НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут прикладного системного аналізу

Кафедра штучного інтелекту

До захисту допущено:

В. о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи і методи штучного інтелекту»

спеціальності 122 «Комп'ютерні науки»

на тему: «Аналіз та прогнозування курсу криптовалют з використанням методів штучного інтелекту»

Виконав:
студент IV курсу, групи КІ-01
Кочедиков Богдан Олександрович _____

Керівник:
доцент, к.ф.-м.н., доцент
Шубенкова Ірина Анатоліївна _____

Консультант з економічного розділу:
професор, д.е.н., проф.кафедкри ЕК ФММ
Шевчук Олена Анатоліївна _____

Консультант з нормоконтролю:
фахівець першої категорії кафедри ШІ,
Кравець Павло Володимирович _____

Рецензент:
професор, д.т.н., доцент
Зайченко Олена Юріївна _____

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

Київ – 2024 року

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

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ
В. о. завідувачки кафедри
_____Ірина ДЖИГИРЕЙ
«31» січня 2024 р.

ЗАВДАННЯ

на дипломну роботу студенту

Кочедикову Богдану Олександровичу

1. Тема роботи «Аналіз та прогнозування курсу криптовалют з використанням методів штучного інтелекту», керівник роботи Шубенкова Ірина Анатоліївна, доцент кафедри математичних методів системного аналізу НН ІПСА, к.ф.-м.н., затверджені наказом по університету від «31» травня 2024 р. № 2240-С

2. Термін подання студентом роботи «10» червня 2024 року.

3. Вихідні дані до роботи: прогнозування цін криптовалютних активів

4. Зміст роботи: аналіз предметної області дослідження, вибір доцільних методів та моделей для роботи, прогнозування цін за допомогою обраного методу та оцінка отриманих результатів.

5. Перелік ілюстративного матеріалу: використані метрики, опис набору даних, принцип роботи та архітектура програмного продукту, результати роботи програми, схеми роботи алгоритмів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук Олена Анатоліївна професор, д.е.н., доцент ЕК ФММ		

7. Дата видачі завдання «05» лютого 2024 року.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Вивчення літератури та джерел за поставленою темою роботи	15.04.2024	Виконано
2	Підготовка першого розділу	01.05.2024	Виконано
3	Підготовка другого розділу	07.05.2024	Виконано
4	Розробка програмного продукту	15.05.2024	Виконано
5	Підготовка третього розділу	22.05.2024	Виконано
6	Підготовка економічної частини	26.05.2024	Виконано
7	Оформлення розділів відповідно до нормоконтролю	30.05.2024	Виконано
8	Підготовка доповіді	04.06.2024	Виконано

Студент Богдан КОЧЕДИКОВ

Керівник Ірина ШУБЕНКОВА

РЕФЕРАТ

Дипломна робота: 102 с., 14 рис., 8 табл., 45 посилань, 1 додаток.

ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ МЕТОДАМИ ШТУЧНОГО ІНТЕЛЕКТУ, ЧАСОВІ РЯДИ, СТАТИСТИЧНІ МЕТОДИ, ГРАДІЄНТНИЙ БУСТИНГ, НЕЙРОННІ МЕРЕЖІ, LSTM, GRU, ARIMA, LightGBM

Об’єкт дослідження – ціни на криптовалютні активи на ринкових біржах.

Предмет дослідження – моделі та методи штучного інтелекту.

Мета роботи – дослідити, реалізувати, оцінити та порівняти роботу моделей та методів штучного інтелекту в задачі прогнозування цін на криптовалютні активи на біржах та реалізувати відповідне програмне забезпечення для цього.

ABSTRACT

Master's thesis: 102 p., 14 figures, 8 tables, 45 references, 1 appendix.

ANALYSIS AND FORECASTING THE PRICE OF CRYPTOCURRENCIES USING ARTIFICIAL INTELLIGENCE METHODS, TIME SERIES, STATISTICAL METHODS, GRADIENT BOOSTING, NEURAL NETWORKS, LSTM, GRU, ARIMA, LightGBM

The object of the study – prices for cryptocurrency assets on market exchanges.

The subject of research – models and methods of artificial intelligence.

The purpose of the work – to investigate, implement, evaluate, and compare the performance of artificial intelligence models and methods in the task of forecasting cryptocurrency asset prices on exchanges and to implement the appropriate software for this purpose.

ЗМІСТ

ВСТУП

Штучний інтелект став важливим інструментом у багатьох галузях, в тому числі і у фінансовому секторі. Що стосується криптовалютних ринків, то вони займають важливе місце в цьому секторі. З кожним роком криптоактиви стають все цікавішими для крупних інвесторів, трейдерів та біржових і фондових аналітиків, а об’єми торгів на біржах невпинно ростуть, тому точне прогнозування руху ціни криптоактивів стає все актуальнішим. Однак прогнозування курсів криптоактивів є складним завданням через широкий спектр факторів, що на них впливають.

Основна мета цієї дипломної роботи – розробити та оцінити ефективність різних моделей штучного інтелекту в прогнозуванні цін на криптовалюти. Для досягнення цієї мети в дослідженні будуть виконані наступні завдання: проаналізовано сучасні методології прогнозування цін на криптовалюти, розроблено програмне забезпечення, яке використовує різні моделі ШІ пристосовані для цього завдання, та оцінено на практиці їхню ефективність та точність на реальних ринкових даних з криптобірж.

Значення цього дослідження полягає в його практичному застосуванні для інвестиційних стратегій та фінансового аналізу на ринку криптоактивів. Крім того, воно має на меті зробити внесок у розвиток та покращення теорій і методологій прогнозування на ринку криптовалют. Завдяки всебічному аналізу, порівнянню та вдосконаленню моделей штучного інтелекту це дослідження може стати цінним ресурсом у сфері аналізу криптовалютного ринку

РОЗДІЛ 1 ОГЛЯД ТА ДОСЛІДЖЕННЯ ЗАДАЧІ ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ

1.1 Означення криптовалюти та як вони влаштовані

Криптовалюти – це цифрові або віртуальні валюти, які використовують криптографію для забезпечення безпеки, що ускладнює їх підробку або подвійне використання. На відміну від традиційних валют, випущених урядами (тобто фіатних валют) та центральними банками, криптовалюти функціонують на децентралізованих платформах з використанням таких технологій, як блокчейн, який є розподіленим реєстром, що забезпечується розрізненою мережею комп'ютерів, відомих як вузли. Біткойн, який був створений у 2009 році анонімною особою, відомою як Сатоші Накамото, часто називають першою і найвідомішою криптовалютою, але зараз на ринку існують тисячі інших варіантів, включаючи Ethereum, Ripple (XRP), Litecoin та багато інших, і вони всі працюють за певними загальноприйнятими принципами роботи.

1. Технологія блокчейн, яка лежить в основі більшості криптовалют. Блокчейн – це децентралізований реєстр усіх транзакцій у мережі. Блокчейн записує транзакції на багатьох комп'ютерах таким чином, що зареєстровані транзакції не можуть бути змінені заднім числом. Це забезпечує безпеку та прозорість.
2. Децентралізація – на відміну від традиційних фінансових систем, які контролюються центральними органами влади, такими як банки та уряди, криптовалюти працюють на децентралізованих платформах. Це означає, що транзакції та випуск одиниць здійснюються мережею колективно, що дає користувачам більше контролю над своїми фінансами.
3. Майнінг і механізми консенсусу – багато криптовалют, зокрема біткоїн, використовують процес, відомий як майнінг, для створення нових монет і підтвердження транзакцій. Майнінг передбачає

розв'язання складних математичних задач для додавання записів про транзакції до блокчейну. Цей процес вимагає значних обчислювальних потужностей та енергії. Різні механізми консенсусу, такі як Proof of Work (PoW) і Proof of Stake (PoS), гарантують, що транзакції надійно додаються в блокчейн і підтверджують їхню дійсність.

4. Гаманці та ключі – для використання криптовалют людям потрібен цифровий гаманець, в якому зберігаються їхні публічні та приватні ключі. Ці ключі необхідні для здійснення транзакцій. Публічний ключ схожий на номер рахунку, в той час як приватний ключ дозволяє власнику відправляти криптовалюту зі свого гаманця. Дуже важливо зберігати приватні ключі в безпеці; якщо їх втратити, пов'язані з ними кошти неможливо відновити.
5. Транзакції – криптовалютні транзакції є одноранговими, тобто відбуваються безпосередньо між користувачами без посередника. Коли транзакція здійснюється, вона транслюється в мережу, перевіряється майнерами або валідаторами, а потім додається до блокчейну. Цей процес може зайняти від кількох секунд до кількох хвилин, залежно від криптовалюти та мережевого трафіку.
6. Смарт-контракти – деякі криптовалюти, такі як Ethereum, підтримують смарт-контракти - самодостатні контракти, умови яких безпосередньо записані в коді. Ці контракти автоматично виконують і забезпечують виконання узгоджених умов при виконанні певних умов, що дозволяє здійснювати більш складні фінансові операції, такі як кредитування, страхування і децентралізовані додатки (DApps), без посередників.

Хоч криптовалюти і є інноваційними та високотехнологічними вони мають як свої переваги, так і не залишені недоліками.

З позитивного боку, криптовалюти забезпечують децентралізований і безпечний спосіб здійснення транзакцій, усуваючи потребу в посередниках,

таких як банки, і, відповідно, знижуючи ризик шахрайства. Крім того, криптовалюти забезпечують більшу доступність фінансових послуг, особливо в регіонах з недостатньо розвиненою банківською системою, дозволяючи особам, які раніше не мали доступу до традиційних банківських систем, брати участь у фінансовій екосистемі. Потенціал високої прибутковості інвестицій завдяки волатильності криптовалют також приваблює значну кількість інвесторів, які шукають прибуткові можливості.

Однак важливо визнати, що криптовалюти не позбавлені значних недоліків. Притаманна цим цифровим активам волатильність може призвести до значних фінансових втрат, що становить значний ризик для інвесторів. Відсутність регулювання на ринку криптовалют викликає значне занепокоєння щодо безпеки та шахрайства. Крім того, анонімність криптовалютних транзакцій асоціюється з незаконною діяльністю, включаючи відмивання грошей і торгівлю наркотиками, що створює складний виклик для правоохоронних органів у всьому світі. Крім того, значне споживання енергії, пов'язане з майнінгом криптовалют, викликає нагальні екологічні проблеми, що сприяють постійним дебатам про стійкість цифрових валют.

У світлі цих міркувань зацікавленим сторонам, включаючи політиків, фінансові установи та інвесторів, вкрай важливо ретельно зважити переваги та ризики, пов'язані з криптовалютами. Постійні дослідження та діалог є важливими для розробки нормативно-правової бази, яка б вирішувала проблеми, пов'язані з цими цифровими активами, одночасно сприяючи інноваціям та забезпечуючи стабільність фінансової системи.

1.2 Історія та сучасний вигляд криптовалют

Криптовалюти беруть свій початок наприкінці 20-го століття, коли почала формуватися концепція анонімних і децентралізованих форм цифрової валюти.

Однією з перших форм криптовалюти була DigiCash, яку представив Девід Чаум у 1989 році. DigiCash мала на меті забезпечити безпечний і приватний спосіб проведення цифрових транзакцій, але зіткнулася зі значними проблемами, включаючи відсутність широкої бази користувачів і регуляторного визнання.

Проте, лише з випуском біткойна у 2009 році анонімною особою або групою осіб, відомою як Сатоші Накамото, ідея криптовалюти набула значної популярності. Біткойн представив концепцію технології блокчейн - децентралізованого реєстру, який безпечно записує всі транзакції через мережу комп'ютерів. Ця проривна технологія забезпечила прозорий і незмінний спосіб управління цифровою валютою, вирішивши проблему подвійних витрат, яка переслідувала попередні спроби.

Успіх біткоїна проклав шлях для розвитку і поширення багатьох інших криптовалют. Ефіріум, запущений у 2015 році Віталіком Бутеріним, розширив можливості технології блокчейн, запровадивши смарт-контракти. Ці самодостатні контракти з умовами угоди, записаними безпосередньо в коді, дозволили автоматизувати складні транзакції та децентралізовані додатки (DApps). Ripple, ще один значний гравець у криптовалютному просторі, націлений на міжнародні грошові перекази, прагнучи забезпечити більш швидкі та економічно ефективні транскордонні платежі.

У наш час вигляд криптовалютного ринку кардинально змінився. Біткойн залишається найвідомішою і найціннішою криптовалютою, яку часто називають «цифровим золотом» через її обмежену пропозицію і передбачувану цінність. Однак зараз існують тисячі альтернативних монет (альткоїнів) з різними варіантами використання. Наприклад, Litecoin пропонує швидший час підтвердження транзакцій порівняно з біткоїном, а Cardano фокусується на стійкості та масштабованості завдяки механізму консенсусу з підтвердженням частки.

Прийняття криптовалют вийшло за рамки лише технологічних ентузіастів і охопило основні фінансові установи та роздрібних інвесторів. Такі великі

компанії, як Tesla і Square, інвестували в біткойн, а платіжні системи, такі як PayPal, тепер дозволяють користувачам купувати, зберігати і продавати криптовалюту. Багато компаній приймають криптовалюти як платіжний засіб, визнаючи їхній потенціал для зниження комісій за транзакції та залучення нових клієнтів.

Уряди також вивчають можливості цифрових валют, зокрема цифрових валют центральних банків (CBDC). Ці підтримувані державою цифрові валюти мають на меті поєднати переваги криптовалют зі стабільністю та довірою до традиційних фіатних валют. Китай, наприклад, активно тестує свій цифровий юань, тоді як інші країни, такі як США та Європейський Союз, перебувають на різних етапах досліджень і розробок.

Поява платформ децентралізованих фінансів (DeFi) свідчить про те, як криптовалюти змінюють традиційні фінансові системи, пропонуючи децентралізовані альтернативи традиційним фінансовим послугам. Платформи DeFi використовують технологію блокчейн для надання таких послуг, як кредитування, запозичення, торгівля та отримання відсотків без посередників. Така демократизація фінансів дозволяє людям мати більший контроль над своїми активами та брати участь у глобальній фінансовій екосистемі.

Загалом, шлях криптовалют від маловідомих цифрових активів до основних фінансових інструментів відображає їхній трансформаційний потенціал. Оскільки технологічний і регуляторний ландшафт продовжує розвиватися, криптовалюти, ймовірно, відіграватимуть дедалі важливішу роль у світовій економіці.

1.3 Актуальність задачі прогнозування курсу криптовалюти

Актуальність прогнозування курсів криптовалют стає все більш значущою в сучасному фінансовому секторі. Оскільки цифрові валюти, такі як

Bitcoin, Ethereum та інші, проникають у масову свідомість, стають все більш популярними, їх ринкова капіталізація в 2023 перевищує трильйони доларів США , а волатильність їхніх ринків створює унікальні виклики та можливості як для інвесторів, трейдерів, так і для політиків. Однією з головних причин важливості точного прогнозування в цій сфері є високий рівень волатильності цін, пов'язаний з цими активами. На відміну від традиційних фінансових інструментів, криптовалюти можуть зазнавати різких цінових коливань протягом коротких періодів, що зумовлено безліччю факторів, включаючи ринкові настрої, регуляторні новини, технологічний прогрес та макроекономічні тенденції.

Ефективні інструменти та моделі прогнозування можуть надати учасникам ринку цінну інформацію, що дозволить їм приймати обґрунтовані рішення та зменшити потенційні ризики. Інвестори, наприклад, можуть використовувати прогнозну аналітику для оптимізації своїх портфелів, встановлення лімітів стоп-лосс і визначення сприятливих моментів для купівлі або продажу активів. Трейдери, особливо ті, що займаються високочастотною торгівлею, можуть скористатися прогнозами ринку в режимі реального часу для розробки надійних торгових стратегій, які використовують короткострокові цінові зміни. У більш широкому масштабі точні прогнози можуть допомогти політикам і регуляторам зрозуміти динаміку ринку, розробити відповідну нормативно-правову базу та виявити системні ризики, які можуть виникнути в криптовалютному секторі, що стрімко розвивається.

Крім того, складні моделі прогнозування, які включають машинне навчання та штучний інтелект, відіграють важливу роль у цій сфері. Ці моделі аналізують величезні масиви даних і виявляють складні закономірності, які люди-аналітики можуть не помітити, тим самим підвищуючи точність прогнозів. Постійний розвиток і вдосконалення цих технологій прокладає шлях до більш надійних і дієвих прогнозів, які мають вирішальне значення для навігації в складних умовах криптовалютних ринків.

Отже, завдання прогнозування курсів криптовалют є ключовим в епоху,

коли цифрові валюти займають значну нішу у світовій фінансовій системі. Це дає учасникам ринку знання, необхідні для орієнтації на нестабільних ринках, допомагає формулювати обґрунтовані інвестиційні та торгові рішення, а також сприяє створенню надійних регуляторних заходів для забезпечення стабільності ринку.

1.4 Огляд основних підходів до прогнозування цін на криптовалюти

Прогнозування цін на криптовалюту - це складна і багатогранна робота, яка вимагає глибокого розуміння різних факторів, що впливають на ринок. Однією з основних особливостей цього завдання є волатильність і непередбачуваність ринку криптовалют. На відміну від традиційних валют, криптовалюти не регулюються центральними банками – тобто є децентралізованими, що призводить до значних цінових коливань на основі ринкових настроїв, спекуляцій та зовнішніх новин.

Крім того, децентралізація криптовалют означає, що на них впливає широкий спектр факторів, включаючи технологічний прогрес, регуляторні зміни та макроекономічні тенденції. Наприклад, запуск нової блокчейн-платформи або оновлення існуючих протоколів може спричинити значні зрушення на ринку. Аналогічно, новини про регуляторні зміни на ключових ринках, таких як Сполучені Штати, Китай або Європейський Союз, можуть мати негайний і глибокий вплив на ціни криптовалют. Крім того, макроекономічні фактори, такі як рівень інфляції, геополітичні події та зміни в глобальних економічних умовах, можуть ще більше ускладнити процес прогнозування. Трейдери також повинні враховувати вплив соціальних мереж і впливових осіб, оскільки один твіт або оголошення можуть кардинально змінити динаміку ринку.

Взаємодія цих різноманітних факторів робить точне прогнозування

складною, але захоплюючою справою, що вимагає як складних аналітичних інструментів, так і гострої інтуїції щодо поведінки ринку.

Зараз існують такі основні методи та підходи до прогнозування цін на криптовалютні активи: фундаментальний аналіз, технічний аналіз, алгоритмічна торгівля, сентимент-аналіз, штучний інтелект.

1.4.1 Фундаментальний аналіз

Фундаментальний аналіз курсів криптовалют передбачає оцінку різних факторів для визначення внутрішньої вартості цифрового активу. На відміну від традиційних фінансових ринків, де фундаментальний аналіз може зосереджуватися на прибутках компаній, співвідношенні ціни до прибутку (P/E) або зростанні доходів, ринок криптовалют вимагає іншого підходу з огляду на його унікальні характеристики.

По-перше, вирішальне значення має технологія, що лежить в його основі. Для будь-якої криптовалюти розуміння технології блокчейну, яку вона використовує, механізмів консенсусу (таких як Proof of Work або Proof of Stake) та рішень щодо масштабування є першочерговими. Наприклад, здатність блокчейну обробляти велику кількість транзакцій в секунду (TPS) може суттєво вплинути на його довгострокову життєздатність і прийняття.

Іншим важливим фактором є команда розробників та спільнота. Криптовалюти з активними та відданими командами, а також сильною підтримкою спільноти, як правило, мають більше шансів на успіх. Оновлення розробки, активність на GitHub та дорожня карта є показниками майбутнього потенціалу проекту.

Корисність криптовалюти – ще один важливий аспект. Криптовалюта, яка пропонує реальні кейси використання, чи то у сфері децентралізованих фінансів (DeFi), управління ланцюгами поставок або цифрової ідентичності, як

правило, має внутрішню цінність, що виходить за рамки спекулятивної торгівлі. Такі проекти, як Ethereum, де розміщуються смарт-контракти і децентралізовані додатки (dApps), продемонстрували значну довговічність і потенціал завдяки широкому спектру застосувань.

Динаміка ринкового попиту та пропозиції також відіграє ключову роль. Загальна пропозиція криптовалюти, рівень інфляції та спосіб розподілу токенів впливають на її ціну. Криптовалюти з обмеженою пропозицією, наприклад, максимальна пропозиція біткоїна становить 21 мільйон монет, можуть створювати дефіцит, що потенційно збільшує вартість у міру зростання попиту.

Нарешті, слід враховувати зовнішні фактори, такі як регуляторні зміни, макроекономічні тенденції та технологічний прогрес. Новини про державне регулювання, інституційні зміни або серйозні порушення безпеки можуть мати значний вплив на ринкові настрої та ціни.

Отже, фундаментальний аналіз курсів криптовалют передбачає багатогранний підхід, який включає оцінку технології, команди розробників, корисності, динаміки ринку та зовнішніх факторів. Розуміючи ці компоненти, інвестори можуть приймати більш обґрунтовані, довгострокові та низькоризикові рішення на цьому нестабільному ринку, що швидко розвивається.

Проте даний підхід має і низку недоліків і складнощів які не можна не брати до уваги.

1. Складність і трудомісткість – проведення фундаментального аналізу вимагає значної кількості часу і зусиль для вивчення різних даних і оцінки різних факторів. Це може стати перешкодою для індивідуальних інвесторів з обмеженими ресурсами.
2. Доступність та надійність даних – доступ до надійних і повних даних про деякі криптовалюти може бути складним. Неточна або неповна інформація може призвести до помилкового аналізу та інвестиційних рішень.

3. Суб'єктивність – деякі аспекти фундаментального аналізу, такі як оцінка команди розробників або корисності криптовалюти, можуть бути суб'єктивними і залежати від точки зору інвестора.
4. Волатильність ринку – незважаючи на ретельний аналіз, непередбачувана природа ринку криптовалют може призвести до значних коливань цін, які важко передбачити. Зовнішні фактори, такі як раптові регуляторні зміни або технологічні збої, можуть швидко змінити ринкові умови.

1.4.2 Технічний аналіз

Технічний аналіз - це методологія, яка використовується для оцінки та прогнозування майбутніх цінових рухів криптовалют, в основному на основі історичних цінових даних та обсягів торгів. На відміну від фундаментального аналізу, який оцінює внутрішню вартість фінансового активу на основі таких факторів, як технологія, команда і ринковий потенціал, технічний аналіз покладається виключно на графіки цін і різні технічні індикатори.

Одним з основних інструментів технічного аналізу є використання свічкових графіків. Кожна свічка відображає рух цін протягом певного періоду часу, показуючи ціни відкриття, закриття, максимуми і мінімуми. Аналізуючи закономірності в цих свічкових формаціях, трейдери можуть визначити потенційні ринкові тенденції та розвороти.

Іншим важливим аспектом є використання ковзних середніх, які допомагають згладжувати цінові дані для формування індикатора, що слідує за трендом. Серед криптовалютних трейдерів особливою популярністю користується ковзаюча середня конвергенція-дивергенція (MACD). Він складається з двох ковзних середніх - швидкої та повільної - і допомагає

виявити зміни імпульсу. Коли швидка ковзаюча середня перетинає повільну, це, як правило, свідчить про бичачий тренд, і навпаки - про ведмежий.

Індекс відносної сили (RSI) – ще один важливий індикатор, який вимірює швидкість і зміну цінових рухів. Він працює за шкалою від 0 до 100, при цьому показники вище 70 вказують на те, що актив перекуплений і може зазнати відкату, в той час як показники нижче 30 вказують на те, що актив перепроданий і може побачити зростання ціни.

Аналіз обсягів додає ще один рівень до технічних торгових стратегій, оскільки він дає уявлення про силу цінового руху. Високі обсяги торгів під час висхідного тренду можуть вказувати на сильний інтерес і стійкість тренду, в той час як низькі обсяги можуть означати потенційний розворот або ослаблення тренду.

Аналізуючи ці та інші елементи, трейдери можуть визначити потенційні точки входу і виходу, управляти ризиками і приймати більш обґрунтовані рішення.

Хоча технічний аналіз дає цінну інформацію, він не позбавлений недоліків. Одним із суттєвих недоліків є його залежність від історичних даних, які не завжди можуть точно передбачити майбутній рух цін, а також волатильність і коливання ринку. На ринкові умови впливає безліч факторів, включаючи новини, регуляторні зміни та макроекономічні тенденції, які можуть бути не відображені в минулих цінових графіках. Крім того, технічний аналіз може бути дуже суб'єктивним, оскільки різні трейдери по-різному інтерпретують одні й ті ж дані, що призводить до суперечливих результатів. Надмірна залежність від індикаторів і моделей також може призвести до того, що трейдери не помітять фундаментальні фактори, які мають вирішальне значення для довгострокових інвестицій. Нарешті, технічний аналіз може забирати багато часу, вимагаючи постійного моніторингу та аналізу цінових графіків, що може бути нездійсненним для всіх трейдерів, особливо тих, хто має інші зобов'язання.

1.4.3 Алгоритмічна торгівля

Алгоритмічна торгівля на криптовалютних ринках передбачає використання автоматизованого програмного забезпечення для здійснення угод на основі заздалегідь визначених критеріїв і торгових стратегій. Це програмне забезпечення може бути запрограмоване на безперервний моніторинг ринкових умов і може приймати рішення про купівлю або продаж активів за частки секунди на основі даних у режимі реального часу. Основна мета алгоритмічної торгівлі – отримати вигоду з неефективності ринку, включаючи розбіжності в цінах і можливості арбітражу, з мінімальним втручанням людини. Розглянемо переваги такого підходу.

1. Швидкість та ефективність – алгоритми можуть обробляти і виконувати угоди набагато швидше, ніж людина-трейдер, що має вирішальне значення в умовах високої волатильності криптовалютного ринку.
2. Точність – автоматизовані системи можуть усунути людські помилки, пов'язані з ручною торгівлею, такі як введення неправильного розміру угоди або неправильний розрахунок рівня ризику.
3. Торгівля без емоцій – алгоритми суворо дотримуються заздалегідь визначених стратегій і не піддаються впливу емоцій, таких як страх або жадібність, що часто призводить до більш дисциплінованої торгівлі.

Також розглянемо основні та найпопулярніші стратегії даного підходу.

1. Маркет-мейкінг – передбачає розміщення ордерів на купівлю і продаж за різними цінами, щоб зафіксувати спред між ними. Ця стратегія спрямована на отримання прибутку від невеликих цінових відхилень.
2. Арбітраж – використовує різницю в цінах на один і той самий актив

на різних біржах. Наприклад, якщо ціна біткоїна на одній біржі нижча, ніж на іншій, алгоритм купує на дешевшій біржі і продає на дорожчій.

3. Відстеження трендів – використовує історичні цінові дані для визначення та відстеження ринкових тенденцій. Ця стратегія має на меті отримати вигоду як від висхідних, так і від низхідних ринкових рухів, купуючи активи, які демонструють висхідний тренд, і продаючи ті, що мають низхідний тренд.

Хоча алгоритмічна торгівля має численні переваги, вона також пов'язана з певними ризиками. Однією з головних проблем є можливість технічних збоїв, таких як програмні збої або проблеми з підключенням, які можуть призвести до значних фінансових втрат. Крім того, алгоритми повинні бути ретельно протестовані в різних ринкових умовах, щоб гарантувати, що вони працюють так, як очікується.

Нарешті, регуляторний контроль – ще один фактор, який слід враховувати, оскільки в різних країнах діють різні закони і правила, що регулюють автоматизовану торгівлю.

1.4.4 Сентимент-аналіз

Сентимент аналіз або аналіз настроїв, який часто називають «видобуванням думок», – це складна методологія, що використовується для визначення настроїв, виражених у тексті. Він передбачає використання передових методів інтелектуального аналізу даних і обробки природної мови для інтерпретації та класифікації емоцій, виражених у текстовому контенті. У сфері криптовалют аналіз настроїв став незамінним інструментом для інвесторів і трейдерів, які прагнуть оцінити ринкові настрої і передбачити рух цін на основі громадської думки і тенденцій в соціальних мережах. Це

передбачає аналіз даних з різних джерел, включаючи Twitter, Reddit, спеціалізовані фінансові форуми і навіть новинні статті, щоб проаналізувати ставлення до різних криптовалют.

Настрої, отримані з цих джерел, можуть мати значний вплив на ринок. Позитивні настрої навколо криптовалюти можуть призвести до збільшення попиту і зростання ціни, тоді як негативні настрої можуть спровокувати розпродаж і подальше падіння ціни. Наприклад, сплеск позитивних твітів про біткойн може спровокувати ажіотажну купівлю, що призведе до зростання ціни, тоді як потік негативних повідомлень про регуляторні заходи може спричинити панічний розпродаж.

Автоматизовані інструменти, що використовують обробку природної мови (NLP) і алгоритми машинного навчання, були розроблені для аналізу настроїв в реальному часі, надаючи трейдерам практичну інформацію для прийняття обґрунтованих рішень. Ці інструменти можуть обробляти величезні обсяги даних майже миттєво, виявляючи тенденції та закономірності, які людина не в змозі виявити вчасно. Досліджуючи ключові слова, фрази та загальний тон контенту, ці інструменти генерують показники настроїв, які допомагають трейдерам інтерпретувати ринкову динаміку.

Наприклад, дослідження, проведене Міттaлом і Гоелом (2012), продемонструвало, що настрої в Твіттері можуть передбачати ціни на фондовому ринку зі значною точністю. Аналогічно, нещодавні досягнення в аналізі настроїв стосовно криптовалют демонструють багатообіцяючі результати в прогнозуванні ринкових тенденцій та поведінки інвесторів. Дослідження показали, що включення аналізу настроїв у торгові стратегії може підвищити прибутковість, надаючи додатковий рівень ринкової інформації [27].

Однак важливо зазначити, що розуміння настроїв не є безпомилковим і повинно використовуватися в поєднанні з іншими аналітичними інструментами. Настрої можуть бути мінливими і залежати від зовнішніх факторів, таких як регуляторні новини, технологічні досягнення та макроекономічні події. Наприклад, позитивний сплеск настроїв, викликаний

новим технологічним оновленням в Ethereum, може бути приглушений, якщо одночасно з'являться негативні новини про посилення регуляторних заходів на великому ринку.

Тому комплексні стратегії, які поєднують сентемент аналіз з іншими підходами, забезпечують більш надійні підходи до торгівлі криптовалютою і можуть досягти більш тонкого і цілісного розуміння ринку, що призводить до прийняття більш обґрунтованих і стратегічно вишідних торгових рішень.

1.4.5 Штучний інтелект

Штучний інтелект (ШІ) продемонстрував значний потенціал у прогнозуванні цін на криптовалюти, використовуючи свою здатність аналізувати величезні обсяги даних і виявляти закономірності, які не під силу людині. Завдяки передовим методам, таким як машинне навчання, нейронні мережі та глибоке навчання, системи ШІ можуть обробляти історичні дані про ціни, ринкові настрої та інші відповідні фактори для прогнозування майбутніх цінових рухів з вищим ступенем точності. Наприклад, алгоритми машинного навчання можуть виявляти тенденції та аномалії, досліджуючи великі масиви даних, нейронні мережі можуть моделювати складні взаємозв'язки між різними змінними, а глибоке навчання може забезпечити багаторівневий підхід до аналізу даних, імітуючи здатність людського мозку вчитися на власному досвіді.

Ці системи штучного інтелекту постійно навчаються і адаптуються до нових даних, з часом покращуючи свої прогностичні можливості. Вони також можуть враховувати інформацію в режимі реального часу, наприклад, оновлення новин, активність у соціальних мережах і геополітичні події, які можуть мати негайний вплив на ціни криптовалют. Така динамічна адаптивність дозволяє ШІ пропонувати більш своєчасні та точні дані порівняно

з традиційними аналітичними методами.

Інтеграція штучного інтелекту в криптовалютну торгівлю може допомогти інвесторам приймати більш обґрунтовані рішення, потенційно збільшуючи прибутковість і знижуючи ризики, пов'язані з високою волатильністю ринку цифрових валют. Використовуючи інструменти на основі штучного інтелекту, трейдери можуть отримати конкурентну перевагу, оптимізувати свої торгові стратегії та ефективніше управляти своїми портфелями. Крім того, штучний інтелект може допомогти в розробці автоматизованих торгових систем, які здійснюють транзакції на основі попередньо встановлених критеріїв, мінімізуючи людські помилки та емоційну упередженість.

Хоча і жодна модель не може гарантувати ідеальну точність через волатильність криптовалют, використання штучного інтелекту дає значні переваги в розробці більш обґрунтованих і стратегічних торгових рішень. Ці технології дозволяють трейдерам залишатися на крок попереду, знижувати ризики і потенційно отримувати вигоду від нових можливостей, що відкриваються у швидкоплинному світі цифрових валют.

Очікується, що з розвитком технології штучного інтелекту її роль у торгівлі криптовалютами стане ще більш важливою, надаючи трейдерам безпрецедентні аналітичні можливості та підвищуючи загальну ефективність ринку. Майбутні досягнення можуть включати в себе включення більш складних алгоритмів, які можуть передбачати довгострокові ринкові тенденції, можливість інтеграції більш широкого спектру джерел даних для ще більш всебічного аналізу, а також розробку все більш інтуїтивно зрозумілих інтерфейсів, які дозволять трейдерам з будь-яким рівнем досвіду використовувати можливості штучного інтелекту. Безперервна еволюція штучного інтелекту здатна революціонізувати ринок криптовалют, зробивши його більш доступним, ефективним і прибутковим для всіх учасників.

1.5 Огляд методів штучного інтелекту для прогнозування цін на криптовалюти

Не всі методи штучного інтелекту підходять для прогнозування цін на криптовалюти, оскільки вони характеризуються високою волатильністю і чутливістю до ринкових настроїв, а це означає, що традиційним статистичним моделям і деяким методам машинного навчання може бути складно ефективно вловити цю динаміку. Хоча такі методи, як лінійна регресія або базові моделі аналізу часових рядів, можуть дати певне розуміння, вони часто не враховують нелінійні та складні закономірності, що спостерігаються на криптовалютних ринках. Тому проаналізуємо та розглялнемо найбільш підходящі методи для поставленої задачі.

1.5.1 LightGBM

Градієнтний бустінг – це просунута техніка машинного навчання, яка будує моделі послідовно. Фундаментальна концепція полягає у створенні нових моделей, які враховують помилки існуючих. Агрегуючи прогнози з декількох слабких моделей, зазвичай дерев рішень, градієнтний бустінг формує надійну прогнозну модель. Кожна нова модель призначена для мінімізації певної функції втрат, наприклад, середньоквадратичної помилки для задач регресії, шляхом концентрації на залишках або помилках попередніх моделей. Ця ітеративна методологія дозволяє градієнтному бустингу досягати успіху в різних завданнях.

Дерева рішень – це популярний метод, який використовується в машинному навчанні та аналізі. По суті, дерева рішень розбивають вхідні дані на гілки в кожному вузлі на основі певних умов, що в кінцевому підсумку

призводить до прийняття рішення або класифікації в кінцевих вузлах, відомих як листя. Ця ієрархічна структура імітує процес прийняття рішень людиною, що робить її інтуїтивно зрозумілою і легкою для інтерпретації. Процес починається з кореневого вузла, що представляє весь набір даних, і просувається через рекурсивні розбиття на основі значень ознак.

LightGBM – це просунутий фреймворк градієнтного бустингу, який використовує алгоритми дерев рішень для ранжування, класифікації та інших завдань машинного навчання. Розроблений компанією Microsoft, LightGBM вирізняється винятковою швидкістю та продуктивністю, особливо при роботі з великими наборами даних з численними характеристиками. Це робить його кращим вибором для багатьох науковців та інженерів, які працюють з великими та складними наборами даних. Його ефективність і масштабованість є ключовими факторами, які відрізняють його від інших методів для підвищення градієнта.

Розлянемо деякі особливості роботи LightGBM.

1. Градієнтний бустинг – метод який використовує LightGBM для покращення точність моделі за рахунок послідовного додавання нових дерев, кожне з яких намагається виправити помилки попередніх дерев.
2. Двійкові дерева рішень – LightGBM використовує двійкові дерева рішень для створення ансамблю моделей. Кожне дерево намагається розділити простір ознак на області, де кожна область містить приклади з подібними значеннями цільової змінної.
3. Вибір кращих розділень – LightGBM застосовує техніку під назвою Leaf-wise Tree Growth, яка вибирає найбільш ефективні розділення для кожного дерева. Це дозволяє моделі швидше збільшувати точність, але може призводити до перетренування, якщо не застосовуються належні методи регуляризації.
4. Оптимізація пам’яті та швидкості – LightGBM спеціально розроблений для ефективного використання пам'яті і швидкості

обчислень. Він використовує методи, такі як histogram-based splitting (поділ на основі гістограм), що значно знижує час тренування та обсяг використовуваної пам'яті.

5. Підтримка паралельних та розподілених обчислень – LightGBM може виконувати паралельні обчислення та розподіляти обчислювальні завдання між різними машинами, що дозволяє працювати з дуже великими наборами даних.
6. Робота з пропущеними значеннями – LightGBM ефективно обробляє пропущені значення у вхідних даних, що дозволяє використовувати його для реальних задач з неповними даними.

Крім того, LightGBM пропонує широкі можливості кастомізації та налаштування гіперпараметрів, що дозволяє користувачам пристосовувати його до своїх конкретних потреб. Така гнучкість робить його придатним для вирішення широкого спектру задач машинного навчання.

Таким чином, LightGBM – це високоефективний і адаптивний метод градієнтного бустингу, який чудово справляється з великими наборами даних, багатими на функції. Поєднання передових методів, таких як навчання на основі гістограм і зростання дерева за листками, а також підтримка паралельного і GPU-навчання робить його про продуктивним в широкому спектрі задач.

1.5.2 LSTM

Нейронна мережа - це обчислювальна модель, призначена для імітації того, як людський мозок обробляє інформацію. Вона складається з взаємопов'язаних вузлів, або нейронів, які працюють разом для аналізу та інтерпретації даних. Кожен вузол виконує базові обчислення, а потім передає результат наступному рівню вузлів. Нейронні мережі особливо ефективні для

виявлення закономірностей і прогнозування на основі великих наборів даних. Вони широко використовуються в різних додатках, таких як розпізнавання зображень і мови, обробка природної мови і навіть фінансове прогнозування.

Рекурентні нейронні мережі (RNN) – це тип нейронних мереж, які особливо добре підходять для послідовного аналізу даних. На відміну від традиційних нейронних мереж, RNN мають зв'язки, які формують спрямовані цикли, що дозволяє їм зберігати «пам'ять» про попередні вхідні дані. Це робить їх ідеальними для задач, де контекст і порядок мають вирішальне значення, таких як прогнозування послідовності, прогнозування часових рядів і моделювання мови. Ключовою особливістю RNN є їхня здатність використовувати свій внутрішній стан (або пам'ять) для обробки послідовностей вхідних даних змінної довжини. Ця унікальна архітектура дозволяє RNN досягати вражаючих результатів у таких додатках, як розпізнавання рукописного тексту, машинний переклад і навіть написання музики.

LSTM (Long Short-Term Memory) – це вдосконалена форма рекурентної нейронної мережі (RNN), спеціально розроблена для управління довгостроковими залежностями в послідовних даних з високою ефективністю. Вона перевершує традиційні RNN, зберігаючи життєво важливу інформацію протягом тривалих періодів часу, усуваючи обмеження, з якими стикалися попередні моделі. LSTM особливо відомі своєю здатністю фіксувати та використовувати контекст попередніх елементів у послідовності, що робить їх високоефективними для складних часових задач.

Розглянемо особливості роботи LSTM.

1. Комірки пам'яті – в основі LSTM лежать комірки пам'яті, які можуть зберігати інформацію протягом тривалого часу. Ці комірки вирішують проблему зникаючих градієнтів, яка є поширеною у звичайних ШНМ, де навчальний сигнал зменшується, коли він поширюється в часі. Підтримуючи більш постійний градієнт помилки, комірки пам'яті дозволяють мережі ефективніше вивчати

довгострокові залежності.

2. Структури воріт – LSTM використовує спеціальні «вентилі» для регулювання потоку інформації через комірки пам'яті. Ці вентилі призначені для вибіркового пропускання інформації, тим самим гарантуючи, що тільки релевантні дані будуть збережені або відкинуті. Існує три основні типи воріт. Вхідні ворота, які Керують кількістю нової інформації, що надходить до комірки пам'яті. Контролюючи вхідний потік, ці ворота допомагають мережі вирішити, яку інформацію варто додати до стану комірки. Вихідні ворота, які визначають обсяг інформації з комірки пам'яті, що використовується для обчислення поточного результату. Цей вентиль впливає на те, яка частина стану комірки повинна бути розкрита на наступному рівні або часовому кроці. Вентиль забуття (Forget Gate), який визначає кількість інформації в комірці пам'яті, яку слід відкинути. Цей вентиль має вирішальне значення для видалення нерелевантної або застарілої інформації, гарантуючи, що комірка пам'яті залишається зосередженою на релевантних даних.
3. Обчислення станів: LSTM підтримує два критичних стани на кожному часовому кроці, які працюють в тандемі для обробки інформації. Стан комірки, який зберігає довготривалу інформацію. Цей стан діє як конвеєр, переносячи інформацію через різні часові кроки з мінімальними змінами, тим самим зберігаючи довгострокові залежності. Прихований стан, який зберігає короткострокову інформацію і використовується для генерації вихідних даних мережі. Прихований стан динамічно оновлюється на кожному часовому кроці, відображаючи безпосередній контекст, а також беручи участь в обчисленнях наступного рівня.
4. Обробка послідовностей – LSTM обробляє послідовності даних, послідовно застосовуючи комірки пам'яті та вентилі до кожного елемента в послідовності. Цей метод дозволяє мережі вивчати

залежності в різних часових масштабах, фіксуючи як короткострокові нюанси, так і довгострокові тенденції. Послідовний характер обробки LSTM робить його ідеальним для завдань, де порядок точок даних має велике значення.

5. Поширення в часі (Backpropagation Through Time, BPTT). Подібно до інших ШНМ, LSTM навчаються за допомогою BPTT, який коригує ваги мережі на основі помилок прогнозування. Цей процес включає в себе розгортання мережі в часі, обчислення градієнтів та оновлення ваг для мінімізації функції втрат. BPTT дозволяє LSTM точно налаштовувати свої параметри, покращуючи їхню здатність передбачати та розуміти послідовні дані.

LSTM широко використовуються в задачах, де інформація про послідовність є критично важливою, таких як аналіз часових рядів і прогнозування. Вміння працювати з довготривалими залежностями робить його незамінним інструментом для управління даними зі складними та розгалуженими зв'язками. В аналізі часових рядів LSTM відмінно виявляють закономірності і роблять прогнози на основі історичних даних, пропонуючи цінну інформацію.

1.5.3 GRU

Закритий рекурентний блок (Gated Recurrent Unit, GRU) - це тип архітектури рекурентної нейронної мережі (RNN), який був представлений Kyunghyun Cho у 2014 році як альтернатива мережі з довгою короткочасною пам'яттю (LSTM). GRU розроблені для вирішення проблеми зникаючого градієнта, яка є типовою для стандартних RNN, що дозволяє їм більш ефективно фіксувати довші залежності в послідовностях. GRU є більш компактним і менш обчислювально інтенсивним, що робить його популярним у

багатьох застосуваннях.

Розглянемо оосновні собливості роботи GRU.

1. Комірки пам'яті та гейти: У GRU, як і в LSTM, використовуються гейти для контролю потоку інформації. Однак, у GRU немає окремої комірки пам'яті; замість цього, єдиний стан (hidden state) відповідає за зберігання довгострокової інформації. У GRU є два основних гейти – це гейт оновлення (Update Gate) та гейт скидання (Reset Gate). Update Gate вирішує, скільки інформації з попереднього стану потрібно зберегти. Він контролює, які частини прихованого стану (hidden state) мають бути оновлені новою інформацією. Гейт скидання (Reset Gate) вирішує, скільки інформації з попереднього стану треба забути. Це дозволяє мережі вирішувати, яку частину попереднього стану використовувати для обчислення нового кандидата стану.
2. Обчислення нового стану: GRU поєднує вхідний сигнал і прихований стан для обчислення нового стану. В процесі оновлення використовується поєднання старого стану, контрольованого гейтами, і нового кандидата стану. Кандидат нового стану (Candidate State) обчислюється на основі поточного вхідного сигналу та скинутого прихованого стану. Кандидат нового стану враховує інформацію з попереднього стану лише в тій мірі, в якій дозволяє гейт скидання. Оновлений стан (Updated State) комбінує попередній стан і кандидат нового стану відповідно до гейту оновлення, щоб отримати новий стан прихованого шару.
3. Обчислення виходу – вихід GRU на кожному кроці часу є функцією поточного прихованого стану, який містить інформацію про всі попередні кроки послідовності.
4. Зворотне поширення помилок через час (Backpropagation Through Time, BPTT). GRU тренується за допомогою зворотного поширення помилок через час, що дозволяє мережі коригувати свої ваги на

основі помилок, які вона робить при прогнозуванні.

До переваг GRU можна віднести наступне:

– спрощена структура GRU, оскільки вона має меншу кількість гейтів і параметрів, ніж LSTM, що робить її більш обчислювально ефективною;
– швидке навчання за рахунок меншої кількості параметрів, що дозволяє GRU швидше тренуватися, особливо на великих наборах даних;
– схожі результати, тобто незважаючи на спрощену структуру, GRU часто дає результати, порівнянні з LSTM, у багатьох задачах обробки послідовностей;

GRU часто використовується в задачах, де важливі послідовності, таких як аналіз часових рядів і прогнозування. Його простота та ефективність роблять його популярним вибором для багатьох задач.

1.5.4 ARIMA

ARIMA, абревіатура від AutoRegressive Integrated Moving Average (авторегресійне інтегроване ковзне середнє), – це широко використовуваний статистичний метод для аналізу та прогнозування даних часових рядів. Аналіз часових рядів має вирішальне значення в різних галузях, включаючи економіку, фінанси, метеорологію та інженерію, де розуміння і прогнозування майбутніх точок даних має першорядне значення. Моделі ARIMA особливо ефективні в розумінні та прогнозуванні майбутніх даних, які демонструють закономірності та кореляції з їхніми історичними значеннями, що робить їх безцінними для стратегічного планування та прийняття рішень.

Розглянемо основні компоненти ARIMA-моделі які вони в себе

включають.

1. Авторегресія (AR) – цей сегмент моделі вказує на те, що поточне значення ряду є лінійною функцією його попередніх значень. По суті, він відображає зв'язок між спостереженням і певною кількістю спостережень із запізненням. AR-частина моделі допомагає зрозуміти, як минулі значення ряду впливають на поточні та майбутні значення, що дає змогу врахувати імпульс і тенденції в даних.
2. Інтегрованість (I) – цей компонент передбачає рознесення точок даних для досягнення стаціонарності, гарантуючи, що властивості ряду не залежать від часу, в який він спостерігається. Усуваючи тренди та сезонність шляхом диференціювання, модель стає більш придатною для надійного прогнозування. Стаціонарність є критично важливою концепцією в аналізі часових рядів, оскільки вона передбачає, що статистичні властивості ряду, такі як середнє значення і дисперсія, залишаються постійними в часі, що є важливим для здійснення точних прогнозів.
3. Ковзаюче середнє (MA) – цей аспект ковзного середнього в ARIMA відображає взаємозв'язок між спостереженням і залишковою похибкою моделі ковзного середнього, застосованої до запізнілих спостережень. Він використовує минулі помилки прогнозу для передбачення майбутніх значень. Цей компонент допомагає згладити короткострокові коливання і виділити довгострокові тенденції або цикли в даних.

Процес ARIMA починається з визначення того, чи є часовий ряд стаціонарним, що є вирішальним елементом для ефективності моделі. Стаціонарність часто оцінюється за допомогою статистичних тестів, таких як розширений тест Дікі-Фуллера (ADF) або тест Квятковського-Філліпса-Шмідта-Шина (KPSS). Якщо ряд не є стаціонарним, для досягнення стаціонарності буде застосовано диференціювання. Це може включати взяття

різниці між послідовними спостереженнями або застосування декількох рівнів диференціювання.

Згодом визначається оптимальна ARIMA модель на основі комбінації членів авторегресії та ковзного середнього, до яких, за необхідності, додається член диференціювання. Вибір моделі, як правило, керується такими критеріями, як інформаційний критерій Акаіке (AIC) або байєсівський інформаційний критерій (BIC), щоб забезпечити найкращу відповідність. Ці критерії допомагають збалансувати складність моделі з точністю прогнозування, уникаючи надмірного припасування, водночас фіксуючи основні закономірності в даних.

Після того, як модель підігнана, вона проходить діагностичні перевірки для підтвердження її адекватності. Аналіз залишків, наприклад, дослідження автокореляції та часткової автокореляції функцій залишків, гарантує, що припущення моделі відповідають дійсності. Якщо залишки не демонструють значущих закономірностей, це свідчить про хорошу відповідність моделі. Нарешті, модель ARIMA може бути використана для прогнозування майбутніх значень, надаючи цінну інформацію та допомагаючи у прийнятті рішень на основі даних у різних сферах.

1.6 Постановка задачі

Ця робота спрямована на дослідження та порівняння прогнозування цін на криптовалюти на криптовалютному ринку за допомогою методів штучного інтелекту, а саме ARIMA, GRU, LSTM та LightGBM. Для досягнення цієї мети буде проведено декілька етапів роботи.

1. Аналіз методів для прогнозування цін на криптовалюти.
2. Розробка вибраних моделей штучного інтелекту, їх налаштування, оптимізація, тренування та покращення.

3. Проведення прогнозування за допомогою розроблених моделей
4. Дослідження ефективності розроблених моделей на основі реальних даних з торгових бірж. Здійснення порівняння точності прогнозів різних моделей, їх збіг з реальними рухами цін та їх оцінка.
5. Аналіз та інтерпретація отриманих результатів і формулювання висновків.

Висновки до розділу 1

У даному розділі було розглянуто означення криптовалюти та її історія, варіативність та принципи роботи. Було досліджено основні підходи, які використовуються для задачі прогнозування цін на криптовалюти їх переваги та недоліки. Представлено методи штучного інтелекту які планується використовувати у даній роботі, описано основні принципи їх роботи. Представлено постановку задачі та активи які планується досліджувати.

РОЗДІЛ 2 МАТЕМАТИЧНІ ОСНОВИ РОБОТИ

2.1 Математичні моделі для прогнозування часових рядів

Часовий ряд – це послідовність точок даних, які зазвичай вимірюються в послідовні моменти часу, розділені рівномірними інтервалами. Ці точки даних можуть представляти найрізноманітніші явища – від щоденних цін на криптовалюти та щомісячних даних про їх об’єми торгів до щорічних кліматичних вимірювань. Основною характеристикою часового ряду є його часова впорядкованість, яка відрізняє його від інших типів даних. Наприклад, у часовому ряді щоденних температур кожна точка даних пов'язана з певною датою, і послідовність дат має вирішальне значення для розуміння тенденцій і закономірностей у часі.

Аналізуючи дані часових рядів, можна виявити основні закономірності, тенденції та сезонні коливання, які можуть бути використані в процесі прийняття рішень і підвищити точність прогнозів. Тому розглянемо декілька основних методів для їх прогнозування.

1. Ковзаюче середнє: цей простий метод передбачає обчислення середнього значення фіксованої кількості минулих точок даних для прогнозування майбутніх значень. Ковзне середнє допомагає згладити коливання і виділити довгострокові тенденції. Наприклад, 7-денне ковзне середнє щоденних даних про ціну може допомогти визначити тижневі тенденції, мінімізуючи щоденні коливання.
2. Експоненціальне згладжування: цей метод призначає минулим спостереженням ваги, що експоненціально зменшуються, що дозволяє останнім даним мати більший вплив на прогноз. Найпоширеніші типи включають просте експоненціальне згладжування, подвійне експоненціальне згладжування та потрійне експоненціальне згладжування. Ці методи особливо корисні для

даних із сильним трендом або сезонними закономірностями, оскільки вони швидше пристосовуються до змін у даних.

3. Авторегресійне інтегроване ковзне середнє (ARIMA): моделі ARIMA є складним поєднанням авторегресії, диференціювання (для врахування нестаціонарності) та компонентів ковзного середнього. Цей метод добре підходить для лінійних часових рядів даних і може бути пристосований до різних характеристик даних шляхом налаштування його параметрів.
4. Сезонна декомпозиція часових рядів (STL): цей метод розкладає часовий ряд на сезонні, трендові та залишкові компоненти. Потім кожен компонент аналізується окремо, щоб зробити прогнози, які враховують сезонність і довгострокові тенденції. STL особливо ефективний для складних часових рядів з декількома сезонними моделями, що перекриваються, наприклад, для даних про споживання електроенергії, яке змінюється залежно від часу доби та сезону.
5. Моделі машинного навчання: з розвитком обчислювальних потужностей методи машинного навчання, такі як LSTM або GRU та згорткові нейронні мережі (CNN), стали популярними для прогнозування часових рядів. Ці моделі можуть вловлювати складні нелінійні закономірності та взаємодії в даних. Наприклад, LSTM-мережі добре підходять для виявлення довгострокових залежностей у даних часових рядів, що робить їх ідеальними для задач прогнозування та аналізу ринків.

2.2 Критерії оцінки моделей

Оцінювання моделей прогнозування часових рядів є важливим етапом для забезпечення точності та надійності прогнозів. Процес оцінювання зазвичай передбачає поділ даних на навчальні та тестові набори, де модель навчається на перших і перевіряється на других. Розглянемо поширені метриками для оцінки ефективності моделей прогнозування часових рядів.

Середня абсолютна похибка (Mean Absolute Error, MAE). Вимірює середню абсолютну різницю між прогнозованими та фактичними значеннями.

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

де y_i – фактичні значення;
$\hat{y}_i$ – прогнозовані значення;
n – кількість спостережень.

Середньоквадратична похибка (Mean Squared Error, MSE). Вимірює середню квадратичну різницю між прогнозованими та фактичними значеннями.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

Квадратичний корінь середньоквадратичної похибки (Root Mean Square Error, RMSE). RMSE є квадратним коренем з MSE і вимірює стандартне відхилення помилок прогнозу.

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

Середня відсоткова похибка (Mean Absolute Percentage Error, MAPE). Вимірює середню відсоткову різницю між прогнозованими та фактичними значеннями.

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

Коефіцієнт детермінації (R^2). Вимірює, яку частку варіації у фактичних значеннях можна пояснити прогнозованими значеннями.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2},$$

де $\bar{y}_i$ – середнє значення фактичних значень.

2.3 Алгоритми моделей штучного інтелекту

В цьому розділі ми детальніше розберемось з алгоритмами роботи вибраних для реалізації методів для виконання поставленої задачі, а саме ARIMA, GRU, LSTM та LightGBM.

2.3.1 LightGBM

LightGBM (Light Gradient Boosting Machine) – це потужний алгоритм градієнтного бустінгу, який використовує в своїй основі дерева рішень. Розглянемо основні кроки алгоритму.

1. Ініціалізація. Початкова модель встановлюється на певне значення, зазвичай це середнє значення цільової змінної для задач регресії або логарифм відношення позитивних і негативних класів для задач класифікації.
2. Побудова дерев рішень.LightGBM будує дерева рішень послідовно, кожне нове дерево намагається зменшити похибку попередніх дерев.
3. Використання leaf-wise стратегії. LightGBM використовує стратегію росту дерев leaf-wise (на відміну від рівневого росту в інших реалізаціях градієнтного бустінгу), що дозволяє скоротити час навчання та зменшити похибку.
4. Оцінка та вибір розщеплень. На кожному кроці вибирається оптимальне розщеплення вузла дерева, яке зменшує втрати (loss).
5. Регуляризація. LightGBM застосовує різні методи регуляризації для запобігання перенавчанню, включаючи L1/L2 регуляризацію, обмеження глибини дерев та мінімальну кількість вибірок у вузлі.
6. Прогнозування. Після навчання всіх дерев, прогнозування для нових даних здійснюється шляхом сумування прогнозів усіх дерев.

LightGBM демонструє високу швидкість, в порівнянні з іншими моделями градієнтного бустингу, завдяки використанню leaf-wise стратегії і може ефективно та швидко працювати з великими обсягами даних використовуючи при цьому менше пам’яті. Ефективні методи навчання дозволяють досягати високої точності моделей та уникати перенавчання моделі.

2.3.2 LSTM

LSTM (Long Short-Term Memory) – це вдосконалена форма рекурентної нейронної мережі (RNN), і тому має більш складний алгоритм який в свою чергу і є більш ефективним.

Розглянемо алгоритм роботи і спрощену діаграму (без ваги та зсувів), щоб дізнатися, як рекурентний блок LSTM обробляє інформацію.

1. Прихований стан (Hidden state) і нові входи – прихований стан з попереднього кроку (h_{t-1}) і вхід на поточному кроці (x_t) об'єднуються перед передачею їх копій через різні ворота.
2. Ворота забуття (Forget gate) – цей вентиль керує тим, яку інформацію слід забути. Оскільки сигмоїдна функція коливається між 0 і 1, вона визначає, які значення у стані комірки слід відкинути (помножити на 0), запам'ятати (помножити на 1) або частково запам'ятати (помножити на деяке значення між 0 і 1).
3. Вхідні ворота (Input gate) допомагають визначити важливі елементи, які потрібно додати до стану комірки. Зауважте, що результати вхідних воріт множаться на кандидата на стан комірки, при цьому до стану комірки додається лише та інформація, яку вхідні ворота вважають важливою.
4. Оновлення стану комірки – спочатку попередній стан комірки (c_{t-1}) множиться на результати воріт забуття. Потім ми додаємо нову інформацію з [вхідні ворота × кандидат на стан комірки], щоб отримати останній стан комірки (c_t).
5. Оновлення прихованого стану – остання частина полягає в оновленні прихованого стану. Останній стан комірки (c_t) пропускається через функцію активації tanh і множиться на результати вихідного вентиля.
6. Нарешті, останній стан комірки (c_t) і прихований стан (h_t)

повертаються назад у рекурентний блок, і процес повторюється на кроці t+1. Цикл продовжується до тих пір, поки ми не досягнемо кінця послідовності.

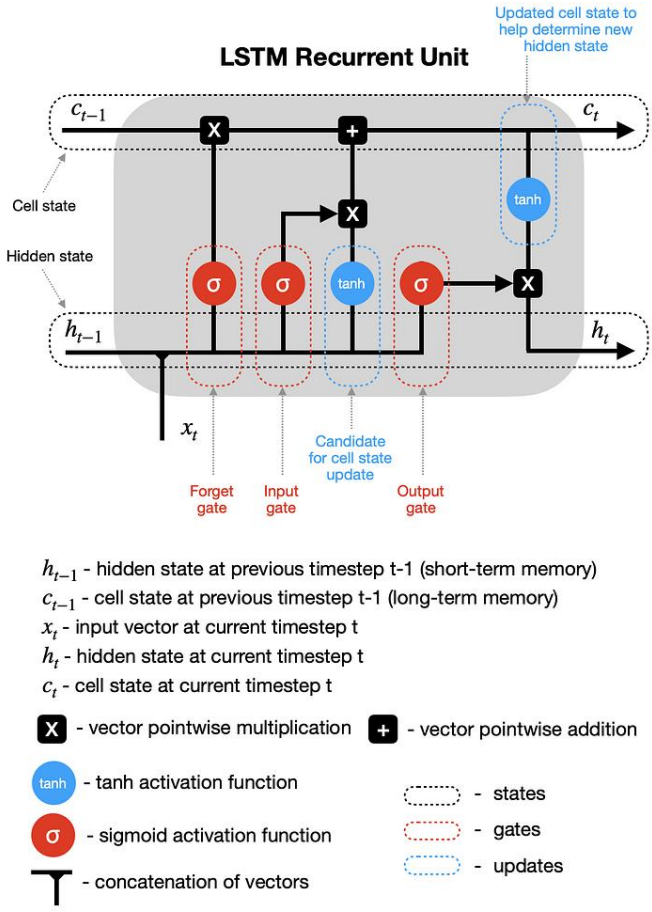

Рисунок 2.1 – Алгоритм LSTM[1]

2.3.3 GRU

Як і інші архітектури рекурентних нейронних мереж, GRU обробляє послідовні дані по одному елементу за раз, оновлюючи свій прихований стан на основі поточного входу та попереднього прихованого стану. На кожному

[1] Джерело зображення: https://miro.medium.com/v2/resize:fit:720/format:webp/1*Zrht4QBK5_hAxif17ED4ew.png

часовому кроці GRU обчислює «вектор активації кандидата», який інтегрує інформацію з вхідних даних і попереднього прихованого стану. Цей вектор-кандидат потім оновлює прихований стан для наступного кроку.

Вектор активації кандидата обчислюється за допомогою двох воріт: воріт скидання і воріт оновлення. Вентиль скидання визначає, які частини попереднього прихованого стану відкинути, а вентиль оновлення вирішує, які частини вектора активації кандидата включити в новий прихований стан.

Алгоритм архітектури GRU:

1. Вентиль скидання r_t та вентиль оновлення z_t обчислюються з використанням поточного входу x та попереднього прихованого стану h_{t-1}

$$r_t = sigmoid(W_r * [h_{t-1}, x_t]),$$
$$z_t = sigmoid(W_z * [h_{t-1}, x_t]),$$

де W_r та W_z – вагові матриці, які навчаються під час навчання.

2. Вектор активації кандидата $\acute{h}_t$ обчислюється за поточним входом x та модифікованою версією попереднього прихованого стану, який «скидається» за допомогою воріт скидання.

$$\acute{h}_t = tanh(W_h * [r_t * h_{t-1}, x_t]),$$

де W_h - інша вагова матриця.

3. Новий прихований стан h_t обчислюється шляхом об'єднання вектора активації кандидата з попереднім прихованим станом, зваженим на вентиль оновлення:

$$h_t = (1 - z) * h_{t-1} + z_t * \acute{h}_t.$$

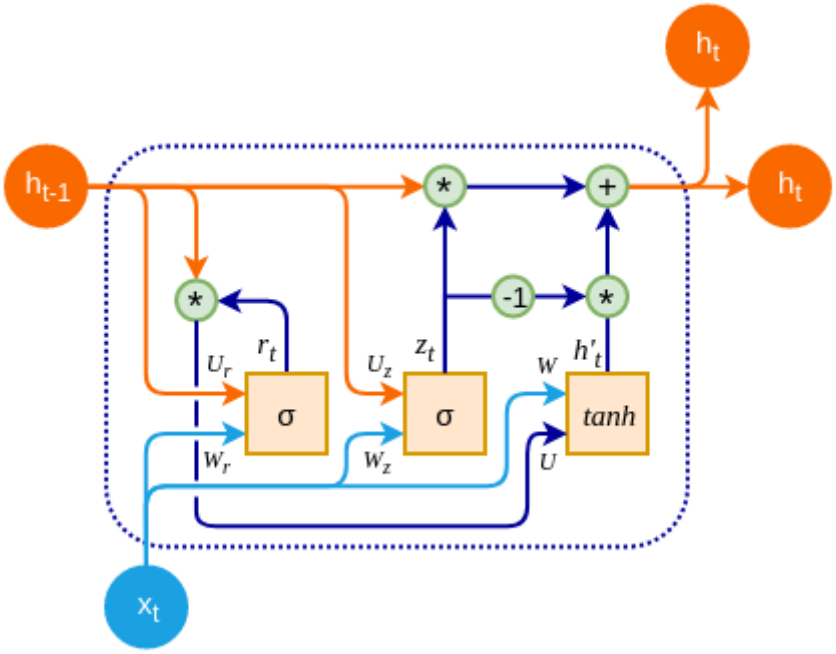

Рисунок 2.2 – Алгоритм GRU [2]

Загалом, вентиль скидання визначає, яку частину попередньогоприхованого стану запам'ятати або забути, тоді як вентиль оновлення визначає, яку частину вектора активації кандидата включити до нового прихованого стану. Результатом є компактна архітектура, яка здатна вибірково оновлювати свій прихований стан на основі вхідних даних та попереднього прихованого стану, без потреби в окремому стані комірки пам'яті, як в LSTM.

2.3.4 ARIMA

Розглянемо загальний алгоритм роботи моделі ARIMA.

1. Попередня обробка даних – ARIMA припускає, що дані часового ряду є стаціонарними, тобто їхні статистичні властивості, такі як

[2] Джерело: https://www.oreilly.com/library/view/advanced-deep-learning/9781789956177/8ad9dc41-3237-483e-8f6b-7e5f653dc693.xhtml

середнє, дисперсія та автокореляція, є постійними в часі. Якщо дані не є стаціонарними, їх потрібно перетворити, щоб зробити їх такими. Зазвичай це передбачає диференціювання даних для усунення тенденцій або сезонних закономірностей. Дані перевіряються на наявність пропущених значень і викидів, які можуть знадобитися перед моделюванням.

2. Ідентифікація моделі – першим кроком є визначення параметрів (p, d, q) моделі ARIMA. Параметр d (порядок диференціювання) визначається шляхом аналізу ступеня диференціювання, необхідного для того, щоб зробити часовий ряд стаціонарним. Це часто робиться шляхом взяття перших різниць і перевірки на стаціонарність за допомогою статистичних тестів, таких як розширений тест Дікі-Фуллера (ADF). Параметри p (порядок авторегресії) та q (порядок ковзного середнього) можна визначити за допомогою графіків автокореляції (ACF) та часткової автокореляції (PACF) різницевих рядів. Ці графіки допомагають визначити порядок компонентів AR та MA.
3. Оцінка параметрів – після того, як визначено порядок моделі ARIMA, наступним кроком є оцінка коефіцієнтів моделі. Зазвичай це робиться за допомогою методу оцінки максимальної правдоподібності (МНК) або методу найменших квадратів.
4. Підгонка моделі – після того, як параметри оцінені, ARIMA модель підганяється під дані. Залишки моделі (різниця між спостережуваними та прогнозованими значеннями) досліджуються, щоб переконатися, що вони є білим шумом (тобто випадковою величиною з постійним середнім значенням та дисперсією). Якщо залишки демонструють закономірності або автокореляцію, це свідчить про те, що модель можна покращити, і, можливо, потрібно внести корективи в оцінки параметрів.
5. Оцінка моделі – підібрана модель ARIMA оцінюється за допомогою

різних метрик, таких як середньоквадратична помилка (MSE), інформаційний критерій Акаіке (AIC), байєсівський інформаційний критерій (BIC) тощо. Ефективність моделі оцінюється на наборі даних або за допомогою перехресної перевірки, щоб забезпечити її узагальнюваність.

6. Прогнозування – після того, як модель ARIMA перевірена, її можна використовувати для прогнозування на майбутні періоди часу. Можна обчислити інтервали прогнозування та інтервали між прогнозами, щоб отримати оцінки невизначеності навколо прогнозів.
7. Уточнення моделі – модель може потребувати періодичного уточнення або оновлення в міру надходження нових даних або зміни процесу, що лежить в основі генерування даних.

Загалом, ARIMA є потужним і гнучким алгоритмом для прогнозування часових рядів, але його ефективність залежить від правильного визначення та оцінки параметрів моделі, а також від того, чи виконуються припущення моделі.

2.4 Обрані криптовалюти

Криптовалюти Ethereum (ETH) та Bitcoin (BTC) були обрані для цього дослідження через їхню значущість на ринку цифрових активів. Ці криптовалюти котируються на всіх основних біржах і демонструють найбільші обсяги торгів, що робить їх високоліквідними та легкодоступними. Крім того, ETH і BTC викликають значний інтерес з боку великих інвесторів та інституційних учасників, що зміцнює їхній статус провідних криптовалют. Їх широке розповсюдження та значний вплив у криптоекосистемі роблять їх ідеальними об'єктами для поглибленого аналізу.

ETH, відомий своєю функціональністю смарт-контрактів та основою численних децентралізованих додатків (dApps), здійснив революцію в способі виконання цифрових контрактів. Ця універсальність зробила Ethereum фаворитом серед розробників і компаній, які прагнуть створювати рішення на основі блокчейну.

BTC, з іншого боку, часто називають цифровим золотом через його першопрохідницьку роль у світі криптовалют та обмежену кількість монет (21 мільйон). Децентралізована природа біткоїна та його репутація як засобу збереження вартості приваблюють широке коло інвесторів - від роздрібних трейдерів до хедж-фондів, які прагнуть диверсифікувати свої портфелі.

Разом Ethereum та Bitcoin не лише домінують у рейтингах ринкової капіталізації, але й задають тенденції та стандарти для всієї криптоіндустрії. Їхній технологічний прогрес, широке розповсюдження та потужна підтримка спільноти створюють багате підґрунтя для всебічного дослідження та аналізу.

2.5 Запропоновані методи

У цій роботі пропонується дослідити точність прогнозування курсу криптовалют за допомогою моделей штучного інтелекту. Алгоритми обраних моделей були описані у розділі 2.4, це – ARIMA, LightGBM, LSTM, GRU. Серед криптовалютних активів для прогнозування було обрано BTC та ETH.

Висновки до розділу 2

В цьому розділі було розібрано, що таке часові ряди, вибрано моделі для дослідження, визначено основні критерії оцінки моделей, та вибрано

криптовалюти на яких буде проведено навчання та прогнозування.

Розібрано алгоритми, структуру та принцип роботи обраних моделей для подальшої їх реалізації для задачі прогнозування.

РОЗДІЛ 3 ОПИС РЕЗУЛЬТАТІВ РОБОТИ

3.1 Опис і передобробка вхідних даних

Для отримання даних було обрано Yahoo Finance. Цей сервіс надає вичерпні та актуальні біржові дані для фінансових активів, в тому числі і криптовалют. Його перевагою є надійна та стабільна робота сервісу, актуальні дані, постійна підтримка та вдосконалення відповідно до потреб користувачів, а також зручне API для отримання даних.

Щоб завантажити дані було використано бібліотеку yfinance, яка працює через API Yahoo Finance. В результаті було завантажено два датасети для Bitcoin з 17 вересня 2014 року по 1 січня 2024 року та Ethereum з 9 грудня 2017 року по 1 січня 2024 року. Ці два відрізняються за розміром, проте для дослідження це може дати цікаві результати, так як датасет для Bitcoin має період даних майже за 10 років, а Ethereum тільки за 6, більше того криптоактив Ethereum є більш стабільним і менш волатильним ніж Bitcoin.

Таблиця 3.1 – Вхідний датасет Bitcoin

	Date	**Open**	**High**	**Low**	**Close**	**Adj Close**	**Volume**
0	17/09/2014	465.9	468.2	452.4	457.3	457.3	21056800
1	18/09/2014	456.9	456.9	413.1	424.4	424.4	34483200
…	...	...	...	...	...	...	...
3391	12/30/2023	42091.8	42584.1	41556.2	42156.9	42156.9	16013925945
3392	12/31/2023	42152.1	42860.9	41998.3	42265.2	42265.2	16397498810

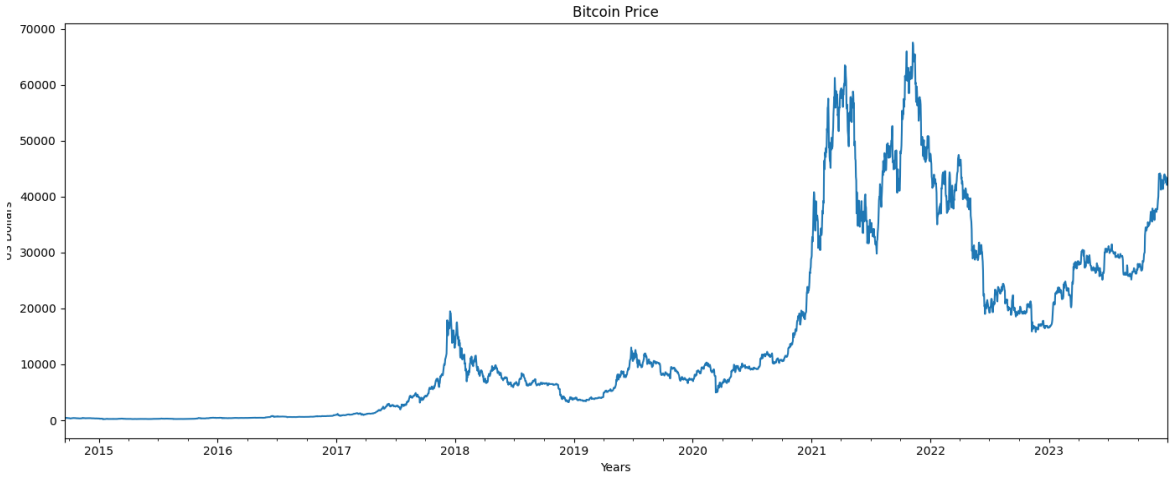

Рисунок 3.1 – Графік ціни Bitcoin побудований на датасеті

Таблиця 3.2 – Вхідний датасет Ethereum

	Date	Open	High	Low	Close	Adj Close	Volume
0	09/11/2017	308.6	329.5	307.1	320.9	320.9	893249984
1	10/11/2017	320.7	324.7	294.5	299.5	299.3	885985984
…	…	…	…	…	…	…	…
2242	12/30/2023	2300.4	2322.0	2270.0	2292.1	2292.1	6888195427
2243	12/31/2023	2291.9	2318.5	2261.4	2281.5	2281.5	6871481744

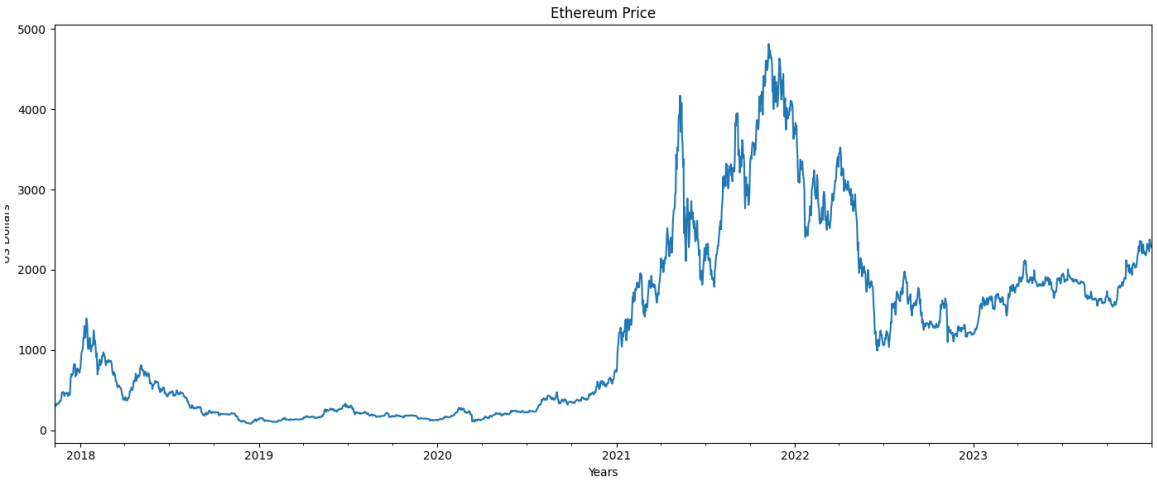

Рисунок 3.2 – Графік ціни Ethereum побудований на датасеті

У вхідниих датасетах отриманий через API Yahoo Finance можемо побачити такі дані, а саме: Date - дату торгів, Open - ціну відкриття торгів, High - найвищу ціну за торговий день , Low - найнижчу ціну за торговий день , Close - ціну закриття торгів, Adj Close - середню ціну за торговий день та Volume - об'єм торгів за день. Для прогнозування буде використано Close – тобто ціну з якою завершився торговий день.

Отримавши та описавши вхідні дані переходимо до етапу передобробки вхідних даних, який включає в себе наступне:

1) Перевірка даних на цілісність на наявність пропусків даних. В отриманих датасетах немає пропусків, тому цей датасет є валідним. Проте у разі нявності таких рядків або стовпців з пропусками – їх потрібно або видалити або заповнити використовуючи імпутацію або інтерполяцію.
2) Нормалізація та стандартизація даних. Цей етап дозволяє прискорити процес навчання та забезпечити кращу продуктивність алгоритів і моделей та привести дані до єдиного масштабу без спотворення відносних відмінностей у значеннях ознак. Цей етап було виконано за допомогою методу MinMaxScaler бібліотеки sklearn.preprocessing.
3) Розбиття датасету на тренувальний та тестовий набори. Розбиття потрібно щоб використати тренувальний набір для навчання моделей, а тестовий набір – для оцінки її якості та точності прогнозування. Стандартний коефіцієнт розбиття, який і був використаний, становить 0.8, тобто 80% від датасету становить тренувальний набір, а інші 20% - тестовий. Також важливо зазначити, що оскільки у нас задача прогнозування цін, розбиваємо датасет послідовно, без перемішування чи випадкової вибірки.

Виконавши вищеописані кроки, наші вхідні дані тепер готові для тренування моделей та оцінки їх прогнозування.

3.2 Опис процесу навчання моделей

Для початку розглянемо статистичний метод ARIMA. Цей метод розрахований на стаціонарність даних, тому для початку потрібно перевірити їх стаціонарність за допомогою тесту Діккі-Фуллера, а за рівень значущості візьмемо стандартне значення в 0.05. Для обидвох датасетів Bitcoin та Ethereum дані виявились не стаціонарні. Тому було вирішено використати перетворення Бокса-Кокса (Box-Cox Transformation) та провести диференціювання (differentiation) для досягнення стаціонарності, в результаті чого обидва ряди стали стаціонарні. Тепер коли у нас ряди стаціонарні, можна переходити до навчання моделі, але тепер потрібно врахувати і сезонність. Проте дана модель може приймати різні значення параметрів порядку P,D,Q. Тому було реалзовано перебір можливих комбінацій параметрів для значень від 1 до 5 щоб вибрати моделі з найменшим значенням критерій Акаіке (Akaike's information criterion, AIC) Методи для приведення до стаціонарного вигляду даних та сама модель була взята з біблеотеки statmodels.

Розглянемо модель градієнтного бустінгу LightGBM. Для покращення точності та врахування часової залежності даних, було вирішено додати до датасетів лагові ознаки (lag features) – це попередні значення часового ряду, які використовуються як додаткові вхідні ознаки для моделі. Вони допомагають моделі враховувати історичні дані для прогнозування майбутніх значень. Лагові ознаки є важливим компонентом у багатьох моделях часового ряду, оскільки вони забезпечують врахування часової залежності між спостереженнями. Було вирішено додати 4 лагові ознаки періодом на 1, 2, 3, 7 днів. Тепер можна переходити до тренування моделі. Оскільки дана модель має велику кількість параметрів для налаштування, було вирішено виконати підбір параметрів за допомогою методу GridSearchCV на основі оцінки RMSE. Після підбору найкращими параметрами для обох датасетів виявились: 'num_leaves': 31,

'learning_rate': 0.1. Модель була взята з бібліотеки lightgbm, а метод для підбору параметрів GridSearchCV з бібліотеки sklearn.

Перейдемо до моделей нейронних мереж і розглянемо GRU та LSTM моделі. Даній нейронні мережі складаються з різних шарів, наявністю дропаутів (Dropout) та їх кількості, відповідно розглянемо архітектуру обидвох моделей окремо. Для побудовии шарів, дропаутів та моделей було використано бібліотеку Keras.

Розглянемо саму архітектуру моделі GRU.

1. Вхідний шар GRU - GRU(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1), activation='tanh'). Цей шар GRU має 50 нейронів (units), використовує функцію активації tanh та очікує вхідні дані у формі (X_train.shape[1], 1), що є одновимірним масивом ціни за ознакою Close з обраних датасетів.
2. Випадкове відключення (dropout) - Dropout(0.2). Для уникнення перенавчання додається шар Dropout, який випадковим чином вимикає 20% нейронів у кожній ітерації навчання.
3. Додатковий шар GRU - GRU(units=50, return_sequences=False, activation='tanh'). Цей шар GRU також має 50 нейронів та функцію активації tanh, return_sequences=False вказує на те, що цей шар не повертає послідовні вихідні дані, оскільки він останній у стеку рекурентних шарів.
4. Додаткове випадкове відключення - Dropout(0.2)
5. Повнозв'язний шар Dense - Dense(units=1). Доданий повнозв'язний шар з одним нейроном без активації, оскільки це завдання регресії, і ми хочемо прогнозувати одне числове значення.

Після побудови архітектури моделі GRU її необхідно скомпілювати та навчити на навчальних даних. Для компіляції використаємо оптимізатор adam і функція втрат mean_squared_error. Для навчання моделі було використано тренування протягом 20 епох, так як це є опитимальним рішенням по часу і точності.

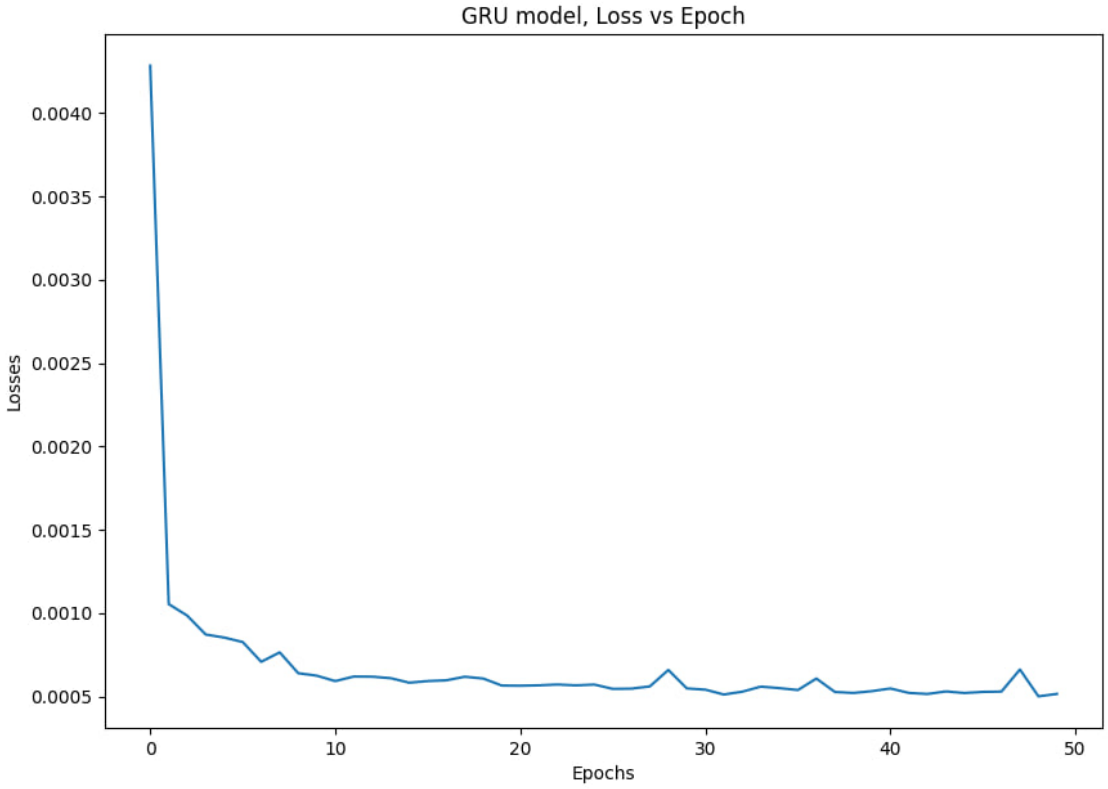

Рисунок 3.3 – Функція втрат відносно епох для моделі GRU

Розглянемо саму архітектуру моделі LSTM.

1. Вхідний шар LSTM – LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1), activation='tanh'). Цей шар LSTM має 50 нейронів (units) та очікує вхідні дані у формі (X_train.shape[1], 1), що є одновимірним масивом ціни за ознакою Close з обраних датасетів.
2. Додатковий шар LSTM – LSTM(units=50, return_sequences=False). Цей шар LSTM також має 50, return_sequences=False вказує на те, що цей шар не повертає послідовні вихідні дані, оскільки він останній у стеку рекурентних шарів.
3. Повнозв'язні шари Dense - Dense(25) та Dense(1). Після шарів LSTM додаються два повнозв'язні шари. Перший має 25 нейронів, а

другий - один нейрон без активації, оскільки це завдання регресії, і ми хочемо прогнозувати одне числове значення.

Для компіляції аналогічно до GRU використовуємо оптимізатор adam і функція втрат mean_squared_error. Для навчання моделі було так само вибрано 20 епох.

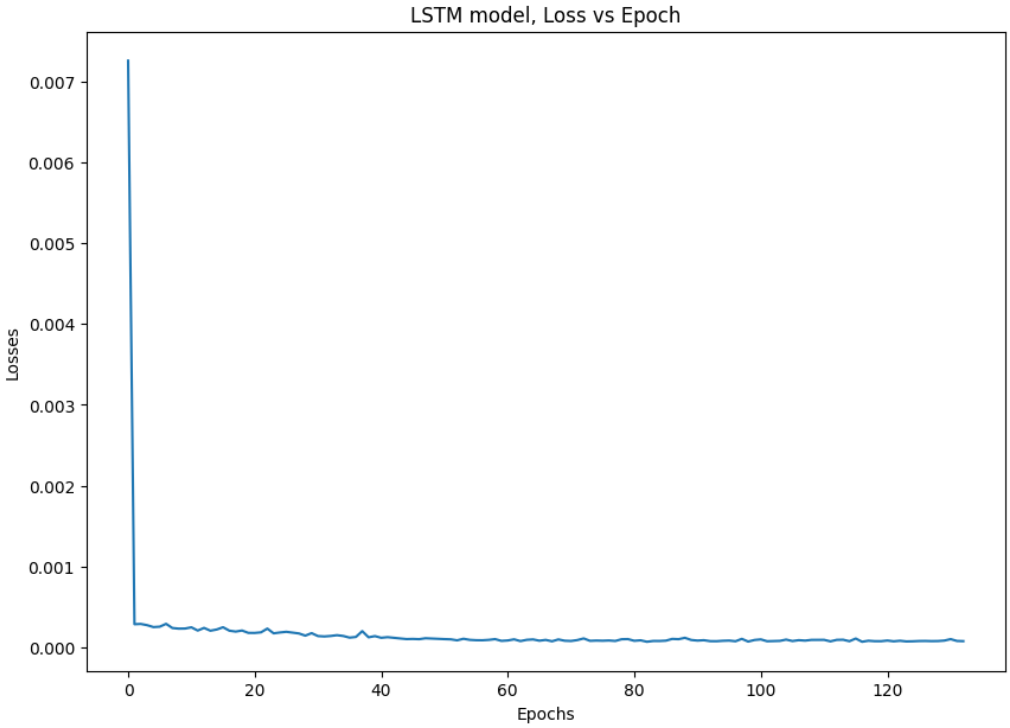

Рисунок 3.4 – Функція втрат відносно епох для моделі LSTM

Додатково можемо порівняти швидкість навчання моделей GRU та LSTM, порівнювати швидкість LightGBM та ARIMA немає сенсу, так як вони навчаються менше 1 секунди і основна частина часу витрачається на підбір параметрів, що вже залежить від вибірки параметрів та їх кількості.

З таблиці 3.3 ми можемо зробити висновок що модель GRU швидша в приблизно 15 разів на обидвох датасетах від моделі LSTM. Це повязано з тим, що GRU має менше параметрів та меншу складність порівняно з LSTM, оскільки в ній відсутні деякі складні механізми як cell state або forget gate.

Таблиця 3.3 – Результати часу навчання моделей нейронних мереж

Модель	Bitcoin	Ethereum
GRU	21.31	14.62
LSTM	336.82	213.32
Розмір датасету	3392	2243

3.3 Оцінка навчання моделей та їх порівняння

Після того як наші моделі пройшли тренування на тренувальних даних перейдемо до прогнозування даних на тестувальних даних та розглянемо основні оціночні показники моделей відносно якості їх прогнозування, а саме R^2, MSE, MAE, MAPE і RMSE. Для розрахунку даних оцінок було використано відповідні методи з бібліотеки sklearn.metrics.

Для початку розглянемо оцінки метрик прогнозів моделей для Bitcoin. Результати метрик показані в таблиці 3.2.

Таблиця 3.4 – Результати метрик прогнозування для Bitcoin

Model	MSE	MAE	MAPE	RMSE	R^2
ARIMA	659570.707039	525.015756	1.891996	812.139586	0.989132
GRU	1644657.914524	969.438557	3.373255	1282.442168	0.972900
LightGBM	762575.027934	648.104941	2.586423	873.255419	0.987425
LSTM	1299552.954783	889.140284	3.232703	1139.979365	0.978586

По даним метрикам можемо побачити, що всі моделі досить добре справились з поставленою задачею. Найкращий результат по всіх метриках показав статистичний метод ARIMA, трішки гірше справився метод градєнтного бустингу LightGBM. Далі за результатами у нас йдуть нейронні

мережі LSTM та GRU, як можна побачити LSTM показала себе набагто краще ніж GRU, врховуючи метрики.

Розглянемо графіки на яких зображено реальні та прогнозовані моделями ціни на Bitcoin на рис. 3.5 – рис. 3.8

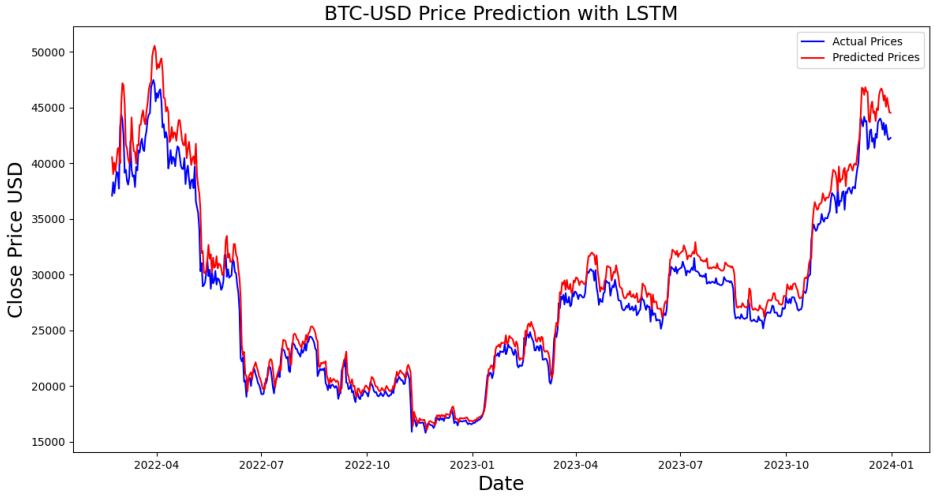

Рисунок 3.5 – Графік реальних значень та прогнозів моделі LSTM на ціну Bitcoin

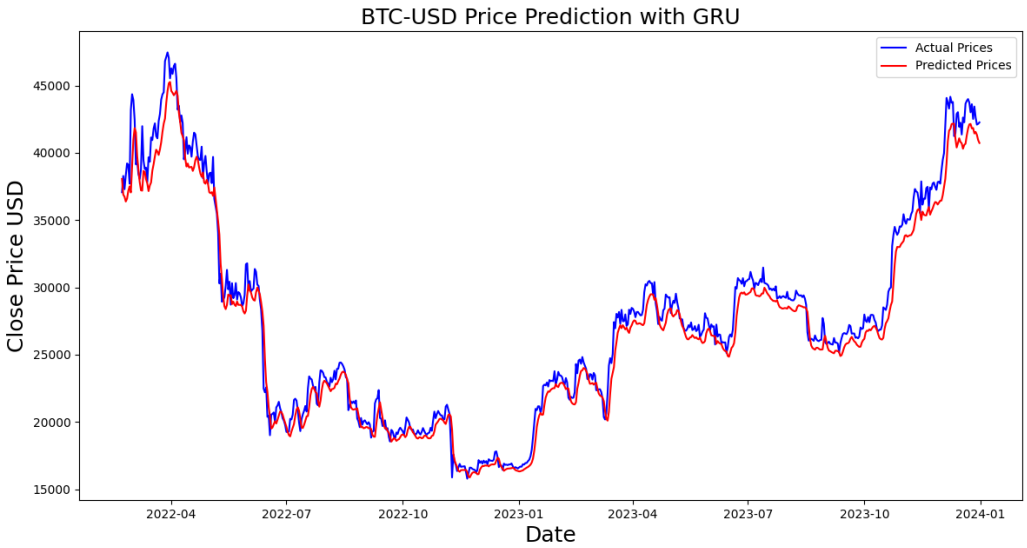

Рисунок 3.6 – Графік реальних значень та прогнозів моделі GRU на ціну Bitcoin

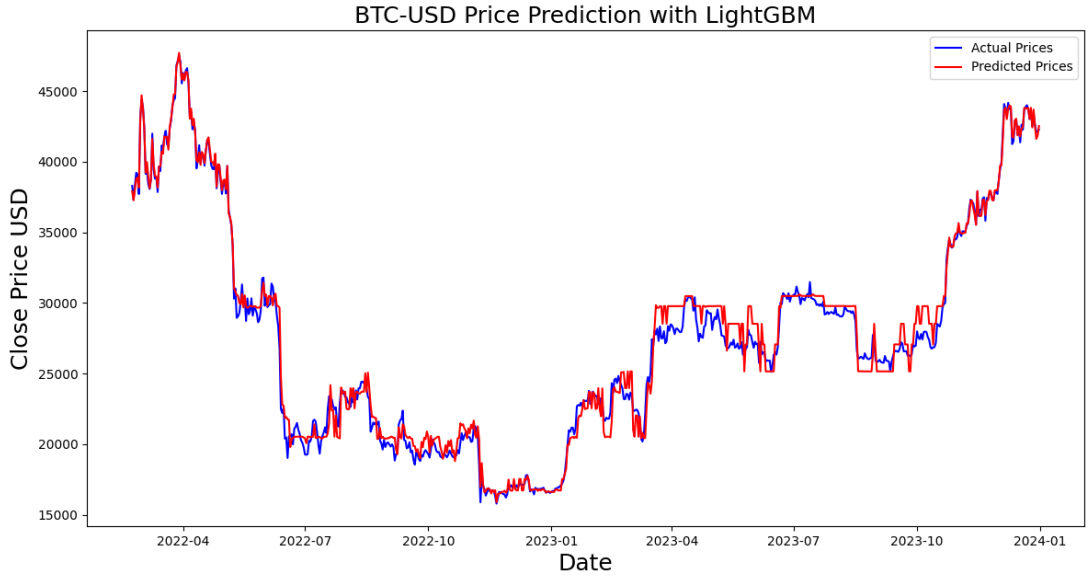

Рисунок 3.7 – Графік реальних значень та прогнозів моделі LightGBM на ціну Bitcoin

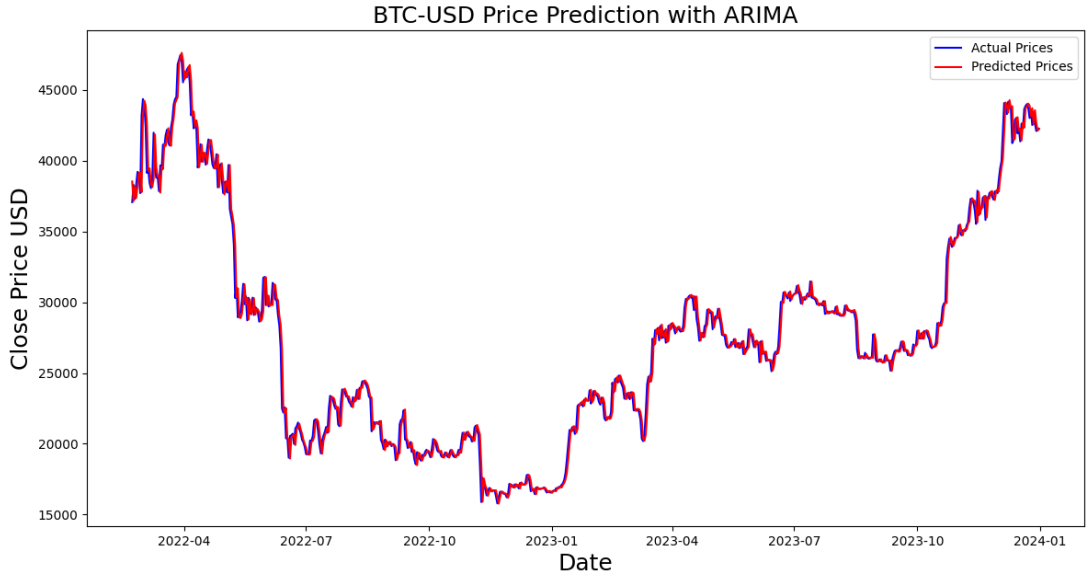

Рисунок 3.8 – Графік реальних значень та прогнозів моделі ARIMA на ціну Bitcoin

Далі розглянемо оцінки метрик прогнозів моделей для Ethereum та подивимось як себе показали моделі з цим криптоактивом. Результати метрик показані в таблиці 3.5.

Таблиця 3.5 – Результати метрик прогнозування для Ethereum

Model	MSE	MAE	MAPE	RMSE	R^2
ARIMA	2213.558729	31.935930	1.910440	47.048472	0.972247
GRU	10083.293665	89.694794	5.446546	100.415604	0.873580
LightGBM	116.136032	8.249529	0.486994	10.776642	0.998541
LSTM	8107.24859	80.068364	4.788397	90.040260	0.898355

По даним метрикам вже бачимо зміни. Найкращий результат по всіх метриках показав метод градєнтного бустингу LightGBM, трохи гірше вже справився статистичний метод ARIMA. Далі аналогічні попередньому криптоактиву результати показали нейронні мережі LSTM та GRU, де LSTM виявився більш точнішим.

Розглянемо графіки на яких зображено реальні та прогнозовані моделями ціни на Bitcoin на рис. 3.9 – рис. 3.12

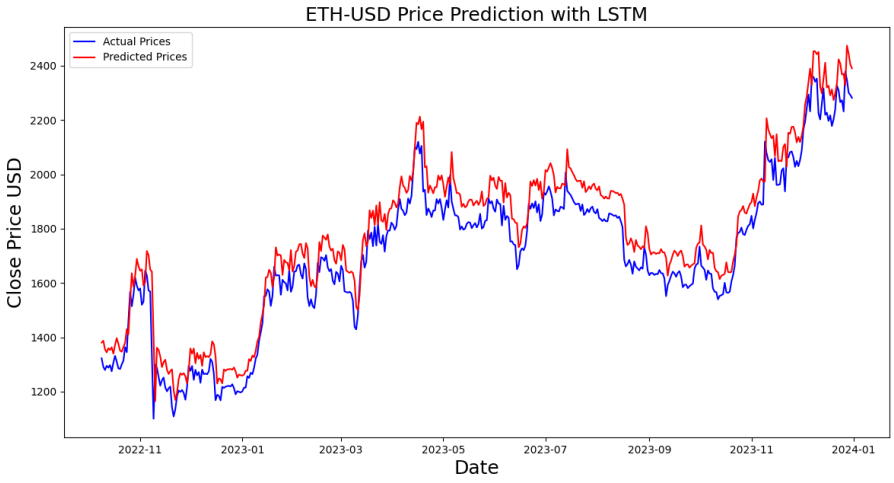

Рисунок 3.9 – Графік реальних значень та прогнозів моделі LSTM на ціну Ethereum

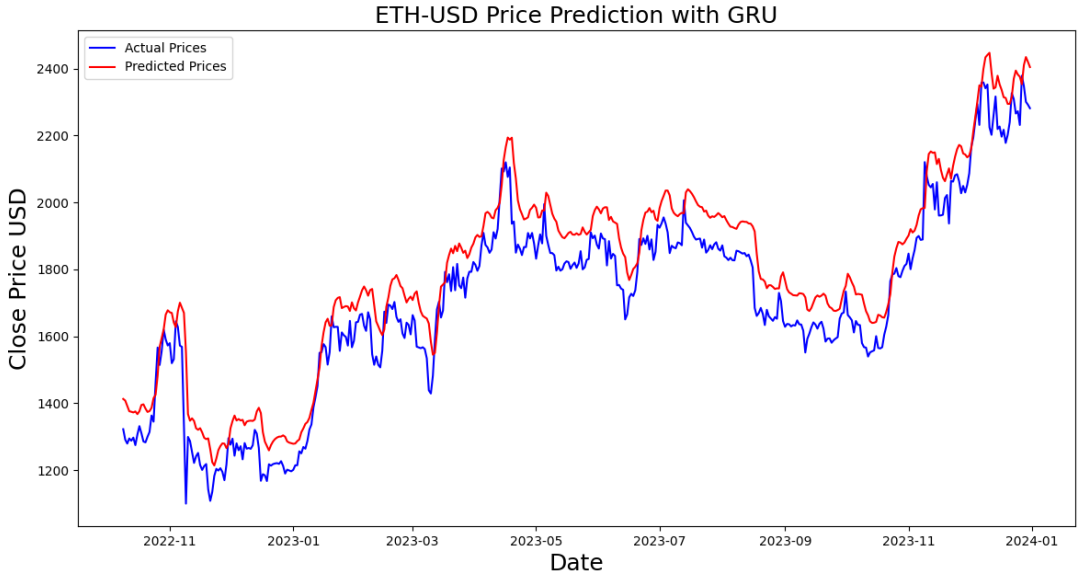

Рисунок 3.10 – Графік реальних значень та прогнозів моделі GRU на ціну Ethereum

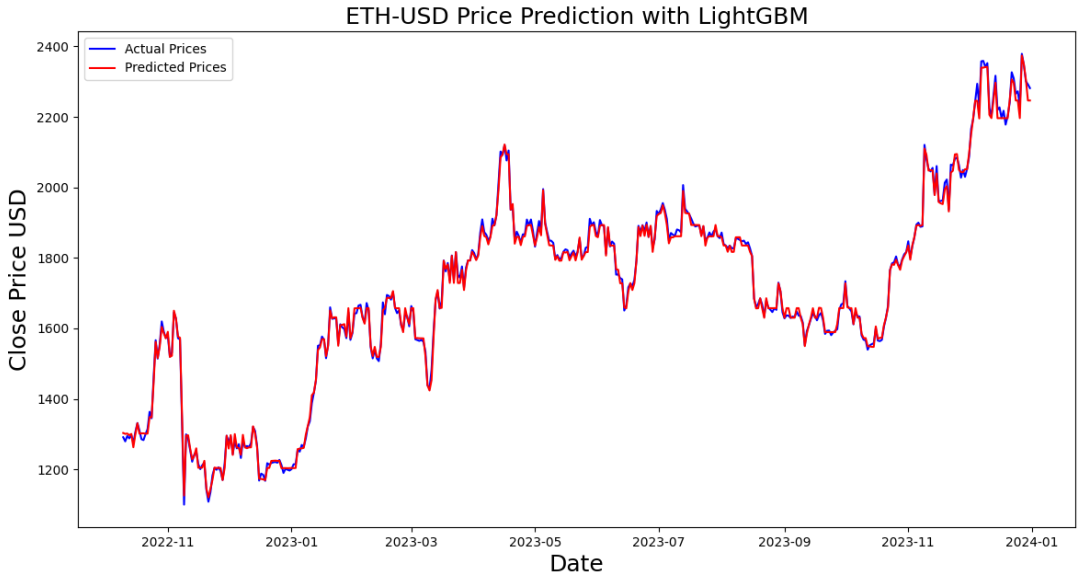

Рисунок 3.11 – Графік реальних значень та прогнозів моделі LightGBM на ціну Ethereum

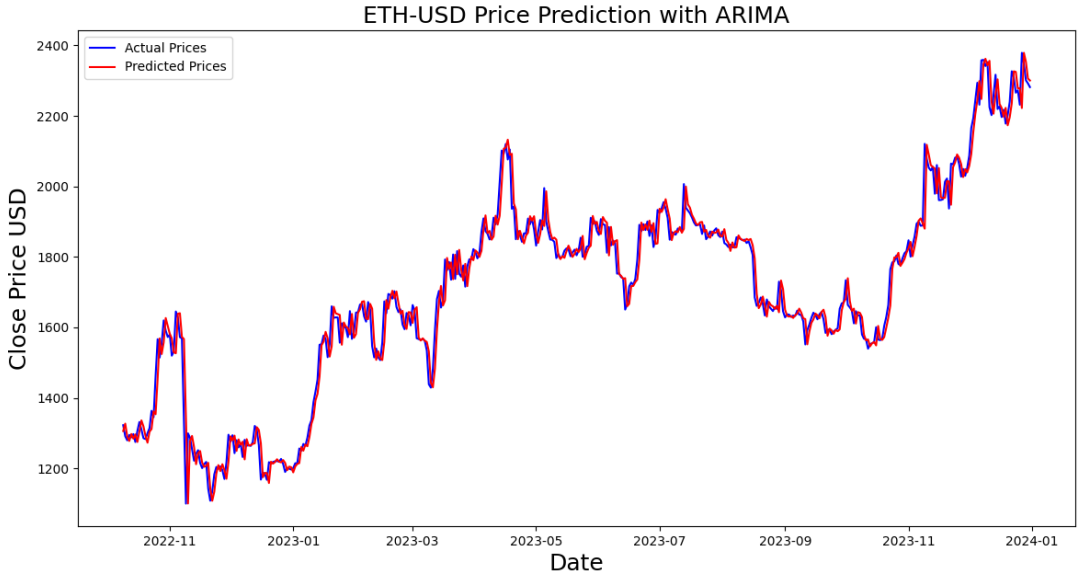

Рисунок 3.12 – Графік реальних значень та прогнозів моделі ARIMA на ціну Ethereum

Оскільки на датасеті Bitcoin та Ethereum ARIMA та LightGBM показали себе по різному, додатково перевіримо як вони себе покажуть на криптоактивах LTC, BNB та XRP.

Таблиця 3.6 – Результати метрик прогнозування для LTC

Model	MSE	MAE	MAPE	RMSE	R^2
LightGBM	0.328692	0.404117	0.490710	0.573317	0.999040
ARIMA	9.613382	2.119001	2.773761	3.100545	0.971987
GRU	34.180458	5.133792	7.104801	5.846405	0.900401
LSTM	10.008946	2.172189	2.860445	3.163691	0.970834

Таблиця 3.7 – Результати метрик прогнозування для BNB

Model	MSE	MAE	MAPE	RMSE	R^2
LightGBM	14.056857	2.140699	0.898446	3.749247	0.991194
ARIMA	58.498001	4.875432	1.778210	7.648398	0.963276
GRU	171.568577	11.100337	4.211427	13.098418	0.892292
LSTM	58.354055	4.753912	1.730072	7.638982	0.963366

Таблиця 3.8 – Результати метрик прогнозування для XRP

Model	MSE	MAE	MAPE	RMSE	R^2
LightGBM	3.0542361e-06	0.0013226	0.262525	0.0017476	0.999707
ARIMA	0.0005334	0.0117854	2.340339	0.0230963	0.948806
GRU	0.0007421	0.0177264	3.629467	0.0272425	0.928776
LSTM	0.0007045	0.0165338	3.422176	0.0265426	0.932389

Як можна побачити з результату метрик в таб. 3.6 – табл. 3.8, в загальному модель градієнтного бустингу LightGBM справляється з прогнозуванням набагато краще ніж ARIMA

3.4 Аналіз отриманих результатів

Підсумовуючи отримані результати, можна сказати що найкраще справляється з задачею прогнозування курсу криптовалют модель градієнтного бустингу LightGBM хоча саме для прогнозування ціни Bitcoin модель ARIMA справилась трохи краще, але дослідивши інші криптовалюти LightGBM виявилась точніше ніж ARIMA. Це може бути зумовлено відмінністю природи тренду ціни Bitcoin від інших криптовалют, так як ARIMA може бути більш ефективною для стабільних і менш шумних даних з явними лінійними

трендами, тоді як LightGBM може краще працювати з даними, які мають складні, нелінійні взаємозв'язки та більше шуму. Трошки гірше з поставленою задачею справились нейронні мережі LSTM та GRU, так як вони є менш точними за попередні моделі, хоча в цілому вони з задачею справились та показали хороші результати. Серед них кращим виявилась LSTM, так як є трохи точнішою, хоча і тренується вона значно довше.

Висновки до розділу 3

Було розроблено програмний продукт для задачі прогнозування цін на криптовалюти за допомогою моделей ARIMA, LightGBM, LSTM та GRU. Надані результати прогнозування на криптовалютах та їх оцінки точності. Проведено аналіз та порівняння результатів прогнозування моделей.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО ВАРСТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

В заданому розділі буде проведено оцінювання основних характеристик для майбутнього програмного продукту, що спеціалізується на дослідженні демографічного стану.

Дана реалізація буде сприяти проведенню усіх необхідних досліджень, що дасть змогу якісно дослідити питання не лише в Україні, проте у всьому світі.

Також в даному дослідженні показано різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що має вплив на економічні фактори та сумісність з майбутнім програмним продуктом. Для цього застосовувався апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

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

Технічні вимоги до програмного продукту є такими:

- функціонування на персональних комп'ютерах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка можливого програмного продукту, яка дозволяє аналізувати різні характеристики, що безпосередньо впливають на стійкість підприємства. Беручи за основу цю функцію, можна виділити такі.

1. F_1 – вибір мови програмування.
2. F_2 – вигляд програмного продукту.
3. F_3 – графічні показники.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1.

a) Python.

b) R.

Функція F_2.

a) Створення графічого додатку.

b) Консольний інтерфейс з графіками

Функція F_3.

a) Застосування існуючих моделей та бібліотек.

b) Створення власних моделей.

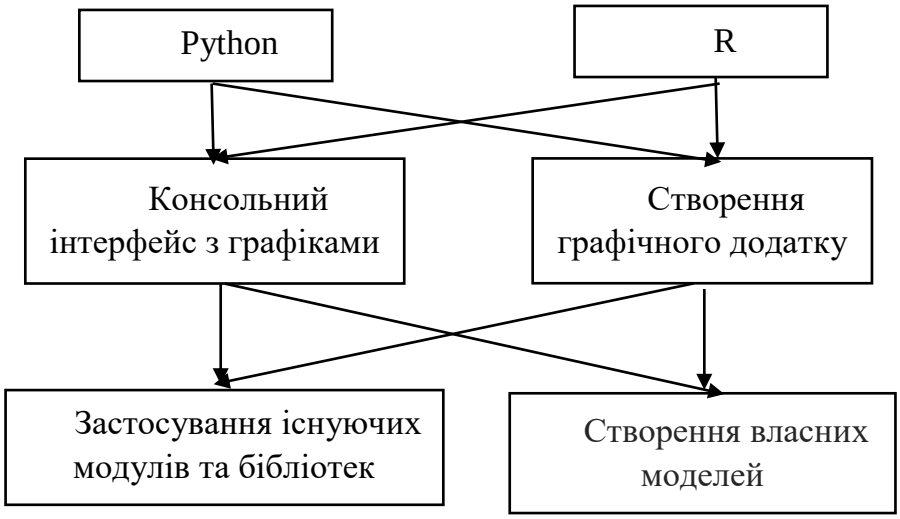

Рисунок 4.1 – Морфологічна карта

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

Таблиця 4.1 – Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	*А*	Швидкість розробки програмного продукту, велика кількість готових бібліотек, методів та методичних вказівок	Швидкість виконання програми, якщо не звернути увагу на покращення швидкодії
	Б	Багато вбудованих функцій для статистичних методів та моделей	Великі затрати часу на реалізацію програмного продукту, складна у вивченні через невелику кількість методичних вказівок
F_2	*А*	Значна економія часу, та більша швидкодія	Можуть бути складнощі у розумінні для користувача, менш приємний вигляд додатку
	Б	Зручність та зрозумілість для користувача	Значні затрати у часі на дихзайн реалізацію та підтримку
F_3	*А*	Значна економія часу при цьому зі збереженням кастомізації моделей за допомогою необхідних вхідних параметрів	Неможливість корегування алгоритму при зосередженні на певній задачі, поза межами параметрів врахованих розробниками.
	Б	Можливість змін в моделі для покращення алгоритму під конкретну поставлену задачу	Значні затрати у часі реалізації

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. Перевагу даємо швидкості, та великій кількості готових бібліотек та модулів для реалізації. Для спрощення роботи по написанню коду варіант Б має бути відкинутий.

Функція F_2. Програма допускає обрання обох варіантів. Можливо використати варіанти А чи Б.

Функція F_3. Реалізація першого варіанту є набагато швидшою для реалізації програмного продукту. Це варіант А.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

$$F_1\text{а} - F_2\text{а} - F_3\text{а},$$

$$F_1\text{а} - F_2\text{б} - F_3\text{а}.$$

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 такі параметри:

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

- *X1* – швидкодія мови програмування;
- *X2* – Точність моделі;
- *X3* – час попередньої обробки 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 мови програмування	X1	оп/мс	20000	25000	30000
Точність моделі	X2	%	70	85	95
Час попередньої обробки даних	X3	мс	350	280	200
Потенційний об’єм програмного коду	X4	кількість рядків коду	2000	1500	800

За даними таблиці 4.3 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

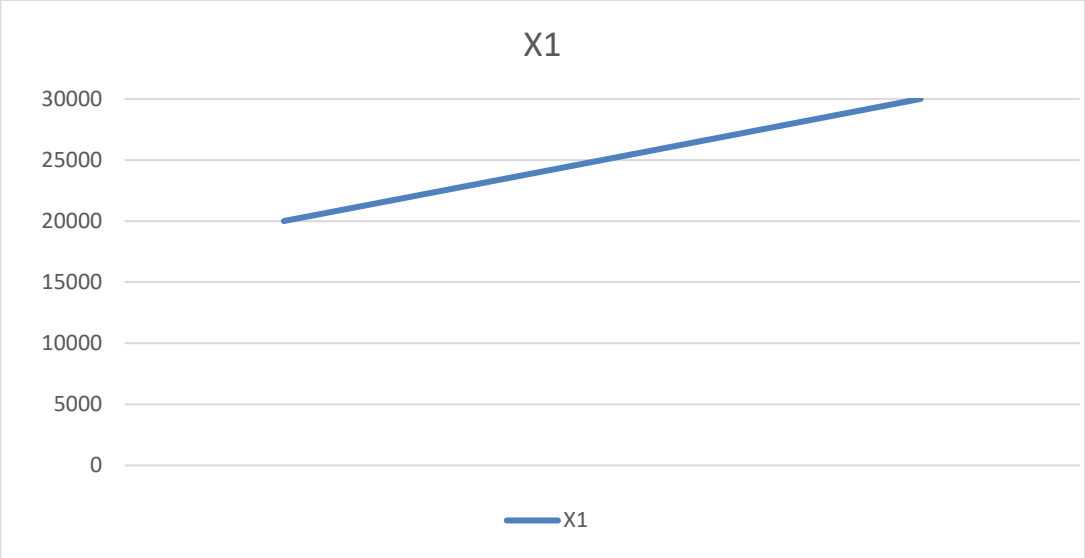

Рисунок 4.2 – X1, швидкодія мови програмування

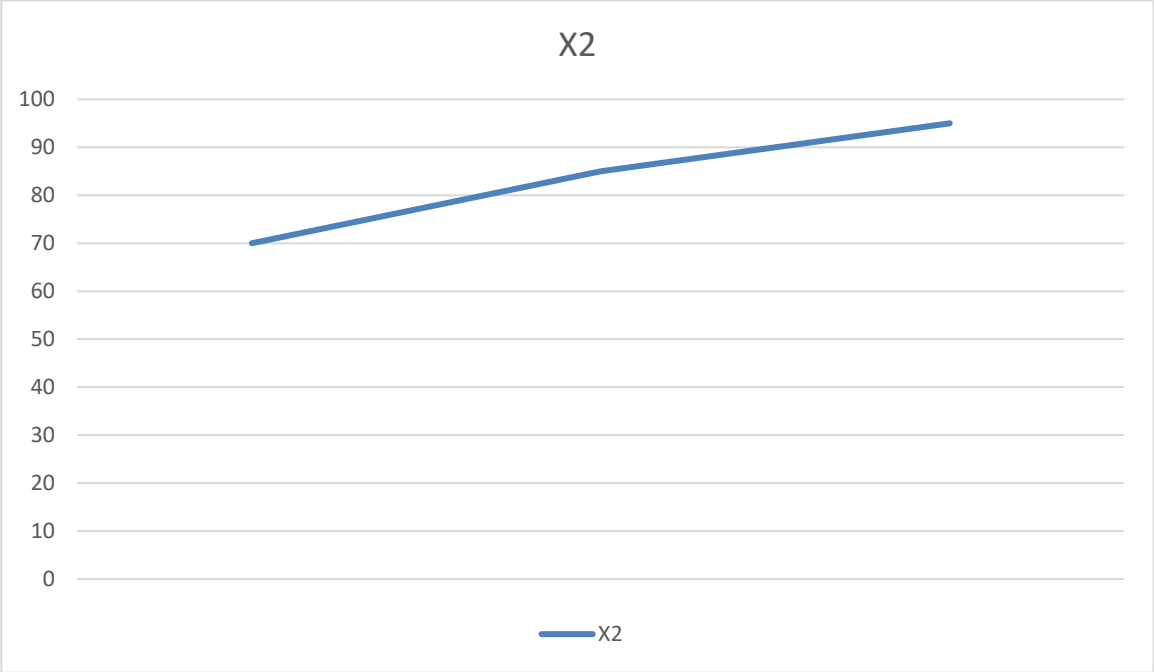

Рисунок 4.3 – X2, Точність моделі

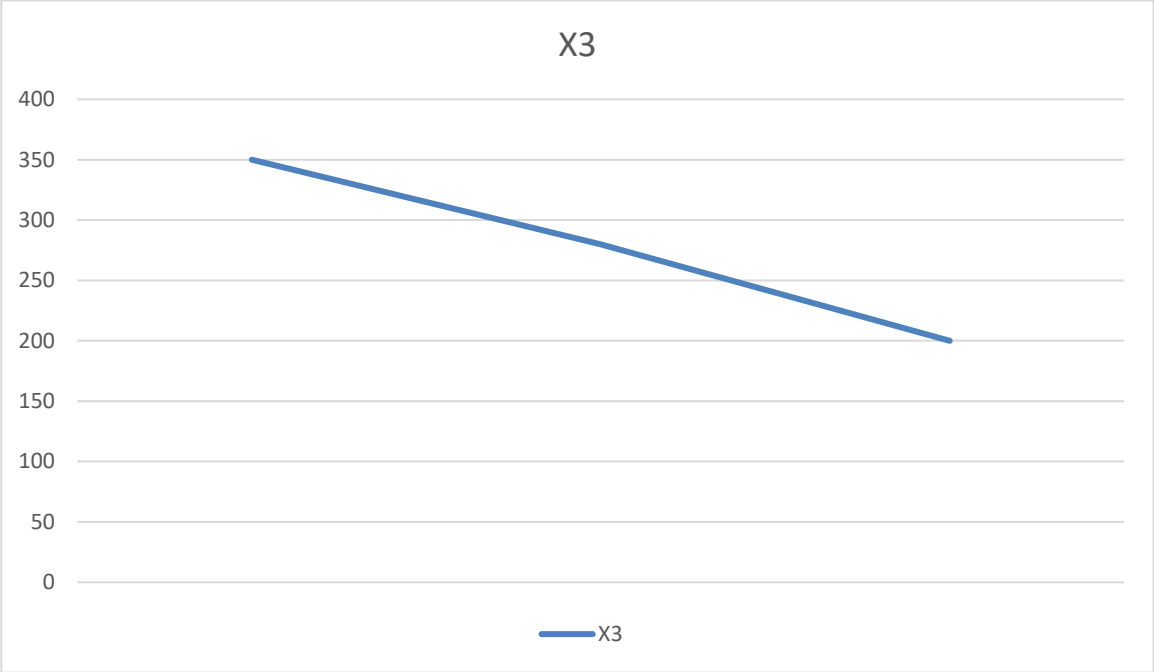

Рисунок 4.4 – X3, час попередньої обробки даних

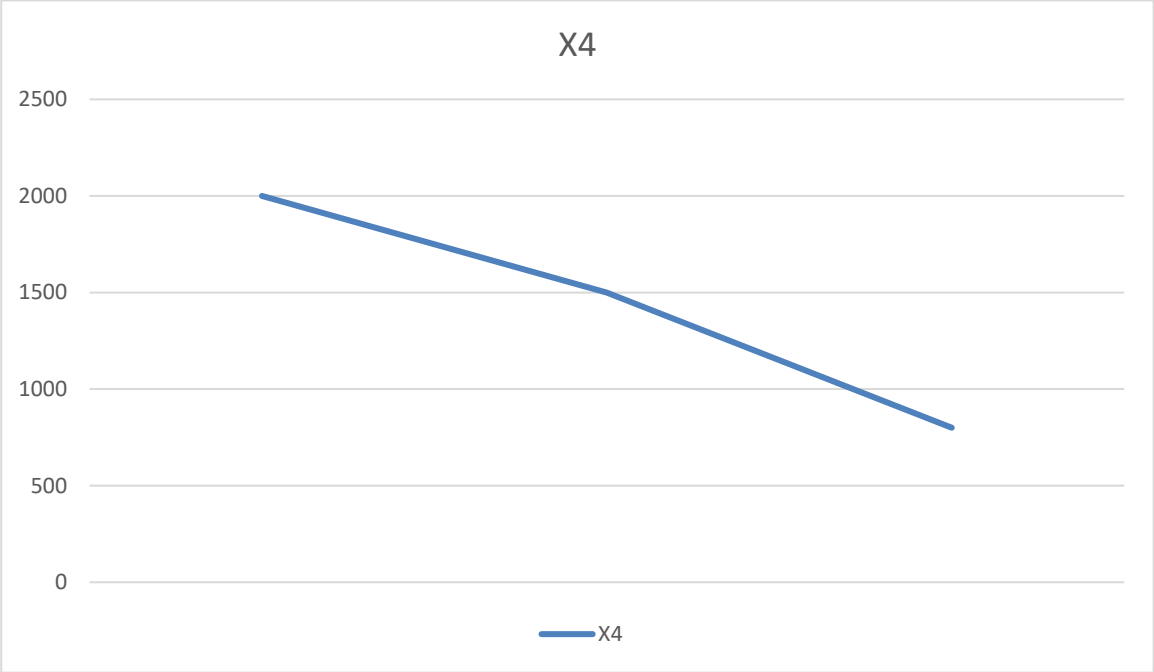

Рисунок 4.5 – X4, потенційний об’єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

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

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

– визначення рівня значимості параметра шляхом присвоєння різних рангів;
– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – 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			
X1	Швидкодія мови програмування	Оп/мс	3	4	3	3	3	5	3	24	6.5	42,25
X2	Точність моделі	%	1	3	1	2	2	3	2	14	-3,5	12,25
X3	Час попередньої обробки даних	мс	1	1	1	1	2	1	1	8	-9,5	90,25
X4	Потенцій-ний об’єм програмного коду	Кількість рядків коду	5	2	5	4	3	1	4	24	6.5	42,25
	Разом		10	10	10	10	10	10	10	70	0	187

Для перевірки степені достовірності експертних оцінок, визначимо такі параметри:

- сума рангів кожного з параметрів і загальна сума рангів:

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

де N – число експертів;

n – кількість параметрів.

- середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17{,}5.$$

- відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T.$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

- загальна сума квадратів відхилення:

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

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 169}{7^2(4^3 - 4)} = 0{,}76 > W_k = 0{,}67.$$

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

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

Таблиця 4.4 – Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	>	>	>	>	>	1,5
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:

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

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

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за такими формулами:

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

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

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

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

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \, .$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 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	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1	1,5	1,5	0,5	4,5	0,28	16,25	59,125	0,28	16,25
X2	0,5	1	1,5	0,5	3,5	0,22	12,25	0,16	34,125	0,16
X3	0,5	0,5	1	0,5	2,5	0,16	9,25	0,21	41,875	0,2
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 варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2* (точність моделі), *X3* (час попередньої обробки даних) та *X4* (потенційний об'єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X1* (швидкість роботи мови програмування) обрано не найгіршим.

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

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

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
F1	А	X1	100	25	0,28	7
F3	А	X2	87	29	0,2	5,8
	Б	X3	27	19	0,16	3,04
F4	А	X4	25	23	0,38	8,74

За даними з таблиці 4.6 за формулою:

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

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 7 + 5{,}8 + 8{,}74 = 21{,}54;$$

$$K_{K2} = 7 + 3{,}04 + 8{,}74 = 18{,}78.$$

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

4.6 Економічний аналіз варіантів розробки

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання.

1. Розробка проекту програмного продукту;
2. Розробка програмної оболонки.

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

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

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

Загальна трудомісткість обчислюється як:

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М},$$

де T_P – трудомісткість розробки ПП;

$K_П$ – поправочний коефіцієнт;

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

$K_М$ – коефіцієнт рівня мови програмування;

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

$K_{СТ.М}$ – 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 =37 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.8.

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 $К_{СТ}$ = 0.9. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$Т_1 = 37 \cdot 1.8 \cdot 0.9 = 59{,}94 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =29 людино-днів, $К_П$ =0.9, $К_{СК}$ = 1, $К_{СТ}$ =0.8:

$$Т_2 = 29 \cdot \ 0.9 \cdot 0.8 = 20.88 \text{ людино-днів.}$$

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

$$Т_I = (59{,}94 \ + 20.88 + 4.8 + 20.88) \cdot 8 = 852 \quad \text{людино-годин.}$$

$$Т_{II} = (59{,}94 \ + 20.88 + 6.91 + 20.88) \cdot 8 = 868{,}88 \quad \text{людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 17000 грн., один аналітик в області даних з окладом 19000. Визначимо середню зарплату за годину за формулою:

$$С_Ч = \frac{М}{T_m \cdot t} \text{грн.,}$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$С_Ч = \frac{17000 + 17000 + 19000}{3 \cdot 21 \cdot 8} = 105{,}16 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою:

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д,$$

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. $С_{ЗП} = 105.16 \cdot 852 \cdot 1.2 = 107515{,}58$ грн.

II. $С_{ЗП} = 105.16 \cdot 868.88 \cdot 1.2 = 109645{,}7$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 107515{,}58 \cdot 0.22 = 23653{,}4$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 109645{,}7 \cdot 0.22 = 24122{,}06$ грн.

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

Так як одна ЕОМ обслуговує одного програміста з окладом 17000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 17000 \cdot 0{,}2 = 40800 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 40800 \cdot (1 + 0.2) = 48960 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 48960 \cdot 0,22 = 10771,2 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 20% та вартості ЕОМ – 10000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.4 \cdot 0.2 \cdot 10000 = 2800 \text{ грн.},$$

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

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

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

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 12 – 16) \cdot 8 \cdot 0.35 = 627,2 \text{ години},$$

де $Д_К$ – календарна кількість днів у році;

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

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

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

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 627,2 \cdot 0,2 \cdot 0,3 \cdot 5,23 = 196,82 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$ – коефіцієнтом зайнятості приладу;

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

Накладні витрати розраховуємо за формулою:

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

Тоді, річні експлуатаційні витрати будуть:

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

$$С_{ЕКС} = 48960 + 10771,2 + 2800 + 1120 + 196,82 + 6700 = 70548,02 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 70548,02 / 627,2 = 112,5 \text{ грн/год.}$$

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:

$$C_М = C_{М-Г} \cdot T,$$

I. $C_М = 112,5 \cdot 852 = 95850$ грн.

II. $C_М = 112,5 \cdot 868,88 = 97749$ грн.

Накладні витрати складають 67% від заробітної плати:

$$C_Н = C_{ЗП} \cdot 0,67,$$

I. $C_Н = 107515,58 \cdot 0,67 = 72035,45$ грн.

II. $C_Н = 109645,70 \cdot 0,67 = 73462,6$ грн.

Отже, вартість розробки ПП за варіантами становить:

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

I. $C_{ПП} = 107515,58 + 23653,4 + 95850 + 72035,45 = 299054.43$ грн.

II. $C_{ПП} = 109645,7 + 24122,06 + 97749 + 73462,6 = 304979.36$ грн.

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР j} = К_{Кj} / С_{Фj},$$

$$К_{ТЕР1} = 21,12 / 300138.95 = 7,038 \cdot 10^{-5},$$

$$К_{ТЕР2} = 18,78 / 304979.36 = 6,158 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1}= 7,203 \cdot 10^{-5}$.

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 $К_{ТЕР} = 7,203 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- вибір програмного продукту – Python;
- реалізація моделей за допомогою існуючих модулів та бібліотек;
- використання консольного інтерфейсу.

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

Висновки до розділу 4

В даній частині було проведено повний функціонально-вартісний аналіз програмного продукту. Також було знайдено оцінку основних функцій програмного продукту.

В результаті виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, було визначено та проведено оцінку основних функцій програмного продукту, а також знайдено параметри, які його характеризують.

На основі аналізу вибрано варіант реалізації програмного продукту.

ВИСНОВКИ

У результаті виконання дипломної роботи проведено дослідження, аналіз та порівняняння точності різних моделей штучного інтелекту у задачах прогнозування курсу криптовалют.

Було описано поняття, що по’язані з криптовалютними ринками, їх історію, різні підходи до прогнозування ціни на криптовалюти, та моделі які використовуються у цій роботі, а саме: ARIMA, LightGBM, LSTM, GRU. Після чого, було детальніше досліджено алгоритми та архітектуру вищенаведених моделей та метрики для оцінки їх точності прогнозування. Відповідно для виконання поставленої задачі та на основі досліджених даних реалізовано програмний продукт на мові Python.

В ході виконання роботи було оцінено якість прогнозування вищезазначених моделей на таких криптовалютах як BTC, ETH, LTC, BNB, XRP за допомогою метрик оцінювання R^2, MSE, MAE, MAPE і RMSE та відповідно порівняно їх. За результатами дослідження найефективнішим та найточнішою з них виявилась модель градієнтного бустингу LightGBM, хоча для BTC найкраще себе показала статистична модель ARIMA, але для інших криптоактивів LightGBM виявився значно точнішим. В загальному всі моделі показали хорошу точність та ефективність, і при налаштуванні моделей під конкретні дані можуть показати дуже хороші результати.

ПЕРЕЛІК ПОСИЛАНЬ

1. The History of Cryptocurrency. URL: https://www.investopedia.com/tech/history-cryptocurrency (дата звернення: 22.05.2024).
2. What Is Blockchain Technology? URL: https://www.ibm.com/topics/blockchain (дата звернення: 22.05.2024).
3. Alber N. Forecasting cryptocurrencies prices using machine learning: The case of the exchange rates of Bitcoin/USD. Computational Economics, 2021, 57(2), 603-617.
4. Alexander C. Market Models: A Guide to Financial Data Analysis. Wiley, 2001.
5. Antonopoulos A. M. Mastering Bitcoin: Unlocking Digital Cryptocurrencies. O'Reilly Media, 2014.
6. Box G. E. P., Jenkins G. M., Reinsel G. C. Time Series Analysis: Forecasting and Control. Wiley, 2015.
7. Breiman L., Friedman J., Stone C. J., Olshen R. A. Classification and Regression Trees. CRC Press, 1984.
8. Brockwell P. J., Davis R. A. Introduction to Time Series and Forecasting. Springer, 2002.
9. Brownlee J. Introduction to Time Series Forecasting with Python: How to Prepare Data and Develop Models to Predict the Future. Machine Learning Mastery, 2017.
10. Buterin V. Ethereum Whitepaper: A Next-Generation Smart Contract and Decentralized Application Platform. Ethereum Foundation, 2013.
11. Carr M. Statistical Arbitrage: A Decision-Making Framework. Wiley, 2002.
12. Chatfield C. The Analysis of Time Series: An Introduction. Chapman and Hall/CRC, 2003.

13. Chen Y., Yuan Y., McAuley J. J. Blockchain for Decentralized Applications: Operational and Legal Considerations. Journal of Management Information Systems, 2019, 36(3), 1042-1056.
14. Chung J., et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv preprint arXiv:1412.3555, 2014.
15. Ciaian P., Rajcaniova M., Kancs D. The Economics of BitCoin Price Formation. Applied Economics, 2016, 48(19), p.1799-1815.
16. European Central Bank. Crypto-Assets: Implications for financial stability, monetary policy, and payments and market infrastructures. 2019.
17. Evans D. S. Economic Aspects of Bitcoin and Other Decentralized Public-Ledger Currency Platforms. University of Chicago Coase-Sandor Institute for Law & Economics Research Paper No. 685, 2014.
18. Gers F. A., Schmidhuber J., Cummins F. Learning to forget: Continual prediction with LSTM. Neural Computation, 2000, 12(10), 2451-2471.
19. Graves A., Schmidhuber J. Framewise phoneme classification with bidirectional LSTM networks. IEEE International Joint Conference on Neural Networks, 2005.
20. Hakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2009. URL: https://bitcoin.org/bitcoin.pdf (дата звернення: 25.05.2024).
21. Hakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System. Bitcoin.org, 2008.
22. Hochreiter S., Schmidhuber J. Long Short-Term Memory. Neural Computation, 1997, 9(8), p.1735-1780.
23. Hochreiter S., Schmidhuber J. Long Short-Term Memory. Neural Computation, 1997, 9(8), p. 1735-1780.
24. Hull J. C. Risk Management and Financial Institutions. Wiley, 2016.
25. Hyndman R. J., Athanasopoulos G. Forecasting: principles and practice. OTexts, 2018.

26. Jang H., Lee J. An Empirical Study on Modeling and Prediction of Bitcoin Prices with Bayesian Neural Networks based on Blockchain Information. 2017.
27. Ke G., et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Advances in Neural Information Processing Systems, 2017.
28. McNally S., Roche J., Caton S. Predicting the price of Bitcoin using Machine Learning. 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), IEEE, 2018, pp. 339-343.
29. Microsoft. LightGBM Documentation. URL: https://lightgbm.readthedocs.io/ (дата звернення: 25.05.2024).
30. Mittal A., Goel A. Stock prediction using Twitter sentiment analysis. Stanford University, 2012.
31. Mougayar W. The Business Blockchain: Promise, Practice, and the Application of the Next Internet Technology. Wiley, 2016.
32. Murphy J. J. Technical Analysis of the Financial Markets. New York Institute of Finance, 1999.
33. Narang R. Inside the Black Box: A Simple Guide to Quantitative and High-Frequency Trading. Wiley, 2013.
34. Narayanan A., Bonneau J., Felten E., Miller A., Goldfeder S. Bitcoin and Cryptocurrency Technologies. Princeton University Press, 2016.
35. Olah C. Understanding LSTM Networks. 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (дата звернення: 26.05.2024).
36. Părăușanu A., Vasilescu L., Dîmbu E. V. Blockchain Technology: Between Hopes and Limiting Factors. Economy and Sociology: Theoretical and Scientifical Journal, 2018, (3), p. 133-142.
37. Pring M. J. Technical Analysis Explained. McGraw-Hill, 2002.
38. Rokach L., Maimon O. Top-Down Induction of Decision Trees Classifiers - A Survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2005, 35(4), 476-487.

39. Rosic A. Proof of Work vs Proof of Stake: Basic Mining Guide. 2020. URL: https://blockgeeks.com/guides/proof-of-work-vs-proof-of-stake/ (дата звернення: 27.05.2024).
40. Schwager J. D. Technical Analysis. Wiley Trading Series, 2012.
41. Tapscott D., Tapscott A. Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World. Penguin, 2016.
42. The Blockchain Technology and Decentralized Finance (DeFi). CoinDesk. URL: https://www.coindesk.com/ (дата звернення: 27.05.2024).
43. Trautman L. J. Virtual Currencies; Bitcoin & What Now After Liberty Reserve, Silk Road, and Mt. Gox? Richmond Journal of Law and Technology, 2014, 20(4).
44. Treleaven P., Galas M., Lalchand V. Algorithmic Trading Review. Communications of the ACM, 2013, 56(11), p. 76-85.
45. Zhang P., et al. A comprehensive survey on KNN-based imputation and us with LightGBM. Applied Intelligence, 2020.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
import warnings
import yfinance as yf
import time
import lightgbm as lgb
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy import stats
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout, GRU
import statsmodels.api as sm
from itertools import product
from statsmodels.tsa.arima.model import ARIMA


def import_data(file_name):
    return pd.read_csv(file_name)


def prepare_data(data):
    data['Date'] = pd.to_datetime(data['Date'])
    data.set_index('Date', inplace=True)
    data = data.dropna()
    return data
```

```
def score_prediction(y_test, predictions, model):
    mae = mean_absolute_error(y_test, predictions)
    mse = mean_squared_error(y_test, predictions)
    mape = np.mean(np.abs((y_test - predictions) / y_test)) * 100
    r2 = r2_score(y_test, predictions)
    rmse = np.sqrt(mse)
    return {'Model': model, 'MAE': mae, 'MSE': mse, 'MAPE': mape, 'R2': r2, 'RMSE':
rmse}



def split_data(data):
    close_prices = data['Close'].values.reshape(-1, 1)

    # Scale the data
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(close_prices)

    # Define the number of previous time steps to use
    time_steps = 60

    # Create training and testing sets
    train_size = int(len(scaled_data) * 0.8)
    train_data = scaled_data[:train_size]
    test_data = scaled_data[train_size - time_steps:]

    def create_dataset(data, time_steps):
        X, y = [], []
        for i in range(time_steps, len(data)):
            X.append(data[i - time_steps:i, 0])
            y.append(data[i, 0])
```

```
        return np.array(X), np.array(y)

    X_train, y_train = create_dataset(train_data, time_steps)
    X_test, y_test = create_dataset(test_data, time_steps)

    # Reshape input to be [samples, time steps, features]
    X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
    X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

    return X_train, y_train, X_test, y_test, scaler


def lstm_model(data, pair):
    # Split data
    X_train, y_train, X_test, y_test, scaler = split_data(data)

    # Build the LSTM model
    model = Sequential()
    model.add(LSTM(50, return_sequences=True, input_shape=(
        X_train.shape[1], 1)))
    model.add(LSTM(50, return_sequences=False))
    model.add(Dense(25))
    model.add(Dense(1))

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

    # Train the model
    start_time = time.time()
    history_lstm = model.fit(X_train, y_train, batch_size=1, epochs=20)
```

```
    end_time = time.time()

    # Plot loses by epochs
    plot_history(history_lstm, 'LSTM')

    # Make predictions
    predictions = model.predict(X_test)
    predictions = scaler.inverse_transform(predictions)

    # Rescale predictions and test price
    y_test_rescaled = scaler.inverse_transform(y_test.reshape(-1, 1))

    # Get model score
    score = score_prediction(y_test_rescaled, predictions, 'LSTM')

    # Visualized the data
    plot_predictions(data, y_test_rescaled, predictions, 'LSTM', pair)

    return score, end_time - start_time


def lightgbm_model(data, pair):
    # Function to create lag features
    def add_lag_features(df, lags):
        for lag in lags:
            df[f'Close_lag_{lag}'] = df['Close'].shift(lag)
        return df

    # Define the lags you want to create
    lags = [1, 2, 3, 7]
```

```
    # Add lag features to the datasets
    data = add_lag_features(data, lags)

    # Drop rows with NaN values resulting from the lagging operation
    data = data.dropna()

    # Drop columns that won't be used as features
    features = data.drop(columns=['Close'])

    # Create feature and target arrays
    X = features.values
    y = data['Close'].values
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
shuffle=False)

    # Create the LightGBM dataset
    train_data = lgb.Dataset(X_train, label=y_train)
    test_data = lgb.Dataset(X_test, label=y_test)
    params = {
        'metric': 'rmse',
        'num_leaves': 31,
        'learning_rate': 0.1,
    }

    # Train the model
    model = lgb.train(params, train_data, valid_sets=[test_data])

    # Make predictions
    y_pred = model.predict(X_test, num_iteration=model.best_iteration)
```

```
    # Evaluate the model
    score = score_prediction(y_test, y_pred, 'LightGBM')

    plot_predictions(data, y_test, y_pred, 'LightGBM', pair)

    return score


def gru_model(data, pair):
    # Split data
    X_train, y_train, X_test, y_test, scaler = split_data(data)

    # Build the GRU model
    model = Sequential()
    model.add(GRU(units=50, return_sequences=True,
input_shape=(X_train.shape[1], 1), activation='tanh'))
    model.add(Dropout(0.2))
    model.add(GRU(units=50, return_sequences=False, activation='tanh'))
    model.add(Dropout(0.2))
    model.add(Dense(units=1))

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

    # Train the model
    start_time = time.time()
    history_gru = model.fit(X_train, y_train, epochs=20, batch_size=32)
    end_time = time.time()
```

```
    # Plot loses by epochs
    plot_history(history_gru, 'GRU')

    # Make predictions
    predictions = model.predict(X_test)

    # Rescale predictions and test price
    predictions = scaler.inverse_transform(predictions)
    y_test_rescaled = scaler.inverse_transform(y_test.reshape(-1, 1))

    # Get model score
    score = score_prediction(y_test_rescaled, predictions, 'GRU')

    # Plot the results of prediction
    plot_predictions(data, y_test_rescaled, predictions, 'GRU', pair)

    return score, end_time - start_time



def plot_predictions(data, y_test, predictions, model, pair):
    plt.figure(figsize=(14, 7))
    plt.plot(data.index[-len(y_test):], y_test, label='Actual Prices', color='blue')
    plt.plot(data.index[-len(predictions):], predictions, label='Predicted Prices',
color='red')
    plt.xlabel('Date', fontsize=18)
    plt.ylabel('Close Price USD', fontsize=18)
    plt.title(f'{pair} Price Prediction with {model}', fontsize=18)
    plt.legend()
    plt.show()
```

```
def plot_history(model_history, model):
    plt.figure(figsize=(10, 7))
    plt.plot(model_history.history["loss"])
    plt.xlabel("Epochs")
    plt.ylabel("Losses")
    plt.title(f"{model} model, Loss vs Epoch")
    plt.show()


def invboxcox(y, lmbda):
    if lmbda == 0:
        return np.exp(y)
    else:
        return np.exp(np.log(lmbda * y + 1) / lmbda)


def arima_model(data, pair):
    # Stationarity check  of the series
    print("Dickey–Fuller test: p=%f" % sm.tsa.stattools.adfuller(data.Close)[1])

    # Box-Cox Transformations
    data['Weighted_Price_box'], lmbda = stats.boxcox(data.Close)
    print("Dickey–Fuller test: p=%f" % sm.tsa.stattools.adfuller(data.Close)[1])

    # Regular differentiation
    data['Weighted_Price_box_diff'] = data.Weighted_Price_box -
data.Weighted_Price_box.shift(1)
    print("Dickey–Fuller test: p=%f" %
sm.tsa.stattools.adfuller(data.Weighted_Price_box_diff[1:])[1])
```

```
    # Initial approximation of parameters
    qs = range(0, 5)
    ps = range(0, 5)
    d = range(0, 5)
    parameters = product(ps, d, qs)
    parameters_list = list(parameters)
    len(parameters_list)

    # Model Selection
    results = []
    best_aic = 0
    best_model = 0
    warnings.filterwarnings('ignore')
    for param in parameters_list:
        try:
            model = ARIMA(data.Weighted_Price_box, order=(param[0], param[1],
param[2])).fit()
        except ValueError:
            print('wrong parameters:', param)
            continue
        aic = model.aic
        if aic < best_aic:
            best_model = model
            best_aic = aic
        results.append([param, model.aic])

    # Best Models
    result_table = pd.DataFrame(results)
    result_table.columns = ['parameters', 'aic']
```

```
    print(result_table.sort_values(by='aic', ascending=True).head())
    print(best_model.summary())

    # Prediction
    train_size = int(len(data) * 0.8)
    y_test = data['Close'].values.reshape(-1, 1)
    y_test = y_test[train_size:]

    y_pred = invboxcox(best_model.predict(start=train_size), lmbda)
    y_pred = y_pred.values.reshape(-1, 1)

    # Get model score
    score = score_prediction(y_test, y_pred, 'ARIMA')

    # Plot the results of prediction
    plot_predictions(data, y_test, y_pred, 'ARIMA', pair)

    return score


if __name__ == '__main__':
    dataBTC = yf.download('BTC-USD', end="2024-01-01").reset_index()
    dataBTC = prepare_data(dataBTC)
    dataBTC.Close.plot(title="Bitcoin Price", figsize=(14, 6))
    plt.tight_layout()
    plt.xlabel('Years')
    plt.ylabel('US Dollars')
    plt.show()
    lstmBTC = lstm_model(dataBTC, 'BTC-USD')
    gruBTC = gru_model(dataBTC, 'BTC-USD')
```

```
lightgbmBTC = lightgbm_model(dataBTC, 'BTC-USD')
arimaBTC = arima_model(dataBTC, 'BTC-USD')

dataETH = yf.download('ETH-USD', end="2024-01-01").reset_index()
dataETH = prepare_data(dataETH)
dataETH.Close.plot(title="Ethereum Price", figsize=(14, 6))
plt.tight_layout()
plt.xlabel('Years')
plt.ylabel('US Dollars')
plt.show()
lstmETH = lstm_model(dataETH, 'ETH-USD')
gruETH = gru_model(dataETH, 'ETH-USD')
lightgbmETH = lightgbm_model(dataETH, 'ETH-USD')
arimaETH = arima_model(dataETH, 'ETH-USD')
print('Bitcoin score:')
print(lstmBTC)
print(gruBTC)
print(lightgbmBTC)
print(arimaBTC)
print('--------------------------------------------------------------------------------')
print('Ethereum score:')
print(lstmETH)
print(gruETH)
print(lightgbmETH)
print(arimaETH)

dataLTC = yf.download('LTC-USD', end="2024-01-01").reset_index()
dataLTC = prepare_data(dataLTC)
lightgbmLTC = lightgbm_model(dataLTC, 'LTC-USD')
arimaLTC = arima_model(dataLTC, 'LTC-USD')
```

```
gruLTC = gru_model(dataLTC, 'LTC-USD')
lstmLTC = lstm_model(dataLTC, 'LTC-USD')
print('------------------------------------------------------------------------')
print('LTC score:')
print(lightgbmLTC)
print(arimaLTC)
print(gruLTC)
print(lstmLTC)

dataBNB = yf.download('BNB-USD', end="2024-01-01").reset_index()
dataBNB = prepare_data(dataBNB)
lightgbmBNB = lightgbm_model(dataBNB, 'BNB-USD')
arimaBNB = arima_model(dataBNB, 'BNB-USD')
gruBNB = gru_model(dataBNB, 'BNB-USD')
lstmBNB = lstm_model(dataBNB, 'BNB-USD')
print('------------------------------------------------------------------------')
print('BNB score:')
print(lightgbmBNB)
print(arimaBNB)
print(gruBNB)
print(lstmBNB)

dataXRP = yf.download('XRP-USD', end="2024-01-01").reset_index()
dataXRP = prepare_data(dataXRP)
lightgbmXRP = lightgbm_model(dataXRP, 'XRP-USD')
arimaXRP = arima_model(dataXRP, 'XRP-USD')
gruXRP = gru_model(dataXRP, 'XRP-USD')
lstmXRP = lstm_model(dataXRP, 'XRP-USD')
print('------------------------------------------------------------------------')
print('XRP score:')
```

```
print(lightgbmXRP)
print(arimaXRP)
print(gruXRP)
print(lstmXRP)
```