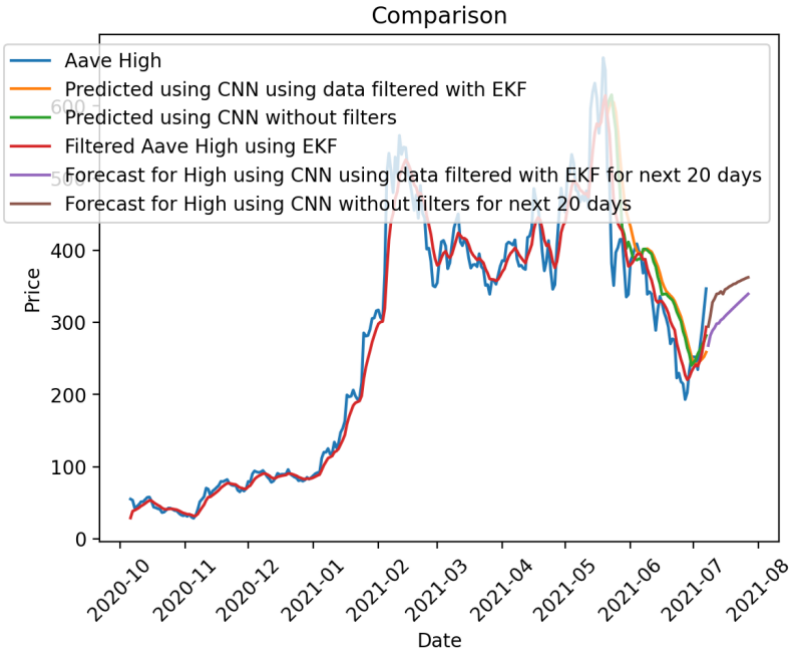

Рис. 3.8 Приклад роботи візуалізації передбачення даних на відфільтрованих та звичайних даних

3.4 Висновки по розділу 3

У ході розробки програмного продукту, що реалізує комплексну модель інвестування на ринку криптовалют, було розроблено різні моделі, часових рядів та нейронних мереж. Користувач застосунку має змогу скористатися як основою комплексною моделлю, так і візуалізацією, щоб наочно побачити графіки роботи різних підходів. Робота моделі є досить довгою, оскільки перебирає всі способи прогнозування та обирає найліпший. Цей аспект зумовлений тим, що у нас представлено велику кількість монет, і в залежності від обраної будуються моделі та прогнози. Покращити ситуацію можна як розпаралеливши навчання моделей, так і для кожної окремої криптовалюти завчасно знайти найпідходящу модель для неї, після чого використовувати тільки її. Загалом існує не тільки описані вище ідеї оптимізації даного програмного продукту, але швидкодія є менш пріоритетним чинником, ніж якість результатів, що отримуються у ході роботи.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі буде проведено оцінювання основних характеристк програмного продукту, що є реалізацією комплексної моделі інвестування на ринку криптовалют. Даний продукт розроблений на мові програмування Python в якості самостійного застосунку. Програмний продукт в першу чергу призначений для широкого використання інвесторами, що прагнуть отримати прогнози, на які можуть покластися.

Функціонально-вартісний аналіз (ФВА) є методологією, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від структури організації. Головною метою ФВА є виявлення можливостей зниження витрат шляхом ефективнішого виробництва та кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовуються економічна, технічна та конструкторська інформація.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи прогнозу стійкості фінансових показників.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка можливого програмного продукту, який дозволить інвесторам скористатися моделями часових рядів та нейронних мереж задля прогнозування цін криптовалют. Беручи за основу цю функцію, можна виділити наступні:

$F_1 -$ бібліотеки для роботи з нейронними мережами

F_2 – бібліотеки для роботи з часовими рядами

F_3 – аналіз якості моделі

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1:

а) Keras

б) TensorFlow

Функція F_2:

а) Statsmodels

б) Prophet

Функція F_3:

а) Sklearn

б) Написання власних

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

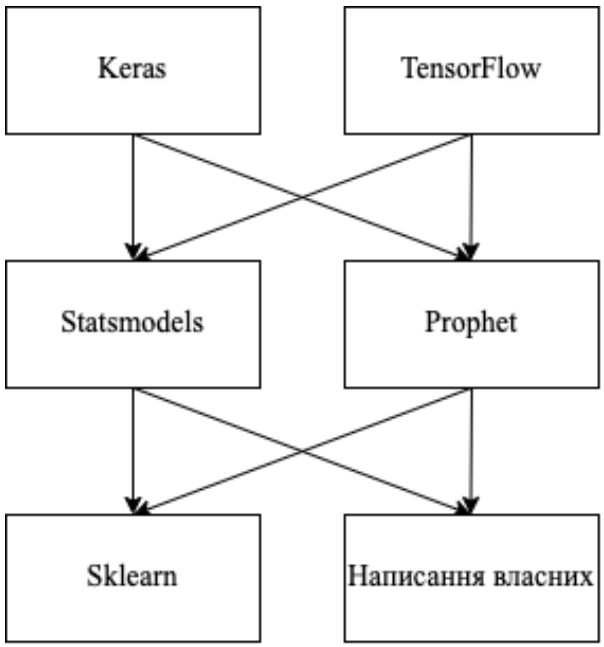

Рисунок 4.1 – Морфологічна карта

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

Таблиця 4.1 - Позитивно-негативна матриця

Функції	Варіант реалізації	Переваги	Недоліки
F_1	А	Гнучкість, висока швидкодія, легке масштабування	Низький рівень абстракції, вимагає великої кількості коду
	Б	Простота, широкий спектр функціоналу, не вимоглива по об’єму пам’яті	Низька гнучкість, не підтримує інтеграцію з новітніми розробками
F_2	А	Широкий набір статистичних моделей, легко інтегрується з іншими пакетами	Обмеженість у використанні сучасних методів, відсутність «зручних» функцій
	Б	Простота використання, врахування сезонності, гнучкість	Обмежена функціональність, має залежність від інших бібліотек
F_3	А	Широкий вибір, простота, швидкодія	Обмеженість у налаштуванні та випадках важких моделей

	Б	Гнучкість, легкість у налаштуванні	Складність реалізації, швидкодія

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

Функція F_1:

Програма допускає використання обох варіантів. Можливо використати варіанти А чи Б.

Функція F_2:

Перевагу даємо можливості інтегруватися з іншими пакетами програмного забезпечення. Це варіант А.

Функція F_3:

Реалізація першого вимагає меншу кількість часу, а також є сприятливою для нашої моделі, тому обираємо варіант А.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

$$\mathrm{F_1а} - F_2\mathrm{а} - F_3\mathrm{а}$$
$$\mathrm{F_1б} - F_2\mathrm{а} - F_3\mathrm{а}$$

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- X1 – швидкодія пакету програмного забезпечення;
- X2 – об’єм пам’яті для обчислень та збереження даних;
- X3 – час навчання даних;
- X4 – потенційний об’єм програмного коду.

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

Таблиця 4.2 - Основні параметри програмного продукту

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія пакету програмного забезпечення	X1	оп/мс	125	150	180
Об’єм пам’яті	X2	Мб	256	128	64
Час попередньої обробки даних	X3	мс	2000	1200	700
Потенційний об’єм програмного коду	X4	кількість рядків коду	2100	1500	1200

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

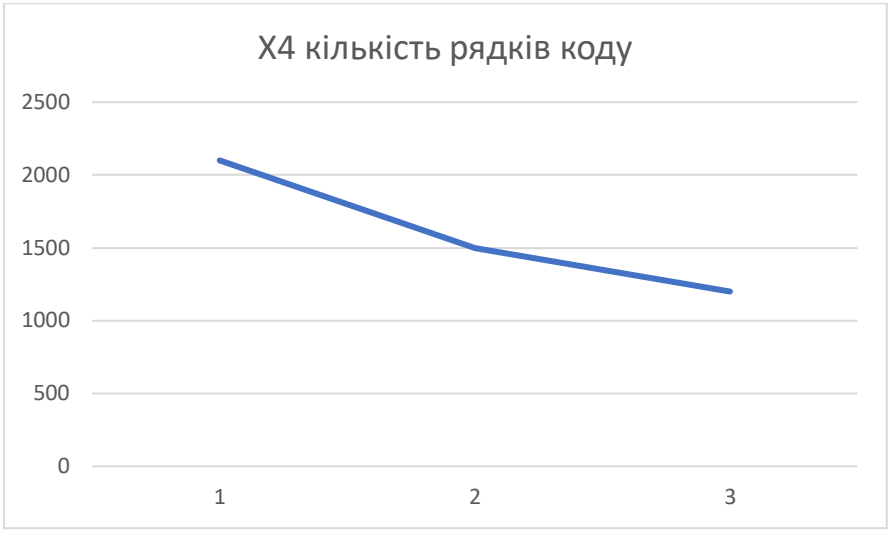

Рисунок 4.2 – X1, швидкодія пакету програмного забезпечення

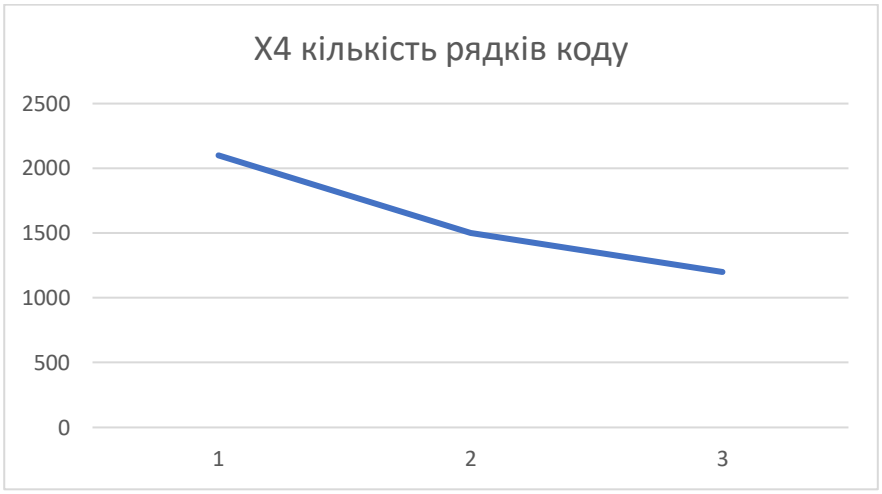

Рисунок 4.3 – X2, об’єм пам’яті

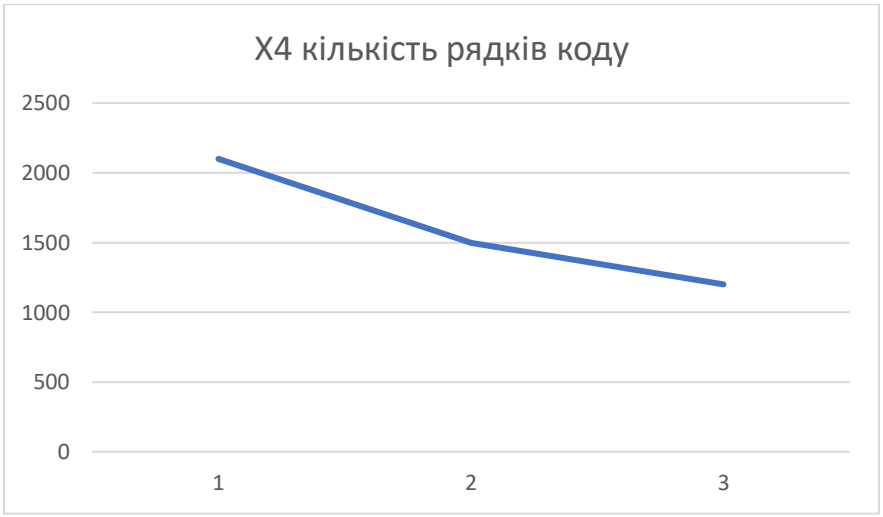

Рисунок 4.4 – X3, час попередньої обробки даних

Рисунок 4.5 – X4, потенційний об’єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення та аналізу, кожен експерт оцінює важливість кожного параметра для розробки програмного продукту, який забезпечує найточніші результати при знаходженні параметрів моделей адаптивного прогнозування та обчисленні прогнозних значень. Визначення значимості кожного параметра здійснюється шляхом попарного порівняння, проводимого експертною комісією з 7 учасників. Процес включає в себе присвоєння різних рангів для визначення рівня значимості параметра, перевірку придатності експертних оцінок для подальшого використання, визначення оцінки попарного пріоритету параметрів та обробку результатів для визначення коефіцієнта значимості. Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 - Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхи-лення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			

Продовження таблиці 4.3

X1	швидкодія пакету програмного забезпечення	Оп/мс	2	3	2	1	1	1	2	12	-5,5	30,25
X2	Об’єм пам’яті	Mb	3	2	3	4	4	3	4	23	5,5	30,25
X3	Час попередньої обробки даних	мс	1	1	1	2	2	2	1	10	-7,5	56,25
X4	Потенційний об’єм програмного коду	Кількість рядків коду	4	4	4	3	3	4	3	25	7,5	56,25
	Разом		10	10	10	10	10	10	10	70	0	173

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

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

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \quad (4.1)$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17{,}5 \quad (4.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T. \tag{4.3}$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 173. \tag{4.4}$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 173}{7^2(4^3 - 4)} = 0{,}706 > W_k \approx 0{,}67. \tag{4.5}$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

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

Таблиця 4.4 - Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	>	<	<	<	<	<	<	0,5
X1 і X3	>	>	>	<	<	<	>	>	1,5
X1 і X4	<	<	<	<	<	<	<	<	0,5
X2 і X3	>	>	>	>	>	>	>	>	1,5
X2 і X4	<	<	<	>	>	<	>	<	0,5
X3 і X4	<	<	<	<	<	<	<	<	0,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}. \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю A=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \quad (4.8)$$

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

$$K_{\text{вi}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \quad (4.9)$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \quad (4.10)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 - Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{вi}}$	b_i^1	$K_{\text{вi}}^1$	b_i^2	$K_{\text{вi}}^2$

Продовження таблиці 4.5

X1	1	0,5	1,5	0,5	3,5	0,22	12,25	0,21	41,875	0,2
X2	1,5	1	1,5	0,5	4,5	0,28	16,25	0,28	59,125	0,28
X3	0,5	0,5	1	0,5	2,5	0,16	9,25	0,16	34,125	0,16
X4	1,5	1,5	1,5	1	5,5	0,34	21,25	0,35	77,875	0,36
Всього:					16	1	59	1	213	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}\,, \qquad (4.11)$$

де n – кількість параметрів; $K_{вi}$– коефіцієнт вагомості i–го параметра; B_i – оцінка i–го параметра в балах.

Таблиця 4.6 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F_1	А	X1	150	4	0,2	0,8
	Б	X2	64	2	0,28	0,56
F_2	А	X3	1200	7	0,16	1,12
F_3	А	X4	1700	1	0,36	0,36

За даними з таблиці 4.6 за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \quad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 0{,}8 + 1{,}12 + 0{,}36 = 2{,}28 ;$$

$$K_{K2} = 0{,}56 + 1{,}12 + 0{,}36 = 2{,}04 .$$

Як видно з розрахунків, кращим є 1 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

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

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$\text{Т}_\text{О} = \text{Т}_\text{Р} \cdot \text{К}_\text{П} \cdot \text{К}_\text{СК} \cdot \text{К}_\text{М} \cdot \text{К}_\text{СТ} \cdot \text{К}_\text{СТ.М}, \quad (4.13)$$

де T_P – трудомісткість розробки ПП;

$К_П$ – поправочний коефіцієнт;

$К_{СК}$ – коефіцієнт на складність вхідної інформації;

$К_М$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P =60 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 60 \cdot 1.8 \cdot 0.9 = 81,6 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто T_P =27 людино-днів, $К_П$ =0.9, $К_{СК}$ = 1, $К_{СТ}$ =0.8:

$$T_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (81,6 + 19.44 + 4.8 + 19.44) \cdot 8 = 1002,24 \text{ людино-годин.}$$

$$T_П = (81,6 + 19.44 + 6.91 + 19.44) \cdot 8 = 1019,12 \quad \text{людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь три програмісти з окладом 20500 грн., один аналітик з окладом 25000. Визначимо середню зарплату за годину за формулою:

$$С_ч = \frac{М}{T_m \cdot t} \text{ грн.}, \qquad (4.14)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$С_ч = \frac{20500 + 20500 + 20500 + 25000}{4 \cdot 21 \cdot 8} = 128,72 \quad \text{грн.} \qquad (4.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д\,, \qquad (4.16)$$

де $С_ч$– величина погодинної оплати праці програміста;

$Т_i$ – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 128,72 \cdot 1002,24 \cdot 1.2 = 154810$ грн.

II. $С_{ЗП} = 128,72 \cdot 1019,12 \cdot 1.2 = 157417,35$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 154810 \cdot 0.22 = 34058,2$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 157417,35 \cdot 0.22 = 34631,82$ грн.

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

Так як одна ЕОМ обслуговує одного програміста з окладом 20500 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 20500 \cdot 0,2 = 49200 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 49200 \cdot (1 + 0.2) = 59040 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 59040 \cdot 0,22 = 12988,8 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 35% та вартості ЕОМ – 35000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.35 \cdot 35000 = 14087,5 \text{ грн.,}$$

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

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 35000 \cdot 0.05 = 2012,5 \quad \text{грн.},$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 12 – 16) \cdot 8 \cdot 0.6 =$$

$$= 1118,4 \text{ години},$$

де $Д_К$ – календарна кількість днів у році;

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

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

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

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1118,4 \cdot 0,4 \cdot 0,55 \cdot 4,86889 = 1197,98 \quad \text{грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

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

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 35000 \cdot 0,67 = 23450 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$C_{ЕКС} = C_{ЗП} + C_{ВІД} + C_{А} + C_{Р} + C_{ЕЛ} + C_{Н}, \quad (4.17)$$

$C_{ЕКС}$ = 59040 + 12988,8 + 14087,5 + 2012,5 + 1197,98 + 23450 = 112776,78 грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{М\text{-}Г} = C_{ЕКС} / T_{ЕФ} = 112776,78 / 1118,4 = 100,83 \text{ грн/год.}$$

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

$$C_{М} = C_{М-Г} \cdot T, \quad (4.18)$$

I. $C_{М} = 100,83 \cdot 1002,24 = 101063,48$ грн.

II. $C_{М} = 100,83 \cdot 1019,12 = 102765,62$ грн.

Накладні витрати складають 67% від заробітної плати:

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

I. $C_{Н} = 154810 \cdot 0,67 = 103722,7$ грн.

II. $C_{Н} = 157417,35 \cdot 0,67 = 105469,39$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_{М} + C_{Н}, \quad (4.20)$$

I. $С_{ПП}$ = 154810 + 34058,2 + 101063,48 + 103722,7 = 393653,68 грн.

II. $С_{ПП}$ = 157417,35 + 34631,82 + 102765,62 + 105469,39 = 400283,36 грн.

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР\,j} = К_{Кj} / С_{Фj}, \quad (4.21)$$

$$К_{ТЕР1} = 2,28 / 393653,68 = 5,79 \cdot 10^{-6},$$

$$К_{ТЕР2} = 2,04 / 400283,36 = 5,1 \cdot 10^{-6}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1}$= 5,79 $\cdot 10^{-6}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР}$ = 5,79 $\cdot 10^{-6}$.

Цей варіант реалізації програмного продукту має такі параметри:

- Бібліотека для роботи з нейронними мережами – Keras;
- Бібліотека для роботи з часовими рядами – Statsmodels;
- Використання Sklearn для аналізу якості моделі.

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

4.8 Висновки по розділу 4

Було проведено повний функціонально-вартісний аналіз програмного продукту. Визначено та проведено оцінку основних функцій програмного продукту. Визначено параметри, які характеризують програмний продукт. Проведено експертне оцінювання параметрів та аналіз якості варіантів реалізації функцій.

Також було проведено економічний аналіз варіантів розробки, це включає в себе оцінку трудомісткості, витрат на заробітну плату, амортизацію та інші витрати.

На основі проведеного дослідження було обрано варіант реалізації програмного продукту, що дозволяє швидко реалізувати доступний, якісний функціонал для роботи.

ВИСНОВКИ

Сьогодення нам показує, що звичний усьому світу ринок для інвестування втрачає актуальність, в свою чергу ринок криптовалют дає змогу людям, що мають знання у інформаційних технологіях, інвестувати та отримувати величезні прибутки.

Інвестування в криптовалютні монети набрало дуже великих обертів, тому розробка комплексної моделі інвестування на цьому ринку є дуже вагомим внеском у розвиток цієї сфери та суміжних з нею напрямків, оскільки ринок криптовалют є хитким та непередбачуваним, що створює складнощі для інвесторів та трейдерів, які на ньому працюють.

В даній роботі було розглянуто предметну область заданої теми, задля повноти розуміння задачі та основних цілей, які стоять для реалізації подібної моделі. Огляд математичних основ розв’язання поставленої задачі зайняв вагоме місце у роботі, оскільки дозволив оглянути теоретичну базу і методологію дослідження інвестиційних рішень. Основною метою наявності цього розділу є обґрунтування і пояснення математичних підходів, моделей та методів.

Був реалізований програмний продукт за допомогою мови програмування Python. Він являє собою застосунок, що дозволяє користувачу зручно скористуватися розробленою комплексною моделлю інвестування та отримати однозначну відповідь на запитання чи варто інвестувати та який можливий прибуток він зможе отримати. Крім того, програмний продукт також дозволяє користувачеві наочно побачити результати роботи фільтрації, прогнозування та передбачення за допомогою графіків. Візуалізація дозволяє інвестору спиратися не тільки на числа, а й наочно оцінити історичні значення криптовалюти та побачити яким є прогнозований результат інвестицій, це дозволить йому зробити більш якісний висновок про цінність вкладення коштів у ту, чи іншу криптовалютну монету.

У четвертому розділі був наведений функціонально-вартісний аналіз програмного продукту. Наявність цього розділу у дипломній роботі дозволила нам оцінити які альтернативні підходи до розробки є, та зробити висновок про цінність кожного з них.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Haykin S. Kalman Filtering and Neural Networks, 2001. 392 p.
2. Julier S.J., Uhlmann J.K. Unscented Filtering and Nonlinear Estimation, 2004. 422 p.
3. Pykalman documentation URL: https://pykalman.github.io/
4. FilterPy documentation URL: https://filterpy.readthedocs.io/en/latest/
5. Hyndman R.J. Forecasting: Principles and Practice, 2018. 405 p.
6. Nelson C.R. Applied Time Series Analysis for Managerial Forecasting, 1994. 231 p.
7. Brockwell P.J. and Davis R.A. Introduction to Time Series and Forecasting, 2002. 255 p.
8. LeCun Y., Bengio Y., Hinton G. Deep learning, 2015. 444 p.
9. Graves A. Supervised sequence labelling with recurrent neural networks, 2008. 137 p.
10. Keras documentation URL: https://keras.io/
11. Numpy documentation URL: https://numpy.org/
12. Tensorflow documentation URL: https://www.tensorflow.org/
13. International journal of forecasting, Another look at measures of forecast accuracy, 2006. 679 p.
14. Journal of the Royal Statistical Society: Series A (General), Accuracy of forecasting: an empirical investigation, 1979. 142 p.
15. Sklearn documentation URL: https://scikit-learn.org/stable/
16. Journal of the American Statistical Association, Forecasting time series with complex seasonal patterns using exponential smoothing, 2011. 106 p.
17. Streamlit documentation URL: https://streamlit.io/

ДОДАТОК А

data_fetcher.py

```
from tsai.all import *
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from constants import trainTestSplitDate

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

class DataFetcher:
    def __init__(self):
        self.COINS = ["Aave", "BinanceCoin", "Bitcoin", "Cardano", "ChainLink",
"Cosmos", "CryptocomCoin", "Dogecoin", "EOS", "Ethereum", "Iota", "Litecoin",
"Monero", "NEM", "Polkadot", "Solana", "Stellar", "Tether","Tron", "Uniswap",
"USDCoin", "WrappedBitcoin", "XRP"]
        self.COINS_MAP = {coin: coin for coin in self.COINS}
        self.FIELDS = ["High","Low","Open","Close","Volume","Marketcap"]
        self.getData()

    def getCoinDF(self, coinName):
        return getattr(self, self.COINS_MAP[coinName])

    def setCoinDF(self, coinName, df):
        setattr(self, coinName, df)

    def getData(self):
        for coin in self.COINS:
            setattr(self, coin, pd.read_csv(f'./coins/coin_{coin}.csv'))
            setattr(getattr(self,                coin),                "index",
pd.to_datetime(getattr(getattr(self, coin),'Date'), format='%Y-%m-%d'))
            del getattr(self, coin)['Date']

    def getTrainTestSplitDf(self, coinName, trainSize):
        coin = self.getCoinDF(coinName)
        train_size = int(len(coin) * trainSize / 100)
        train = coin[:train_size]
        test = coin[train_size:]
```

```
        return [train, test]

    def getTrainTestSplit(self, coinName, field):
        [train, test] =  self.getTrainTestSplitDf(coinName, trainSize=95)
        return [train[field], test[field]]
```

filterer.py

```
import numpy as np
from pykalman import KalmanFilter
from pykalman import UnscentedKalmanFilter
from scipy.signal import windows, convolve
from filterpy.kalman import UnscentedKalmanFilter
from filterpy.kalman import JulierSigmaPoints
from filterpy.kalman import ExtendedKalmanFilter



class Filterer:
    FILTERS = ["Kalman", "ParzenWindow", "UKF", "EKF"]
    def __init__(self,
                 # ! Kalman props
                 kalman_initial_state_covariance=np.eye(1),
                 kalman_n_dim_obs=1,

                 # ! Parzen window props
                 window_size=15,
                 std=3,

                 # ! UKF props
                ukf_sigmas_n=2,
                ukf_sigmas_kappa=10,



                 # ! EKF props
                 ekf_initial_state=np.array([0., 0.]),
                 ekf_initial_covariance=np.eye(2) * 0.1,
                 ekf_process_noise_covariance=np.eye(2) * 0.01,
                 ekf_measurement_noise_covariance=np.array([[0.1]])
                 ):

        # ! Kalman filter
```

```
    self.kf = KalmanFilter(
        initial_state_covariance=kalman_initial_state_covariance,
        n_dim_obs=kalman_n_dim_obs
        )

    # ! ParzenWindow filter
    self.kernel = windows.gaussian(window_size, std=std)


    # ! UKF
    sigmas = JulierSigmaPoints(n=ukf_sigmas_n, kappa=ukf_sigmas_kappa)
    def fx(x, dt):
        xout = np.empty_like(x)
        xout[0] = x[1] * dt + x[0]
        xout[1] = x[1]
        return xout

    def hx(x):
        return x[:1]

    # Инициализация фильтра
    self.ukf = UnscentedKalmanFilter(
        dim_x=2,
        dim_z=1,
        dt=1.,
        hx=hx,
        fx=fx,
        points=sigmas
    )

    # ! EKF
    self.ekf = ExtendedKalmanFilter(dim_x=2, dim_z=1)
    self.ekf.x = ekf_initial_state
    self.ekf.P = ekf_initial_covariance
    self.ekf.Q = ekf_process_noise_covariance
    self.ekf.R = ekf_measurement_noise_covariance
    self.ekf.F = np.eye(2)
    self.ekf.H = np.array([[1., 0.]])


def getKalmanFilterData(self, coinDF, field):
    kf = self.kf
    coinField = coinDF[field]
```

```
    noise = np.std(np.diff(coinField))
    kf.observation_covariance = noise ** 2

    df = coinDF.copy()
    filteredField, _ = kf.filter(coinDF[field])
    df[f'{field}_Kalman'] = filteredField
    return df

def getParzenWindowFilterData(self, coinDF, field):
    kernel = self.kernel
    data = coinDF[field]
    smoothed = convolve(data, kernel, mode='same') / sum(kernel)

    df = coinDF.copy()
    df[f'{field}_ParzenWindow'] = smoothed
    return df

def getUKFFilterData(self, coinDF, field):
    data = coinDF[field]

    filtered_data = []

    for z in data.values:
        self.ukf.predict()
        self.ukf.update(z)
        filtered_data.append(self.ukf.x[0])


    df = coinDF.copy()
    df[f'{field}_UKF'] = np.array(filtered_data)

    return df


def getEKFFilterData(self, coinDF, field):
    data = coinDF[field]
    filtered_data = []

    def measurement_update(measurement):
        def HJacobian(x):
            # Jacobian of the measurement function
            return np.array([[1., 0.]])
```

```
            def Hx(x):
                # Expected measurement
                return np.dot(self.ekf.H, x)

            self.ekf.update(z=measurement, HJacobian=HJacobian, Hx=Hx)

        for z in data.values:
            self.ekf.predict()
            measurement_update(z)
            filtered_data.append(self.ekf.x[0])


        df = coinDF.copy()
        df[f'{field}_EKF'] = np.array(filtered_data)
        return df
```

drawer.py

```
import matplotlib.pyplot as plt
import streamlit as st

from data_fetcher import DataFetcher

class Drawer():
    def __init__(self, parent):
        self.parent = parent

    def drawData(self, coinName, fields):
        fig, ax = plt.subplots()
        ax.set_ylabel(f'{coinName} Price')
        ax.set_xlabel('Date')
        ax.tick_params(axis='x', labelrotation=45) # используем tick_params для
поворота меток по оси x
        coinDF = self.parent.getCoinDF(coinName)
        for field in fields:
            ax.plot(coinDF.index, coinDF[field], label=field)
        ax.set_title(coinName)
        ax.legend()
        # plt.show()
        return fig
        # st.pyplot(fig)

    def drawComparison(self, dfs, fields, labels):
```

```
        fig, ax = plt.subplots()
        ax.set_ylabel('Price')
        ax.set_xlabel('Date')
        ax.tick_params(axis='x', rotation=45)
        for i in range(len(dfs)):
            for j in range(len(fields[i])):
                df = dfs[i]
                field = fields[i][j]
                ax.plot(df.index, df[field], label=labels[i][j])
        ax.set_title('Comparison')
        ax.legend()
        # plt.show()
        return fig

    def drawTrainTestSplit(self, train, test):
        plt.plot(train, color = "black")
        plt.plot(test, color = "red")
        plt.ylabel('Price')
        plt.xlabel('Date')
        plt.xticks(rotation=45)
        plt.title("Train/Test split Data")
        plt.show()
```

main.py

```
from data_fetcher import DataFetcher
from drawer import Drawer
from filterer import Filterer
from predictor import Predictor
import streamlit as st
import pandas as pd
import math

def procentageIncome(future_price, current_price):
   return round(((future_price - current_price) / current_price) * 100, 2)

# Базові налаштування вебсторінки
st.set_page_config(
    page_title='Diploma',
    layout='wide'
    )

st.title('Model')
```

```
dataFetcher = DataFetcher()
drawer = Drawer(dataFetcher)
filterer = Filterer()

select_column, _, _ = st.columns(spec=[1, 1, 1])


INVESTMENT_MODELS = ["Short Term", "Long Term", "Investment packages"]
selectedCoinName = None
# NN_MODELS = ["RNN", "CNN", "DFN", "LSTM", "GRU", "Transformer"]
MODELS = ["RNN", "CNN"]

# ALL_MODELS = ["ARMA", "ARIMA", "SARIMA", "RNN", "CNN", "DFN", "LSTM", "GRU",
"Transformer"]

MODELS = ["ARMA", "ARIMA", "SARIMA", "RNN", "CNN", "DFN", "LSTM", "GRU",
"Transformer"]
# DATA_FILTERS = ["Kalman", "ParzenWindow", "UKF", "EKF"]
DATA_FILTERS = ["Kalman", "EKF"]
TRAIN_SIZE = 80
FORECAST_DATA = 7
WINDOW_SIZE = 7
SELECTED_FIELD = "High"
MODELS_DF = {}
MODELS_ERROR = {}

MODEL_PROGRESS_INCREMENT = math.ceil(100/((len(DATA_FILTERS) +
1)*len(MODELS)))


# Bitcoin - ParzenWindow, DFN
# Aave - NONE, RNN
progress_bar = st.progress(0)
selectedInvestmentModel = st.selectbox("Select preferred investment model: ",
options=INVESTMENT_MODELS)
if(selectedInvestmentModel == INVESTMENT_MODELS[0] or selectedInvestmentModel
== INVESTMENT_MODELS[1]):
    Long_Short_Data_Fetcher = DataFetcher()
    Long_Short_Drawer = Drawer(Long_Short_Data_Fetcher)
    Long_Short_Filterer = Filterer()

    if(selectedInvestmentModel == INVESTMENT_MODELS[0]):
```

```
        TRAIN_SIZE = 85
        FORECAST_DATA = st.number_input("Number of days to forecast: ", step=1,
value=7, max_value=30, min_value=1)
        RMSE_ERROR_WEIGHT = 0.5
        SMAPE_ERROR_WEIGHT = 0.4
        DS_ERROR_WEIGHT = 0.1
    else:
        TRAIN_SIZE = 65
        NUMBER_OF_MONTHES = st.number_input("Number of monthes to forecast: ",
step=1, value=1, max_value=12, min_value=1)
        FORECAST_DATA = 30 * NUMBER_OF_MONTHES
        RMSE_ERROR_WEIGHT = 0.15
        SMAPE_ERROR_WEIGHT = 0.35
        DS_ERROR_WEIGHT = 0.5


    selectedCoinName    =    st.selectbox("Select    coin    to    invest:    ",
options=Long_Short_Data_Fetcher.COINS)
    originalCoinDF = Long_Short_Data_Fetcher.getCoinDF(selectedCoinName)
    Long_Short_Predictor        =        Predictor(coinName=selectedCoinName,
parent=Long_Short_Data_Fetcher,                            trainSize=TRAIN_SIZE,
forecast_data=FORECAST_DATA, window_size=WINDOW_SIZE)

    run_button = st.button("Run model")
    if run_button:
        PROGRESS = 0

        FORECAST_DATA_FRAME_OBJECT = {"NONE": {}}
        for model in MODELS:
            getattr(Long_Short_Predictor,
f'getPredictionModel{model}')(f'{SELECTED_FIELD}')
            predictionDF            =            getattr(Long_Short_Predictor,
f'prediction{SELECTED_FIELD}{model}_DF')

            forcastDF             =             getattr(Long_Short_Predictor,
f'forecast{SELECTED_FIELD}{model}_DF')
            FORECAST_DATA_FRAME_OBJECT["NONE"][model] = forcastDF

            MODELS_DF[model] = predictionDF
            MODELS_ERROR[model]                                                =
Long_Short_Predictor.getAllErrors(Long_Short_Predictor.testDF,   predictionDF,
SELECTED_FIELD, "Predictions", window_size=WINDOW_SIZE)
            PROGRESS += MODEL_PROGRESS_INCREMENT
```

```
            progress_bar.progress(PROGRESS)

        print(MODELS_ERROR)
        ModelsErrorsWithoutFilters                                          =
Long_Short_Predictor.getWeightedErrors(MODELS_ERROR,        RMSE_ERROR_WEIGHT,
SMAPE_ERROR_WEIGHT, DS_ERROR_WEIGHT)

        FILTER_ERROR_DATA_OBJECT = {"NONE": ModelsErrorsWithoutFilters}

        for current_filter in DATA_FILTERS:
            modelsDF = {}
            modelsError = {}
            FORECAST_DATA_FRAME_OBJECT[current_filter] = {}
            for model in MODELS:
                filteredCoinDF          =          getattr(Long_Short_Filterer,
f'get{current_filter}FilterData')(originalCoinDF, SELECTED_FIELD)
                originalCoinDF[f'{SELECTED_FIELD}_{current_filter}']        =
filteredCoinDF[f'{SELECTED_FIELD}_{current_filter}']
                Long_Short_Data_Fetcher.setCoinDF(selectedCoinName,
originalCoinDF)
                Current_Filter_Predictor                                    =
Predictor(coinName=selectedCoinName,           parent=Long_Short_Data_Fetcher,
trainSize=TRAIN_SIZE, forecast_data=FORECAST_DATA, window_size=WINDOW_SIZE)

                getattr(Current_Filter_Predictor,
f'getPredictionModel{model}')(f'{SELECTED_FIELD}_{current_filter}')
                predictionDF        =        getattr(Current_Filter_Predictor,
f'prediction{SELECTED_FIELD}_{current_filter}{model}_DF')
                forcastDF          =          getattr(Current_Filter_Predictor,
f'forecast{SELECTED_FIELD}_{current_filter}{model}_DF')

                FORECAST_DATA_FRAME_OBJECT[current_filter][model] = forcastDF

                modelsDF[model] = predictionDF
                modelsError[model]                                          =
Current_Filter_Predictor.getAllErrors(Current_Filter_Predictor.testDF,
predictionDF, SELECTED_FIELD, "Predictions", window_size=WINDOW_SIZE)
                PROGRESS += MODEL_PROGRESS_INCREMENT
                if(PROGRESS > 100):
                    progress_bar.progress(100)
                else:
                    progress_bar.progress(PROGRESS)
            print(modelsError)
```

```
            ModelsErrorsWithCurrentFilter                                    =
Current_Filter_Predictor.getWeightedErrors(modelsError,     RMSE_ERROR_WEIGHT,
SMAPE_ERROR_WEIGHT, DS_ERROR_WEIGHT)
            FILTER_ERROR_DATA_OBJECT[current_filter]                         =
ModelsErrorsWithCurrentFilter

        print(FILTER_ERROR_DATA_OBJECT)
        bestFilter,                          bestModel                       =
Long_Short_Predictor.getBestFilterAndModelCombination(FILTER_ERROR_DATA_OBJEC
T)
        print(bestFilter, bestModel)

        procent                                                              =
procentageIncome(FORECAST_DATA_FRAME_OBJECT[bestFilter][bestModel]["Forecast"
].iloc[-1], Current_Filter_Predictor.testDF[SELECTED_FIELD].iloc[-1])
        if(selectedInvestmentModel == INVESTMENT_MODELS[0]):
            if(procent < 0):
                st.warning(f'If  you  invest  today  in  {selectedCoinName}  in
{FORECAST_DATA} days you will lose {procent}% of your investment budget')
            else:
                st.success(f'If  you  invest  today  in  {selectedCoinName}  in
{FORECAST_DATA} days you will earn {procent}% of your investment budget')
        else:
            if(procent < 0):
                st.warning(f'If  you  invest  today  in  {selectedCoinName}  in
{NUMBER_OF_MONTHES} monthes you will lose {procent}% of your investment budget')
            else:
                st.success(f'If  you  invest  today  in  {selectedCoinName}  in
{NUMBER_OF_MONTHES} monthes you will earn {procent}% of your investment budget')


button_optoins = ["Data visualization", "Data filtration visualization", "Data
prediction visualization", "Data prediction with filtration visualization",
"Data forecasting visualization"]

st.title('Visualization')
selected_button = st.selectbox('Select visualization', options=button_optoins)

if(selected_button == button_optoins[0]):
    selectedCoin1       =       st.selectbox(label="Select       coin:       ",
options=dataFetcher.COINS, key="selectCoin1")
    selectedFields1     =     st.multiselect(label="Select     fields:     ",
options=dataFetcher.FIELDS, default=["High"], key="selectFieldDraw")
```

```
    if(st.button("Visualize data")):
        if(len(selectedFields1) == 0):
            st.error("Field selection is required")
        else:
            st.pyplot(drawer.drawData(selectedCoin1, selectedFields1))
elif(selected_button == button_optoins[1]):
    selectedCoin2       =       st.selectbox(label="Select       coin:       ",
options=dataFetcher.COINS, key="selectCoin2")
    selectedFilters2     =     st.multiselect(label="Select     filter:     ",
options=filterer.FILTERS, default="UKF", key="selectFilter2")
    selectedField2      =      st.selectbox(label="Select      field:      ",
options=dataFetcher.FIELDS, key='selectField2')

    if(st.button("Visualize data filtration")):
        df2 = dataFetcher.getCoinDF(selectedCoin2)
        filteredFields2 = []
        filteredLables2 = []
        if(len(selectedFilters2) != 0):
            for selectedFilter2 in selectedFilters2:
                filteredCoinDF               =               getattr(filterer,
f'get{selectedFilter2}FilterData')(df2, selectedField2)
                df2[f'{selectedField2}_{selectedFilter2}']                    =
filteredCoinDF[f'{selectedField2}_{selectedFilter2}']
                filteredFields2.append(f'{selectedField2}_{selectedFilter2}')
                filteredLables2.append(f'Filtered              {selectedCoin2}
{selectedField2} using {selectedFilter2}')
                dataFetcher.setCoinDF(selectedCoin2, df2)
            figur = drawer.drawComparison(
                [df2, filteredCoinDF],
                [[f'{selectedField2}'], filteredFields2],
                [[f'{selectedCoin2} {selectedField2}'], filteredLables2]
                )
            st.pyplot(figur)
        else:
            st.error("Filter selection is required")
elif(selected_button == button_optoins[2]):
    WINDOW_SIZE = 7
    selectedCoin3       =       st.selectbox(label="Select       coin:       ",
options=dataFetcher.COINS, key="selectCoin3")
    selectedPredictionModel3 = st.selectbox(label="Select prediction model",
options=Predictor.PREDICTION_MODELS, index=4, key="selectModel3")
```

```
    selectedField3 = st.selectbox(label="Select field: ", options=dataFetcher.FIELDS, key='selectField3')
    selectedTrainSize3 = st.selectbox(label="Select train sample size %: ", options=[i for i in range(97, 1, -1)], index=4,  key="selectTrainSize3")

    if(st.button("Visualize data prediction")):
        df3 = dataFetcher.getCoinDF(selectedCoin3)
        coinPredictor3 = Predictor(coinName=selectedCoin3, parent=dataFetcher, trainSize=selectedTrainSize3, window_size=WINDOW_SIZE)

        getattr(coinPredictor3, f'getPredictionModel{selectedPredictionModel3}')(selectedField3)
        predictionDF3 = getattr(coinPredictor3, f'prediction{selectedField3}{selectedPredictionModel3}_DF')

        print("coinPredictor3.testDF.head", coinPredictor3.testDF)
        print('predictionDF3["Predictions"]', predictionDF3["Predictions"])
        errorRMSE = coinPredictor3.getErrorRMSE(coinPredictor3.testDF, predictionDF3, selectedField3, "Predictions", WINDOW_SIZE)
        errorSMAPE = coinPredictor3.getErrorSMAPE(coinPredictor3.testDF, predictionDF3, selectedField3, "Predictions", WINDOW_SIZE)
        errorDS = coinPredictor3.getErrorDS(coinPredictor3.testDF, predictionDF3, selectedField3, "Predictions", WINDOW_SIZE)


        figur = drawer.drawComparison(
            [df3, predictionDF3],
            [[{selectedField3}], ["Predictions"]],
            [[f'{selectedCoin3} {selectedField3}'], [f'{selectedCoin3} {selectedField3} prediction {selectedPredictionModel3} model']]
        )
        st.pyplot(figur)
        st.text(f'RMSE ERROR: {errorRMSE}')
        st.text(f'SMAPE ERROR: {errorSMAPE}')
        st.text(f'DS ERROR: {errorDS}')

elif(selected_button == button_optoins[3]):
    WINDOW_SIZE = 7
    error_table = {}

    selectedCoin4 = st.selectbox(label="Select coin: ", options=dataFetcher.COINS, key="selectCoin4")
```

```
    selectedPredictionModel4 = st.selectbox(label="Select prediction model",
options=Predictor.PREDICTION_MODELS, index=4, key="selectModel4")

    optionFilters4 = filterer.FILTERS.copy()
    optionFilters4.append("NONE")
    selectedFilters4     =     st.multiselect(label="Select     filter:     ",
options=optionFilters4, default="UKF", key="selectFilter4")
    selectedField4      =      st.selectbox(label="Select      field:      ",
options=dataFetcher.FIELDS, key='selectField4')
    selectedTrainSize4 = st.selectbox(label="Select train sample size %: ",
options=[i for i in range(99, 1, -1)], index=4, key="selectTrainSize4")
    if(st.button("Visualize data prediction with filtration")):
        df4 = dataFetcher.getCoinDF(selectedCoin4)
        filteredFields4 = []
        filteredLables4 = []
        predictedFields4 = [f'{selectedField4}']
        predictedLables4 = [f'{selectedCoin4} {selectedField4}']
        if(len(selectedFilters4) != 0):
            for selectedFilter4 in selectedFilters4:
                if(selectedFilter4 != "NONE"):
                    filteredCoinDF            =            getattr(filterer,
f'get{selectedFilter4}FilterData')(df4, selectedField4)
                    df4[f'{selectedField4}_{selectedFilter4}']              =
filteredCoinDF[f'{selectedField4}_{selectedFilter4}']

filteredFields4.append(f'{selectedField4}_{selectedFilter4}')
                    filteredLables4.append(f'Filtered          {selectedCoin4}
{selectedField4} using {selectedFilter4}')
                    dataFetcher.setCoinDF(selectedCoin4, df4)

            coinPredictor4         =         Predictor(coinName=selectedCoin4,
parent=dataFetcher, trainSize=selectedTrainSize4, window_size=WINDOW_SIZE)
            for selectedFilter4 in selectedFilters4:
                if(selectedFilter4 == "NONE"):
                    getattr(coinPredictor4,
f'getPredictionModel{selectedPredictionModel4}')(f'{selectedField4}')
                    predictionDF          =          getattr(coinPredictor4,
f'prediction{selectedField4}{selectedPredictionModel4}_DF')
                    predictedLables4.append(f'Predicted                  using
{selectedPredictionModel4} without filters')
                    forcastDF            =            getattr(coinPredictor4,
f'forecast{selectedField4}{selectedPredictionModel4}_DF')
```

```
                else:
                    getattr(coinPredictor4,
f'getPredictionModel{selectedPredictionModel4}')(f'{selectedField4}_{selected
Filter4}')
                    predictionDF          =          getattr(coinPredictor4,
f'prediction{selectedField4}_{selectedFilter4}{selectedPredictionModel4}_DF')
                    predictedLables4.append(f'Predicted                  using
{selectedPredictionModel4} using data filtered with {selectedFilter4}')
                    forcastDF            =            getattr(coinPredictor4,
f'forecast{selectedField4}_{selectedFilter4}{selectedPredictionModel4}_DF')


                error_table[selectedFilter4] = [
                    coinPredictor4.getErrorRMSE(coinPredictor4.testDF,
predictionDF, selectedField4, "Predictions", WINDOW_SIZE),
                    coinPredictor4.getErrorSMAPE(coinPredictor4.testDF,
predictionDF, selectedField4, "Predictions", WINDOW_SIZE),
                    coinPredictor4.getErrorDS(coinPredictor4.testDF,
predictionDF, selectedField4, "Predictions", WINDOW_SIZE)
                ]
                df4[f'Predictions_{selectedFilter4}']                        =
predictionDF["Predictions"]
                predictedFields4.append(f'Predictions_{selectedFilter4}')
            figur = drawer.drawComparison(
                [df4, filteredCoinDF],
                [predictedFields4, filteredFields4],
                [predictedLables4, filteredLables4]
                )
            st.pyplot(figur)
            df = pd.DataFrame(error_table, index=["RMSE", "SMAPE", "DS"])
            st.table(df)
        else:
            st.error("Filter selection is required")

elif(selected_button == button_optoins[4]):
            WINDOW_SIZE = 7
            error_table = {}

            selectedCoin5     =     st.selectbox(label="Select     coin:     ",
options=dataFetcher.COINS, key="selectCoin5")
            selectedPredictionModel5 = st.selectbox(label="Select prediction
model", options=Predictor.PREDICTION_MODELS, index=4, key="selectModel5")
```

```
            optionFilters = filterer.FILTERS.copy()
            optionFilters.append("NONE")
            selectedFilters5 = st.multiselect(label="Select filter: ",
options=optionFilters, default="UKF", key="selectFilter5")
            selectedField5 = st.selectbox(label="Select field: ",
options=dataFetcher.FIELDS, key='selectField5')
            selectedTrainSize5 = st.selectbox(label="Select train sample size
%: ", options=[i for i in range(99, 1, -1)], index=4, key="selectTrainSize5")
            selectedForecastDays = st.number_input("Number of days to predict:
", step=1, value=7)

            if(st.button("Visualize data forecastiong")):
                df5 = dataFetcher.getCoinDF(selectedCoin5)
                filteredFields5 = []
                filteredLables5 = []
                predictedFields5 = [f'{selectedField5}']
                predictedLables5 = [f'{selectedCoin5} {selectedField5}']
                drawDf = []
                drawFileds = []
                drawLables = []
                if(len(selectedFilters5) != 0):
                    for selectedFilter5 in selectedFilters5:
                        if(selectedFilter5 != "NONE"):
                            filteredCoinDF = getattr(filterer,
f'get{selectedFilter5}FilterData')(df5, selectedField5)
                            df5[f'{selectedField5}_{selectedFilter5}'] =
filteredCoinDF[f'{selectedField5}_{selectedFilter5}']

filteredFields5.append(f'{selectedField5}_{selectedFilter5}')
                            filteredLables5.append(f'Filtered {selectedCoin5}
{selectedField5} using {selectedFilter5}')
                            dataFetcher.setCoinDF(selectedCoin5, df5)

                    coinPredictor5 = Predictor(coinName=selectedCoin5,
parent=dataFetcher, trainSize=selectedTrainSize5,
forecast_data=selectedForecastDays, window_size=WINDOW_SIZE)
                    for selectedFilter5 in selectedFilters5:
                        print(selectedFilter5)
                        if(selectedFilter5 == "NONE"):
                            getattr(coinPredictor5,
f'getPredictionModel{selectedPredictionModel5}')(f'{selectedField5}')
                            predictionDF = getattr(coinPredictor5,
f'prediction{selectedField5}{selectedPredictionModel5}_DF')
```

```
                            predictedLables5.append(f'Predicted using
{selectedPredictionModel5} without filters')
                            forcastDF = getattr(coinPredictor5,
f'forecast{selectedField5}{selectedPredictionModel5}_DF')
                            drawLables.append([f'Forecast for {selectedField5}
using {selectedPredictionModel5} without filters for next
{selectedForecastDays} days'])

                        else:
                            getattr(coinPredictor5,
f'getPredictionModel{selectedPredictionModel5}')(f'{selectedField5}_{selected
Filter5}')
                            predictionDF = getattr(coinPredictor5,
f'prediction{selectedField5}_{selectedFilter5}{selectedPredictionModel5}_DF')
                            predictedLables5.append(f'Predicted using
{selectedPredictionModel5} using data filtered with {selectedFilter5}')
                            forcastDF = getattr(coinPredictor5,
f'forecast{selectedField5}_{selectedFilter5}{selectedPredictionModel5}_DF')
                            drawLables.append([f'Forecast for {selectedField5}
using {selectedPredictionModel5} using data filtered with {selectedFilter5} for
next {selectedForecastDays} days'])
                        drawDf.append(forcastDF)
                        drawFileds.append(["Forecast"])
                        df5[f'Predictions_{selectedFilter5}'] =
predictionDF["Predictions"]

predictedFields5.append(f'Predictions_{selectedFilter5}')

                        error_table[selectedFilter5] = [

coinPredictor5.getErrorRMSE(coinPredictor5.testDF, predictionDF,
selectedField5, "Predictions", WINDOW_SIZE),

coinPredictor5.getErrorSMAPE(coinPredictor5.testDF, predictionDF,
selectedField5, "Predictions", WINDOW_SIZE),
                            coinPredictor5.getErrorDS(coinPredictor5.testDF,
predictionDF, selectedField5, "Predictions", WINDOW_SIZE)
                        ]

                    drawDf[:0] = [df5, filteredCoinDF]
                    drawFileds[:0] = [predictedFields5, filteredFields5]
                    drawLables[:0] = [predictedLables5, filteredLables5]
                    figur = drawer.drawComparison(
```

```
                    drawDf,
                    drawFileds,
                    drawLables
                    )
                st.pyplot(figur)

                df = pd.DataFrame(error_table, index=["RMSE", "SMAPE",
"DS"])
                st.table(df)
            else:
                st.error("Filter selection is required")
```