МІНІСТЕСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КІБЕРНЕТИКИ,
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ІНЖЕНЕРІЇ
НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ
КАФЕДРА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ ТА ЕКОНОМІЧНОЇ КІБЕРНЕТИКИ

Допущено до захисту:

Завідувач кафедри

_____ д. е. н., проф. П. М. Грицюк

«_____» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня «магістр»

за освітньо-професійною програмою «Інформаційні системи і технології»

спеціальності 126 «Інформаційні системи та технології»

на тему: «Інтелектуальна система прогнозування вартості криптовалюти на основі нейронних мереж LSTM»

Виконав:

здобувач вищої освіти 2 курсу, групи ІТБ-61м

Антонюк Богдан Сергійович

Керівник:

д. е. н., проф. П. М. Грицюк

Рецензент:

Рівне 2025

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ПК	–	персональний комп’ютер
БД	–	база даних
AI	–	Artificial intelligence (штучний інтелект)
АСУ	–	автоматизована система управління
АІС	–	автоматизована інформаційна система
ТЗ	–	технічне завдання

ВСТУП

Штучний інтелект став одним із найзначніших технологічних досягнень XXI століття, вплинувши на розвиток людства так само масштабно, як свого часу поява всесвітньої павутини. Це не просто набір алгоритмів чи інструментів — це окремий напрям сучасних інформаційних технологій, метою якого є моделювання процесів мислення і прийняття рішень. Щодня системи зі штучним інтелектом допомагають нам виконувати безліч рутинних і складних завдань, спрощуючи доступ до інформації, оптимізуючи робочі процеси, пришвидшуючи аналіз даних і навіть створюючи новий контент.

Серед найвідоміших представників інтелектуальних систем можна виділити ChatGPT, Gemini, DeepSeek, Marlin та інші сучасні моделі, здатні згенерувати текст, створити реалістичне зображення, обробити відео, провести аналітику чи надати експертні рекомендації. Достатньо лише короткого запиту, і такі системи виконують складні обчислення, що раніше вимагали участі великої кількості фахівців.

Актуальність обраної теми полягає в тому, що однією з найбільш перспективних сфер застосування штучного інтелекту є прогнозування часових рядів. Сучасні нейронні мережі здатні аналізувати великі обсяги історичних даних, виявляти приховані закономірності та з високою точністю передбачати майбутні показники. Ця технологія набуває особливої актуальності у фінансовій сфері, де точність прогнозування визначає ефективність інвестиційних рішень. Саме ринок криптовалют це ринок, що активно використовує алгоритми прогнозування. Висока волатильність цифрових активів, швидкі зміни попиту та пропозиції, залежність від глобальних подій і новин роблять традиційні методи прогнозування малоефективними. Саме тому застосування нейромереж, здатних навчатися на складних і нестабільних даних, відкриває нові можливості для аналізу та передбачення вартості криптовалют.

Розробленість теми роботи. Проблематика прогнозування вартості криптовалют активно досліджується науковою спільнотою та комерційними

аналітичними компаніями. Завдяки високій волатильності цифрових активів, задача точного передбачення їхньої динаміки стала особливо актуальною для трейдерів, інвесторів та фінансових організацій.

Найбільш помітні дослідження у цій сфері здійснюють провідні криптобіржі (Binance Research, Coinbase Institutional), аналітичні платформи (Glassnode, Santiment), а також наукові центри, що працюють із методами машинного навчання та нейронних мереж.

Попри значні досягнення, існує низка відкритих проблем, які потребують більш детального вивчення. Серед них:

- складність багатофакторного аналізу, що включає не лише історичні ціни, а й новинний фон, ринкову ліквідність, макроекономічні показники та активність трейдерів;
- підвищена похибка моделей під час різких стрибків або падінь ціни, викликаних непередбачуваними подіями;
- недостатня адаптивність класичних статистичних методів до умов нестабільного крипторинку;
- обмеженість наявних моделей при прогнозуванні на кілька кроків уперед, особливо в умовах швидкої зміни трендів.

Незважаючи на використання сучасних алгоритмів штучного інтелекту, питання точного та стабільного прогнозування криптовалют залишається відкритим. Це визначає актуальність подальших досліджень у напрямі застосування глибоких нейронних мереж та комбінованих моделей для підвищення точності прогнозів. Саме ці виклики стали основою для проведення даної роботи та розробки спеціалізованої нейромережі для аналізу та передбачення вартості криптовалют.

Об’єктом дослідження даної магістерської роботи є динаміка цін криптовалют у вигляді часових рядів.

Предметом дослідження є методи прогнозування вартості криптовалют на основі штучних нейронних мереж та їх інтеграція у програмне забезпечення у вигляді браузерного розширення.

Метою дослідження є розробка власної нейронної мережі та програмного рішення у форматі браузерного розширення, яке здійснює прогнозування вартості криптовалют у реальному часі та може використовуватися кінцевими користувачами.

Для досягнення мети необхідно виконати такі завдання:

- Проаналізувати сучасні підходи прогнозування фінансових часових рядів за допомогою штучних нейронних мереж.
- Дослідити особливості ринку криптовалют та фактори, які впливають на зміну їх вартості.
- Розробити архітектуру нейронної мережі, оптимізовану для задач прогнозування курсу криптовалют.
- Здійснити підбір та підготовку даних для навчання моделі, включаючи нормалізацію, формування ознак та генерацію вибірок.
- Реалізувати програмне модульне рішення на основі Python для навчання нейронної мережі.
- Створити браузерне розширення з інтегрованою моделлю нейромережі для відображення прогнозів та виконання локальних обчислень або взаємодії з сервером.
- Провести тестування роботи нейромережі та браузерного розширення, оцінити точність прогнозів і зручність використання.
- Надати рекомендації щодо можливого практичного застосування розробленого рішення.

Методи дослідження У роботі застосовано такі методи:

Аналіз літературних та наукових джерел. Використовується для оцінки сучасних підходів до прогнозування криптовалют, розробки нейронних мереж та

створення інтегрованих програмних рішень (десктопних, веб та браузерних інструментів).

Методи моделювання. Застосовуються для проєктування нейронної мережі:

- розробка архітектури моделі (наприклад: LSTM, GRU, CNN-LSTM, hybrid models);
- визначення функцій активації, оптимізаторів, метрик та параметрів навчання; формування множини ознак на підставі цінових показників та технічних індикаторів.

Методи програмування. Використовуються для реалізації комплексу програмного забезпечення:

- розробка нейромережі у середовищі Python із застосуванням бібліотек (TensorFlow / PyTorch, Pandas, Scikit-learn);
- створення браузерного розширення, що взаємодіє з нейромережею (через локальне виконання або API);
- реалізація механізмів обробки даних, отримання котирувань та візуалізації прогнозів.

Методи тестування. Спрямовані на оцінку якості прогнозів та роботи програмного продукту:

- оцінка точності прогнозу за показниками MAE, RMSE, MAPE;
- тестування продуктивності та стабільності моделі;
- тестування функціоналу браузерного розширення (коректність даних, швидкість відображення, UX/UI).

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Історія розвитку ринку ціних паперів.

Перші передумови появи ринку цінних паперів беруть свій початок ще в Європі наприкінці XVI – на початку XVII століття. У цей період із активізацією міжнародної торгівлі та зростанням потреб держав у фінансуванні виникають перші організовані місця для обміну фінансовими документами. Поширення боргових інструментів, таких як векселі та облігації, сприяло формуванню стабільної інфраструктури для купівлі-продажу цінних паперів, що згодом стало основою сучасного біржового механізму.

У подальшому фондові ринки активно розвивалися в багатьох країнах світу, зокрема у Великій Британії, Нідерландах та США. Саме в цих державах були сформовані відомі фондові біржі, серед яких Нью-Йоркська фондова біржа (NYSE) та Лондонська фондова біржа (LSE), які з часом перетворилися на глобальні центри міжнародної торгівлі цінними паперами – джерело [1].

З розвитком телекомунікацій та комп’ютерних технологій у другій половині XX століття фондові ринки стали ще більш доступними, що сприяло зростанню обсягів операцій та глобалізації інвестиційних процесів.

На території України становлення біржової торгівлі розпочалося значно пізніше. Уже в XIX – на початку XX століття у великих містах — Києві, Одесі, Харкові та Львові — почали діяти товарно-фондові біржі, на яких проводилися операції з акціями місцевих комерційних структур та муніципальними позиками. Проте в 1920–1930-х роках, у зв’язку із запровадженням командно-адміністративної економічної моделі, діяльність фондових бірж була повністю припинена, що фактично поставило розвиток українського ринку цінних паперів на тривалу паузу- джерело [2].

Сучасним етапом еволюції ринку ціних паперів стала поява криптовалютних бірж, що функціонують як цифрове продовження фондових майданчиків. На цих платформах замість класичних фінансових інструментів — акцій, облігацій чи деривативів — предметом торгівлі є криптовалюти та інші види цифрових активів. Такі біржі створили можливість залучення капіталу в

новому, децентралізованому форматі та відкрили доступ до інноваційних інвестиційних продуктів, серед яких токенізовані активи, цифрові валюти, NFT-інструменти та активи, прив'язані до реальних товарів чи цінних паперів. Розвиток криптобірж суттєво змінив структуру світових фінансових потоків, зробивши ринок більш демократичним та доступним для широкого кола учасників, оскільки для інвестування більше не потрібні посередники у вигляді банків чи традиційних брокерів. Сучасні дослідження в Україні підтверджують тенденцію поступового злиття традиційного фондового ринку з ринком цифрових активів, що формує новий тип фінансової екосистеми, у якій блокчейн-технології доповнюють класичні механізми торгівлі та інвестування.

1.2 Класичні фондові біржі.

Класичні фондові ринки є ключовим елементом сучасної фінансової системи, забезпечуючи обіг цінних паперів, залучення інвестицій та формування вартості компаній. Саме через біржові механізми бізнес отримує капітал для розвитку, а інвестори — можливість отримувати прибуток, володіючи корпоративними правами або фінансовими активами.

На світовій арені діють низка біржових центрів, які визначають тенденції глобального інвестування. До них належать:

- New York Stock Exchange (NYSE) — найбільший у світі біржовий майданчик за капіталізацією, що перевищує 25 трильйонів доларів. Тут представлені провідні корпорації, серед яких Apple, Coca-Cola, Boeing. На NYSE формуються важливі індикатори ринку, зокрема Dow Jones, S&P 500 і NYSE Composite. Діяльність біржі ґрунтується на максимальній прозорості, що дозволяє відстежувати дії учасників торгів у режимі реального часу та зменшує імовірність спекуляцій.
- NASDAQ — друга за масштабом біржа США, відома концентрацією високотехнологічних компаній, як-от Google, Microsoft, Tesla, Meta. На відміну від NYSE, NASDAQ з початку свого існування працює

виключно в електронному форматі та використовує дилерську модель, у якій ціни формуються маркет-мейкерами. Біржа орієнтована на інноваційні компанії та стартапи, що згодом можуть стати ринковими лідерами.

- Tokyo Stock Exchange (TSE) — головна біржа Азії, на якій торгуються акції промислових і технологічних гігантів — Toyota, Sony, Honda. London Stock
- Exchange (LSE) — один із найстаріших світових торговельних майданчиків, що виконує роль європейського фінансового центру та об’єднує бізнес із різних країн.
- Shanghai Stock Exchange (SSE) та Shenzhen Stock Exchange (SZSE) — провідні китайські біржі, які відображають швидке зростання економіки Китаю та розвиток інноваційної індустрії.
- Euronext — об’єднана європейська платформа, що функціонує одразу в кількох країнах: Франції, Нідерландах, Португалії, Бельгії та Італії. Завдяки цифровізації та технологічному прогресу доступ до біржової торгівлі став значно простішим.
- Онлайн-платформи, такі як eToro, Robinhood, Interactive Brokers, Plus500, Degiro, надали можливість інвестувати навіть користувачам із невеликим стартовим капіталом, пропонуючи навчальні матеріали, мобільні застосунки та копіювання стратегій досвідчених трейдерів.

Класичний ринок приваблює інвесторів значною різноманітністю інструментів: акцій, облігацій, деривативів, ETF, індексних фондів, REIT. Акції дозволяють отримувати дивіденди та брати участь у розвитку компанії, облігації забезпечують більш стабільний дохід, а ETF дають змогу інвестувати в цілі галузі або індекси з низькими витратами.

Попри численні переваги — потенціал зростання, пасивний дохід, ліквідність та прозорість — ринок залишається ризиковим. Він чутливий до економічних подій, вимагає фінансової грамотності та правильного підходу до оцінки активів.

1.3 Особливості криптовалютних бірж.

Криптовалютні біржі є відносно новим елементом фінансової інфраструктури, який сформувався на перетині технологій блокчейну та ринкових механізмів торгівлі активами. Вони забезпечують купівлю, продаж та обмін цифрових активів, виконуючи роль посередника між учасниками ринку так само, як і класичні фондові біржі. Водночас криптобіржі мають унікальні особливості, що суттєво відрізняють їх від традиційних фінансових торговельних майданчиків.

Насамперед, криптовалютні біржі функціонують на основі децентралізованих технологій, що дозволяє здійснювати операції з активами без участі державних регуляторів чи банківських структур.

На криптовалютних біржах торгується широкий спектр цифрових активів, кожен з яких має власні функції та механізми забезпечення таблиця 1.1:

Таблиця 1.1 Функції та механізми забезпечення цифрових активів

Тип активу	**Приклади**	**Що забезпечує цінність**
Криптовалюти загального призначення	Bitcoin, Litecoin	Дефіцитність, децентралізована емісія, попит інвесторів; BTC обмежений 21 млн монет
Платформні токени	Ethereum, Solana, Cardano	Використання як «палива» для виконання смарт-контрактів та транзакцій у мережі
Стейблкоїни	USDT, USDC, DAI	Мають прив’язку до долара, золота або кошика активів; забезпечені резервами або алгоритмічно
DeFi-токени	UNI, AAVE, COMP	Використовуються у децентралізованих фінансових протоколах; дозволяють брати кредити, надавати ліквідність

NFT та токенізовані цифрові об’єкти	Колекційні NFT, токени ігор	Цінність визначається унікальністю, рідкістю чи корисністю в цифрових сервісах
Токенізовані реальні активи (RWA)	золото (PAXG), акції, нерухомість	Цифрові копії реальних активів, що мають юридичне підтвердження забезпечення

Такий спектр активів перетворює криптовалютні біржі з майданчиків торгівлі монетами на універсальні фінансові екосистеми, що дублюють та доповнюють класичні ринки.

На ринку існують десятки великих бірж, проте кілька з них задають стандарти функціональності, регуляції та інновацій.

- Binance — найбільша криптовалютна біржа у світі за обсягами торгів. Вона підтримує сотні криптовалют та похідних інструментів, надає стейкінг, ф’ючерсні контракти, кредитування, автоматизовану торгівлю. Binance також впровадила власну блокчейн-екосистему (BNB Chain), має токен BNB, використовується як інструмент комісій і технічна валюта екосистеми. Біржа вирізняється високою ліквідністю, що дозволяє виконувати великі угоди з мінімальними ринковими втратами.
- Coinbase — найбільш регульована біржа США, орієнтована на легальний та безпечний обіг цифрових активів. Компанія стала першою криптобіржею, що вийшла на фондовий ринок NASDAQ (2021 р.), затвердивши свій статус як легального фінансового оператора. Coinbase має спрощений інтерфейс, систему зберігання депозитів у холодних гаманцях, що забезпечує високий рівень захисту, і тісно співпрацює з урядовими органами США та ЄС.
- Kraken — одна з найстаріших криптобірж, відома жорсткими стандартами безпеки та довготривалим аудиторським контролем. Kraken пропонує маржинальну торгівлю, майнінг-пул, ф’ючерси та

індексні фонди криптовалют. Її інфраструктура орієнтована на професійних трейдерів та інституційних інвесторів.

- Децентралізовані біржі (DEX) такі як Uniswap, PancakeSwap, SushiSwap функціонують без централізованого управління. Торгівля відбувається шляхом взаємодії зі смарт-контрактами. Ліквідність формується користувачами через механізм пулів ліквідності. DEX-моделі виключають посередників, але потребують більших технічних знань та несуть ризики смарт-контрактів.

Системні переваги та недоліки криптовалютних бірж.

Ключові переваги:

- глобальний доступ без географічних обмежень;
- 24/7 торгівля без вихідних;
- можливість інвестувати без участі банків;
- швидкі транзакції та низькі комісії;
- широкий спектр активів та фінансових інструментів.

Суттєві ризики:

- недостатня регуляція в окремих країнах;
- ризик зламів і втрати коштів на централізованих серверах;
- висока волатильність активів;
- можливі маніпуляції цінами на низьколіквідних ринках;
- помилки чи вразливості у смарт-контрактах на DEX.

Таким чином, криптовалютні біржі є не лише альтернативою класичним фондовим ринкам, а й новим етапом їх еволюції, розширюючи перелік доступних інвестиційних інструментів та технологій. Їх розвиток сприяє поширенню цифрової економіки, а високий рівень автоматизації створює умови для застосування алгоритмічних систем прогнозування, зокрема моделей штучних нейронних мереж.

1.4 Опис криптобіржи Binance.

У рамках виконання цієї роботи для подальшого аналізу та проведення практичних експериментів було обрано криптовалютну біржу Binance. Це рішення пояснюється тим, що Binance утримує статус найбільшої та однієї з ключових бірж цифрових активів у світі, демонструючи найвищі показники за добовими обсягами торгів, кількістю користувачів та різноманіттям доступних інструментів. Крім значної популярності, платформа вирізняється продуманим та інтуїтивно зрозумілим інтерфейсом, що забезпечує комфортну роботу як новачкам, так і трейдерам з досвідом. В межах цієї роботи особливо корисною є можливість зручного експорту історичних даних у форматі CSV, що дозволяє отримувати точні ринкові вибірки за потрібними часовими інтервалами та використовувати їх для подальшого аналізу й побудови моделей прогнозування. Важливою перевагою Binance є наявність кількох типів ринків, серед яких основними виступають спотовий та ф'ючерсний.

Спотовий ринок рис. 1.1 передбачає купівлю та продаж криптовалют за поточною ринковою ціною з фактичним отриманням активу на баланс.

Ф'ючерсний ринок рис. 1.2, своєю чергою, дозволяє укладати контракти на зміну вартості активів у майбутньому, що відкриває можливості для хеджування, використання кредитного плеча та торгівлі як на зростанні, так і на падінні ціни.

Таке різноманіття торгових механізмів робить Binance універсальним інструментом для комплексного вивчення поведінки ринку, оцінки ризиків та побудови моделей прогнозування.

Рис. 1.1 Вигляд вікна спотового ринку BTC/USDT

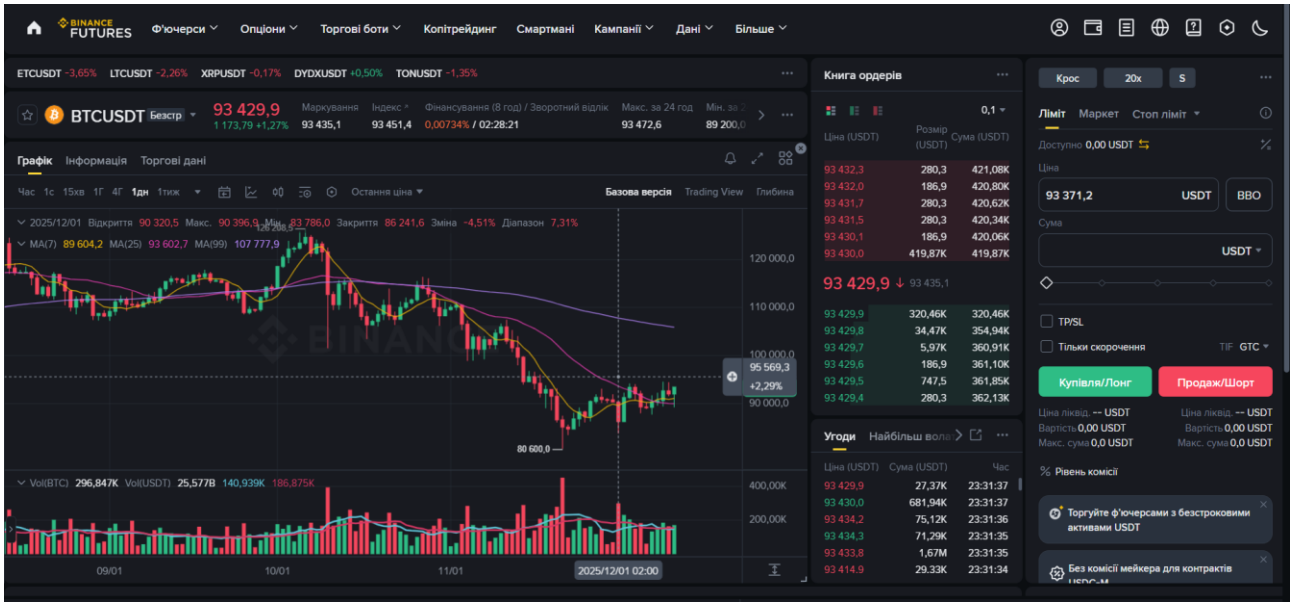

Рис. 1.2 Вигляд вікна фючерсного ринку BTC/USDT

Як видно з рисунків, графіки цін різних типів практично не відрізняються за основними тенденціями, відмінсть поляє лише у візуальному розташувані інформації про обсяги торгів та кнопок купівлі/ продажу.

Для детального аналізу обрано найбільш інформативний формат — графік японських свічок. Цей тип графіка складається з окремих сегментів — свічок, кожна з яких відображає рух ціни за певний часовий проміжок. Інтервали свічок можна гнучко налаштовувати залежно від потреб трейдера: від 15 хвилин і 1 години до 4 годин, 1 дня або 1 тижня (рис. 1.3). Японські свічки дозволяють не лише відстежувати відкриття, закриття, максимуми та мінімуми ціни за обраний період, але й швидко виявляти ринкові патерни, оцінювати силу тренду та зміну

настроїв учасників ринку. Завдяки такій структурі графіка трейдер отримує наочний та детальний інструмент для аналізу коротко- та довгострокових рухів ціни.

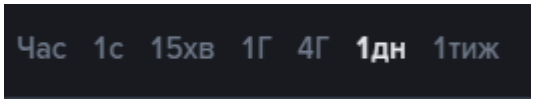

Рис. 1.3 Панель вибору часового проміжку

Дана свічка має певну будову рис. 1.4, а саме тіло свічки та її тінь, тіло показує ціну в момент відкриття свічки та ціну в момент закриття, так званна тінь вказує на мінімуми та максимуми ціни за певний обраний проміжок часу. В свою чергу колір свіки вказує на різницю між ціною відкриття та ціною закриття, якщо ціна на початку часовго проміжку є нижчою ніж на кінці дана свічка буде зеленого кольору, і навпаки якщо ціна на початок певного проміжку є більшою ніж при закінені свічка буде червноною.

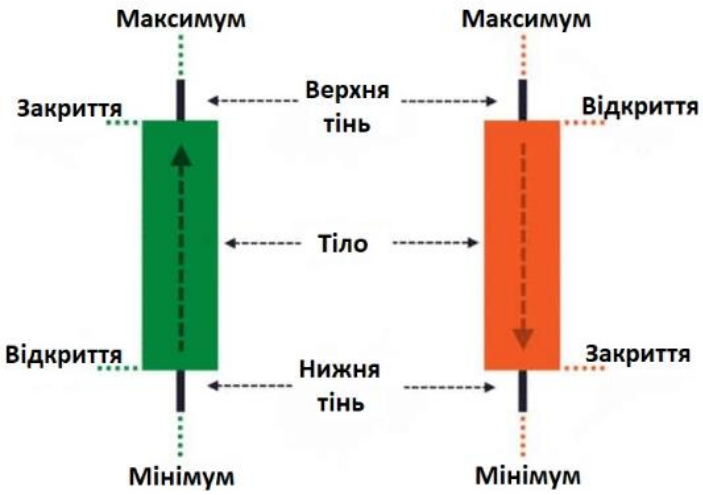

Рис. 1.4 Будова свічки

Представлений графік рис. 1.3 є одним із ключових інструментів трейдера, оскільки містить максимальний обсяг корисної ринкової інформації. Він дозволяє своєчасно помічати характерні патерни руху ціни, відстежувати поведінку покупців і продавців щодо конкретного цифрового активу та оцінювати рівень ринкового інтересу чи ажіотажу.

Аналізуючи динаміку графіка з огляду на попередні періоди, трейдер отримує можливість виявляти глибші закономірності, прогнозувати потенційні

коливання та визначати ключові зони підтримки й опору - області, в яких ціна традиційно сповільнює рух або повертається у протилежний бік.

Такі рівні відіграють важливу роль у розумінні поведінки учасників ринку. Наприклад, якщо ціна опускається до значної зони підтримки, покупці зазвичай проявляють підвищену активність, що зменшує пропозицію на ринку та штовхає вартість активу вгору.

У випадку стрімкого підвищення ціни ситуація може бути протилежною: при наближенні до сильної зони опору обсяги продажів зростають, оскільки багато трейдерів фіксують прибуток, і ціна не здатна пройти цей рівень через нестачу покупців. У результаті формується відкат або розворот.

Детальне дослідження графіку в декількох часових горизонтах дозволяє глибше зрозуміти структуру ринкових рухів, оцінити баланс попиту й пропозиції та робить процес прогнозування більш обґрунтованим. Якщо бажаєте, можу додати коротку довідку про індикатори, які допомагають аналізувати такі рівні.

Для трейдерів, які працюють у високому темпі та здійснюють операції в коротких часових інтервалах, надзвичайно важливо розуміти структуру та динаміку торгових обсягів (рис. 1.5). Саме ці дані відображають поточні настрої ринку, показуючи, наскільки активно діють покупці та продавці у певний момент часу.

Обсяги допомагають трейдерам оцінити силу руху ціни: чи підкріплене зростання реальним попитом, чи зниження відбувається на тлі масового розпродажу.

Рис. 1.5 Обсяг торгів в моменті BTC/USDT

Крім цього, на платформі Binance обсяги торгів візуалізуються у вигляді вертикальних стовпців під кожною свічкою (рис. 1.6), що значно спрощує аналіз піків активності. Таке графічне відображення дозволяє швидко виявляти моменти різкого збільшення інтересу до активу, визначати фази накопичення або розподілу, а також розуміти, коли ціна рухається «на об’ємі», а коли - без достатньої підтримки. Подібні спостереження допомагають трейдеру приймати більш обґрунтовані рішення, особливо під час торгівлі всередині дня чи scalp-операцій.

Рис. 1.6 Історія обсягів торгів BTC/USDT

У межах цієї наукової роботи передбачається створення браузерного розширення, основна мета якого - підвищити ефективність роботи трейдерів, що здійснюють операції в коротких часових інтервалах. Такі користувачі працюють у високодинамічному середовищі, де рішення необхідно приймати швидко та на основі максимально точних сигналів.

Розроблюване розширення інтегруватиме результати роботи нейронної мережі, яка на основі аналізу історичних ринкових даних здатна оперативно

виявляти повторювані патерни та прогнозувати поведінку ціни на наступний часовий проміжок, тобто формування наступної свічки.

Це забезпечить трейдерів інструментом, що допоможе своєчасно реагувати на зміни ринку, зменшити частку суб'єктивності в прийнятті рішень і підвищити точність короткострокових торгових стратегій.

1.5 Висновок до розділу.

Ринок цінних паперів пройшов довгий еволюційний шлях, поступово адаптуючись до розвитку технологій та змін у глобальній економіці. Його формування відбувалося через створення традиційних бірж, механізмів торгівлі, інституцій інвесторів та регуляторних органів, що забезпечували безпеку операцій і прозорість фінансової діяльності.

Розвиток цифрової економіки став потужним каталізатором для появи нових фінансових інструментів, які розширили поняття активів та підходи до інвестування. Особливо важливу роль у цьому процесі відіграли криптовалюти та платформи для їх обігу — криптовалютні біржі, що фактично стали новим видом інфраструктури ринку капіталу. Криптовалютні біржі логічно продовжили розвиток ринку цінних паперів, запропонувавши децентралізовані інструменти обміну та інвестування. Вони сформували простір, у якому цифрові активи, такі як Bitcoin, Ethereum, Solana та інші токени, набули статусу інвестиційних і платіжних активів.

Цінністю нових видів активів можуть виступати алгоритмічні механізми, попит користувачів, реальні резерви (як у випадку зі стейблкоінами USDT, USDC), або ж внутрішні економічні моделі блокчейн-екосистем. Саме можливість оперувати активами, що не мають фізичної форми, але підтверджують цінність технологією, зробила криптовалюти новим етапом у фінансовій еволюції.

Найбільші криптобіржі світу стали ключовими елементами цієї системи, забезпечуючи ліквідність, зберігання активів, аналітичні інструменти та

механізми торгівлі. Такі платформи, як Binance, Coinbase, Kraken, Bybit, OKX, перетворилися на глобальні фінансові центри цифрової економіки. Вони не лише об’єднують мільйони трейдерів та інституційних інвесторів, а й розробляють власні інноваційні продукти, включаючи NFT-майданчики, децентралізовані сервіси кредитування, деривативні ринки та навчальні програми.

Більшість провідних бірж активно взаємодіють з регуляторами, що свідчить про поступову інтеграцію крипторинків у формальну фінансову систему світу. На сьогоднішній день, криптовалютні біржі не лише змінили уявлення про торгівлю активами, але й сформували сучасний тренд цифрових інвестицій, що розширює можливості як приватних користувачів, так і міжнародних фінансових установ. Вони стали невід’ємним продовженням еволюції фондового ринку, надаючи нові інструменти, нові правила та новий рівень фінансової свободи.

Однак разом із тим зростає й потреба у вдосконаленні правового регулювання, посиленні безпеки та розробці глобальних стандартів, які забезпечать стабільний та справедливий розвиток цього сектора.

Криптовалютні біржі можна вважати не тимчасовою тенденцією, а потужною складовою сучасного світового фінансового ринку, який формує майбутнє цифрової економіки.

РОЗДІЛ 2. ЗАСТОСУВАННЯ НЕЙРОНИХ МЕРЕЖ ДЛЯ АНАЛІЗУ РИНКУ

2.1 Аналіз наявних бібліотек для обробки даних.

Розвиток фінансових технологій та активне використання криптовалютних бірж у наукових та практичних задачах привели до зростання потреби у точних методах обробки великих обсягів даних. Прогнозування вартості криптовалют є складним процесом, що потребує роботи зі структурованими та неструктурованими часовими рядами, числовими індикаторами, історичними даними торгів та ринковими метриками. Для виконання таких задач особливого значення набувають програмні інструменти, здатні забезпечити ефективне збирання, очищення, аналіз та підготовку даних до моделювання нейронними мережами.

Сучасні мови програмування, зокрема Python пропонують широкий спектр бібліотек, призначених для роботи з економічними та фінансовими даними.

Серед найпоширеніших бібліотек для аналізу й обробки даних на Python варто виділити Pandas, що забезпечує зручні інструменти для опрацювання таблиць і часових рядів, NumPy для високошвидкісних математичних операцій, а також Matplotlib і Seaborn, які використовуються для побудови графіків та візуалізації ринкової динаміки.

Для фінансової аналітики особливого значення набувають бібліотеки TA-Lib та Pandas-TA, які дозволяють автоматично будувати індикатори технічного аналізу (RSI, MACD, CCI, EMA тощо) та виявляти торговельні сигнали.

Для нормалізації початкових даних добре підходить бібліотека Sklearn, яка виконує важливу роль у підготовці фінансових даних до моделювання. Вона дозволяє застосовувати такі методи, як MinMaxScaler, StandardScaler, RobustScaler, що зменшують вплив різких ринкових стрибків і підвищують стабільність навчання моделей. Завдяки цим інструментам дані набувають стандартизованого вигляду, що є критично важливим для коректної роботи нейронних мереж та алгоритмів машинного навчання.

Для побудови моделей на основі нейронних мереж активно використовують бібліотеку Keras, що дозволяє створювати складні архітектури буквально в один рядок коду. Завдяки своїй простоті та гнучкості вона ідеально підходить для задач прогнозування часових рядів, зокрема цін криптовалют.

Окрему увагу слід приділити також бібліотеці TensorFlow.keras.optimizers, яка містить повний набір сучасних оптимізаторів (Adam, Nadam, AdamW, RMSprop, Adagrad, Radam, SWATS та інші), що легко підключаються до моделей та дають змогу підвищити точність навчання, мінімізувати втрати й адаптувати нейромережу до динамічних ринкових умов.

Сучасні технології аналізу даних та машинного навчання відкрили можливість глибоко досліджувати фінансові ринки та прогнозувати вартість криптоактивів. У роботі з часовими рядами особливо ефективними вважаються архітектури, здатні враховувати послідовність значень та їхній взаємозв’язок у часі.

До таких моделей належать рекурентні нейронні мережі (RNN) та їх удосконалені модифікації — LSTM (Long Short-Term Memory) і GRU (Gated Recurrent Unit). Їхня здатність запам’ятовувати важливі зміни в історії даних робить ці алгоритми провідним інструментом для прогнозування нестабільних криптовалютних ринків.

Рекурентні нейронні мережі (RNN) RNN — це ключовий тип моделей, здатний аналізувати дані, у яких важлива черговість елементів: фінансові котирування, новинні потоки, послідовності транзакцій тощо. На відміну від звичайних повнозв’язних мереж, RNN мають механізм внутрішньої пам’яті (hidden state) рис 2.1, який дозволяє зберігати інформацію з попередніх кроків та використовувати її при обчисленні наступних. Такий принцип роботи забезпечує поетапний аналіз часової послідовності.

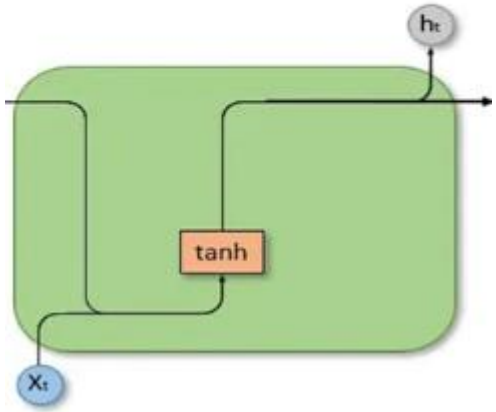

Рис. 2.1 Архітектура RNN

Переваги RNN:

- Дозволяють відслідковувати динаміку змін фінансових показників з урахуванням минулих значень.
- Можуть обробляти послідовності довільної довжини.
- Використовуються не лише в прогнозуванні, а й у текстовому аналізі новин чи соціальних мереж, що може впливати на ринкові настрої.

Недоліки RNN:

- Схильні до зникнення або вибуху градієнтів при аналізі довгих послідовностей.
- Погано працюють із тривалими залежностями, не здатні утримувати важливі дані на великих проміжках часу.
- Вимагають значних обчислювальних ресурсів у разі великих часових рядів.

LSTM — покращені рекурентні моделі LSTM були створені як відповідь на нестабільність класичних RNN і дозволяють утримувати інформацію довше завдяки складній системі керування пам’яттю Рис. 2.1 . Їхня архітектура містить три ключові блоки: forget gate, input gate та output gate, які вирішують, що варто забути, запам’ятати або використати як результат.

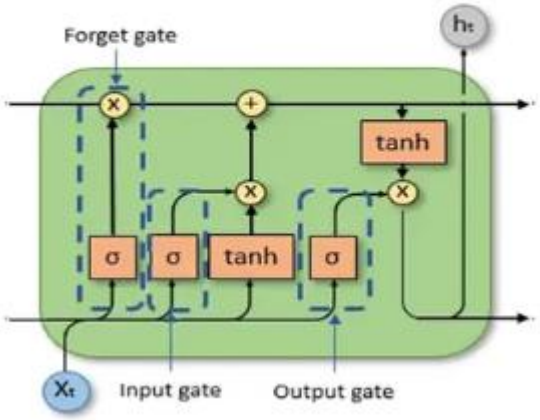

Рис. 2.2 Архітектура LSTM

Така структура забезпечує здатність моделі фіксувати як короткострокові, так і довгострокові закономірності.

LSTM особливо корисні для:

- Виявлення трендів та циклічності ринку.
- Аналізу багатофакторних даних (ціна, обсяг, індикатори, волатильність).
- Прогнозування не лише цін, а й ймовірності певних рухів на ринку.

GRU — легка й швидка альтернатива LSTM GRU повторюють функції LSTM, але мають компактнішу структуру рис. 2.3, що робить їх менш вимогливими до ресурсів. Вони містять лише два основних елементи: update gate та reset gate, які визначають, яку частину даних зберігати, а яку скидати.

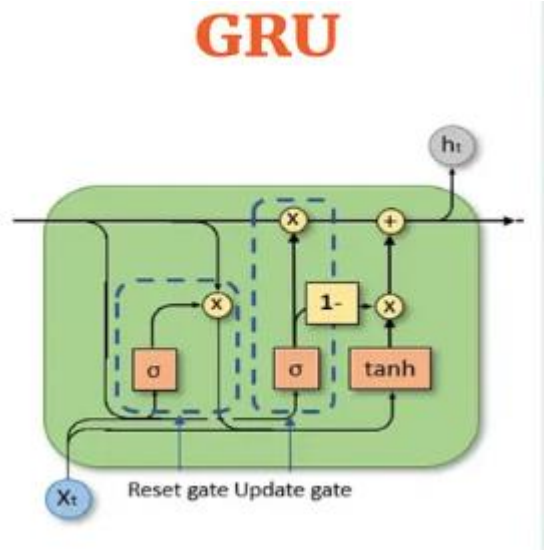

Рис. 2.3 Архітектура GPU

Переваги GRU:

- Менше параметрів — швидше тренування.
- Потребують менше пам’яті, придатні для мобільних або реальних торгових систем.
- Часто показують точність, подібну до LSTM, особливо при невеликій кількості даних.

Недоліки GRU:

- Не мають окремого стану пам’яті, як LSTM — менш гнучкі.
- Можуть бути нестійкими при аналізі дуже довгих залежностей.
- Не завжди перевершують LSTM у складних фінансових моделях.

CNN — згорткові мережі для часових рядів. Хоча згорткові мережі зазвичай асоціюються з обробкою зображень, CNN довели свою ефективність у роботі з криптовалютними та фінансовими часовими рядами. Саме завдяки своїй структурі рис. 2.4 вони аналізують локальні закономірності (патерни) в даних, наприклад, короткі трендові рухи або пікові зміни.

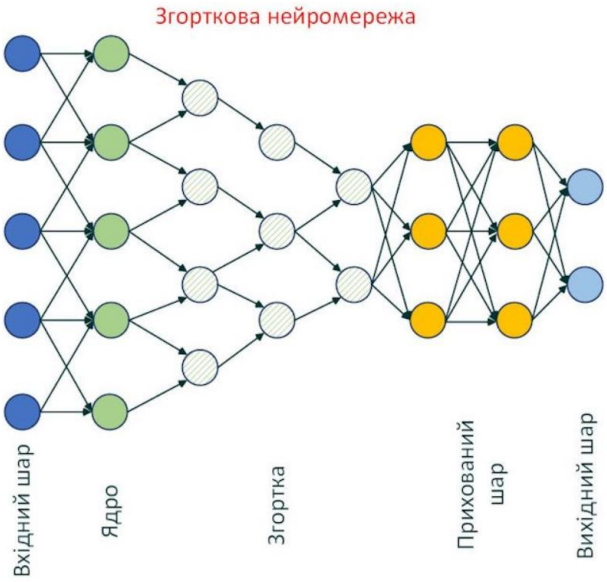

Рис. 2.4 Архітектура CNN

Переваги CNN:

- Значно швидше за LSTM/GRU.
- Добре виявляють локальні тренди та фрактальні закономірності ринку.
- Ефективно працюють у поєднанні з RNN/LSTM/GRU як модулі попередньої обробки.

Недоліки Conv1D:

- Не здатні фіксувати довгострокову пам’ять без інших шарів.
- Потребують правильно підібраної довжини згортки (kernel size).
- Самостійно можуть пропустити залежності, що тривають тривалий час.

Таким чином, для прогнозування ціни криптовалют доцільно комбінувати можливості RNN, LSTM, GRU або CNN залежно від глибини аналізу, обсягу даних і доступних обчислювальних ресурсів. У дослідженні [3] показано, що в задачах валютного прогнозування глибинні мережі LSTM та GRU здатні перевершувати традиційні методи, як-от Random Forest та VAR, що підкреслює їхню важливу роль у сучасних фінансових дослідженнях.

При створенні нейронних мереж для аналізу й прогнозування цін криптовалют важливе значення має не лише тип архітектури (RNN, LSTM, GRU чи Conv1D), а й алгоритм, який відповідає за навчання мережі. Такий алгоритм називають оптимізатором.

Його головна задача – правильно та ефективно оновлювати ваги моделі під час тренування, мінімізуючи похибку прогнозу. Іншими словами, оптимізатор визначає шлях, швидкість та якість навчання нейронної мережі. У фінансових задачах тренування є особливо складним, адже ринок криптовалют відзначається високою волатильністю, шумністю даних та нелінійністю трендів.

Вибір оптимізатора безпосередньо впливає на те:

- як швидко нейромережа навчиться;
- чи зможе вона уникнути переобучення;
- наскільки точним буде прогноз;
- чи зможе модель адаптуватися до різких цінових коливань.

Саме тому правильний підбір оптимізатора є ключовим етапом під час проєктування моделі машинного навчання.

Основні оптимізатори для прогнозування вартості криптовалют:

SGD – класичний метод стохастичного градієнтного спуску, який оновлює ваги моделі поступово, аналізуючи невеликі частини даних.

Переваги:

- простота та мінімальна витрата ресурсів;
- працює стабільно на малих моделях;
- корисний при великому шумі даних.

Недоліки:

- повільне зближення до оптимального розв’язку;
- може «застрягти» в локальному мінімумі;
- не підходить для складних архітектур (LSTM, GRU) у чистому вигляді.

Momentum SGD. Ця модифікація SGD додає «момент» — інерцію руху в напрямку мінімуму.

Переваги:

- прискорює зближення до мінімуму;
- рідше застряє в локальних мінімумів;
- краще працює з нерівними поверхнями втрат, характерними для прогнозів цін.

Недоліки:

- потребує ретельно підібраного коефіцієнта моменту;
- при неправильних параметрах може «перестрибувати» мінімум.

Adam (Adaptive Moment Estimation). Adam є одним із найпопулярніших оптимізаторів для задач з ринковими даними, оскільки поєднує переваги Momentum та адаптивної зміни швидкості навчання.

Переваги:

- адаптивна швидкість навчання для кожного параметра;
- швидке зближення на нестабільних даних крипторинку;
- чудово підходить для LSTM, GRU.

Недоліки:

- легко перенавчається;
- не завжди досягає найкращого глобального мінімуму.

Nadam (Nesterov-accelerated Adam). Розширення Adam з використанням поправок Nesterov Momentum.

Переваги:

- більш точне оновлення ваг;
- швидше зближення, ніж Adam;
- краще передбачає різкі стрибки на ринках.

Недоліки:

- може бути нестабільним при поганій підготовці даних;
- параметри складніше налаштовувати.

AdamW (Adam with Weight Decay). У цьому оптимізаторі передбачено додаткове «розкладання ваг», що зменшує переобучення.

Переваги:

- набагато краща генералізація (менше «запам’ятовує» шум);
- ідеальний для волатильних криптовалютних даних;
- покращує точність довгострокових прогнозів LSTM/GRU.

Недоліки:

- трохи повільніший за Adam;
- потребує правильно підібраного коефіцієнта weight decay.

Radam (Rectified Adam). Розв’язує проблему нестабільного початку навчання в Adam.

Переваги:

- стабільний старт моделі;
- точніший рух до мінімуму, менше «скачків»;
- краща робота на складних архітектурах.

Недоліки:

- трохи повільніший від Adam у простих завданнях.

Adagrad / RMSProp. Застосовуються для адаптивної зміни швидкості навчання.

Adagrad:

- добре працює для рідких ознак (об’єми, новинні дані), але дуже швидко «сповільнюється» — ускладнює прогноз.

RMSProp:

- краще працює у фінансових часових рядах;
- стабільний на LSTM/GRU;
- потребує обережного налаштування.

2.2 Підготовка даних для нейроної мережі

У цьому розділі розглядається етап підготовки даних, який є фундаментальним під час побудови ефективної моделі нейронної мережі. Якість

вибірки, її обсяг та статистичні характеристики безпосередньо впливають на точність і стабільність прогнозів, особливо при роботі з часовими рядами вартості криптовалют, що характеризуються високою волатильністю та наявністю шумів.

Коректна попередня обробка не лише покращує здатність моделі виявляти закономірності, а й запобігає появі некоректних градієнтів, нестабільному навчанню або недостатній узагальнювальній здатності.

Нижче буде детальніше представлено кілька способів нормалізації даних, кожен з яких по-своєму впливає на стабільність і якість навчання моделі. Кожен метод має власні сильні сторони та недоліки, тому вибір конкретного підходу повинен відповідати структурі вихідного набору та особливостям задачі прогнозування.

Проведений аналіз формує надійну основу для визначення оптимальної стратегії підготовки вибірки під час проєктування нейронної мережі для передбачення цінових коливань.

Окрім цього, для підвищення ефективності навчання планується розширити базовий набір даних, отриманий із Binance, шляхом додавання низки технічних індикаторів. Це дозволить моделі краще вловлювати приховані закономірності та підвищити точність прогнозу.

Під час підготовки вибірки для навчання нейронної мережі важливим етапом є нормалізація даних. Оскільки значення різних ознак можуть мати різні масштаби, це впливає на швидкість та стабільність збіжності моделі, а також на якість прогнозу. Існує кілька поширених методів масштабування, кожен з яких по-різному трансформує дані та підходить для різних типів задач.

MinMaxScaler Найпопулярніший метод нормалізації, який перетворює значення у діапазон [0, 1] або інший заданий інтервал.

Переваги:

- зберігає форму розподілу даних, що важливо для часових рядів; підходить для моделей, які чутливі до масштабу (LSTM, GRU, CNN);
- забезпечує стабільні градієнти та прискорює навчання;

- легко інтерпретується та широко підтримується бібліотеками Python.

Недоліки:

- дуже чутливий до викидів (outliers): один аномальний пік може суттєво змістити масштабування;
- при надходженні нових значень за межами початкового діапазону дані можуть "виходити" за межі [0, 1];
- інколи втрачає інформацію про абсолютні масштаби змін ціни.

Через свою простоту і ефективність MinMaxScaler найчастіше застосовується для нормалізації криптовалютних часових рядів, але потребує попереднього очищення даних від аномальних значень.

StandardScaler (z-нормалізація) Цей метод масштабує дані так, щоб їх середнє дорівнювало нулю, а стандартне відхилення - одиниці.

Переваги:

- менш чутливий до викидів, ніж MinMaxScaler;
- добре працює, коли дані мають приблизно нормальний розподіл;
- зберігає відносні відхилення та варіації.

Недоліки:

- не гарантує обмежений діапазон значень;
- підвищені або експоненційні значення можуть впливати на масштабування;
- інколи ускладнює стабільний градієнт у рекурентних нейромережах.

RobustScaler Масштабує дані, використовуючи медіану та інтерквартильний розмах (IQR). Це робить метод стійким до аномальних значень.

Переваги:

- практично нечутливий до викидів; підходить для “шумних” фінансових даних;
- зберігає структуру змін без впливу поодиноких піків.

Недоліки:

- іноді зменшує контрастність даних;
- може гірше працювати для глибоких моделей, що очікують компактного діапазону значень.

MaxAbsScaler Масштабує дані в діапазон [-1; 1], ділячи на максимальне абсолютне значення.

Переваги:

- простий та ефективний для даних без сильних аномалій;
- добре працює з симетричними рядам, де є як позитивні, так і негативні значення; не змінює форму розподілу.

Недоліки:

- чутливий до пікових значень;
- рідше застосовується для крипторинків, де домінують невід'ємні показники.

Логарифмічна нормалізація Використовується для даних, що мають експоненційне зростання або великі розриви.

Переваги:

- згладжує різкі коливання та великі стрибки ціни;
- добре підходить для індикаторів із сильними змінами.

Недоліки:

- непридатний, якщо дані містять нульові або від'ємні значення;
- не завжди зручно інтерпретувати.

Серед усіх методів нормалізації для своєї роботи я обрав саме MinMaxScaler, він є найбільш поширеним для криптовалютних часових рядів, оскільки забезпечує компактний інтервал значень та покращує стабільність роботи LSTM-та CNN-моделей.

Єдиним істотним недоліком використання MinMaxScaler є його висока чутливість до викидів та різких пікових значень, які можуть спотворювати

масштабування та впливати на якість навчання моделі. Щоб мінімізувати цей ефект, у своїй роботі передбачається застосування кількох окремих нормалізаторів MinMaxScaler - кожен з них працюватиме лише з одним конкретним стовпцем даних.

Такий підхід є важливим, оскільки показники, пов’язані з обсягами торгів, мають зовсім інші діапазони та динаміку порівняно з цінами криптовалют, що може додавати небажаний шум у вибірку та негативно впливати на навчання моделі.

У побудованій нейронній мережі для кожного типу даних буде створено окремий масштабувальний об’єкт: індивідуальні нормалізатори для ціни відкриття, мінімальної та максимальної ціни, ціни закриття, окремий нормалізатор для стовпця обсягів торгів, а також окремі нормалізатори для кожного технічного індикатора.

Така модульна схема нормалізації дозволяє зберегти структуру кожної ознаки та забезпечує більш стабільні умови для навчання нейромережі. Додатково це підвищує точність зворотного перетворення, що є критично важливим для подальшої інтерпретації прогнозованих значень у вихідному масштабі.

Ще одним ключовим кроком у процесі підготовки даних є розширення базового датасету за рахунок обчислення додаткових технічних індикаторів:

Одним із ключових інструментів згладжування та аналізу трендів є експоненційні рухомі середні (EMA). На відміну від простої середньої, EMA приділяє більшу вагу останнім значенням, що дає змогу швидше реагувати на зміни ціни. Кожне нове значення EMA обчислюється шляхом комбінування попереднього значення середньої та поточної ціни закриття із використанням коефіцієнта згладжування. У моделі застосовуються дві короткі EMA - із періодом 5 та 10, які допомагають визначити локальні зміни напрямку руху ціни.

Формула EMA:

$p =$ ціна закриття; $\qquad EMA_t = EMA_{t-1} + \alpha(p_t - EMA_{t-1})$

$N -$ період середньої (5 чи 10 у даному випадку) $\qquad \alpha = \dfrac{2}{N+1}$

Поряд із цим обчислюється проста рухома середня (SMA), яка відображає усереднене значення ціни за вибраний проміжок часу. На відміну від EMA, усі значення вікна мають однакову вагу, тому SMA є менш чутливою до останніх коливань. У роботі використовується SMA із періодом 20, що дозволяє оцінити загальний середньостроковий тренд.

Формула SMA:

$$p = \text{ціна закриття}; N = 20 \text{ (період)} \qquad SMA_t = \frac{1}{N}\sum_{i=0}^{N-1} p_{t-i}$$

Для більш повного аналізу трендових змін включено індикатор MACD, який базується на різниці двох експоненційних середніх - короткої (12) та довгої (26). Отримана різниця формує основну лінію MACD, а додатково поверх неї обчислюється сигнальна лінія- EMA з періодом 9 від значень MACD. Таким чином, індикатор MACD складається з трьох елементів (рис. 2.5), що рухаються навколо нульової лінії:

Лінія MACD: допомагає визначити висхідний або низхідний імпульс (ринковий тренд). Вона розраховується шляхом віднімання двох експоненційних ковзних середніх (EMA).

Сигнальна лінія : EMA лінії MACD (9-періодна EMA). Комбінований аналіз сигнальної лінії з лінією MACD може бути корисним для виявлення потенційних розворотів або точок входу та виходу.

Гістограма: графічне уявлення розбіжності та сходження лінії MACD і сигнальної лінії. Інакше кажучи, гістограма розраховується з урахуванням відмінностей між двома лініями.

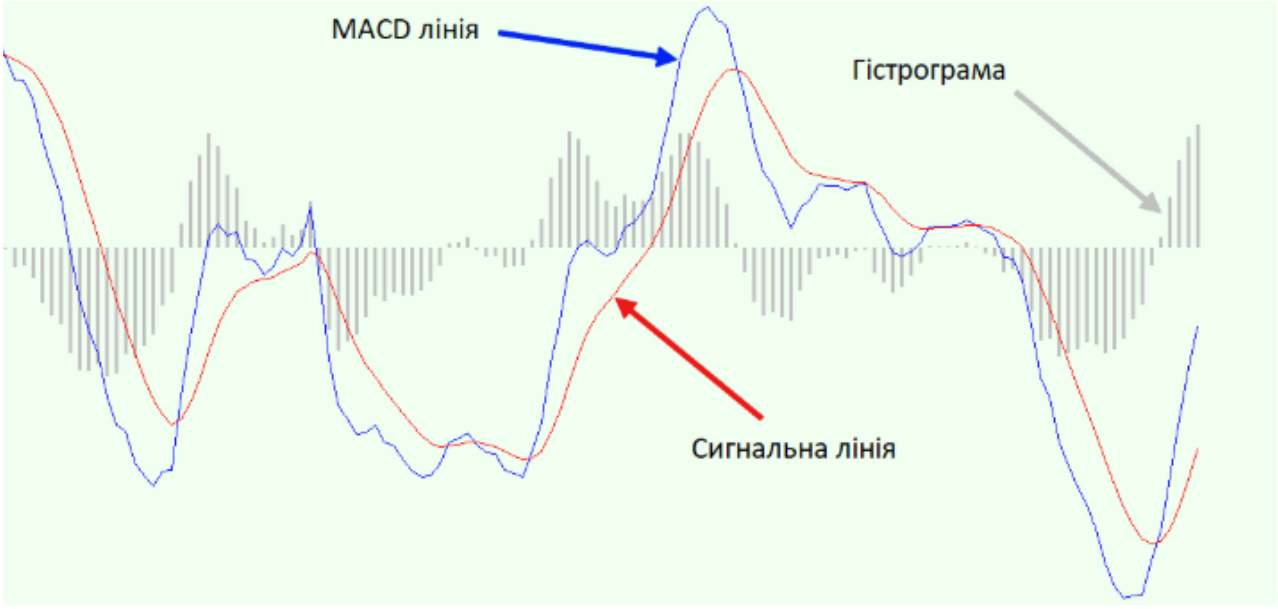

Рис. 2.5 Структура індекатора MACD.

Формула MACD:

$$MACD = EMA_{12} - EMA_{26}$$

$$Signal\ line = EMA_9(MACD)$$

Ще одним важливим інструментом є RSI - індикатор відносної сили, який оцінює співвідношення середніх величин приростів та спадів ціни за певний період. Для цього обчислюється різниця між послідовними значеннями ціни, після чого позитивні та негативні зміни усереднюються за методом Вайлдера. На основі отриманого відношення визначається значення RSI, яке змінюється в діапазоні від 0 до 100 і показує, наскільки інтенсивно ціна зростає чи падає в поточний момент. У роботі використовується RSI з періодом 7, що робить індикатор чутливим до короткострокових змін - це особливо корисно для криптовалютних ринків, які характеризуються високою волатильністю.

Кроки розрахунку:

Обчислюємо зміну ціни $\Delta_t = prise_t - prise_{t-1}$

Визначаємо позитивні та негативні зміни

$$Up_t = \max(\Delta_t, 0); Down_t = \max(-\Delta_t, 0)$$

Обчислюємо середні значення за період N (в даному випадку 7)

$SMAUP = SMA(Up, N);$ $SMADown = SMA(Down, N);$

Відношення сили $RS = \frac{SMAUp}{SMADown}$

Остаточна формула $RSI = 100 - \frac{100}{1+RS}$

Щоб оцінити відносне положення поточної ціни у межах недавнього діапазону коливань, до даних додається стохастичний осцилятор. Спочатку визначається мінімальна(low) та максимальна(hihg) ціна за останні 14 періодів, після чого поточне значення ціни(price) нормується в їх межах. Отриманий показник %K демонструє, наскільки близько ціна розташована до локального максимуму або мінімуму. Для зменшення шумів додатково обчислюється лінія %D - згладжене середнє значення %K за три періоди.

Фрмула $\%K = 100 * \frac{price - low_{min}}{high_{max} - low_{min}}$

Формула $\%D = SMA(\%K, 3)$

Для аналізу ширини цінового діапазону та пошуку зон потенційного перекуплення чи перепроданості використовуються смуги Болліджера. Спершу обчислюється проста середня SMA(20), після чого визначається стандартне відхилення за той самий період. Верхня та нижня межі формуються шляхом додавання та віднімання двох стандартних відхилень від середньої, що створює канал, у межах якого ціна перебуває більшу частину часу. Різке наближення ціни до меж каналу може сигналізувати про посилення волатильності.

Окремо у вибірку додається індикатор ATR, призначений для вимірювання реального діапазону цінових коливань. Для його обчислення визначається максимальне значення з трьох компонентів: різниці між максимумом(higt) і мінімумом(low) поточної свічки, а також модулів різниці між поточними екстремумами та ціною закриття(close) попереднього періоду. Отриманий істинний діапазон усереднюється за N періодів(в даному випадку 14), що дозволяє оцінити поточний рівень волатильності ринку.

Кроки обрахунку ATR

$$TR_1 = high_t - low_t$$
$$TR_2 = |high_t - close_{t-1}|$$
$$TR_1 = |low_t - close_{t-1}|$$
$$TR = \max(TR_1, TR_2, TR_3)$$
$$ATR = \text{SMA}(TR, N)$$

Сукупність наведених технічних індикаторів забезпечує модель нейронної мережі широким спектром характеристик - від трендових та імпульсних властивостей до оцінки ринкових діапазонів і волатильності. Така комплексна підготовка даних сприяє кращому виявленню прихованих закономірностей у часових рядах і підвищує якість прогнозування.

2.3 Методика побудови та навчання нейронної мережі

Процес прогнозування фінансових ринків з використанням нейронних мереж завжди починається з визначення правильної стратегії моделювання та вибору математичних інструментів, здатних адекватно відобразити поведінку ринкових процесів. Особливо це важливо у випадку криптовалют, адже вони характеризуються непередбачуваною динамікою, високою волатильністю, залежністю від новинних подій, глобальної економічної активності та психології учасників ринку. Моделювання такої системи вимагає врахування нелінійних залежностей, складних стохастичних коливань та великої кількості взаємодіючих ознак.

Саме тому вибір методів машинного навчання, структур мережі та способів оцінювання якості моделі суттєво впливає на точність кінцевих прогнозів.

Прогнозування вартості криптовалют – це багатовимірна задача з високим шумом, нерівномірним розподілом даних у часі та частими змінами напрямків тренду. Вибір правильної моделі визначає здатність алгоритму не лише відтворювати минулі дані, а й успішно передбачати майбутні значення під

впливом ринкових коливань. Для цього застосовуються чотири основні підходи до обробки даних у машинному навчанні: регресія, класифікація, кластеризація та крос-валідація, кожен з яких займає важливе місце у побудові нейронних моделей.

Регресія — основа для прогнозування цін. У задачі передбачення майбутньої вартості криптовалюти ключовою метою є отримання числового прогнозу, тобто оцінки ціни через певний часовий проміжок: годину, день або тиждень. Ця задача формалізується як регресія, де модель навчається відтворювати залежності між історичними даними та майбутніми цінами. Регресійні методи на основі RNN, LSTM, GRU здатні враховувати час, кореляції між послідовними значеннями та нелінійні зміни ринку. Саме тому регресія є базовим математичним механізмом, на якому будується прогнозування криптовалют.

Класифікація — прогноз поведінки ринку. У деяких випадках точне числове значення ціни не є необхідним. Трейдеру важливіше знати, який буде напрямок руху ринку — зростання, падіння або стабільність. У такому випадку задача моделі перетворюється на класифікацію, де система прогнозує тренд і формує торгові сигнали. Класифікаційні алгоритми дозволяють оцінити ризики і приймати рішення без точного передбачення вартості, що особливо корисно в умовах високої волатильності.

Кластеризація — пошук закономірностей та прихованих патернів. Кластеризація не дає прямої відповіді щодо майбутньої ціни, проте дозволяє виявити приховані структури в ринкових даних. Такий підхід класифікує поведінку ринку на групи: періоди різкої волатильності, спокійні флетові рухи, сильні трендові імпульси, а також об’єднує криптовалюти за схожістю динаміки. Часто цей метод використовується як підготовчий етап, після якого для кожного кластера окремо навчається регресійна модель. Таким чином покращується точність прогнозів і стійкість до шумів.

Крос-валідація — гарантія якості та узагальнення. Щоб переконатися, що модель не просто «вивчила» історію, а має здатність адекватно прогнозувати нові дані, застосовується крос-валідація. Вона розбиває вибірку на окремі підмножини тренування та тестування. Для часових рядів використовуються спеціалізовані схеми, які не порушують хронологію даних, щоб уникнути спотворення причинно-наслідкових зв’язків. Це дозволяє виявити перенавчання, оцінити стабільність та ефективність моделі.

Комбіновані підходи — синтез для максимальної точності Сучасні алгоритми прогнозування фінансових ринків дедалі частіше поєднують декілька методів одночасно. Наприклад, спершу виконується кластеризація ринкової поведінки, після чого для кожного кластера окремо формується регресійна модель на основі LSTM, GRU або Conv1D, а на завершальному етапі застосовується класифікація для визначення оптимальної торгової стратегії. Такий підхід дозволяє враховувати різні стани ринку, підвищує точність прогнозів та робить модель більш стійкою до шумів.

У рамках розробки моделі для прогнозування часових рядів було обрано гібридний підхід, що поєднує згорткову нейромережу (CNN) та рекурентну мережу LSTM. Така архітектура дозволяє водночас виділяти локальні закономірності у даних та враховувати довгострокові залежності між значеннями в часовій послідовності. Згорткова складова моделі реалізована за допомогою шару Conv1D, який належить до архітектури CNN і призначений для обробки одновимірних сигналів. Саме тому він часто використовується для часових рядів, де кожне спостереження складається з набору ознак, а важливими є короткі локальні шаблони, наприклад, коливання цін у криптовалютних даних.

Conv1D аналізує локальні вікна послідовності та автоматично виділяє характерні патерни, які людина не завжди може помітити вручну. Завдяки цьому шар навчається розпізнавати короткі тренди, різкі зміни, піки та циклічні коливання.

Після згорткової обробки інформація передається до рекурентної частини моделі, представленої шарами LSTM. Їх завданням є запам’ятовування важливих залежностей у ширшому часовому контексті, забезпечення стійкого прогнозу навіть при довгих послідовностях та мінімізація проблеми затухаючих градієнтів, яка присутня в простих RNN.

Conv1D–LSTM модель – це оптимальне поєднання методів для прогнозування криптовалют через її здатність поєднувати виявлення локальних патернів та моделювання довгострокових залежностей. Conv1D-шари виконують роль первинних фільтрів, які виокремлюють короткострокові закономірності та локальні тренди в часовому ряді. Завдяки згортковим операціям модель може виявляти повторювані патерни, такі як імпульси, хвилі або мікро-тренди, що важко зафіксувати рекурентними мережами самостійно.

Застосування BatchNormalization після згорток допомагає стабілізувати процес навчання та зменшити внутрішнє ковзання розподілів ознак.

Dropout використовується для зниження ризику перенавчання, вимикаючи випадкові нейрони під час тренування і змушуючи модель навчитися більш загальних представлень.

MaxPooling1D зменшує розмірність послідовності, видаляючи надлишкову інформацію і прискорюючи обробку, при цьому зберігаючи найважливіші характеристики.

Після виявлення локальних патернів згорткові шари формують набір ознак, який передається до LSTM-блоків для аналізу їхньої часової еволюції. LSTM-шари відповідають за захоплення довгострокових залежностей, які часто присутні у фінансових часових рядах через циклічність чи вплив макроекономічних подій.

Перший LSTM шар з великим числом нейронів (256) дозволяє зберегти широкий спектр інформації про послідовність і формувати багатовимірний контекст.

Другий LSTM шар з меншою кількістю нейронів (128) виступає як агрегатор і спрощувач інформації, готуючи її до фінальних шарів прийняття рішення.

Шари Dense на кінці мережі перетворюють згорнуті й рекурентно оброблені ознаки у безпосередньо прогнозовані значення або вектори прогнозів.

Використання активації ReLU виявляється ефективним у підвищенні нелінійності моделі і дозволяє швидко зближуватися під час оптимізації.

Архітектура поєднує переваги згорток у виявленні локальних характеристик та переваги LSTM у збереженні довготривалої пам’яті. Такий підхід робить модель більш стійкою до шуму, оскільки Conv1D відфільтровує короткострокові випадкові коливання, а LSTM фокусується на суттєвих трендах.

BatchNormalization і Dropout у поєднанні з правильно підібраними оптимізаторами зменшують ризик перенавчання навіть при відносно обмежених обсягах даних.

Гібридна модель показує кращу здатність до узагальнення, ніж чисто згорткові або чисто рекурентні архітектури, особливо на фінансових часових рядах.

Conv1D-шари є обчислювально ефективними, що дозволяє зменшити загальний час тренування мережі в порівнянні з лише глибокими рекурентними архітектурами.

LSTM-шари, у свою чергу, більш толерантні до пропусків та нерегулярних часових інтервалів, якщо такі трапляються у даних.

Комбінована архітектура дає можливість використовувати різні типи ознак:

- цінові;
- об’ємні;

- індикаторні та навіть зовнішні фактори такі як індекси чи текстові сигнали.

Двошарові LSTM дозволяють моделі навчитися абстрагованим презентаціям послідовності, спочатку в широкому просторі, а потім у більш компактному.

Останній Dense-64 шар виступає як шар проміжної нелінійної агрегації, передаючи стиснуту репрезентацію в кінцевий вихід.

Вихідний Dense шар без активації або з лінійною активацією підходить для регресійних задач, коли необхідно передбачити безпосередні числові значення.

Таке проектування спрощує адаптацію моделі до різних задач — від прогнозу числових значень до передбачення напрямку руху або ймовірності подій.

Інтеграція BatchNormalization доцільна для пришвидшення зближення і вирівнювання масштабу ознак між шарами.

Conv1D-фільтри з експериментально підібраними розмірами кернела дозволяють контролювати чутливість до тривалості локальних патернів. Вибір кількості фільтрів (64 і 128) забезпечує баланс між представницькою здатністю та обчислювальною складністю.

Dropout значення 0.5 застосоване для сильного регуляризаційного ефекту, що корисно при невеликих наборах даних або високому шумі, що характерно для фінасових даних.

LSTM з return_sequences=True у першому шарі дозволяє зберегти послідовний вихід для наступного LSTM шару, що покращує навчання ієрархічних часових структур. Вбудована гнучкість цієї архітектури дозволяє легко додавати додаткові шари або модулі, такі як attention-механізми або residual-зв’язки.

Для задач в реальному часі модель можна оптимізувати шляхом зменшення числа параметрів або заміни LSTM на GRU у другому шарі.

У практичній реалізації важливо налаштувати оптимізатор, learning rate scheduler та ранню зупинку для запобігання перенавчанню та збереження стабільності.

2.4 Комп’ютерні експерименти та дослідження моделей

З метою вибору найбільш ефективного оптимізатора для моєї нейроної мережі було проведено серію комп’ютерних експериментів (рис. 2.6–2.21), у межах яких здійснювалося порівняння різних підходів до оптимізації процесу навчання.

У дослідженні розглядалися як класичні методи градієнтного спуску, зокрема стохастичний градієнтний спуск (SGD), так і сучасні адаптивні оптимізатори, серед яких Adam, Nadam, AdamW та Radam. Оцінювання ефективності кожного з методів здійснювалося на основі швидкості збіжності, стабільності навчання та якості прогнозування на тестових даних.

Результати експериментів показали, що окремі адаптивні оптимізатори, зокрема AdamW, забезпечують швидке зменшення функції втрат і прискорене досягнення локального мінімуму. Водночас у ряді випадків така перевага супроводжувалася зниженням здатності моделі до узагальнення, що проявлялося у менш стабільних результатах на валідаційній вибірці.

Це підтверджує доцільність більш глибокого аналізу не лише швидкості навчання, а й узагальнюючих властивостей моделі. У зв’язку з цим додаткову увагу було приділено гібридним стратегіям оптимізації, які поєднують переваги адаптивних методів і класичних підходів.

Зокрема, як зазначається у дослідженні [4], перехід від оптимізаторів типу Adam або споріднених до нього методів до SGD на пізніх етапах навчання може суттєво покращити здатність моделі до узагальнення та підвищити стабільність кінцевих результатів. Такий підхід було враховано під час проведення експериментів і подальшого аналізу отриманих результатів.

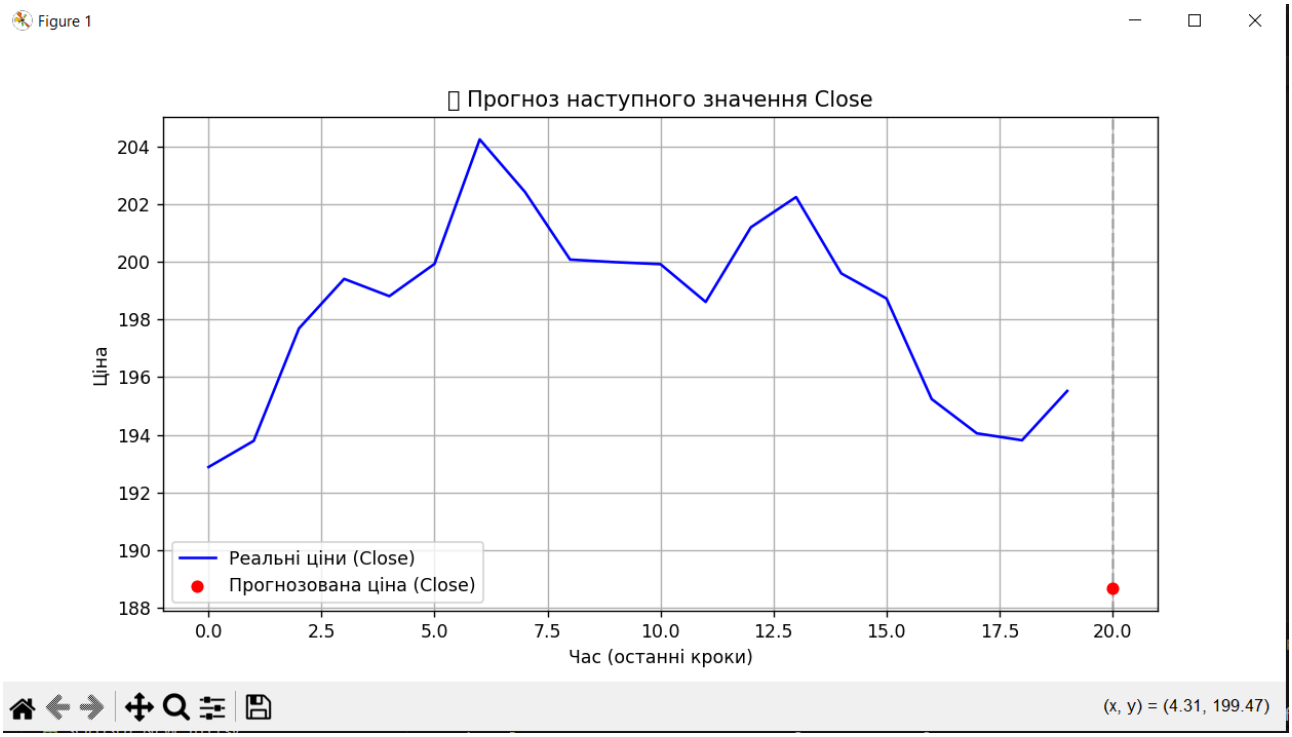

Рис. 2.6 Графік точності прогнозу оптимізатор Adam перший етап навчання

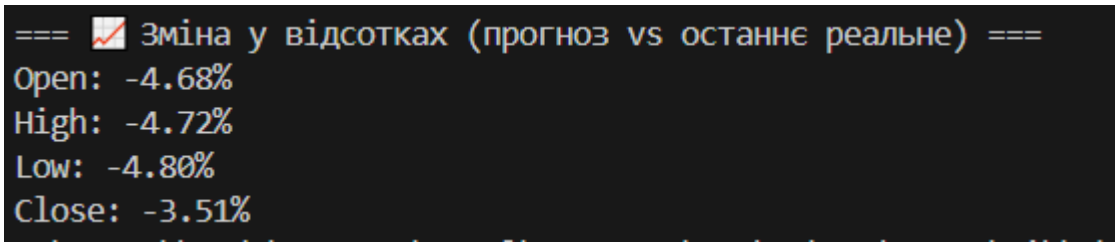

Рис. 2.7 Точність оптимізатора Adam перший етап навчання

Перший етап навчання нейронної мережі було здійснено з використанням адаптивного оптимізатора Adam. На даному етапі модель навчалася з фіксованими гіперпараметрами, що дозволило оцінити базову динаміку процесу оптимізації, швидкість збіжності та стабільність зменшення функції втрат.

Отримані результати(рис. 2.6 - 2.7) дають змогу проаналізувати ефективність обраного оптимізатора на початковій фазі навчання, а також

слугують базою для подальшого порівняння з іншими методами оптимізації та гібридними стратегіями.

На рис. 2.6 представлено графік результатів оцінювання точності прогнозування ціни закриття (Close). Синя крива відображає фактичні значення ціни за останні 20 часових інтервалів (свічок), тоді як червона точка відповідає прогнозованому значенню на наступний часовий крок. Як видно з наведеного графіка, після завершення першого етапу навчання нейронна мережа сформувала прогноз, що знаходиться у безпосередній близькості до фактичного значення ціни. Це свідчить про те, що модель на початковій стадії навчання змогла коректно відтворити основні закономірності часової динаміки даних та адекватно апроксимувати поведінку ринку на короткому горизонті прогнозування. Невелике відхилення прогнозованого значення від реального не має системного характеру та може бути зумовлене високою волатильністю ринку, а також наявністю шуму у вхідних даних.

На рис. 2.7 наведено відносні похибки прогнозування, поданої у відсотковому вираженні. Відхилення обчислюється окремо для кожного прогнозованого параметра, зокрема для ціни відкриття (Open), мінімального (Low), максимального (High) та кінцевого (Close) значень. На основі отриманих числових результатів можна зробити висновок, що вже після першого етапу навчання модель демонструє відносно низьку похибку прогнозування, яка не перевищує 5%.

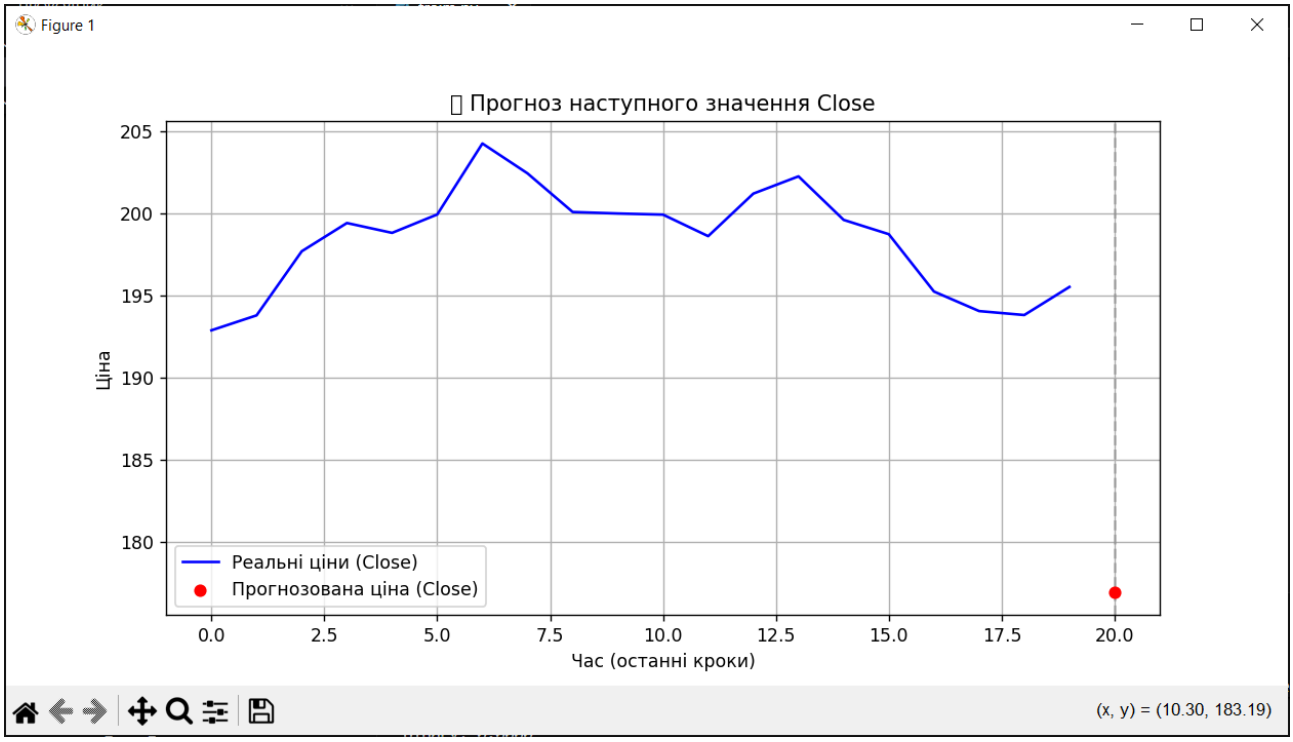

Рис. 2.8 Графік точності прогнозу оптимізатор Nadam другий етап навчання

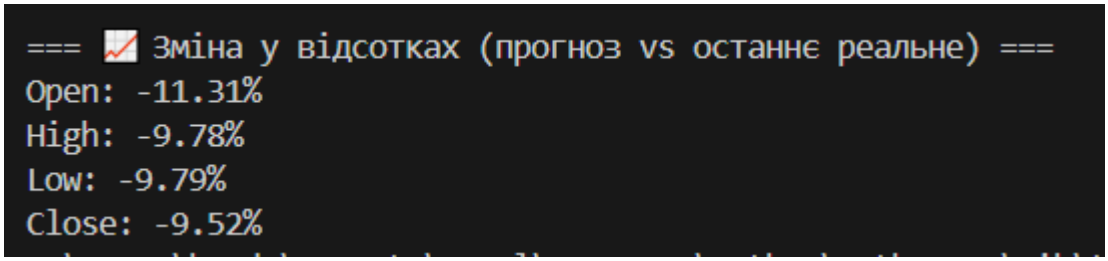

Рис. 2.9 Точність оптимізатора Nadam другий етап навчання

Другий етап навчання нейронної мережі було реалізовано зі зміною оптимізатора: замість Adam було використано Nadam. Такий підхід дозволив оцінити вплив модифікованого адаптивного методу оптимізації на якість прогнозування моделі.

Як наочно видно з графіка, поданого на рис. 2.8, прогнозоване значення ціни закриття суттєво віддалилося від реального значення у порівнянні з результатами, отриманими на першому етапі навчання.

Відповідно до результатів, наведених на рис. 2.9, спостерігається зростання відносної похибки не лише для ціни закриття, але й для інших прогнозованих показників, зокрема ціни відкриття, мінімального та

максимального значень. Це свідчить про погіршення загальної якості прогнозування та зниження стабільності роботи моделі після зміни оптимізатора.

Таким чином, на основі проведеного експерименту можна зробити висновок, що другий етап навчання з переходом від оптимізатора Adam до Nadam чинить негативний вплив на точність прогнозування та загальну ефективність даної моделі.

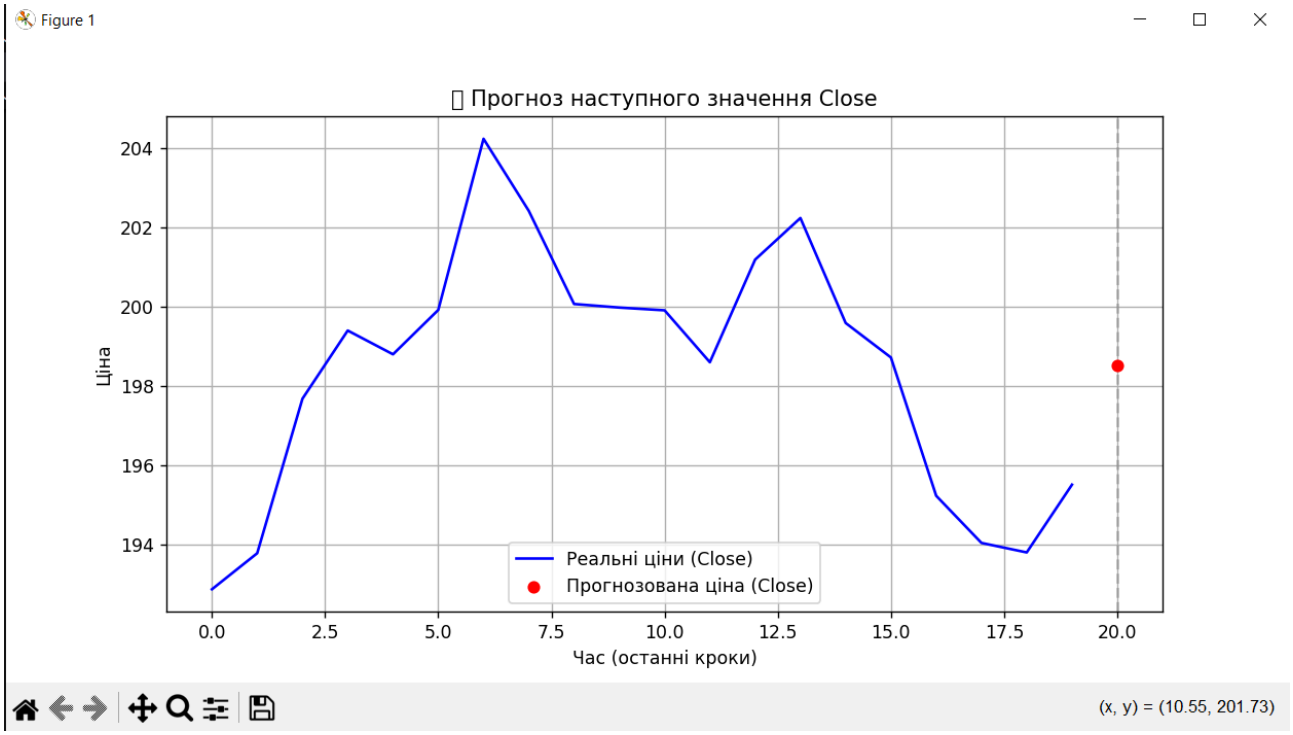

Рис. 2.10 Графік точності прогнозу оптимізатор Radam перший етап навчання

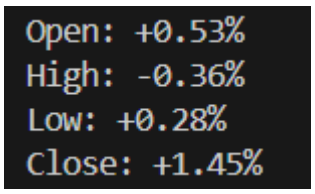

Рис. 2.11 Точність оптимізатора Radam перший етап навчання

Перший етап навчання нейронної мережі було проведено з використанням оптимізатора RAdam (Rectified Adam), який поєднує переваги адаптивних методів оптимізації та підвищену стабільність процесу навчання на початкових ітераціях.

Відповідно до графіка, наведеного на рис. 2.10, можна спостерігати, що модель надзвичайно точно відтворила загальну тенденцію зміни ціни та сформувала прогноз, який знаходиться у безпосередній близькості до фактичного значення.

Аналіз результатів, поданих на рис. 2.11, додатково підтверджує високу ефективність обраного оптимізатора: вже на першому етапі навчання зафіксовано надзвичайно низький рівень відносної похибки для інших прогнозованих показників, зокрема ціни відкриття, мінімального та максимального значень. Отримані результати свідчать про високу стабільність навчання та здатність моделі до коректного узагальнення навіть на ранній стадії оптимізації.

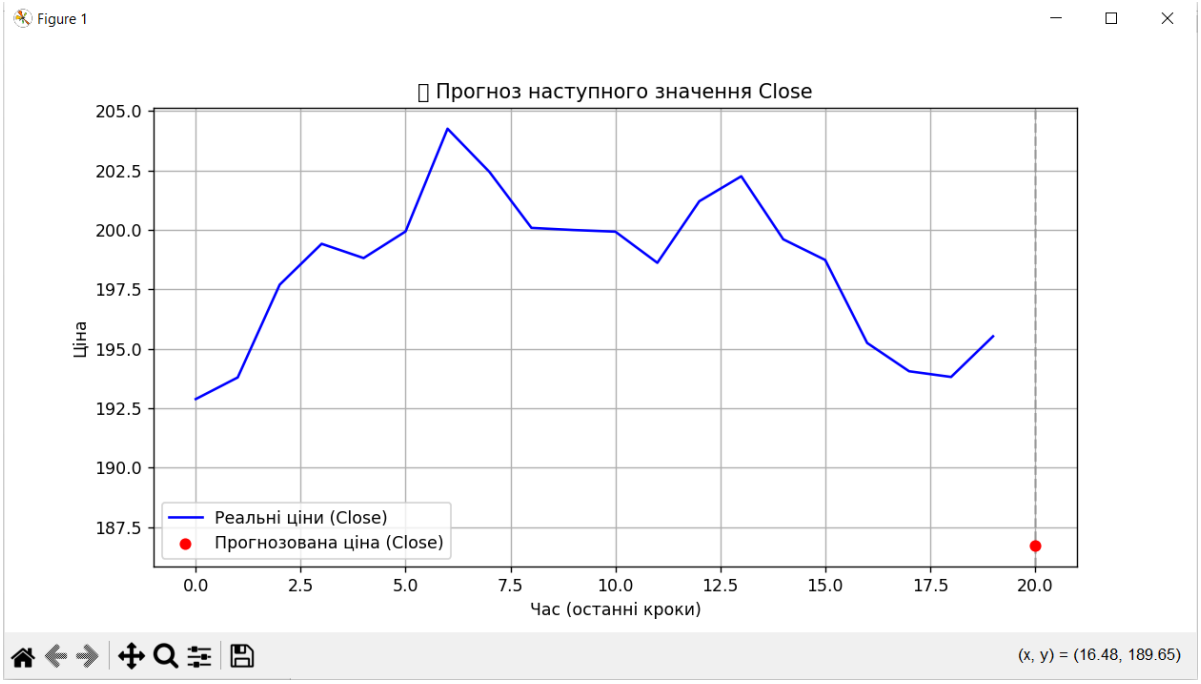

Рис. 2.12 Графік точності прогнозу оптимізатор Nadam другий етап навчання

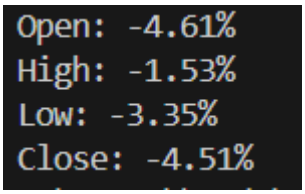

Рис. 2.13 Точність оптимізатора Nadam другий етап навчання

Другий етап навчання нейронної мережі було виконано зі зміною оптимізатора: замість RAdam застосовано Nadam. Такий підхід дозволив проаналізувати вплив даної заміни на стабільність навчання та точність прогнозування.

Як наочно видно з графіка, наведеного на рис. 2.12, після переходу до оптимізатора Nadam модель почала втрачати точність прогнозу, що проявляється у збільшенні відхилення прогнозованого значення від фактичного.

Додатково, результати, подані на рис. 2.13, свідчать про зростання відносної похибки не лише для ціни закриття, але й для інших прогнозованих показників, зокрема ціни відкриття, мінімального та максимального значень. Це вказує на загальне погіршення якості прогнозування та зниження узагальнюючих властивостей моделі.

За результатами проведеного експерименту можна зробити висновок, що перехід від оптимізатора RAdam до Nadam призвів до збільшення похибки прогнозування та негативно вплинув на ефективність даної моделі.

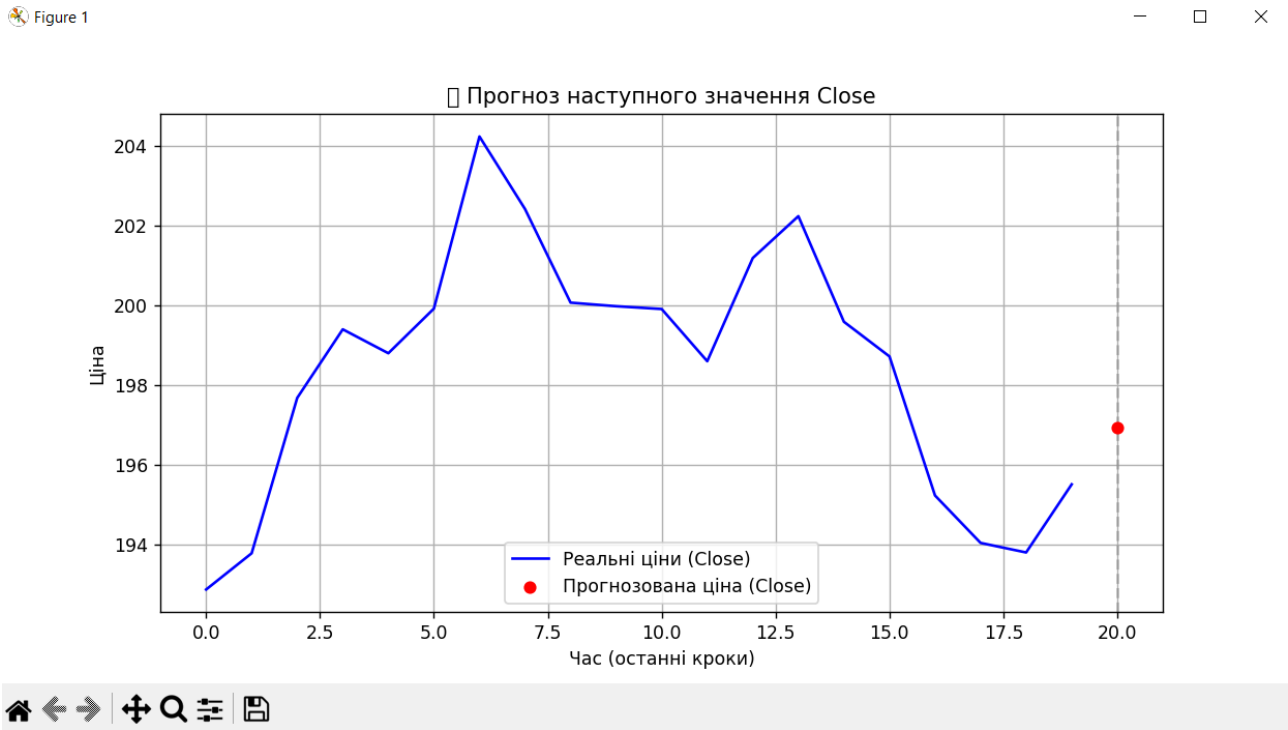

Рис. 2.14 Графік точності прогнозу оптимізатор AdamW перший етап навчання

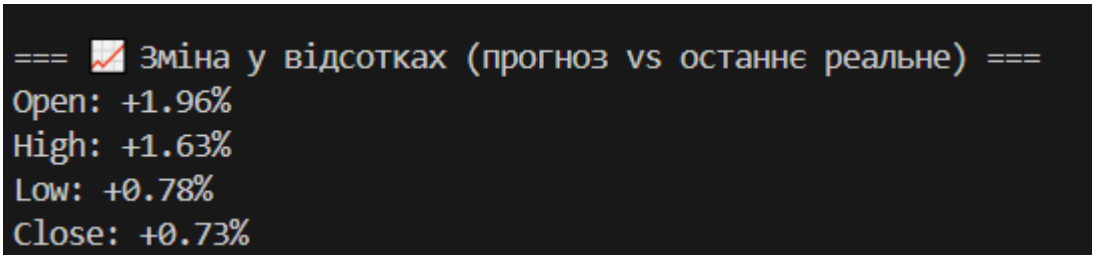

Рис. 2.15 Точність оптимізатора AdamW перший етап навчання

Перший етап навчання нейронної мережі було проведено з використанням оптимізатора AdamW, який є модифікацією класичного Adam із відокремленою ваговою регуляризацією (weight decay).

Як наочно видно з графіка, наведеного на рис. 2.14, модель демонструє швидку збіжність та достатньо близьке наближення прогнозованого значення ціни до фактичного.

Прогноз коректно відтворює загальну тенденцію зміни ринку, однак як видно на рис. 2.15, відносні похибки при прогнозувані всіх показників, дещо більші, у порівнянні з результатами, отриманими при використанні оптимізатора Radam.

Загалом результати першого етапу навчання з оптимізатором AdamW свідчать про його ефективність на початковій стадії оптимізації та здатність моделі адекватно вловлювати основні закономірності часових рядів, що робить доцільним подальший аналіз його поведінки на наступних етапах навчання та у порівнянні з іншими оптимізаторами.

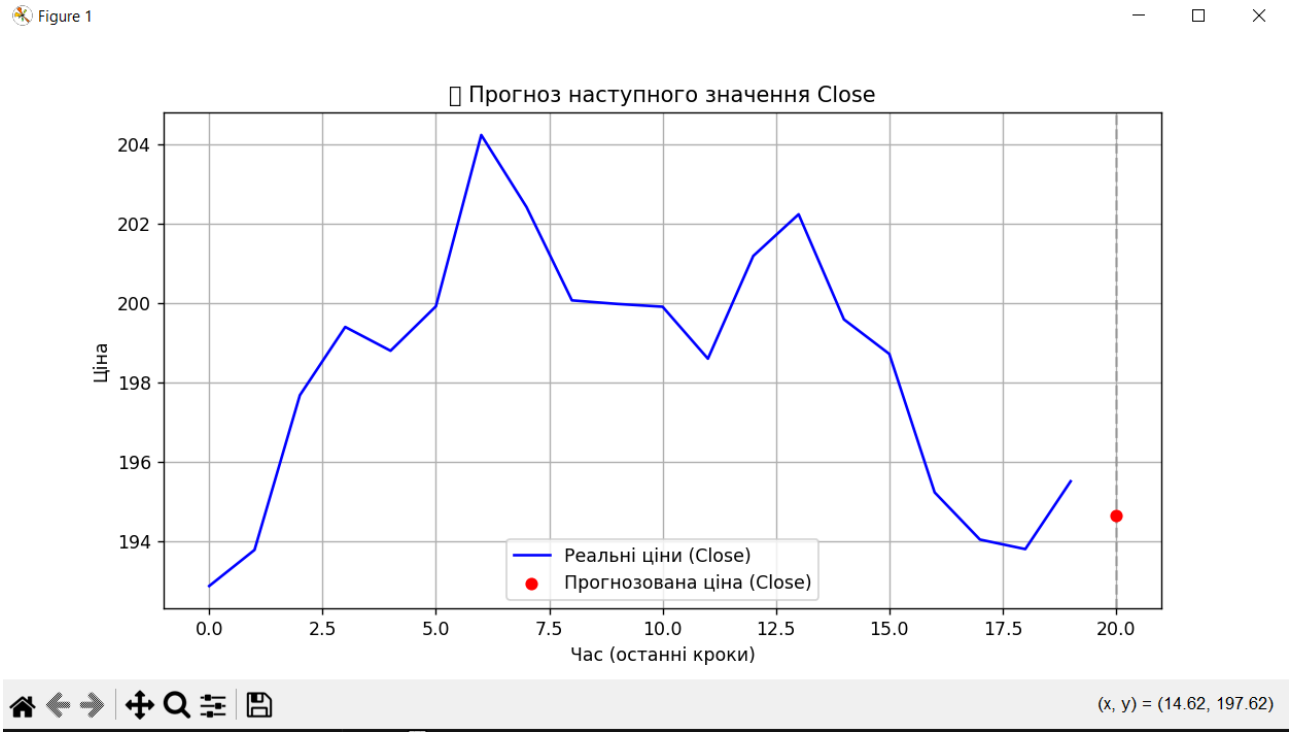

Рис. 2.16 Графік точності прогнозу оптимізатор Nadam другий етап навчання

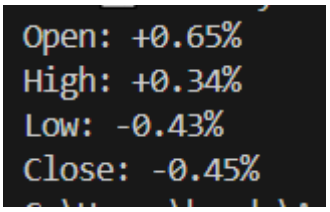

Рис. 2.17 Точність оптимізатора Nadam другий етап навчання

Другий етап навчання нейронної мережі було проведено із заміною оптимізатора: замість AdamW використано Nadam. Мета цього підходу полягала у покращенні стабільності та точності прогнозування шляхом поєднання переваг двох адаптивних методів оптимізації.

Як видно з графіка, наведеного на рис. 2.16, прогнозоване значення ціни наблизилося до фактичного, демонструючи помітне підвищення точності порівняно з першим етапом навчання.

Аналіз відносної похибки, представленої на рис. 2.17, підтверджує зростання точності прогнозування для всіх основних показників, включаючи ціни відкриття, мінімум, максимум та закриття.

Можна зробити висновок, що послідовне використання оптимізаторів AdamW і Nadam дозволило знизити похибку прогнозування та підвищити ефективність моделі, забезпечивши кращу узагальнюючу здатність на тестових даних.

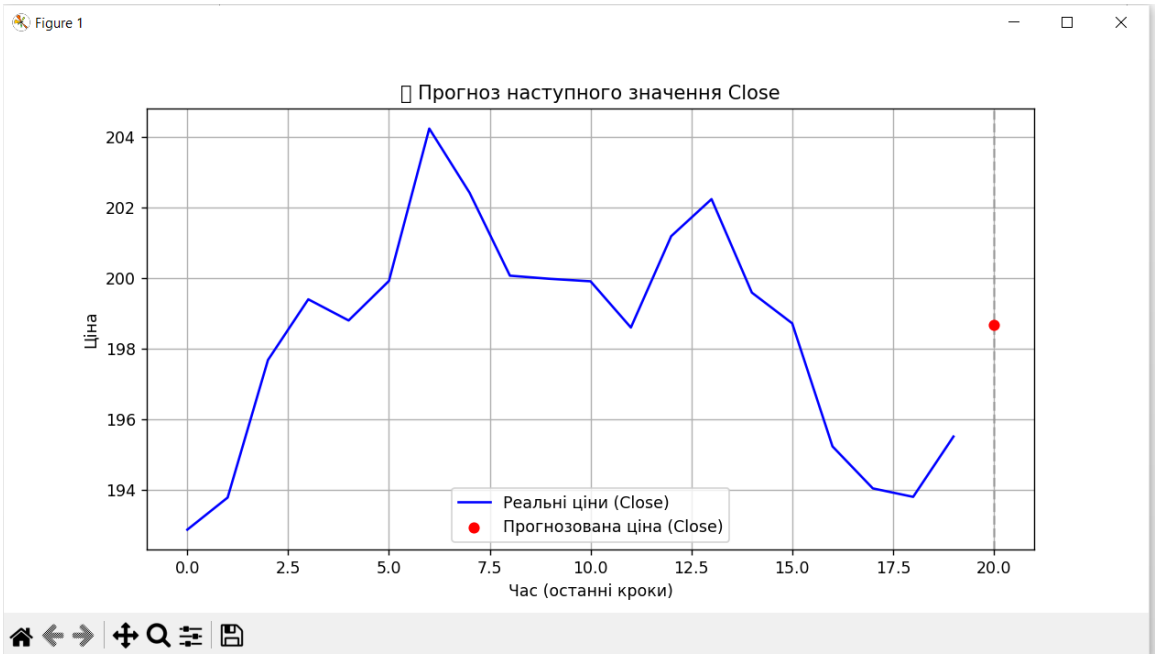

Рис. 2.18 Графік точності прогнозу оптимізатор AdamW перший етап навчання

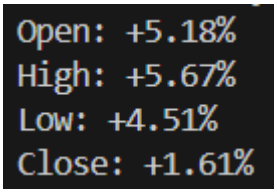

Рис. 2.19 Точність оптимізатора AdamW перший етап навчання

Перший етап навчання нейронної мережі було проведено з використанням оптимізатора AdamW.

Як видно з графіка, наведеного на рис. 2.18, модель здатна відтворювати основні патерни поведінки ринку, вловлюючи тенденції зміни ціни.

Однак аналіз відносної похибки, поданої на рис. 2.19, показує, що точність прогнозу поки що є відносно низькою, і відхилення прогнозованих значень від фактичних є помітним. Це свідчить про необхідність подальшого вдосконалення моделі.

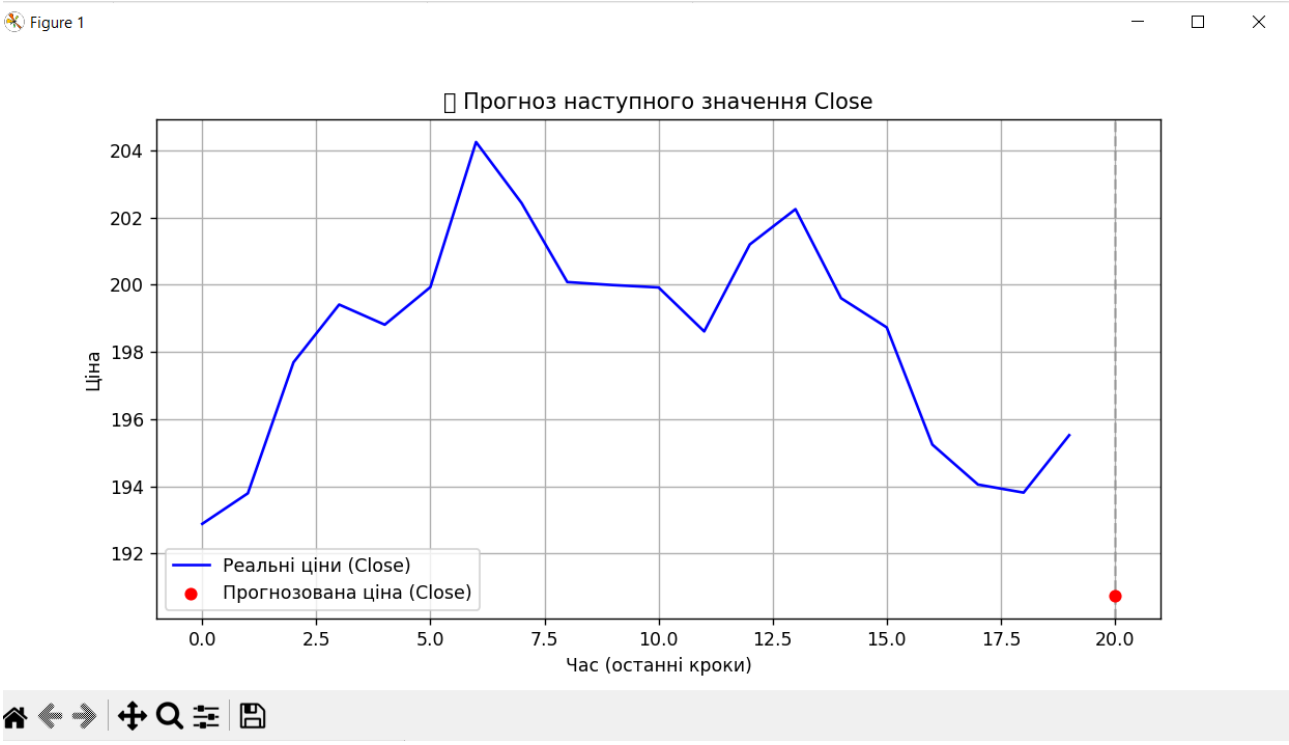

Рис. 2.20 Графік точності прогнозу оптимізатор SGD другий етап навчання

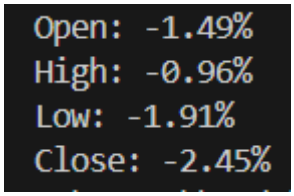

Рис. 2.21 Точність оптимізатора другий етап навчання

Другий етап навчання нейронної мережі було проведено з переходом на оптимізатор SGD.

Як видно з графіка, наведеного на рис. 2.20, модель дещо неточно оцінює напрямок руху ринку на деяких часових інтервалах, що проявляється у незначних відхиленнях прогнозованих значень від фактичних.

Водночас аналіз відносної похибки, представлений на рис. 2.21, демонструє загальне зниження похибки та підвищення точності прогнозування для всіх основних показників, включаючи ціни відкриття, мінімум, максимум та закриття.

Отже, можна зробити висновок, що перехід на оптимізатор SGD сприяв покращенню узагальнюючих властивостей моделі та підвищенню точності

прогнозованих значень, незважаючи на окремі неточності у відтворенні короткострокових коливань ринку.

Аналізуючи результати експериментів (рис. 2.6–2.21), можна помітити, що комбінування двох оптимізаторів не завжди покращує якість навчання моделі. У низці випадків використання зв’язки Adam із Nadam або поєднання Radam з Nadam призводило до підвищення помилки, що свідчить про їхню недостатню взаємодоповнюваність.

Водночас інші комбінації демонстрували протилежний ефект. Зокрема, послідовне застосування AdamW з Nadam, а також стратегія перемикання з AdamW на Namam забезпечували зменшення похибки та більш стабільну збіжність моделі на валідаційних даних.

Це вказує на те, що правильно підібрані гібридні алгоритми здатні компенсувати недоліки один одного: AdamW покращує контроль над регуляризацією та швидко стабілізує ранні етапи навчання, тоді як Nadam дозволяє ефективніше коригувати градієнти та активніше реагувати на зміну напрямку оптимізації.

З огляду на отримані результати, для своєї моделі я обираю схему гібридного навчання, у якій послідовно використовуватимуться два оптимізатори: AdamW та Nadam . Такий підхід забезпечує швидку збіжність на старті, краще узагальнення на завершальних етапах і мінімізує ризик «застрягання» у локальних мінімумах, що є критично важливим для прогнозування часових рядів у фінансовій сфері.

2.5 Висновок до розділу.

У даному розділі було досліджено ключові аспекти побудови моделі прогнозування вартості криптовалют на основі методів машинного навчання та нейронних мереж.

Значну увагу приділено питанню попередньої обробки даних, оскільки саме якість вхідної інформації визначає ефективність будь-якої моделі. Було підкреслено важливість обсягу, структури та статистичних характеристик даних, що впливають на стабільність навчання та здатність мережі виявляти закономірності у високоволатильних часових рядах. Було детально розглянуто різні підходи до нормалізації та стандартизації даних, проаналізовано їх сильні сторони й обмеження. Особливий акцент зроблено на застосуванні засобів бібліотеки MinMaxScaler, яка дозволяє масштабувати значення до уніфікованих діапазонів, зменшуючи вплив відмінностей масштабу між ознаками. Також досліджено роль технічних індикаторів - EMA, SMA, MACD, RSI, стохастичного осцилятора, смуг Болліджера та ATR -які доповнюють вихідний набір даних важливими сигналами щодо імпульсу, волатильності та напрямку тренду. Дане посилення інформаційної насиченості даних дає змогу нейромережі ефективніше виявляти приховані нелінійні залежності.

Особливу увагу приділено аналізу специфіки криптовалютних часових рядів, що характеризуються високою волатильністю, шумністю та динамічно змінюваними закономірностями. Через це використання простих статистичних моделей є недостатнім - виникає необхідність у застосуванні глибоких нейронних архітектур, здатних працювати з нелінійними структурами та мінливими ринковими патернами.

У межах роботи було обґрунтовано доцільність застосування регресійного підходу для прогнозування числових значень ціни. Окремо зазначено, що класифікаційні методи можуть виступати допоміжним інструментом для визначення напрямку руху ринку, а методи кластеризації - підвищувати якість попередньої обробки даних, виявляючи приховані групи станів ринку. Метод крос-валідації визначено як ключовий механізм оцінки стійкості моделі, особливо з огляду на властиву часовим рядам структурну нестабільність.

Значущим результатом стало обґрунтування вибору архітектури CNN–LSTM, яка поєднує здатність згорткових шарів виявляти локальні закономірності у вибірках із можливістю LSTM-шарів запам’ятовувати довготривалі

залежності. Така конструкція дає змогу створити модель, здатну адаптивно реагувати на короткострокові сигнали, зберігаючи при цьому широкий контекст ринкових змін.

Також було показано, що вибір оптимізатора суттєво впливає на поведінку моделі. Аналіз продемонстрував переваги гібридної стратегії, яка поєднує AdamW та Nadam: перший покращує здатність до узагальнення за рахунок коректної L2-регуляризації, тоді як другий забезпечує швидшу й чутливішу адаптацію до змінних трендів завдяки модифікованій імпульсній корекції.

Обраний підхід дозволяє досягти нижчої похибки та стабільнішої збіжності під час навчання. У сукупності всі описані методи формують системний підхід до моделювання криптовалютних ринків.

Вибір індикаторів, ретельна нормалізація, класифікаційні та кластеризаційні механізми, оцінка стабільності через крос-валідацію, гібридна архітектура CNN–LSTM та комбіновані оптимізатори створюють глибоку, адаптивну та надійну модель прогнозування. У підсумку можна стверджувати, що розроблена модель не лише враховує складність і мінливість криптовалютних часових рядів, а й здатна ефективно адаптуватися до умов ринку. Це визначає її практичну цінність та відкриває перспективи для подальших досліджень і застосувань у реальних інвестиційних задачах.

РОЗДІЛ. РОЗРОБКА РОЗШИРЕННЯ ДЛЯ CHROME

3.1 Особливості розробки розширення для браузера.

У попередніх розділах було детально розглянуто методи моделювання нейронної мережі та підходи до її оптимізації. На цьому етапі настає перехід від теоретичного та математичного підґрунтя до створення прикладного інструмента. Іншими словами, тепер необхідно інтегрувати розроблену модель у зручну «обгортку» та надати їй практичної форми, придатної для використання кінцевими користувачами.

Саме тому цей розділ присвячений розробці браузерного розширення, яке взаємодіє з сервером, виконує обчислення та забезпечує інтуїтивний інтерфейс для роботи з прогнозами.

Розробка розширення для браузера має як спільні риси, так і суттєві відмінності порівняно зі створенням звичайної настільної програми. Подібність проявляється у використанні класичних підходів до структурування логіки, роботі з даними, застосуванні API та побудові клієнт-серверної архітектури.

Проте особливість браузерного розширення полягає у глибокій інтеграції з середовищем браузера, обмеженнях безпеки, спеціальній файловій структурі та вимозі працювати в умовах sandbox-оточення. Розширення не є повноцінною програмою у традиційному розумінні — воно функціонує всередині браузера, підпорядковуючись його політикам, API та механізмам взаємодії з вебсередовищем.

Структура браузерного розширення включає кілька ключових компонентів:

- manifest.json — основний конфігураційний файл, що описує дозволи, скрипти, доступ до сторінок та налаштування.
- background (service worker) — логічний центр, який працює у фоновому режимі, керує обміном даних із сервером, WebSocket-з’єднаннями, таблицями параметрів і викликами API.

- content scripts - скрипти, що взаємодіють із вебсторінками, перехоплюють або відображають інформацію.
- UI-компоненти (popup, options, окремі HTML-сторінки) - інтерфейсна частина, доступна користувачу.

Саме фоновий скрипт виконує роль «з'єднувального вузла» між клієнтом і сервером, координуючи потоки інформації. Комунікація зазвичай відбувається через HTTP-запити або WebSocket-підключення, що дозволяє отримувати прогноз у реальному часі, надсилати вхідні дані на сервер та обробляти відповіді нейронної мережі.

Таким чином вибудовується повноцінна асинхронна система, де браузерне розширення виступає фронтендом, а сервер - обчислювальним ядром. Особливу увагу слід приділити тому, що розширення обмежене політиками браузера, включно з CORS, дозволами доступу, правилами використання локального сховища та обмеженнями на виконання коду. Це вимагає ретельного планування архітектури рис. 3.1 та коректної організації взаємодії між компонентами.

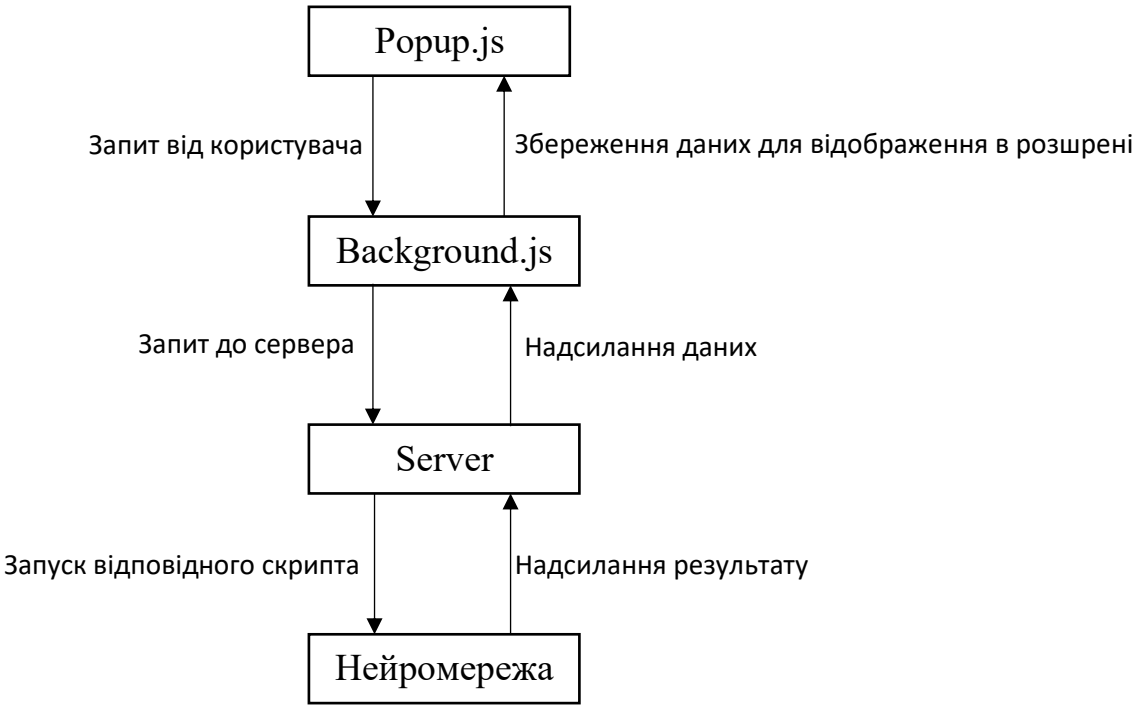

Рис. 3.1 Архітектура розширення

Для створення проєкту використовувалося середовище розробки Visual Studio Code, яке є гнучким інструментом, призначеним для веборієнтованих проєктів. Воно забезпечує підтримку JavaScript, HTML, JSON, TypeScript, а також плагінів для налагодження, форматування коду та роботи з WebSocket-з'єднаннями. Завдяки інтеграції Git, автодоповненню та можливості працювати одночасно з клієнтським та серверним кодом, Visual Studio Code є оптимальним вибором для створення браузерних розширень.

Важливо зазначити, що під час розробки розширення доводиться враховувати не лише логіку інтерфейсу, а й ефективність обробки даних, стабільність зв’язку з сервером та безпеку роботи. Унікальність таких застосунків полягає в тому, що вони об’єднують вебтехнології та механізми роботи операційної системи через інструменти браузера.

Створення браузерного розширення - це комплексний процес, який поєднує веброзробку, клієнтсько-серверну взаємодію та інтеграцію з інструментами машинного навчання. Завдяки такому підходу стає можливим забезпечити зручну, доступну та функціональну оболонку для використання нейронної мережі у реальних умовах.

3.2 Запуск сервера для функціонування розширення

Сервер у архітектурі «браузерне розширення ↔ сервер ↔ модель» виконує ключову роль - він є обчислювальним ядром і центром комунікації. Саме сервер приймає вхідні дані від розширення (віконні фрагменти часу, ознаки, запити на прогноз), передає їх у нейромережу, виконує обчислення (інференс або навчання), формує результати й повертає їх клієнту у зручному форматі. Окрім чисто обчислювальної функції, сервер також відповідає за: підготовку та валідацію вхідних даних, кешування результатів, логування, обробку черг

запитів, а також за механізми автентифікації та контролю доступу у випадку розгортання в мережі.

Існує кілька типів серверних рішень, придатних для інтеграції з браузерним розширенням:

- Локальний сервер (on-premise) - запускається на локальному ПК користувача або на сервері в локальній мережі.

  Переваги:

  - низька затримка;
  - повний контроль над даними;
  - простота налагодження;
  - приватність (дані не залишають машину).

  Недоліки:

  - обмежена доступність ззовні;
  - залежність від ресурсів локальної машини;
  - необхідність локального супроводу (оновлення, запуск).
- Віддалений (cloud) сервер / VPS — розгорнутий у хмарі (AWS, GCP, Azure, DigitalOcean).

  Переваги:

  - масштабування;
  - висока доступність;
  - простота інтеграції з CI/CD;
  - можна обслуговувати багато користувачів.

  Недоліки:

  - мережеві затримки;
  - витрати на хмару;
  - питання приватності та безпеки даних.
- Serverless / функції як сервіс (FaaS) — AWS Lambda, Google Cloud Functions тощо.

  Переваги:

  - оплата лише за використання;

- автоматичне масштабування.

Недоліки:

- холодні старти;
- обмеження часу виконання;
- менш підходить для довготривалих процесів інференсу або WebSocket-з'єднань.

- Протокол-орієнтовані рішення:
  - REST API - класичний HTTP/HTTPS підхід для запит-відповідь. Добре підходить для періодичних запитів або ненав'язливих запитів на інференс.
  - WebSocket / socket.io / ws - двостороннє постійне з'єднання, оптимальне для реального часу, стрімінгових даних та швидких оновлень.
  - gRPC - бінарний, високопродуктивний протокол, корисний для сервіс-до-сервісу зв'язку в мікросервісній архітектурі.

Кожне рішення має свої сильні та слабкі сторони. REST простий у впровадженні, сумісний з будь-яким клієнтом, але повільніший для частих оновлень через накладні витрати HTTP-запитів.

WebSocket підтримує низьколатентний двосторонній обмін, що робить його придатним для передачі сигналів у реальному часі (стріми цін, миттєві прогнозні оновлення).

gRPC - відмінний вибір при побудові внутрішньої сервісної шини з високою пропускною спроможністю, але складніший для прямої інтеграції в браузер (вимагає proxy або додаткових бібліотек).

Оптимальне рішення для моєї магістерської роботи (браузерне розширення з вбудованою/локальною нейромережею для прогнозування криптовалют), є локальний сервер на базі WebSocket.

Основні переваги :

- Мінімальна затримка. Інференс часто повинен виконуватися швидко - WebSocket дозволяє передавати дані й отримувати відповіді миттєво, без накладних витрат на відкриття HTTP-з'єднання.
- Підтримка двосторонньої комунікації. Розширення може не лише надсилати запит, а й отримувати події від сервера (оновлення прогнозів, повідомлення про готовність моделі після перенавчання).
- Приватність та контроль. Локальний сервер гарантує, що чутливі ринкові або користувацькі дані не виходять за межі машини користувача.
- Простота розробки та налагодження. Розгортання локального WebSocket-сервера зручно відлагоджувати на одному ПК разом із моделлю, логами та інструментами розробника.

Однією з поширених, простих у використанні ідеї для локального WebSocket-сервера є бібліотека websockets (Python).

Основні можливості та функції цієї бібліотеки:

- Асинхронний сервер і клієнт - websockets побудований на asyncio, що дозволяє обслуговувати багато з'єднань одночасно без блокувань. Прийом і відправка повідомлень (text/binary) - сервер може отримувати текстові або бінарні повідомлення від клієнта і відправляти у відповідь прогнозні дані або статуси.
- Підтримка ping/pong та таймаутів - механізми для перевірки живучості з'єднання та виявлення обірваних клієнтів.
- Легкість у визначенні протоколу обміну (JSON-формат) - зазвичай обмін відбувається у вигляді JSON-повідомлень із полями type, payload, request_id тощо.
- Обробка багатьох клієнтів - можна створити логіку для мультивідправки (broadcast) або адресної відповіді.

- Шифрування (WSS) можливе при використанні SSL-сертифікатів - хоча для локального середовища це не завжди необхідно, у продакшн-режимі слід використовувати WSS.

Локальний WebSocket-сервер - ідеальний для розробки, тестування та персональних застосунків. Якщо ж потрібно розширити систему для багатьох користувачів або забезпечити віддалений доступ до моделі, варто розглянути перехід до cloud-серверів із балансуванням навантаження, контейнеризацією (Docker, Kubernetes) та використанням високопродуктивних протоколів (gRPC) або брокерів повідомлень (RabbitMQ, Kafka) для асинхронної обробки.

Для даної реалізації браузерного розширення з локальною нейромережею найбільш оптимальним та зручним рішенням є запуск локального WebSocket-сервера (на базі Python websockets). Основні властивості WebSocket-сервера подані в таблиці 3.1.

Таблиця 3.1 Основні властивосі WebSocket-сервера

Властивість	**Пояснення**
Двосторонність	Клієнт і сервер надсилають дані
Реальний час	Немає затримок опитування
Постійне з’єднання	Без повторних запитів
Низький overhead	Менше трафіку
Формат даних	JSON / текст / binary

Для кращого розуміння роботи WebSocket, варто розглянути його роботу поетапно рис.3.2.

1. Встановлення з’єднання (Handshake).

- Клієнт (зазвичай браузер) ініціює запит на встановлення WebSocket-з’єднання до сервера через стандартний HTTP/HTTPS запит.
- В HTTP-запиті вказується заголовок Upgrade: websocket, який сигналізує серверу про намір перейти з протоколу HTTP на WebSocket.

- Сервер відповідає підтвердженням, підтверджуючи підтримку WebSocket-протоколу та погоджуючись на оновлення з’єднання.
- Після цього встановлюється двостороннє TCP-з’єднання, яке підтримується відкритим.

2. Двостороння комунікація (Full-Duplex)

- Після успішного «handshake» обидві сторони можуть надсилати дані одночасно і без повторного HTTP-запиту.
- Кожне повідомлення передається у вигляді фреймів (frame), які містять: тип повідомлення (текст, бінарні дані, ping/pong), довжину даних, самі дані.
- Клієнт і сервер можуть миттєво обмінюватися повідомленнями, що робить WebSocket ефективним для реального часу, наприклад чатів, онлайн-ігор або фінансових котирувань.

3. Підтримка стану з’єднання.

- З’єднання WebSocket залишається відкритим, поки одна зі сторін не ініціює його закриття.
- Протокол підтримує спеціальні фрейми ping/pong, щоб перевіряти, чи все ще активне з’єднання.

4. Закриття з’єднання

- Будь-яка зі сторін може ініціювати закриття з’єднання, надіславши Close frame.
- Інша сторона відповідає Close frame, після чого TCP-з’єднання закривається. Після закриття з’єднання обмін повідомленнями припиняється.

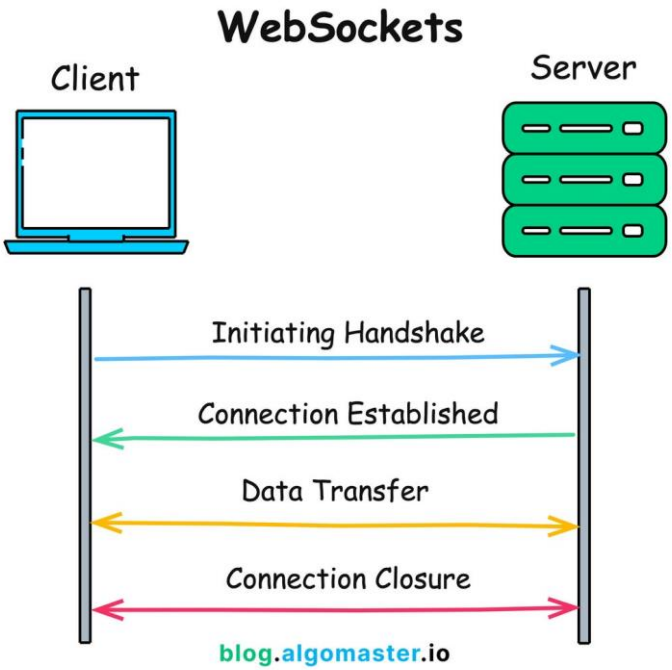

Рис. 3.2 Поетапна робота WebSocket

Такий сервер забезпечує мінімальні затримки, двосторонній реальний час обміну, приватність даних та простоту налагодження - саме ті характеристики, що роблять його найкращим вибором для реалізації описаної в роботі системи.

Код сервера на базі Python websockets .

```
import asyncio
import websockets
import json
import os
import subprocess

async def handler(websocket):
    print("🔗 Client connected")

    try:
        async for message in websocket:
            print(f"📨 Отримано запит: {message}")

            # message очікується як номер (рядок або JSON)
            try:
                request = json.loads(message)
                number = request.get("data")[0]
```

```
                msg_type=request.get("type")[0]
            except json.JSONDecodeError:
                # якщо прийшов просто рядок
                number = message
                msg_type="0"



            BASE_DIR = os.path.dirname(__file__)
            if(msg_type=="1"):
                print(f"!!! Запуск скрипта 1 !!! {msg_type}  {number}")
                main_file = os.path.join(BASE_DIR, "main.py")
            else:
                print(f"!!! Запуск скрипта 2 !!! {msg_type}  {number}")
                main_file = os.path.join(BASE_DIR, "train.py")

            result = subprocess.run(["python", main_file, number],
capture_output=True, text=True,encoding="utf-8")

            if(result.stderr):
                print(f"!!!нАДСИЛАННЯ ... !!!")
                filename = os.path.join(BASE_DIR, f"{number}.json")

                if os.path.exists(filename):
                    data = json.load(open(filename, "r", encoding="utf-8"))
                    data["filename"] = f"{number}.json"
                    await websocket.send(json.dumps(data))
                    print(f"📤 Відправлено файл: {filename}")

                else:
                    data = {"error": f"Файл {filename} не знайдено."}

            else: print(result.stderr)


    except websockets.exceptions.ConnectionClosed:
        print("❌ Client disconnected")
    finally:
        print("❌ З'єднання закрито")
```

```
async def main():
    async with websockets.serve(handler, "localhost", 8765):
        print("🚀 WebSocket server started at ws://localhost:8765")
        await asyncio.Future()

if __name__ == "__main__":
    asyncio.run(main())
```

Цей код створює WebSocket-сервер, який уміє приймати повідомлення від клієнта та виконувати Python-скрипти залежно від отриманих даних. Після запуску сервер відкриває асинхронний слухач на порту 8765 і чекає на підключення клієнтів.

Коли новий клієнт під’єднується, запускається функція handler, яка одразу виводить повідомлення про встановлення з’єднання. Усередині цієї функції сервер переходить у режим постійного прослуховування вхідних повідомлень.

Кожне повідомлення друкується в консолі, що дає можливість відстежувати роботу сервера в реальному часі. Далі сервер намагається інтерпретувати повідомлення як JSON. Якщо розбір вдається, він отримує два параметри: номер та тип запиту. Саме ці значення визначають, який із наявних скриптів слід запустити. Якщо ж JSON розібрати не вдалося, повідомлення сприймається як проста строка, а тип за замовчуванням встановлюється у “0”.

Такий механізм забезпечує надійність роботи сервера навіть у випадках, коли надходять некоректні або частково пошкоджені дані. Після визначення типу запиту сервер обирає, який саме скрипт потрібно запускати, формуючи шлях або до main.py, або до train.py. Логіка розподілу проста: якщо тип дорівнює “1”, викликається main.py - це модуль, який працює з уже завчасно навченою моделлю. Він отримує вікно даних, передає їх моделі та оперативно повертає готовий прогноз, оскільки не виконує важких обчислень і не змінює ваги нейромережі. Якщо ж тип інший, запускається train.py, що відповідає за повторне навчання моделі. Він завантажує поточну конфігурацію нейронної мережі,

коригує її ваги на основі нових даних і оновлює модель для підвищення точності майбутніх прогнозів. Таким чином сервер гнучко перемикається між швидким отриманням результату та повним оновленням моделі.

Для запуску скрипта використовується subprocess.run, у який передається номер як аргумент командного рядка. Вивід скрипта зберігається, щоб сервер міг пізніше перевірити наявність помилок. Кодування встановлено в UTF-8, що запобігає проблемам з українськими чи іншими Unicode-символами. Після завершення роботи скрипта сервер перевіряє, чи є щось у потоці помилок. Якщо помилки є, він переходить до спроби надіслати клієнту JSON-файл, пов'язаний із номером запиту.

У каталозі сервера він шукає файл з назвою <number>.json. Якщо файл існує, його вміст зчитується, до нього додається поле з назвою файлу, після чого ці дані миттєво надсилаються клієнту. Якщо ж файлу немає, сервер формує об’єкт із текстом помилки про його відсутність. Якщо ж скрипт не сформував жодних помилок, сервер просто друкує порожній потік stderr, що допомагає розуміти поточну логіку програми.

Після обробки одного повідомлення сервер залишається в циклі й готовий приймати наступні запити без розриву з’єднання. Якщо клієнт раптово закриває WebSocket, виникає виняток, який обробляється, і сервер повідомляє про відключення. Після цього з’єднання коректно закривається.

Функція main відповідає за створення та запуск WebSocket-сервера. При запуску програми сервер ініціалізується і переходить у нескінченний режим очікування через нерозв’язану asyncio.Future. Таким чином, сервер працює постійно, доки його не завершити примусово.

У підсумку програма реалізує легкий та асинхронний сервер, здатний приймати команди, запускати скрипти та повертати результати. Такий підхід є практичним для браузерних розширень, автоматизації або систем обробки даних. Структура коду добре організована, що полегшує підтримку та подальше розширення функціоналу.

3.3 Опис коду нейромережі.

У попередніх розділах було сформовано архітектуру моделі гібридного типу та визначено оптимальні підходи до її навчання й підвищення точності прогнозування. Спираючись на отримані результати моделювання, було розроблено програмну реалізацію нейромережі, яка забезпечує обробку часових рядів та формування короткострокового прогнозу вартості криптовалюти. У цьому розділі зосереджується увага не лише на теоретичних аспектах, а й на безпосередньому розгляді структури коду, що лежить в основі системи.

В дданому розділі поданий детельний описи ключових скриптів, які забезпечують роботу нейромережі.

Зокрема розглянуто:

main.py – основний скрипт для швидкого прогнозування, що служить "вхідною точкою" для отримання результатів без повторного навчання моделі.

```
import time
import numpy as np
import os
import sys
import matplotlib.pyplot as plt
import pandas as pd
import json
import joblib
import subprocess
from tensorflow.keras.losses import Huber # type: ignore
from sklearn.preprocessing import MinMaxScaler # type: ignore
from tensorflow.keras.models import Sequential # type: ignore
from tensorflow.keras.layers import Conv1D, MaxPooling1D, LSTM, Dense, Dropout #
type: ignore
from tensorflow.keras.callbacks import ModelCheckpoint # type: ignore
from tensorflow.keras.models import load_model # type: ignore
from tensorflow.keras.metrics import MeanSquaredError # type: ignore
from tensorflow.keras.optimizers import Nadam, RMSprop,AdamW, SGD  # type: ignore
from tensorflow.keras.models import Sequential, load_model # type: ignore
from tensorflow.keras.layers import Conv1D, MaxPooling1D, LSTM, Dense, Dropout,
BatchNormalization # type: ignore
```

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau # type: ignore

if(len(sys.argv)>1):
    name  = sys.argv[1]

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
download_file = os.path.join(BASE_DIR, "My_scrypt.py")
result = subprocess.run(["python", download_file], capture_output=True,
text=True,encoding="utf-8")

if(os.path.exists(os.path.join(BASE_DIR, f"{name}.csv"))):
    df = pd.read_csv(os.path.join(BASE_DIR, f"{name}.csv"))
else:
    download_file = os.path.join(BASE_DIR, "My_scrypt.py")
    result = subprocess.run(["python", download_file], capture_output=True,
text=True,encoding="utf-8")
    df = pd.read_csv(os.path.join(BASE_DIR, f"{name}.csv"))


df = df.drop(columns=["timestamp"])
df = df[:-1]
df["to_bit"] = (df["close"] - df["open"] > 0).astype(int)
cols = list(df.columns)
cols.insert(4, cols.pop(-1))
df = df[cols]
def add_indicators(df):
    # Щоб уникнути зсувів, копіюємо
    df = df.copy()

    # EMA короткі
    df['ema_5'] = df['close'].ewm(span=5, adjust=False).mean()
    df['ema_10'] = df['close'].ewm(span=10, adjust=False).mean()

    # SMA
    df['sma_20'] = df['close'].rolling(window=20).mean()

    # MACD та сигнал
    ema12 = df['close'].ewm(span=12, adjust=False).mean()
    ema26 = df['close'].ewm(span=26, adjust=False).mean()
```

```
    df['macd'] = ema12 - ema26
    df['macd_signal'] = df['macd'].ewm(span=9, adjust=False).mean()

    # RSI (використуємо “wilders” метод)
    delta = df['close'].diff()
    up = np.where(delta > 0, delta, 0)
    down = np.where(delta < 0, -delta, 0)
    period = 7  # короткий для скальпінгу
    # усереднення “wilders”
    avg_up = pd.Series(up).rolling(window=period).mean()
    avg_down = pd.Series(down).rolling(window=period).mean()
    rs = avg_up / (avg_down + 1e-9)
    df['rsi_7'] = 100 - (100 / (1 + rs))

    # Stochastic %K, %D
    low_min = df['low'].rolling(window=14).min()
    high_max = df['high'].rolling(window=14).max()
    df['stoch_k'] = 100 * ((df['close'] - low_min) / (high_max - low_min + 1e-9))
    df['stoch_d'] = df['stoch_k'].rolling(window=3).mean()

    # Bollinger Bands (напр. SMA20 +/- 2 std)
    df['bb_sma'] = df['close'].rolling(window=20).mean()
    df['bb_std'] = df['close'].rolling(window=20).std()
    df['bb_upper'] = df['bb_sma'] + 2 * df['bb_std']
    df['bb_lower'] = df['bb_sma'] - 2 * df['bb_std']

    # ATR (average true range)
    df['tr1'] = df['high'] - df['low']
    df['tr2'] = np.abs(df['high'] - df['close'].shift(1))
    df['tr3'] = np.abs(df['low'] - df['close'].shift(1))
    df['true_range'] = df[['tr1', 'tr2', 'tr3']].max(axis=1)
    df['atr_14'] = df['true_range'].rolling(window=14).mean()

    # Видалити проміжні стовпці, які були допоміжні
    df = df.drop(columns=['tr1', 'tr2', 'tr3', 'true_range', 'bb_std'])

    return df
df2 = add_indicators(df)
df2 = df2.dropna().reset_index(drop=True)
# Завантажуємо скейлери
```

```
while True:
    try:
        scaler_small =
joblib.load(os.path.join(BASE_DIR,f"{name}scaler_small.pkl"))
        scaler_large =
joblib.load(os.path.join(BASE_DIR,f"{name}scaler_large.pkl"))
        scaled_indicators = []
        for i, col in enumerate(df2.columns[6:], start=7):
            scaler_name = f'{name}scaler_{i}'
            scaler = joblib.load(os.path.join(BASE_DIR,f"{scaler_name}.pkl"))
            df2[col] = scaler.transform(df2[[col]])
            scaled_indicators.append(df2[[col]])
        data_small = scaler_small.transform(df2.iloc[:, 0:4])
        data_binary = df2.iloc[:, 4:5]
        data_large = scaler_large.transform(df2.iloc[:, 5:6])
        data = np.hstack([data_small, data_large, data_binary]+scaled_indicators)
        break
    except FileNotFoundError:
        global_file = os.path.join(BASE_DIR, "global.py")
        result = subprocess.run(["python", global_file, name], capture_output=True,
text=True,encoding="utf-8")
        time.sleep(5)



window_size = 20
X, y = [], []

for i in range(len(data) - window_size):
    X.append(data[i:i+window_size])   # вікно
    y.append(data[i+window_size])     # цільовий рядок

X = np.array(X)  # (кількість_вікон, 20, 6)
y = np.array(y)[:, 0:5]  # (кількість_вікон, 5)



model = load_model(os.path.join(BASE_DIR,f"{name}best_model.h5"),
custom_objects={"mse": MeanSquaredError})
```

```
last_window_interest = data[-21:-1]
last_window_interest = np.expand_dims(last_window_interest, axis=0)
predicted_interest =model.predict(last_window_interest)
interest_small=scaler_small.inverse_transform(predicted_interest[:, 0:4])
last_real_norm = data[-1, 0:4].reshape(1, -1)
last_real = scaler_small.inverse_transform(last_real_norm)[0]
insert_numbers=[]
for i in range(4):
    insert_numbers.append(((float(interest_small[0][i])-
float(last_real[i]))/float(last_real[i]))*100)



# Беремо останнє вікно (20 рядків, якщо так вчилась модель)
last_window = data[-20:]
last_window = np.expand_dims(last_window, axis=0)

# Прогноз у нормалізованому масштабі
predicted_norm = model.predict(last_window)

# --- Відновлюємо значення ---
# Перші 4 стовпці -> через scaler_small
pred_small = scaler_small.inverse_transform(predicted_norm[:, 0:4])


# Об’єднуємо назад
insert_numbers = np.array(insert_numbers).reshape(1, -1)
predicted_original = np.hstack([pred_small, insert_numbers])

# Вивід
columns = ["open", "max", "min", "end","open_i", "max_i", "min_i", "end_i"]

# створюємо словник {ключ: значення}
data = {columns[i]: float(predicted_original[0][i]) for i in range(len(columns))}

# записуємо у файл
with open(os.path.join(BASE_DIR,f"{name}.json"), "w", encoding="utf-8") as f:
    json.dump(data, f, ensure_ascii=False, indent=4)

print(1)
```

Даний скрипт реалізує повний цикл обробки фінансових даних, підготовки ознак, масштабування, завантаження навченої нейронної мережі та формування прогнозу на основі останнього доступного фрагмента цінової історії.

На самому початку програма отримує від системи аргумент командного рядка, який ідентифікує поточний торговий актив. Це дозволяє універсально працювати з будь-яким тікером або набором історичних котирувань, запускаючи одну й ту ж логіку для різних файлів.

Після цього визначається робоча директорія скрипта, а також виконується допоміжна програма, яка відповідає за завантаження актуальних даних, якщо вони відсутні або застарілі. Такий підхід гарантує, що перед початком обчислень модель отримає свіжі та коректно сформовані дані.

Далі код перевіряє наявність CSV-файлу з даними. Якщо файл існує, він завантажується у вигляді таблиці. Якщо ж його немає, скрипт повторно запускає процедуру завантаження зовнішніх даних, після чого файл читається повторно. Це забезпечує автономність та відмовостійкість процесу прогнозування.

Після завантаження таблиця очищається: видаляється стовпець timestamp, а останній рядок відкидається, оскільки він містить неповні дані. У таблицю додається бінарний індикатор to_bit, який визначає, чи закрився актив вище за ціну відкриття.

Реорганізація стовпців забезпечує правильний порядок ознак, що буде узгоджений з майбутнім масштабуванням та структурою нейромережі. На наступному етапі запускається функція генерації технічних індикаторів. Вона додає до набору даних такі показники, як EMA, SMA, MACD, RSI, стохастичний осцилятор, смуги Болліджера та ATR. Ці індикатори є ключовими у фінансовому аналізі, оскільки розкривають імпульс, тренд, волатильність і потенційні точки розвороту ринку.

Після обчислення допоміжні проміжні колонки видаляються, а набір ознак стає чистим і узгодженим. Таблиця проходить очищення від відсутніх даних та оновлює індексацію, що гарантує безперервність часової послідовності.

Наступний критичний блок відповідає за завантаження масштабаторів MinMaxScaler, які були попередньо збережені під час навчання моделі. Скрипт постійно перевіряє наявність всіх потрібних scaler-файлів, і якщо якогось бракує, запускає global.py, що генерує модель з усіма відповідними та потрібними скалерів.

Поступово всі стовпці проходять масштабування, кожен із використанням свого унікального scaler-файлу. Такий рівень деталізації гарантує точне відтворення умов, при яких модель навчалася. Після цього формується остаточний масив data, який містить як нормалізовані значення основних цінових стовпців, так і всі нормалізовані індикатори.

Далі формується вибірка у форматі, придатному для рекурентної моделі. Дані нарізаються на перекриваючі вікна фіксованої довжини - 20 періодів. Для кожного вікна формується відповідний цільовий рядок. У підсумку формується тривимірний масив X, який згодовуватиметься мережі.

Модель завантажується з файлу best_model.h5. Це навчена нейромережа, яка включає у себе шари LSTM, згорткові шари, нормалізацію та інші компоненти. Завдяки параметру custom_objects скрипт може коректно інтерпретувати користувацькі функції лосів або метрик, які були використані під час навчання. Коли модель завантажена, скрипт формує останнє вікно даних, яке представляє собою найновіші 20 спостережень. Воно передається в модель, і та генерує прогноз у нормалізованому просторі.

Отримані значення інвертуються через, щоб повернути їх у вихідний масштаб цін. Далі скрипт додатково обчислює відсоткові зміни між прогнозованими та останніми реальними значеннями. Так формується масив insert_numbers, який містить прогноз динаміки руху ціни у відсотках. Після цього скрипт з’єднує прогнозовані значення та відсоткові зміни у єдину таблицю. Для

зручності результат трансформується у словник, де кожен ключ відповідає конкретному прогнозованому параметру: open, max, min, end та їх процентним змінним версіям.

Це дозволяє легко інтегрувати результат у зовнішні системи, особливо у WebSocket-сервер, що викликає даний скрипт. У фіналі результат записується у JSON-файл з іменем активу, що дає можливість клієнтським застосункам швидко отримати готовий структурований прогноз. Завершується робота скрипта виведенням числа 1, яке використовується як сигнал успішного виконання.

train.py – модуль, відповідальний за перенавчання моделі на оновлених даних, що дає змогу адаптувати мережу до актуальних умов ринку;

```
import time
import numpy as np
import os
import sys
import matplotlib.pyplot as plt
import pandas as pd
import json
import joblib
import subprocess
from tensorflow.keras.losses import Huber # type: ignore
from sklearn.preprocessing import MinMaxScaler # type: ignore
from tensorflow.keras.models import Sequential # type: ignore
from tensorflow.keras.layers import Conv1D, MaxPooling1D, LSTM, Dense, Dropout #
type: ignore
from tensorflow.keras.callbacks import ModelCheckpoint # type: ignore
from tensorflow.keras.models import load_model # type: ignore
from tensorflow.keras.metrics import MeanSquaredError # type: ignore
from tensorflow.keras.optimizers import Nadam, RMSprop,AdamW, SGD  # type: ignore
from tensorflow.keras.models import Sequential, load_model # type: ignore
from tensorflow.keras.layers import Conv1D, MaxPooling1D, LSTM, Dense, Dropout,
BatchNormalization # type: ignore
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau # type: ignore

if(len(sys.argv)>1):
```

```
    name  = sys.argv[1]

BASE_DIR = os.path.dirname(__file__)
download_file = os.path.join(BASE_DIR, "My_scrypt.py")
result = subprocess.run(["python", download_file], capture_output=True,
text=True,encoding="utf-8")

if(os.path.exists(os.path.join(BASE_DIR, f"{name}.csv"))):
    df = pd.read_csv(os.path.join(BASE_DIR, f"{name}.csv"))
else:
    download_file = os.path.join(BASE_DIR, "My_scrypt.py")
    result = subprocess.run(["python", download_file], capture_output=True,
text=True,encoding="utf-8")
    df = pd.read_csv(os.path.join(BASE_DIR, f"{name}.csv"))


df = df.drop(columns=["timestamp"])
df = df[:-1]
df["to_bit"] = (df["close"] - df["open"] > 0).astype(int)
cols = list(df.columns)
cols.insert(4, cols.pop(-1))
df = df[cols]
def add_indicators(df):
    # Щоб уникнути зсувів, копіюємо
    df = df.copy()

    # EMA короткі
    df['ema_5'] = df['close'].ewm(span=5, adjust=False).mean()
    df['ema_10'] = df['close'].ewm(span=10, adjust=False).mean()

    # SMA
    df['sma_20'] = df['close'].rolling(window=20).mean()

    # MACD та сигнал
    ema12 = df['close'].ewm(span=12, adjust=False).mean()
    ema26 = df['close'].ewm(span=26, adjust=False).mean()
    df['macd'] = ema12 - ema26
    df['macd_signal'] = df['macd'].ewm(span=9, adjust=False).mean()

    # RSI (використуємо “wilders” метод)
```

```
    delta = df['close'].diff()
    up = np.where(delta > 0, delta, 0)
    down = np.where(delta < 0, -delta, 0)
    period = 7  # короткий для скальпінгу
    # усереднення “wilders”
    avg_up = pd.Series(up).rolling(window=period).mean()
    avg_down = pd.Series(down).rolling(window=period).mean()
    rs = avg_up / (avg_down + 1e-9)
    df['rsi_7'] = 100 - (100 / (1 + rs))

    # Stochastic %K, %D
    low_min = df['low'].rolling(window=14).min()
    high_max = df['high'].rolling(window=14).max()
    df['stoch_k'] = 100 * ((df['close'] - low_min) / (high_max - low_min + 1e-9))
    df['stoch_d'] = df['stoch_k'].rolling(window=3).mean()

    # Bollinger Bands (напр. SMA20 +/- 2 std)
    df['bb_sma'] = df['close'].rolling(window=20).mean()
    df['bb_std'] = df['close'].rolling(window=20).std()
    df['bb_upper'] = df['bb_sma'] + 2 * df['bb_std']
    df['bb_lower'] = df['bb_sma'] - 2 * df['bb_std']

    # ATR (average true range)
    df['tr1'] = df['high'] - df['low']
    df['tr2'] = np.abs(df['high'] - df['close'].shift(1))
    df['tr3'] = np.abs(df['low'] - df['close'].shift(1))
    df['true_range'] = df[['tr1', 'tr2', 'tr3']].max(axis=1)
    df['atr_14'] = df['true_range'].rolling(window=14).mean()

    # Видалити проміжні стовпці, які були допоміжні
    df = df.drop(columns=['tr1', 'tr2', 'tr3', 'true_range', 'bb_std'])

    return df
df2 = add_indicators(df)
df2 = df2.dropna().reset_index(drop=True)
print(df2.head())
# Завантажуємо скейлери

while True:
```

```
    if(os.path.exists(os.path.join(BASE_DIR, f"{name}scaler_small.pkl")) and
os.path.exists(os.path.join(BASE_DIR, f"{name}scaler_large.pkl"))):
        scaler_small =
joblib.load(os.path.join(BASE_DIR,f"{name}scaler_small.pkl"))
        scaler_large =
joblib.load(os.path.join(BASE_DIR,f"{name}scaler_large.pkl"))
        scaled_indicators = []
        for i, col in enumerate(df2.columns[6:], start=7):
            scaler_name = f'{name}scaler_{i}'
            scaler = joblib.load(os.path.join(BASE_DIR,f"{scaler_name}.pkl"))
            df2[col] = scaler.transform(df2[[col]])
            scaled_indicators.append(df2[[col]])
        data_small = scaler_small.transform(df2.iloc[:, 0:4])
        data_binary = df2.iloc[:, 4:5]
        data_large = scaler_large.transform(df2.iloc[:, 5:6])
        data = np.hstack([data_small, data_large, data_binary]+scaled_indicators)
        break
    else:
        global_file = os.path.join(BASE_DIR, "global.py")
        result = subprocess.run(["python", global_file, name], capture_output=True,
text=True,encoding="utf-8")
        time.sleep(5)


window_size = 20
X, y = [], []

for i in range(len(data) - window_size):
    X.append(data[i:i+window_size])   # вікно
    y.append(data[i+window_size])     # цільовий рядок

X = np.array(X)  # (кількість_вікон, 20, 6)
y = np.array(y)[:, 0:5]  # (кількість_вікон, 5)


# Найкраща модель
model = load_model(os.path.join(BASE_DIR,f"{name}best_model.h5"),
custom_objects={"mse": MeanSquaredError})
model.load_weights(os.path.join(BASE_DIR,f"{name}best_model.h5"))
# ===================== Модель ======================
```

```
model.compile(
optimizer=AdamW(learning_rate=1e-3, weight_decay=1e-4),
loss=Huber(delta=1.0),
metrics=['mae'])

checkpointA = ModelCheckpoint(
    "best_phaseA.h5",
     monitor="val_loss",
     save_best_only=True,
     mode="min",
     verbose=1)
earlyA = EarlyStopping(
    monitor="val_loss",
    patience=15,
    restore_best_weights=True,
    verbose=1)
reduceA = ReduceLROnPlateau(
    monitor="val_loss",
    factor=0.5,
    patience=10,
    min_lr=1e-7,
    verbose=1)

historyA = model.fit(X, y,
                     epochs=200,
                     batch_size=2,
                     validation_split=0.2,
                     callbacks=[checkpointA, earlyA, reduceA],
                     shuffle=False,
                     verbose=1)

# ====================== Перший прогноз ======================

# Завантажуємо найкращу модель
model = load_model("best_phaseA.h5", custom_objects={"mse": MeanSquaredError})

# Після Phase A: завантажити найкращі ваги
model.load_weights("best_phaseA.h5")

# ------- Переходимо до Phase B: Nadam -------
```

```
# Зберігаємо вага, але НЕ стан оптимізатора (несумісні класи)
model.compile(optimizer=Nadam(learning_rate=1e-5), loss=Huber(), metrics=['mae'])

checkpointB = ModelCheckpoint(f"{name}best_model.h5", monitor="val_loss",
save_best_only=True)
earlyB = EarlyStopping(monitor="val_loss", patience=15, restore_best_weights=True)
reduceB = ReduceLROnPlateau(monitor="val_loss", factor=0.5, patience=8, min_lr=1e-
8)

historyB = model.fit(X, y, epochs=40, batch_size=2, validation_split=0.2,
                     callbacks=[checkpointB, earlyB, reduceB], shuffle=False,
verbose=1)

model = load_model(os.path.join(BASE_DIR,f"{name}best_model.h5"),
custom_objects={"mse": MeanSquaredError})
last_window_interest = data[-21:-1]
last_window_interest = np.expand_dims(last_window_interest, axis=0)
predicted_interest =model.predict(last_window_interest)
interest_small=scaler_small.inverse_transform(predicted_interest[:, 0:4])
last_real_norm = data[-1, 0:4].reshape(1, -1)
last_real = scaler_small.inverse_transform(last_real_norm)[0]
insert_numbers=[]
for i in range(4):
    insert_numbers.append(((float(interest_small[0][i])-
float(last_real[i]))/float(last_real[i]))*100)


# Беремо останнє вікно (20 рядків, якщо так вчилась модель)
last_window = data[-20:]
last_window = np.expand_dims(last_window, axis=0)

# Прогноз у нормалізованому масштабі
predicted_norm = model.predict(last_window)

# --- Відновлюємо значення ---
# Перші 4 стовпці -> через scaler_small
pred_small = scaler_small.inverse_transform(predicted_norm[:, 0:4])


# Об’єднуємо назад
insert_numbers = np.array(insert_numbers).reshape(1, -1)
```

```
predicted_original = np.hstack([pred_small, insert_numbers])

# Вивід
columns = ["open", "max", "min", "end","open_i", "max_i", "min_i", "end_i"]

# створюємо словник {ключ: значення}
data = {columns[i]: float(predicted_original[0][i]) for i in range(len(columns))}

# записуємо у файл
with open(os.path.join(BASE_DIR,f"{name}.json"), "w", encoding="utf-8") as f:
    json.dump(data, f, ensure_ascii=False, indent=4)

print(1)
```

Поданий скрипт реалізує повний цикл використання нейронної мережі для прогнозування поведінки фінансового ринку, починаючи від отримання даних і закінчуючи формуванням підсумкового файлу з прогнозом.

На початку він зчитує аргумент командного рядка, який визначає, для якого торгового активу буде виконуватися обробка. Це дозволяє гнучко запускати систему для різних криптовалютних пар без зміни основної логіки коду.

Далі програма визначає розташування базової директорії, готує шлях до допоміжного скрипта і запускає його для завантаження або оновлення актуального CSV-файлу з котируваннями. Таким чином забезпечується, що подальша аналітика виконуватиметься на найсвіжіших даних.

Після отримання CSV-файлу дані завантажуються у pandas-таблицю. З них видаляється стовпець часу, що не впливає на прогноз, і останній рядок, який часто містить ще не завершену свічку та може знизити якість навчання. Далі формується бінарний індикатор напряму руху ціни, що використовується як одна з ознак моделі.

Для покращення інформативності даних застосовується окрема функція, яка обчислює технічні індикатори. Вона розширює базовий набір характеристик такими показниками, як ковзні середні, MACD, RSI, стохастики, смуги

Боллінджера та величина ринньої волатильності через ATR. Усі ці індикатори доповнюють ручні спостереження торгівців автоматизованими числовими патернами, які можуть суттєво покращити здатність моделі виявляти закономірності. Функція також видаляє тимчасові стовпці, які використовувалися лише для проміжних розрахунків.

Після цього з таблиці прибираються рядки з пропусками, що виникли через обчислення ковзних показників. Наступним кроком програма переходить до підготовки даних для нейромережі.

Всі ознаки мають бути нормалізовані раніше збереженими скейлерами. Для цього код чекає на наявність файлів зі скейлерами та у разі їх відсутності запускає глобальний скрипт, відповідальний за створення цих нормалізаторів.

Такий підхід гарантує узгодженість масштабів між різними запусками моделі та уникнення перекосу в навчанні. Після появи всіх необхідних файлів скейлери завантажуються, і кожна група ознак масштабується окремо.

Зокрема, основні ціни проходять через один скейлер, ключова велика ознака - через інший, а технічні індикатори - кожен через свій окремий нормалізатор. Потім дані зводяться у єдину матрицю, яка буде введенням для нейромережі.

На основі цієї матриці формується послідовність вікон: кожне вікно містить 20 останніх спостережень, і до нього додається рядок, який йде одразу після нього. Цей прицільний обсяг даних відтворює структуру історичного контексту, потрібного для короткострокових прогнозів.

Коли дані готові, скрипт переходить до завантаження базової версії моделі. Вона має бути збереженою у вигляді файлу найкращої конфігурації, отриманої в попередніх навчаннях. Модель компілюється з оптимізатором AdamW, використовуючи Huber-функцію як втрату та середню абсолютну похибку як метрику. Поєднання цих параметрів забезпечує збалансовану роботу мережі в початковий період навчання.

Для контролю процесу встановлюються три колбеки: збереження найкращих ваг, зупинка при відсутності покращень та зниження швидкості навчання при уповільненні прогресу. Перший етап навчання триває достатньо довго, щоб мережа змогла вирівняти ваги та навчитися базовим закономірностям ринкових рухів.

Коли цей етап завершено, найкраща модель завантажується повторно. Після цього запускається другий етап навчання, який використовує оптимізатор Nadam.

Він дозволяє точніше коригувати ваги на завершальній стадії підготовки моделі. Другий етап коротший, але більш спрямований на дрібні корекції, які можуть суттєво вплинути на фінальну точність. Результатом є новий файл із найкращими вагами, що зберігається для подальшого практичного застосування.

На завершення навчання скрипт переходить до прогнозування. Він формує вікно з останніх 20 нормалізованих рядків і пропускає його через модель, отримуючи прогноз у масштабі нормалізації. Далі проводиться зворотне перетворення першої частини прогнозу через відповідний скейлер.

Після цього обчислюються процентні зміни між очікуваними величинами та останніми реальними даними. Таке поєднання абсолютних значень та відносних змін робить прогноз більш корисним для подальшого використання. Для отримання остаточного результату скрипт формує об’єднаний масив, який включає прогнозовані ціни та обчислені відсоткові зміни. Потім створюється словник з ключами, які визначають параметри прогнозу. Цей словник зберігається у JSON-файл з назвою активу. Завершуючи роботу, скрипт виводить числовий індикатор успішного виконання, що спрощує інтеграцію з іншими модульними компонентами.

global.py – скрипт, у якому реалізовано повний цикл навчання моделі "з нуля", включно з підготовкою даних, ініціалізацією архітектури та запуском процесу оптимізації;

```
import numpy as np
```

```
import os
import sys
import matplotlib.pyplot as plt
import pandas as pd
import json
import joblib
import subprocess
from tensorflow.keras.losses import Huber # type: ignore
from sklearn.preprocessing import MinMaxScaler # type: ignore
from tensorflow.keras.models import Sequential # type: ignore
from tensorflow.keras.layers import Conv1D, MaxPooling1D, LSTM, Dense, Dropout #
type: ignore
from tensorflow.keras.callbacks import ModelCheckpoint # type: ignore
from tensorflow.keras.models import load_model # type: ignore
from tensorflow.keras.metrics import MeanSquaredError # type: ignore
from tensorflow.keras.optimizers import Nadam, RMSprop,AdamW, SGD  # type: ignore
from tensorflow.keras.models import Sequential, load_model # type: ignore
from tensorflow.keras.layers import Conv1D, MaxPooling1D, LSTM, Dense, Dropout,
BatchNormalization # type: ignore
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau # type: ignore

if(len(sys.argv)>1):
    name  = sys.argv[1]
BASE_DIR = os.path.dirname(__file__)
download_file = os.path.join(BASE_DIR, "My_scrypt.py")
result = subprocess.run(["python", download_file], capture_output=True,
text=True,encoding="utf-8")

if(os.path.exists(os.path.join(BASE_DIR, f"{name}.csv"))):
    df = pd.read_csv(os.path.join(BASE_DIR, f"{name}.csv"))
else:
    download_file = os.path.join(BASE_DIR, "My_scrypt.py")
    result = subprocess.run(["python", download_file], capture_output=True,
text=True,encoding="utf-8")
    df = pd.read_csv(os.path.join(BASE_DIR, f"{name}.csv"))

df = pd.read_csv("SOLUSDT_TEST_4h.csv")
df = df.drop(columns=["timestamp"])
df = df[:-1]
```

```
df["to_bit"] = (df["close"] - df["open"] > 0).astype(int)

cols = list(df.columns)
cols.insert(4, cols.pop(-1))  # переносимо binary на 5-й стовпець
df = df[cols]
def add_indicators(df):
    # Щоб уникнути зсувів, копіюємо
    df = df.copy()

    # EMA короткі
    df['ema_5'] = df['close'].ewm(span=5, adjust=False).mean()
    df['ema_10'] = df['close'].ewm(span=10, adjust=False).mean()

    # SMA
    df['sma_20'] = df['close'].rolling(window=20).mean()

    # MACD та сигнал
    ema12 = df['close'].ewm(span=12, adjust=False).mean()
    ema26 = df['close'].ewm(span=26, adjust=False).mean()
    df['macd'] = ema12 - ema26
    df['macd_signal'] = df['macd'].ewm(span=9, adjust=False).mean()

    # RSI (використуємо “wilders” метод)
    delta = df['close'].diff()
    up = np.where(delta > 0, delta, 0)
    down = np.where(delta < 0, -delta, 0)
    period = 7  # короткий для скальпінгу
    # усереднення “wilders”
    avg_up = pd.Series(up).rolling(window=period).mean()
    avg_down = pd.Series(down).rolling(window=period).mean()
    rs = avg_up / (avg_down + 1e-9)
    df['rsi_7'] = 100 - (100 / (1 + rs))

    # Stochastic %K, %D
    low_min = df['low'].rolling(window=14).min()
    high_max = df['high'].rolling(window=14).max()
    df['stoch_k'] = 100 * ((df['close'] - low_min) / (high_max - low_min + 1e-9))
    df['stoch_d'] = df['stoch_k'].rolling(window=3).mean()

    # Bollinger Bands (напр. SMA20 +/- 2 std)
```

```
    df['bb_sma'] = df['close'].rolling(window=20).mean()
    df['bb_std'] = df['close'].rolling(window=20).std()
    df['bb_upper'] = df['bb_sma'] + 2 * df['bb_std']
    df['bb_lower'] = df['bb_sma'] - 2 * df['bb_std']

    # ATR (average true range)
    df['tr1'] = df['high'] - df['low']
    df['tr2'] = np.abs(df['high'] - df['close'].shift(1))
    df['tr3'] = np.abs(df['low'] - df['close'].shift(1))
    df['true_range'] = df[['tr1', 'tr2', 'tr3']].max(axis=1)
    df['atr_14'] = df['true_range'].rolling(window=14).mean()

    # Видалити проміжні стовпці, які були допоміжні
    df = df.drop(columns=['tr1', 'tr2', 'tr3', 'true_range', 'bb_std'])

    return df

# Використання:
df2 = add_indicators(df)

# Видалити рядки з NaN (вже через індикатори)
df2 = df2.dropna().reset_index(drop=True)

print(df2.head())

#print("Перші 5 рядків таблиці:")
#print(df.head())

# 0-3 стовпці: нормалізуємо разом
scaler_small = MinMaxScaler()
data_small = scaler_small.fit_transform(df2.iloc[:, 0:4])
joblib.dump(scaler_small, "scaler_small.pkl")

# 4-й стовпець: бінарний
data_binary = df2.iloc[:, 4].values.reshape(-1, 1)

# 5-й стовпець: велике число, окремий скейлер
scaler_large = MinMaxScaler()

data_large = scaler_large.fit_transform(df2.iloc[:, 5:6])
```

```
joblib.dump(scaler_large, "scaler_large.pkl")

scaled_indicators = []
for i, col in enumerate(df2.columns[6:], start=7):
    scaler_name = f'scaler_{i}'
    scaler = MinMaxScaler()
    df2[col] = scaler.fit_transform(df2[[col]])
    scaled_indicators.append(df2[[col]])
    # збереження скейлера у файл
    joblib.dump(scaler, f"{scaler_name}.pkl")
    print(f"Збережено {scaler_name}.pkl")


# Збираємо назад


data = np.hstack([data_small, data_binary, data_large]+scaled_indicators)

window_size = 20
X, y = [], []

for i in range(len(data) - window_size):
    X.append(data[i:i+window_size])   # вікно
    y.append(data[i+window_size])     # цільовий рядок

X = np.array(X)  # (кількість_вікон, 20, 6)
y = np.array(y)[:, 0:5]  # (кількість_вікон, 5)

print("X shape:", X.shape)
print("y shape:", y.shape)


# ====================== Модель ======================
def create_model():
    model = Sequential([
        Conv1D(64, 3, activation='relu', input_shape=(window_size, 18)),
        BatchNormalization(), Dropout(0.5),
        Conv1D(128, 3, activation='relu'),
        BatchNormalization(), MaxPooling1D(2), Dropout(0.5),
        LSTM(256, return_sequences=True), Dropout(0.5),
        LSTM(128), Dropout(0.5),
```

```
        Dense(64, activation='relu'),
        Dense(5)
    ])
    return model
# ------- Фаза A: RAdam -------
model = create_model()

model.compile(
optimizer=AdamW(learning_rate=1e-3, weight_decay=1e-4),
loss=Huber(delta=1.0),
metrics=['mae'])

checkpointA = ModelCheckpoint(
    "best_phaseA.h5",
     monitor="val_loss",
     save_best_only=True,
     mode="min",
     verbose=1)
earlyA = EarlyStopping(
    monitor="val_loss",
    patience=15,
    restore_best_weights=True,
    verbose=1)
reduceA = ReduceLROnPlateau(
    monitor="val_loss",
    factor=0.5,
    patience=10,
    min_lr=1e-7,
    verbose=1)

historyA = model.fit(X, y,
                     epochs=200,
                     batch_size=2,
                     validation_split=0.2,
                     callbacks=[checkpointA, earlyA, reduceA],
                     shuffle=False,
                     verbose=1)

# ===================== Перший прогноз =====================
```

```
# Завантажуємо найкращу модель
model = load_model("best_phaseA.h5", custom_objects={"mse": MeanSquaredError})

# Після Phase A: завантажити найкращі ваги
model.load_weights("best_phaseA.h5")

# ------- Переходимо до Phase B: Nadam -------
# Зберігаємо вага, але НЕ стан оптимізатора (несумісні класи)
model.compile(optimizer=Nadam(learning_rate=1e-5), loss=Huber(), metrics=['mae'])

checkpointB = ModelCheckpoint(f"{name}best_model.h5", monitor="val_loss",
save_best_only=True)
earlyB = EarlyStopping(monitor="val_loss", patience=15, restore_best_weights=True)
reduceB = ReduceLROnPlateau(monitor="val_loss", factor=0.5, patience=8, min_lr=1e-
8)

historyB = model.fit(X, y, epochs=40, batch_size=2, validation_split=0.2,
                     callbacks=[checkpointB, earlyB, reduceB], shuffle=False,
verbose=1)
```

Цей програмний модуль виконує повний цикл підготовки даних, розширення датасету індикаторами та навчання глибокої нейромережі для прогнозування криптовалютних ринків.

На самому початку він імпортує численні бібліотеки, необхідні для роботи з масивами даних, побудови моделі, масштабування параметрів і автоматичного керування процесом тренування. Далі скрипт перевіряє параметри командного рядка й за потреби викликає допоміжний файл, який відповідає за підготовку вхідного CSV-файлу.

Після цього програма завантажує датасет, видаляє стовпець із часовою позначкою та створює бінарний маркер, що відображає напрямок руху ціни. Окремий блок коду формує правильне розташування стовпців у таблиці, що спрощує подальшу нормалізацію. Важливою частиною є функція додавання технічних індикаторів, яка розширює дані середніми ковзними, осциляторами, смугами Болліджера та іншими статистичними характеристиками.

Після обчислення індикаторів скрипт очищує дані від пропусків, щоб забезпечити коректне навчання. Далі починається масштабування: різні групи ознак нормалізуються окремими скейлерами, оскільки мають різну природу та різні діапазони значень.

Кожен скейлер зберігається у файл для подальшого використання під час прогнозування. Після цього дані об'єднуються в єдину матрицю, що містить як цінові значення, так і технічні індикатори. Створюються вікна даних фіксованого розміру, які використовуються як вхід нейромережі, та формується цільовий вектор для навчання.

Далі програма визначає архітектуру моделі, що поєднує згорткові шари, LSTM-блоки й повнозв'язні шари для комплексного аналізу часових рядів. У першій фазі тренування модель оптимізується за допомогою AdamW та втрати Huber, а процес супроводжується ранньою зупинкою та динамічним зменшенням швидкості навчання.

Після завершення першого етапу скрипт завантажує найкращу модель та переходить до другої фази донавчання. На цьому етапі оптимізатор змінюється на Nadam, який може покращити збіжність і точність на фінішних ітераціях. Важливо, що ваги залишаються сталими, але стан оптимізатора перезапускається, оскільки різні алгоритми не є сумісними між собою.

У другій фазі модель донавчається з меншою швидкістю, що дозволяє тонко підлаштувати ваги без ризику переосмислення патернів. У результаті формується файл моделі з найкращими параметрами, придатний для подальшого використання в системі прогнозування. Таким чином, скрипт реалізує не лише повний конвеєр підготовки даних, а й двофазну стратегію навчання, спрямовану на максимальну точність прогнозів.

My_scrypt – допоміжний модуль, призначений для обробки та завантаження нових даних, що надходять від розширення браузера. У рамках цього розділу кожен зі скриптів буде детально проаналізований, а їхня функціональна роль у системі - чітко окреслена. Це дозволить повністю

зрозуміти взаємодію між компонентами, а також загальну логіку роботи програмної частини розробленої моделі прогнозування.

```
import pandas as pd
import requests
import time
import os

def fetch_klines(symbol, interval, limit=1000):
    url = "https://fapi.binance.com/fapi/v1/klines"
    params = {
        "symbol": symbol,
        "interval": interval,
        "limit": limit
    }
    response = requests.get(url, params=params)
    data = response.json()

    df = pd.DataFrame(data, columns=[
        "timestamp", "open", "high", "low", "close", "volume", "close_time",
        "quote_asset_volume", "number_of_trades", "taker_buy_base",
"taker_buy_quote", "ignore"
    ])

    df["timestamp"] = pd.to_datetime(df["timestamp"], unit='ms')
    df = df[["timestamp", "open", "high", "low", "close", "volume"]]
    return df
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
def save_15m_to_csv(symbol):
    df = fetch_klines(symbol, "15m")
    filename = os.path.join(BASE_DIR,f"{symbol}_text_15m_button.csv")
    df.to_csv(filename, index=False)

def save_1h_to_csv(symbol):
    df = fetch_klines(symbol, "1h")
    filename = os.path.join(BASE_DIR,f"{symbol}_text_1h_button.csv")
    df.to_csv(filename, index=False)


def save_4h_to_csv(symbol):
```

```
    df = fetch_klines(symbol, "4h")
    filename = os.path.join(BASE_DIR,f"{symbol}_text_4h_button.csv")
    df.to_csv(filename, index=False)


def save_1d_to_csv(symbol):
    df = fetch_klines(symbol, "1d")
    filename = os.path.join(BASE_DIR,f"{symbol}_text_1d_button.csv")
    df.to_csv(filename, index=False)


symbols = ["ETHUSDT", "SOLUSDT","BTCUSDT"]
for symbol in symbols:
    save_15m_to_csv(symbol)
    time.sleep(3)
    save_1h_to_csv(symbol)
    time.sleep(3)
    save_4h_to_csv(symbol)
    time.sleep(3)
    save_1d_to_csv(symbol)
    time.sleep(3)
```

Цей програмний фрагмент реалізує повний цикл автоматизованого завантаження біржових даних із сервера Binance та збереження їх у формат CSV.

На самому початку імпортуються необхідні бібліотеки, що забезпечують роботу з таблицями, виконання HTTP-запитів, затримок між операціями та файловою системою.

Основна функція, яка відповідає за отримання даних, називається fetch_klines і виконує запит до API Binance по заданому торговому символу та часовому інтервалу. Вона формує словник параметрів, що включає пару активів, таймфрейм та кількість повернутих рядків.

Після виконання запиту функція перетворює відповідь сервера на список і одразу будує з нього DataFrame. Отримана таблиця містить усі стандартні поля японських свічок, проте частина з них не використовується в подальшій роботі.

Тому після приведення часу до зрозумілого формату мілісекунди перетворюються на datetime-об’єкти. Таблиця очищується від зайвих стовпців, щоб у результаті залишити лише ключові цінові та об’ємні характеристики. Підготовлений DataFrame повертається у викликаючу частину коду й може бути збережений у файл. Далі визначається базова директорія виконання скрипта, яка допомагає створювати файли незалежно від того, звідки запускається програма. Наступні чотири функції виконують подібну структуру: кожна з них викликає fetch_klines з певним інтервалом, а потім зберігає отримані дані у CSV-файл. Кожен із цих допоміжних методів має у назві часовий інтервал, що значно спрощує використання коду.

Формат імені файлу сформований так, щоб чітко ідентифікувати актив та таймфрейм. Такий підхід дозволяє створювати різні набори даних для аналізу без ризику перезапису попередніх результатів.

Головна частина коду містить список символів, для яких потрібно виконати завантаження. Кожен символ проходить через послідовність викликів функцій, що формують таймфрейми 15 хвилин, 1 годину, 4 години та 1 день. Після кожного звернення до API вставляється пауза, що захищає програму від перевищення лімітів Binance та можливих помилок через занадто часті запити.

Ці штучні затримки також дозволяють отримувати дані стабільно та без зайвого навантаження на сервер. Завдяки покроковій структурі скрипт має добру читабельність та легко розширюється за потреби. Наприклад, можна швидко додати нові таймфрейми або ж додати інші криптовалютні активи.

Важливо, що дані доступні у вигляді CSV-файлів, що робить їх сумісними з інструментом аналізу складною нейромережевою моделею. Автоматизація процесу завантаження знімає необхідність ручного збору свічкової історії, що суттєво економить час під час дослідження ринків.

Крім того, скрипт легко можна інтегрувати у більшу систему або запускати з інтервалом через планувальники. Завдяки модульності він дає можливість використовувати лише частину логіки без переписування всього коду. Загалом цей код є прикладом практичного, компактного та надійного інструмента для формування локального архіву ринкових даних. Він виконує своє завдання без зайвих дій та демонструє правильний підхід до взаємодії з API біржових платформ. Саме така структура дозволяє в майбутньому масштабувати його без значних змін.

3.4 Опис коду розширення.

У попередніх підрозділах було розглянуто загальні принципи побудови серверної частини та особливості взаємодії між нейромережею та браузерним середовищем. Тепер варто перейти безпосередньо до аналізу коду розширення, яке забезпечує зручний доступ користувача до можливостей прогнозування, формує інтерфейс взаємодії та відповідає за передачу даних між браузером і локальним сервером.

Розширення має власну архітектуру рис. 3.3 та виконує роль "клієнтської оболонки", що поєднує декілька важливих компонентів: інтерфейс користувача рис. 3.4, фонові процеси, обмін повідомленнями через WebSocket та обробку отриманих даних.

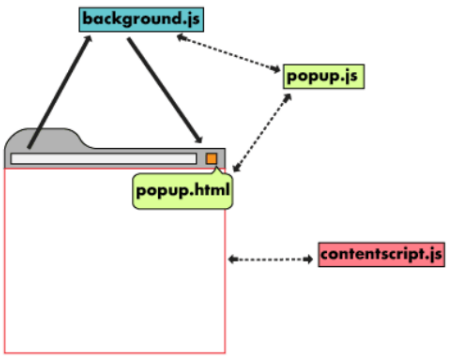

Рис. 3.3 Архітектура розширення

Саме воно виступає проміжною ланкою між користувачем і нейромережею, забезпечуючи автоматичне надсилання запитів, отримання результатів прогнозу та оновлення інформації на вебсторінці.

Далі буде детально описано структуру коду розширення, логіку роботи основних скриптів, механізми зв’язку з сервером, а також функції, які відповідають за збір і передачу даних. Це дозволить краще зрозуміти внутрішню організацію розширення та роль кожного файла в загальному функціонуванні системи прогнозування.

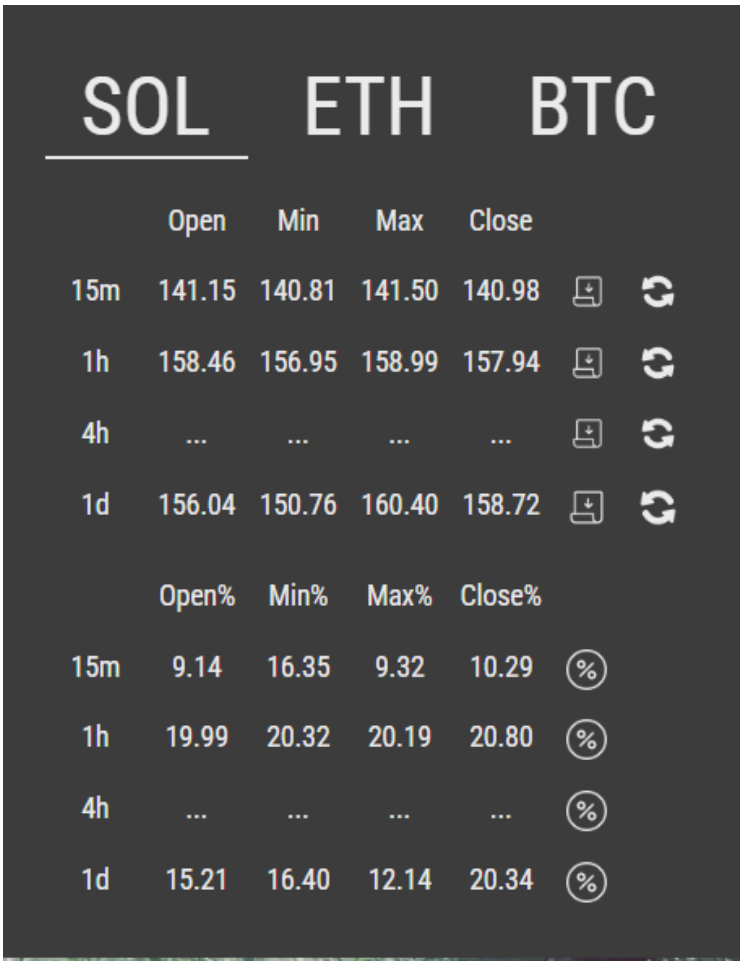

Рис. 3.4 Інтерфейс користувача

Як показано на рис. 3.4, розширення містить три окремі вкладки, кожна з яких відповідає певній криптовалюті. У межах вибраної вкладки користувач бачить поточні ціни, подані у доларах США за одну одиницю активу. Праворуч розташовані кнопки оновлення прогнозу для різних часових інтервалів - 15 хвилин, 1 година, 4 години та 1 день.

Це дає змогу отримувати прогнози з різною глибиною залежно від потреб користувача. Нейромережа, інтегрована в систему, визначає чотири ключові показники: ціну відкриття, максимально можливу ціну, мінімальне значення та прогнозовану ціну закриття для вибраного часовового проміжку. Користувач має можливість як швидко оновити прогноз на основі вже навченої моделі, так і виконати перенавчання на свіжих даних у разі потреби більш актуального результату.

Після формування прогнозу у нижній частині вкладки відображається похибка у відсотках, що дає змогу оцінити якість передбачення в реальних ринкових умовах. На зображенні похибка становить приблизно 20%, що є доволі значним відхиленням. Це пояснюється тим, що ринок криптовалют характеризується високою нестабільністю: глобальні новини, політичні події або економічні зміни можуть миттєво спричиняти різкі коливання ціни. Попри це, нейромережа здатна виявляти типові закономірності, а показник похибки відіграє важливу роль у прийнятті рішення користувачем, адже дає змогу оцінити ступінь ризику.

Для реалізації свого браузерного розширення я обрав середовище розробки Visual Studio, оскільки воно забезпечує зручну роботу з HTML, CSS, JavaScript та дає змогу ефективно керувати проєктною структурою. Створене розширення відповідає стандартній архітектурі, притаманній більшості сучасних браузерних рішень, і містить набір файлів, кожен з яких виконує власну функцію в процесі обробки та відображення даних.

Структура розширення складається з таких основних елементів:

manifest.json — головний конфігураційний файл, який описує властивості розширення: його назву, версію, дозволи, доступ до ресурсів, а також список скриптів і сторінок, що використовуються. Саме через manifest браузер знає, як коректно завантажувати та виконувати розширення. Код файлу manifest.json:

```
{
  "manifest_version": 3,
```

```
  "name": "THelper",
  "version": "1.0",
  "description": "Receive live data from Python WebSocket server",
  "background": {
  "service_worker": "js/background.js"
  },
  "action": {
    "default_popup": "html/popup.html",
    "default_title": "Thelper"
  },
  "icons": {
    "16": "img/icons/16x16.png",
    "48": "img/icons/48x48.png",
    "128": "img/icons/128x128.png"
  },
  "content_scripts": [
    {
      "matches": ["<all_urls>"],
      "js": ["js/content.js"]
    }
  ],

  "permissions": ["storage", "tabs", "activeTab", "scripting"]
}
```

Даний фрагмент коду представляє собою файл manifest.json, який визначає структуру, поведінку та ключові можливості Chrome-розширення під назвою THelper. У першому блоці встановлюється версія маніфесту - число 3, яке відповідає сучасним вимогам браузера та забезпечує підвищену безпеку та ефективність роботи. Назва розширення, його версія та короткий опис дозволяють браузеру ідентифікувати програму й пояснюють користувачу її призначення - отримання потокових даних через WebSocket-сервер Python.

Далі визначається background-скрипт, який виконується як service worker і працює у фоновому режимі незалежно від відкритих вкладок. Саме він координує обмін даними, обробку запитів і зв’язок розширення з сервером.

У наступному блоці конфігурується action, тобто поведінка кнопки розширення на панелі браузера. Вказується, що при натисканні відкриватиметься спеціальне вікно popup.html, а сам елемент матиме зрозумілий підпис. Окремо задається набір піктограм різних розмірів, що забезпечує коректне відображення логотипу програми в усіх частинах інтерфейсу Chrome. Значущим елементом є секція content_scripts, у якій визначається сценарій, що буде впроваджений у всі веб-сторінки. Він дозволяє взаємодіяти з DOM, перехоплювати події та передавати інформацію між сторінками та фоновою частиною розширення.

Остання частина містить перелік дозволів, які потрібні для повноцінної роботи THelper. Дозвіл на storage відкриває доступ до локального сховища браузера, де можуть зберігатися результати або тимчасові дані. Права tabs та activeTab дозволяють працювати з інформацією про відкриті сторінки та одержувати доступ до активної вкладки під час взаємодії користувача. Дозвіл scripting надає можливість динамічно виконувати JavaScript-код у контенті сторінок, що забезпечує розширенню гнучкість і дозволяє впроваджувати складні механізми обробки подій.

Усі ці компоненти разом формують каркас Chrome-розширення, визначають взаємодію його частин і гарантують стабільну роботу в межах браузера. Manifest.json служить точкою входу, яка описує призначення, межі доступу та внутрішню структуру інструмента, що на практиці забезпечує можливість безпечного та керованого обміну даними з Python-сервером.

popup.html - Це основна HTML-сторінка, яку користувач бачить після відкриття розширення. Головне вікно, представлене на рис. 3.5, демонструє прогнозовані значення для криптовалюти Solana.

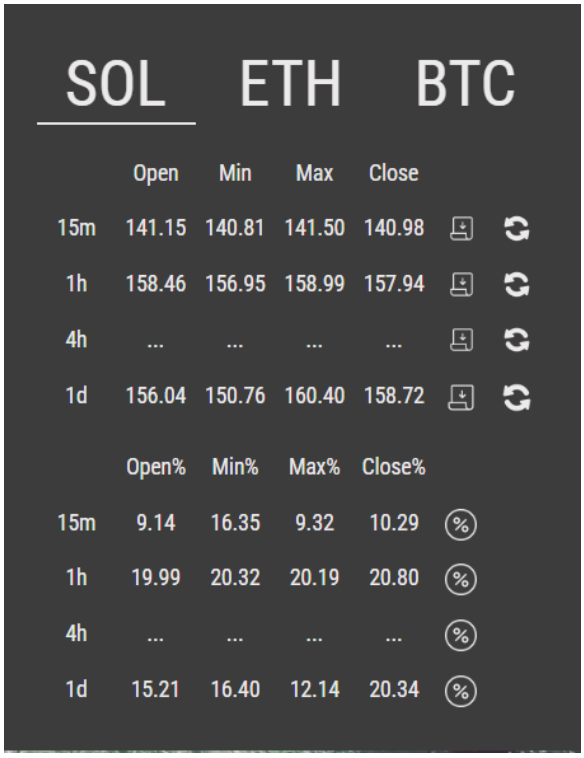

Рис. 3.5 Головне вікно розшрення

У верхній частині розміщено перелік вкладок, що дає змогу швидко перемикатися між іншими криптовалютами. Структура коду цих сторінок майже однакова, відмінності стосуються лише індексів елементів та кнопок, що дозволяє уникати помилок під час обробки даних для різних активів. Нижче наведено повний код сторінки popup.html.

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8" />
  <title>Python WebSocket Data</title>
  <link rel="stylesheet" href="/css/style.css">
  <link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Roboto+Condensed:ital,wght@0,100..900;1,100..900&display=swap" rel="stylesheet">
</head>
<body>
  <div id="all_button_top">
    <a href="popup.html"  data-page="button_SOL"  class="top_button">SOL</a>
```

```
    <a href="eth.html"  data-page="button_ETH" class="top_button">ETH</button>
    <a href="btc.html"  data-page="button_BTC"  class="top_button">BTC</a>
  </div>



  <div id="main_conteiner">
    <div class="main_conteiner_head">
      <div id="text_main_1" class="main_box">Open</div>
      <div id="text_main_2" class="main_box">Min</div>
      <div id="text_main_3" class="main_box">Max</div>
      <div id="text_main_4" class="main_box">Close</div>
    </div>
    <div id="main_conteiner_15m" class="main_conteiner_">
      <div id="text_15m" class="main_box">15m</div>
      <div id="text_15m_open" class="main_box"></div>
      <div id="text_15m_min" class="main_box"></div>
      <div id="text_15m_max" class="main_box"></div>
      <div id="text_15m_close" class="main_box"></div>
      <div id="SOLUSDT_text_15m_button" class="main_box_button">
        <img src="/img/icons/dow_w.png" style="max-width: 18px; height: auto;
align-items: center;" alt="" >
      </div>
      <div id="SOLUSDT_text_15m_button_f" class="main_box_button">
         <img src="/img/icons/restart_w.png" style="max-width: 18px; height: auto;
align-items: center;" alt="" >
      </div>
    </div>
    <div id="main_conteiner_1h" class="main_conteiner_">
      <div id="text_1h" class="main_box">1h</div>
      <div id="text_1h_open" class="main_box"></div>
      <div id="text_1h_min" class="main_box"></div>
      <div id="text_1h_max" class="main_box"></div>
      <div id="text_1h_close" class="main_box"></div>
      <div id="SOLUSDT_text_1h_button" class="main_box_button">
        <img src="/img/icons/dow_w.png" style="max-width: 18px; height: auto;
align-items: center;" alt="" >
      </div>
      <div id="SOLUSDT_text_1h_button_f" class="main_box_button">
```

```
        <img src="/img/icons/restart_w.png" style="max-width: 18px; height: auto;
align-items: center;" alt="" >
      </div>
    </div>
    <div id="main_conteiner_4h" class="main_conteiner_">
      <div id="text_4h" class="main_box">4h</div>
      <div id="text_4h_open" class="main_box"></div>
      <div id="text_4h_min" class="main_box"></div>
      <div id="text_4h_max" class="main_box"></div>
      <div id="text_4h_close" class="main_box"></div>
      <div id="SOLUSDT_text_4h_button" class="main_box_button">
        <img src="/img/icons/dow_w.png" style="max-width: 18px; height: auto;
align-items: center;" alt="" >
      </div>
      <div id="SOLUSDT_text_4h_button_f" class="main_box_button">
        <img src="/img/icons/restart_w.png" style="max-width: 18px; height: auto;
align-items: center;" alt="" >
      </div>
    </div>
    <div id="main_conteiner_1d" class="main_conteiner_">
      <div id="text_1d" class="main_box">1d</div>
      <div id="text_1d_open" class="main_box"></div>
      <div id="text_1d_min" class="main_box"></div>
      <div id="text_1d_max" class="main_box"></div>
      <div id="text_1d_close" class="main_box"></div>
      <div id="SOLUSDT_text_1d_button" class="main_box_button">
        <img src="/img/icons/dow_w.png" style="max-width: 20px; height: auto;
align-items: center;" alt="" >
      </div>
      <div id="SOLUSDT_text_1d_button_f" class="main_box_button">
        <img src="/img/icons/restart_w.png" style="max-width: 20px; height: auto;
align-items: center;" alt="" >
      </div>
    </div>
        <div class="main_conteiner_head">
      <div id="text_main_1" class="main_box">Open%</div>
      <div id="text_main_2" class="main_box">Min%</div>
      <div id="text_main_3" class="main_box">Max%</div>
      <div id="text_main_4" class="main_box">Close%</div>
        </div>
```

```
        <div id="main_conteiner_15m" class="main_conteiner_">
      <div id="text_15m_p" class="main_box">15m</div>
      <div id="text_15m_open_p" class="main_box"></div>
      <div id="text_15m_min_p" class="main_box"></div>
      <div id="text_15m_max_p" class="main_box"></div>
      <div id="text_15m_close_p" class="main_box"></div>
      <div id="SOLUSDT_text_15m_button_p" class="main_box_button">
        <img src="/img/icons/proc_w.png" style="max-width: 20px; height: auto;
align-items: center;" alt="" >
      </div>
      <div id="text_P" class="main_box_button"></div>
    </div>
    <div id="main_conteiner_1h" class="main_conteiner_">
      <div id="text_1h_p" class="main_box">1h</div>
      <div id="text_1h_open_p" class="main_box"></div>
      <div id="text_1h_min_p" class="main_box"></div>
      <div id="text_1h_max_p" class="main_box"></div>
      <div id="text_1h_close_p" class="main_box"></div>
      <div id="SOLUSDT_text_1h_button_p" class="main_box_button">
        <img src="/img/icons/proc_w.png" style="max-width: 20px; height: auto;
align-items: center;" alt="" >
      </div>
        <div id="text_P" class="main_box_button"></div>
    </div>
    <div id="main_conteiner_4h" class="main_conteiner_">
      <div id="text_4h_p" class="main_box">4h</div>
      <div id="text_4h_open_p" class="main_box"></div>
      <div id="text_4h_min_p" class="main_box"></div>
      <div id="text_4h_max_p" class="main_box"></div>
      <div id="text_4h_close_p" class="main_box"></div>

      <div id="SOLUSDT_text_4h_button_p" class="main_box_button">
        <img src="/img/icons/proc_w.png" style="max-width: 20px; height: auto;
align-items: center;" alt="" >
      </div>
      <div id="text_P" class="main_box_button"></div>
    </div>
    <div id="main_conteiner_1d" class="main_conteiner_">
      <div id="text_1d_p" class="main_box">1d</div>
      <div id="text_1d_open_p" class="main_box"></div>
```

```
      <div id="text_1d_min_p" class="main_box"></div>
      <div id="text_1d_max_p" class="main_box"></div>
      <div id="text_1d_close_p" class="main_box"></div>
      <div id="SOLUSDT_text_1d_button_p" class="main_box_button">
        <img src="/img/icons/proc_w.png" style="max-width: 20px; height: auto;
align-items: center;" alt="" >
      </div>
      <div id="text_P" class="main_box_button"></div>
    </div>
  </div>

  <script src="/js/popup.js"></script>
</body>
</html>
```

У верхній частині даного HTML-коду підключаються базові налаштування, такі як кодування UTF-8, заголовок сторінки та зовнішній CSS-файл, що відповідає за стилізацію. Додатково підключено шрифт Roboto Condensed з Google Fonts, що надає інтерфейсу сучасного, компактного вигляду.

Одразу після відкриття тіла документа формується блок верхньої навігації. Це панель із трьома кнопками-вкладками - SOL, ETH та BTC. Кожна з них відкриває окрему сторінку відповідної криптовалюти, забезпечуючи швидке перемикання між даними. Така структура робить інтерфейс максимально інтуїтивним, адже користувач одразу розуміє, дані якого активу переглядає.

Після панелі вкладок розташовується головний контейнер - центральний елемент сторінки. Перший його блок відображає заголовки колонок: Open, Min, Max і Close.

Далі йдуть секції, присвячені різним часовим пеіодам: 15m, 1h, 4h та 1d. Кожен з цих блоків складається з назви інтервалу та чотирьох пустих полів, куди в реальному часі підставлятимуться значення, отримані через WebSocket.

У кожному рядку також містяться дві кнопки-іконки: одна відповідає за завантаження прогнозу від перенвченої моделі на основі нових даних, інша - за швидке оновлення на навченій моделі з оновленими даними. Після відображення абсолютних значень повторюється аналогічна структура для відображення процентних змін.

У другому блоці також є заголовки Open%, Min%, Max% та Close%. Нижче - чотири таймфрейми з такими ж текстовими полями, у які скрипт підставляє обчислені відсоткові відхилення прогнозу від реальних значень. Кожен рядок іконку з символом відсотків, для інтуєтивного розуміння, що дані показники обраховані у відсотках. Такий підхід дозволяє легко порівнювати динаміку й оцінювати волатильність ринку за різний час.

Весь інтерфейс логічно поділений на блоки, що спрощує сприйняття та дозволяє уникати перевантаження інформацією. Кожен елемент має власний унікальний ID, завдяки чому JavaScript-код у файлі popup.js може безпомилково оновлювати значення на сторінці.

Наприкінці документа підключається скрипт popup.js, що відповідає за роботу WebSocket-з'єднання, обробку отриманих даних та реакцію на натискання кнопок. Загалом код формує повноцінний інтерактивний інтерфейс, який відображає прогнозні значення у зручному табличному форматі, дозволяє перемикатися між криптовалютами, завантажувати нові дані й аналізувати зміни у відсотках. Структура сторінки є продуманою, зрозумілою і легко адаптується під інші активи.

style.css - файл стилів для оформлення інтерфейсу. Він відповідає за візуальну складову: кольори, шрифти, розташування блоків, адаптивність. Нижче наведено повний код файлу style.css.

```
body {
  width: wrap-content;
  padding: 5px;
  background-color: #3c3c3c;
}
```

```css
#all_button_top{
  height: 75px;
  width: wrap-content;
  display: flex;
  justify-content: center;
  align-items: center;
  gap: 10px;
}
.top_button{
  border: none;
  width: 100px;
  display: inline-block;
  background-color: transparent;
  color: #eceaea;
  text-align: center;
  font-size: 40px;
  font-family: "Roboto Condensed", sans-serif;
  position: relative;
  text-decoration: none;
  transition: color 0.3s ease ;
}
.top_button::before{
  content: "";
  position: absolute;
  bottom : -2px;
  left: 0%;
  width: 100%;
  height: 2px;
  background:#eceaea;
  transform: scaleX(0);
  transition: transform 0.3s ease ;
}
.top_button:hover{
  color: #eceaea;
}
.top_button:hover::before{
  transform: scaleX(1);
}

.top_button.active::before{
```

```
  transform: scaleX(1);
}


#center_conteiner{
  margin-top: 10px;
  display: flex;
  justify-content: center;
  align-items: center;
}

.main_conteiner_head{
  margin-top: 10px;
  margin-left: 60px;
  margin-right: 80px;
  display: flex;
  text-align: center;
  justify-content: center;
  align-items: center;
}
.main_box_button{
  width: 35px;
  height: 35px;
  justify-content: center;
  align-items: center;
  text-align: center;
}
.main_box{
  width: 50px;
  height: 35px;
  font-size:14px;
  color: #eceaea;
  font-family: "Roboto Condensed", sans-serif;
  text-align: center;
  justify-content: center;
  align-items: center;
}
.main_conteiner_{
  display: flex;
  justify-content: center;
```

```
  align-items: center;
}


#dataContainer {
  margin-top: 10px;
  background: #f1f1f1;
  padding: 8px;
  border-radius: 8px;
}
#setColors, #colors {
  margin-bottom: 8px;
}

#popupContent {
  margin-top: 8px;
  font-size: 14px;
}
```

Цей фрагмент стилів формує візуальну основу інтерфейсу розширення, задаючи єдину естетику та зручність взаємодії з елементами сторінки. Стилі побудовані так, щоб створити лаконічний, контрастний та інтуїтивний дизайн, де кожний елемент має чітко визначене функціональне місце. Оформлення сторінки починається з базових параметрів для тіла документа, де задається темний фон та відступи, що створюють акуратну рамку навколо вмісту. У верхній частині розташований блок з кнопками переключення між сторінками.

Він вирівняний по центру й має адаптивну структуру: кнопки знаходяться поруч і розділені рівномірними проміжками. Це створює відчуття впорядкованості та дозволяє користувачу швидко орієнтуватися серед доступних розділів.

Кожна верхня кнопка має власний стиль - великі літери, мінімалістичність, контрастний колір і плавна анімація. За рахунок псевдоелемента підкреслення з’являється лише при наведенні, а також закріплюється для активної вкладки.

Такі дрібні анімаційні ефекти не лише роблять інтерфейс приємнішим, а й підсилюють інформативність. Основні контейнери даних також підпорядковані єдиному задуму - матеріали розміщені в ряд, вирівняні по центру й зберігають однаковий розмір.

Блоки з текстовою інформацією мають компактні розміри, чітку типографіку та світлий колір шрифту, який контрастує з темним фоном. Це створює візуальну стабільність і зручність сприйняття навіть при великій кількості числових даних.

Додаткові кнопки, розміщені поруч із рядами даних, мають квадратну форму та спеціально відведене місце для іконок - так вони стають помітними, але не відволікають від основного змісту.

У нижній частині стилів передбачено оформлення додаткових елементів, що можуть відображати службову інформацію або проміжні дані. Вони подані у світлій рамці з округленими краями, що вирізняє їх з-поміж темного фону. Такі блоки призначені для візуального згрупування аналітичної інформації й забезпечення загальної чистоти інтерфейсу.

Наведений CSS формує збалансовану структуру сторінки, яка одночасно виглядає сучасно та виконує практичну функцію, простіше кажучи робить взаємодію з прогнозними даними зручною і комфортною. Кожен стиль працює на те, щоб інтерфейс залишався логічним, впорядкованим і легко читаним, забезпечуючи приємну взаємодію користувача з розширенням.

popup.js - основний скрипт взаємодії з інтерфейсом. Він обробляє події натискань кнопок, оновлює інформацію у вкладках та встановлює зв’язок із сервером через WebSocket. Даний скрипт працює лише коли відкрите розширення тому вся логіка передачі та отримання даних від серевера перенесена до скрипту background.js. На сторінках з іншими криптовалютами є відповідні скрипти які за сруктурою подібні до цього єдиною відміністтю є різні ідекатори, для уникнення помилок підчас виведення на екран користувача. Нище наведений код скрипту popup.js.

```
document.addEventListener('DOMContentLoaded', () => {
  const button_15m = document.getElementById('SOLUSDT_text_15m_button');
  const button_1h = document.getElementById('SOLUSDT_text_1h_button');
  const button_4h = document.getElementById('SOLUSDT_text_4h_button');
  const button_1d = document.getElementById('SOLUSDT_text_1d_button');

  const button_15m_f = document.getElementById('SOLUSDT_text_15m_button_f');
  const button_1h_f = document.getElementById('SOLUSDT_text_1h_button_f');
  const button_4h_f = document.getElementById('SOLUSDT_text_4h_button_f');
  const button_1d_f = document.getElementById('SOLUSDT_text_1d_button_f');

  getStoredData();
  getProcData();

  chrome.runtime.onMessage.addListener((msg, sender, sendResponse) => {
    if (msg.command === "updateData") {
      getStoredData();
      getProcData();
    }
  });


  button_15m.addEventListener("click", () => {
    chrome.runtime.sendMessage({ type: "0", data: button_15m.id });
    console.log("📤 Відправлено повідомлення:", button_15m.id);
  });

  button_1h.addEventListener("click", () => {
    chrome.runtime.sendMessage({ type: "0", data: button_1h.id });
    console.log("📤 Відправлено повідомлення:", button_1h.id);
  });
   button_4h.addEventListener("click", () => {
    chrome.runtime.sendMessage({ type: "0", data: button_4h.id });
    console.log("📤 Відправлено повідомлення:", button_4h.id);
  });
    button_1d.addEventListener("click", () => {
    chrome.runtime.sendMessage({ type: "0", data: button_1d.id });
    console.log("📤 Відправлено повідомлення:", button_1d.id);
  });
    button_15m_f.addEventListener("click", () => {
```

```
    chrome.runtime.sendMessage({ type: "1", data: button_15m.id });
    console.log("📤 Відправлено повідомлення:", "button_15m_f");
  });
      button_1h_f.addEventListener("click", () => {
    chrome.runtime.sendMessage({ type: "1", data:button_1h.id });
    console.log("📤 Відправлено повідомлення:", "button_1h_f");
  });
        button_4h_f.addEventListener("click", () => {
    chrome.runtime.sendMessage({ type: "1", data: button_4h.id });
    console.log("📤 Відправлено повідомлення:", "button_4h_f");
  });
          button_1d_f.addEventListener("click", () => {
    chrome.runtime.sendMessage({ type: "1", data: button_1d.id });
    console.log("📤 Відправлено повідомлення:", "button_1d_f");
  });


function getStoredData() {

  chrome.storage.local.get(
  ["SOL_open_15m", "SOL_min_15m", "SOL_max_15m", "SOL_end_15m"],
  (result) => {
    document.getElementById('text_15m_open').textContent = result.SOL_open_15m ??
"-";
    document.getElementById('text_15m_min').textContent = result.SOL_min_15m ?? "-
";
    document.getElementById('text_15m_max').textContent = result.SOL_max_15m ?? "-
";
    document.getElementById('text_15m_close').textContent = result.SOL_end_15m ??
"-";
  }
  );
  chrome.storage.local.get(["SOL_open_1h","SOL_min_1h","SOL_max_1h","SOL_end_1h"],
(result) => {
    document.getElementById('text_1h_open').textContent = result.SOL_open_1h ?? "-
";
    document.getElementById('text_1h_min').textContent = result.SOL_min_1h ?? "-";
    document.getElementById('text_1h_max').textContent = result.SOL_max_1h ?? "-";
```

```
    document.getElementById('text_1h_close').textContent = result.SOL_end_1h ?? "-
";
  });

  chrome.storage.local.get(["SOL_open_4h","SOL_min_4h","SOL_max_4h","SOL_end_4h"],
(result) => {
    document.getElementById('text_4h_open').textContent = result.SOL_open_4h ?? "-
";
    document.getElementById('text_4h_min').textContent = result.SOL_min_4h ?? "-";
    document.getElementById('text_4h_max').textContent = result.SOL_max_4h ?? "-";
    document.getElementById('text_4h_close').textContent = result.SOL_end_4h ?? "-
";
  });
    chrome.storage.local.get(["SOL_open_1d","SOL_min_1d","SOL_max_1d","SOL_end_1d"]
, (result) => {
    document.getElementById('text_1d_open').textContent = result.SOL_open_1d ?? "-
";
    document.getElementById('text_1d_min').textContent = result.SOL_min_1d ?? "-";
    document.getElementById('text_1d_max').textContent = result.SOL_max_1d ?? "-";
    document.getElementById('text_1d_close').textContent = result.SOL_end_1d ?? "-
";
  });
}

const buttons = document.querySelectorAll('.top_button');

// Підсвічування після перезавантаження
let saved = localStorage.getItem('activeBtn');
if (saved) {
  document.querySelector(`.top_button[data-
page="${saved}"]`)?.classList.add('active');
}
// Клік
buttons.forEach(btn => {
  btn.addEventListener('click', () => {
    localStorage.setItem('activeBtn', btn.dataset.page);
  });
});

function getProcData() {
```

```
  chrome.storage.local.get(
  ["SOL_open_15m_p", "SOL_min_15m_p", "SOL_max_15m_p", "SOL_end_15m_p"],
  (result) => {
    document.getElementById('text_15m_open_p').textContent = result.SOL_open_15m_p
?? "-";
    document.getElementById('text_15m_min_p').textContent = result.SOL_min_15m_p ??
"-";
    document.getElementById('text_15m_max_p').textContent = result.SOL_max_15m_p ??
"-";
    document.getElementById('text_15m_close_p').textContent = result.SOL_end_15m_p
?? "-";
  }
  );
  chrome.storage.local.get(["SOL_open_1h_p","SOL_min_1h_p","SOL_max_1h_p","SOL_end_
1h_p"], (result) => {
    document.getElementById('text_1h_open_p').textContent = result.SOL_open_1h_p ??
"-";
    document.getElementById('text_1h_min_p').textContent = result.SOL_min_1h_p ??
"-";
    document.getElementById('text_1h_max_p').textContent = result.SOL_max_1h_p ??
"-";
    document.getElementById('text_1h_close_p').textContent = result.SOL_end_1h_p ??
"-";
  });

  chrome.storage.local.get(["SOL_open_4h_p","SOL_min_4h_p","SOL_max_4h_p","SOL_end_
4h_p"], (result) => {
    document.getElementById('text_4h_open_p').textContent = result.SOL_open_4h_p ??
"-";
    document.getElementById('text_4h_min_p').textContent = result.SOL_min_4h_p ??
"-";
    document.getElementById('text_4h_max_p').textContent = result.SOL_max_4h_p ??
"-";
    document.getElementById('text_4h_close_p').textContent = result.SOL_end_4h_p ??
"-";
  });
    chrome.storage.local.get(["SOL_open_1d_p","SOL_min_1d_p","SOL_max_1d_p","SOL_en
d_1d_p"], (result) => {
    document.getElementById('text_1d_open_p').textContent = result.SOL_open_1d_p ??
"-";
```

```
    document.getElementById('text_1d_min_p').textContent = result.SOL_min_1d_p ??
"-";
    document.getElementById('text_1d_max_p').textContent = result.SOL_max_1d_p ??
"-";
    document.getElementById('text_1d_close_p').textContent = result.SOL_end_1d_p ??
"-";
  });
}

});
```

Даний код формує поведінкову логіку клієнтської частини розширення Chrome. Загальна структура коду побудована навколо події завантаження документа, після якої сторінка ініціалізує всі елементи інтерфейсу, підключає обробники натискань та налаштовує взаємодію зі сховищем браузера і фоновими процесами скрипту background.js.

Першим кроком відбувається отримання посилань на всі кнопки, що відповідають за різні часові інтервали. Для кожного інтервалу передбачено дві кнопки: одна відправляє запит на отримання прогнозу з уже натренованої моделі, а друга - ініціює процес перенавчання моделі. Такий поділ дозволяє користувачу швидко обирати між швидким розрахунком і повною перебудовою ваг моделі, що підвищує точність подальших прогнозів.

Одразу після ініціалізації сторінка викликає два ключові методи — getStoredData() і getProcData(). Перший відповідає за відображення основних прогнозних значень, другий за виведення обрахованої похибки прогнозу. Обидві функції отримують дані зі сховища браузера Chrome, де їх попередньо розміщує бекенд (скрипт background.js).

Для зручності відображення кожне значення підставляється безпосередньо в DOM, а у випадку відсутності даних встановлюється символ «-», що дозволяє підтримувати чистий і структурований інтерфейс.

Наступна частина коду відповідає за комунікацію з фоновою частиною розширення. Через chrome.runtime.onMessage.addListener сторінка реагує на надходження нового повідомлення з серверу. Коли бекенд надсилає команду оновити інформацію, сторінка одразу повторно викликає обидві функції отримання даних, забезпечуючи актуальність прогнозів без ручного перезавантаження.

Окремо налаштована логіка обробки натискань на кнопки. При кліку кожна кнопка відправляє у фоновий процес повідомлення, яке містить тип операції та її ідентифікатор. Таким чином, система чітко розрізняє, чи потрібно просто виконати прогнозування, чи повністю перенавчити модель. В консолі браузера ці дії фіксуються, що дозволяє здійснювати швидку діагностику та відстежувати роботу інтерфейсу.

Важливою частиною коду є механізм підсвічування активної вкладки. Завдяки збереженню ідентифікатора кнопки у локальному сховищі браузера інтерфейс пам’ятає, яку вкладку користувач відкрив останньою.

Після перезавантаження сторінки цей вибір автоматично підсвічується, забезпечуючи комфортну навігацію та відчуття цілісності взаємодії. Функція getProcData() дублює структуру основної функції отримання даних, але працює з показниками похибки обчислення, що мають індекс _p. Це дозволяє розділяти сирі прогнозні значення та похибку прогнозу у відсотках.

Цей код створює комплексну, але добре структуровану логіку відображення даних у розширенні Chrome. Він поєднує асинхронну взаємодію з бекендом, управління інтерфейсом і локальне зберігання інформації. Завдяки цьому користувач отримує плавний доступ до прогнозів моделі та можливість миттєво запускати обчислення або перенавчання. Логіка написана так, щоб максимально спростити роботу з даними та підтримувати інтерфейс завжди актуальним і інтуїтивним.

background.js - фоновий скрипт, що працює незалежно від відкритого інтерфейсу розширення. Він покликаний підтримувати WebSocket-з’єднання,

оновлювати дані у фоновому режимі та передавати їх до popup.js. Нище наведений код скрипту background.js.

```
  chrome.runtime.onMessage.addListener((msg, sender, sendResponse) => {
    ws = new WebSocket('ws://localhost:8765');
    ws.onopen = () =>  {console.log("вІДКРИТО ");
    ws.send(JSON.stringify({
    type: [msg.type],
    data:[msg.data]
     }));
    console.log("Відправлено запит", msg.data);
    chrome.runtime.sendMessage({ command: "updateData" });
  switch (msg.data) {

  case "SOLUSDT_text_15m_button":
    chrome.storage.local.set({
      SOL_open_15m: "...",
      SOL_min_15m: "...",
      SOL_max_15m: "...",
      SOL_end_15m: "...",
      SOL_open_15m_p: "...",
      SOL_min_15m_p: "...",
      SOL_max_15m_p: "...",
      SOL_end_15m_p: "..."
    });
    setupButton(msg.data, 'SOL_open_15m', 'SOL_min_15m', 'SOL_max_15m',
'SOL_end_15m');
    break;

  case "SOLUSDT_text_1h_button":
    chrome.storage.local.set({
      SOL_open_1h: "...",
      SOL_min_1h: "...",
      SOL_max_1h: "...",
      SOL_end_1h: "...",
      SOL_open_1h_p: "...",
      SOL_min_1h_p: "...",
      SOL_max_1h_p: "...",
      SOL_end_1h_p: "..."
```

```
    });
    setupButton(msg.data, 'SOL_open_1h', 'SOL_min_1h', 'SOL_max_1h', 'SOL_end_1h');
    break;

  case "SOLUSDT_text_4h_button":
    setupButton(msg.data, 'SOL_open_4h', 'SOL_min_4h', 'SOL_max_4h', 'SOL_end_4h');
    chrome.storage.local.set({
      SOL_open_4h: "...",
      SOL_min_4h: "...",
      SOL_max_4h: "...",
      SOL_end_4h: "...",
      SOL_open_4h_p: "...",
      SOL_min_4h_p: "...",
      SOL_max_4h_p: "...",
      SOL_end_4h_p: "..."
    });
    break;

  case "SOLUSDT_text_1d_button":
    chrome.storage.local.set({
      SOL_open_1d: "...",
      SOL_min_1d: "...",
      SOL_max_1d: "...",
      SOL_end_1d: "...",
      SOL_open_1d_p: "...",
      SOL_min_1d_p: "...",
      SOL_max_1d_p: "...",
      SOL_end_1d_p: "..."
    });
    setupButton(msg.data, 'SOL_open_1d', 'SOL_min_1d', 'SOL_max_1d', 'SOL_end_1d');
    break;

  case "BTCUSDT_text_1d_button":
    chrome.storage.local.set({
      BTC_open_1d: "...",
      BTC_min_1d: "...",
      BTC_max_1d: "...",
      BTC_end_1d: "...",
      BTC_open_1d_p: "...",
      BTC_min_1d_p: "...",
```

```
      BTC_max_1d_p: "...",
      BTC_end_1d_p: "..."
    });
    setupButton(msg.data, 'BTC_open_1d', 'BTC_min_1d', 'BTC_max_1d', 'BTC_end_1d');
    break;
  case "BTCUSDT_text_4h_button":
    chrome.storage.local.set({
      BTC_open_4h: "...",
      BTC_min_4h: "...",
      BTC_max_4h: "...",
      BTC_end_4h: "...",
      BTC_open_4h_p: "...",
      BTC_min_4h_p: "...",
      BTC_max_4h_p: "...",
      BTC_end_4h_p: "..."
    });
    setupButton(msg.data, 'BTC_open_4h', 'BTC_min_4h', 'BTC_max_4h', 'BTC_end_4h');
    break;
    case "BTCUSDT_text_1h_button":
    chrome.storage.local.set({
      BTC_open_1h: "...",
      BTC_min_1h: "...",
      BTC_max_1h: "...",
      BTC_end_1h: "...",
      BTC_open_1h_p: "...",
      BTC_min_1h_p: "...",
      BTC_max_1h_p: "...",
      BTC_end_1h_p: "..."
    });
    setupButton(msg.data, 'BTC_open_1h', 'BTC_min_1h', 'BTC_max_1h', 'BTC_end_1h');
    break;
    case "BTCUSDT_text_15m_button":
    chrome.storage.local.set({
      BTC_open_15m: "...",
      BTC_min_15m: "...",
      BTC_max_15m: "...",
      BTC_end_15m: "...",
      BTC_end_15m_p:"...",
      BTC_min_15m_p: "...",
      BTC_max_15m_p: "...",
```

```
      BTC_end_15m_p: "..."

    });
    setupButton(msg.data, 'BTC_open_15m', 'BTC_min_15m', 'BTC_max_15m',
'BTC_end_15m');
    break;
    case "ETHUSDT_text_1d_button":
    chrome.storage.local.set({
      ETH_open_1d: "...",
      ETH_min_1d: "...",
      ETH_max_1d: "...",
      ETH_end_1d: "...",
      ETH_open_1d_p: "...",
      ETH_min_1d_p: "...",
      ETH_max_1d_p: "...",
      ETH_end_1d_p: "..."
    });
    setupButton(msg.data, 'ETH_open_1d', 'ETH_min_1d', 'ETH_max_1d', 'ETH_end_1d');
    break;
    case "ETHUSDT_text_4h_button":
    chrome.storage.local.set({
      ETH_open_4h: "...",
      ETH_min_4h: "...",
      ETH_max_4h: "...",
      ETH_end_4h: "...",
      ETH_open_4h_p: "...",
      ETH_min_4h_p: "...",
      ETH_max_4h_p: "...",
      ETH_end_4h_p: "..."
    });
    setupButton(msg.data, 'ETH_open_4h', 'ETH_min_4h', 'ETH_max_4h', 'ETH_end_4h');
    break;
    case "ETHUSDT_text_1h_button":
    chrome.storage.local.set({
      ETH_open_1h: "...",
      ETH_min_1h: "...",
      ETH_max_1h: "...",
      ETH_end_1h: "...",
      ETH_open_1h_p: "...",
      ETH_min_1h_p: "...",
```

```
      ETH_max_1h_p: "...",
      ETH_end_1h_p: "..."
    });
    setupButton(msg.data, 'ETH_open_1h', 'ETH_min_1h', 'ETH_max_1h', 'ETH_end_1h');
    break;
    case "ETHUSDT_text_15m_button":
    chrome.storage.local.set({
      ETH_open_15m_p: "...",
      ETH_min_15m_p: "...",
      ETH_max_15m_p: "...",
      ETH_end_15m_p: "...",
      ETH_open_15m: "...",
      ETH_min_15m: "...",
      ETH_max_15m: "...",
      ETH_end_15m: "..."
    });
    setupButton(msg.data, 'ETH_open_15m', 'ETH_min_15m', 'ETH_max_15m',
'ETH_end_15m');
    break;
}


}});




  function setupButton(buttonId, text1, text2, text3, text4) {
    console.log("Стан WebSocket:", ws.readyState);
        ws.onmessage = event => {
          const data = JSON.parse(event.data);
          chrome.storage.local.get(null, (data) => {
          console.table(data);
          });
          if (data.error) {
          console.log({error:"Неотримано дані "});
          alert(data.error);
          return;}
```

```
            if (data.filename === `${buttonId}.json`) {
              console.log({success:"✅ Дані отримано"});
              chrome.storage.local.set({
                  [text1]: parseFloat(data.open).toFixed(2),
                  [text2]: parseFloat(data.min).toFixed(2),
                  [text3]: parseFloat(data.max).toFixed(2) ,
                  [text4]: parseFloat(data.end).toFixed(2),
                  [text1+"_p"]:parseFloat(data.open_i).toFixed(2),
                  [text2+"_p"]:parseFloat(data.min_i).toFixed(2),
                  [text3+"_p"]:parseFloat(data.max_i).toFixed(2) ,
                  [text4+"_p"]:parseFloat(data.end_i).toFixed(2)});
                  chrome.runtime.sendMessage({ command: "updateData" });
            } else {
              console.log({error:"❌ Не той файл"});
            }
        }
}
```

Представлений фрагмент JavaScript-коду описує взаємодію Chrome-розширення з локальним сервером через WebSocket та організовує оновлення даних щодо вибраних криптовалютних пар. Його робота побудована навколо принципу обробки користувацьких дій, переданих у вигляді повідомлень між частинами розширення, і подальшого отримання актуальної інформації з сервера.

Першим ключовим етапом є реєстрація обробника повідомлень chrome.runtime.onMessage.addListener. Кожне отримане повідомлення ініціює створення нового WebSocket-з’єднання з локальним сервером. Щойно канал відкривається, розширення відправляє запит, що містить тип дії та її аргументи. Це дозволяє серверу зрозуміти, яку саме криптовалюту та таймфрейм потрібно прогнозувати.

Одразу після цього викликається повідомлення про оновлення даних, щоб інші частини розширення були поінформовані про зміни. Фрагмент великого оператора switch відображає те, як розширення реагує на кожну кнопку

користувача. Для кожного конкретного таймфрейму 15 хвилин, 1 година, 4 години або доба - перезаписуються значення в chrome.storage.local. Це робиться для уникнення помилок, спричинених залишковими даними з попередніх операцій.

Кожен випадок передає у функцію setupButton набір ключів, які відповідають відкриттю, мінімуму, максимуму та закриттю ціни відповідної валютної пари.

Допоміжна функція setupButton відповідає за обробку відповіді, що надходить по WebSocket. Після отримання повідомлення розширення розпаковує JSON-дані, перевіряє коректність результату та зіставляє отримане ім’я файлу з очікуваним. Якщо все відповідає запиту, дані з точністю до сотих записуються у сховище.

Крім основних показників, зберігаються похибки прогнозу з індексом «_p». На завершення надсилається системне повідомлення, яке дає іншим компонентам змогу оновити інтерфейс розширення.

Усі частини коду працюють як єдина система: інтерфейс повідомляє про дію, сервер повертає результати, а сховище забезпечує синхронізацію між сторінками розширення. Така структура робить застосунок гнучким, прозорим і здатним легко масштабуватися для додавання нових валют або таймфреймів.

Img/icons - папка з наборами іконок різних розмірів, потрібних для коректного відображення розширення в браузері, структуру файлів необхідних іконок подано на рис. 3.3.

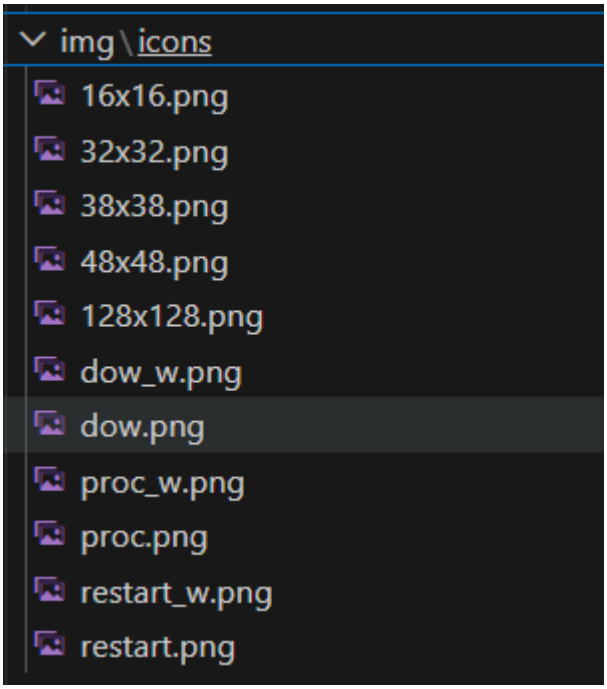

Рис. 3.3 Структура іконок розширення

3.5 Висновок до розділу.

В даному розіділі я комплексно розглянуто процес створення та організації роботи браузерного розширення, що забезпечує інтеграцію розробленої нейромережевої моделі у зручний і зрозумілий інтерфейс для кінцевого користувача.

Важливо підкреслити, що розробка розширення суттєво відрізняється від створення звичайної настільної програми: тут ключову роль відіграє взаємодія з браузером, застосування вебтехнологій, система дозволів та більш гнучке, модульне розділення компонентів. Розширення працює в контейнері браузера, де всі процеси розділені на інтерфейсні, фонові та логічні, що дозволяє оптимізувати продуктивність та забезпечити безперервне оновлення даних.

Окрему увагу приділено запуску та ролі сервера, який виступає центральним елементом у взаємодії між розширенням та нейромережею. Були розглянуті основні типи серверів, що можуть застосовуватися для подібних

рішень, а також їх переваги та недоліки. Найбільш ефективним для цього проєкту виявився локальний сервер, що працює на основі WebSocket-підключення. Такий підхід забезпечує постійний двосторонній зв’язок, мінімальну затримку передачі даних, можливість миттєвого отримання результатів прогнозу та швидке перенавчання моделі при потребі.

Було детально описано саму WebSocket-бібліотеку, принципи встановлення з’єднання, обміну повідомленнями та підтримки активної сесії між розширенням і сервером.

В межах даного розділу структуровано проаналізовано роботу всіх основних файлів, що забезпечують функціонування розширення. Зокрема, manifest.json визначає конфігурацію, дозволи, фонові процеси та інтерфейсні сторінки, закладаючи фундамент взаємодії розширення з браузером.

Файл popup.html відповідає за побудову візуальної частини інтерфейсу - вкладок, кнопок, областей для відображення прогнозованих значень та похибок при прогнозувані.

Стилі у style.css забезпечують зручний, впорядкований і візуально зрозумілий вигляд застосунку.

Скрипт popup.js реалізує клієнтську логіку обробку подій користувача, підключення до сервера, відправку запитів, відображення прогнозів і розрахунок похибок.

Для підтримки фонової роботи, збереження стану та постійного WebSocket-з’єднання використовується background.js, який функціонує незалежно від відкритої вкладки та забезпечує стабільність обміну даними.

У результаті всі компоненти системи формують цілісний інструмент, де послідовність роботи виглядає наступним чином:

- запуск локального WebSocket-сервера;
- встановлення з’єднання з фоновим скриптом background.js;
- передача запитів на прогноз чи перенавчання моделі;

- обробка даних сервером і повернення готових результатів;
- оновлення інтерфейсу через popup.js відповідно до інформації, отриманої від сервера;
- відображення достовірних прогнозів та похибок у popup.html для кінцевого користувача.

Таким чином, третій розділ надав цілісне розуміння того, як відбувається інтеграція складної системи машинного навчання у практичний інструмент, орієнтований на користувача. Були розкриті як технічні аспекти взаємодії клієнта та сервера, так і архітектурні особливості браузерного розширення. У підсумку сформовано ефективне програмне рішення, яке поєднує сучасні вебтехнології та потужність нейронної мережі, забезпечуючи оперативний доступ до прогнозів і сприяючи ухваленню більш точних рішень у роботі з цифровими активами.

ЗАГАЛЬНІ ВИСНОВКИ

Під час виконання даної магістерської роботи було всебічно досліджено та реалізовано повний цикл створення інструменту для прогнозування вартості криптовалют - від теоретичних основ і аналізу ринку до розробки програмних рішень і практичної інтеграції нейронної мережі у браузерне розширення.

На початковому етапі увага була зосереджена на вивченні специфіки ринку цінних паперів та його ключових відмінностей від криптовалютних бірж. Було встановлено, що класичні фінансові ринки регулюються жорсткими правилами, мають централізовану структуру та обмежену волатильність, тоді як криптовалютний сегмент характеризується децентралізованістю, цілодобовою торгівлею, високою нестабільністю та суттєвим впливом глобальних подій.

Саме ці відмінності формують унікальні виклики для побудови моделей прогнозування, які мають ефективно працювати в умовах швидких змін і нерівномірності даних.

Далі було проведено розгорнутий аналіз сучасних бібліотек для обробки даних, зокрема pandas, NumPy, Matplotlib, scikit-learn, TensorFlow та Keras. Ці інструменти забезпечують комплексну підтримку на всіх етапах роботи з часовими рядами. Наприклад очищення, підготовка даних до побудови нейромереж та візуалізації результатів прогнозу, прорахунок похибки даного прогнозу.

Особливе значення мають модулі sklearn.preprocessing, що дозволяють коректно масштабувати та нормалізувати дані, тим самим підвищуючи якість навчання моделі.

Було розглянуто можливості високорівневих архітектур, які дають змогу формувати складні мережеві структури буквально кількома рядками коду, зокрема LSTM та оптимізатори з бібліотеки keras.optimizers.

На наступному етапі було проведено моделювання нейронної мережі, де обґрунтовано використання гібридної архітектури CNN-LSTM. Одновимірні

згорткові шари застосовано для виділення локальних патернів і короткострокових залежностей у часових рядах, тоді як рекурентні LSTM-шари дали змогу ефективно опрацьовувати довгі послідовності та враховувати контекст.

Було детально описано роботу кожного шару, їх функціональне призначення, проблему перенавчання та методи її зменшення за допомогою Dropout і BatchNormalization.

Окремим етапом стало дослідження оптимізаторів. Порівняння SGD, Adam, Nadam, AdamW та RAdam дало змогу визначити найбільш ефективну комбінацію, а результати тестування продемонстрували доцільність використання гібридного навчання із поєднанням двох оптимізаторів.

Завершальна частина роботи була присвячена розробці браузерного розширення, яке виконує роль інтерактивної оболонки для нейромережевої моделі. Було вивчено специфіку розробки розширень, відмінність цього процесу від створення звичайних програм та особливості роботи з браузерним середовищем.

Детально розглянуто структуру файлів, роль manifest.json, логіку відображення інформації в popup.html, організацію стилів у style.css, реалізацію клієнтської логіки у popup.js та роботу background.js, який підтримує постійний канал зв’язку.

Для забезпечення швидкого та стабільного обміну даними між нейромережевою моделлю та інтерфейсом було розгорнуто локальний WebSocket-сервер. Він забезпечує двосторонній зв’язок у режимі реального часу, що дозволяє користувачу миттєво оновлювати прогнози або перенавчати модель на актуальних даних. У підсумку реалізовано повноцінну систему, яка поєднує глибокий аналіз ринку, сучасні підходи до машинного навчання та інструменти веброзробки.

Створена нейромережа здатна генерувати прогнози вартості криптовалют, а розширення браузера робить цей функціонал доступним, наочним та зручним у використанні. Завдяки інтеграції усіх компонентів - від попередньої обробки даних до інтерфейсу користувача - сформовано гнучкий, модульний та ефективний інструмент, здатний допомагати трейдерам та дослідникам приймати обґрунтовані рішення в умовах високої динамічності криптовалютного ринку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Buklib – [Електронний ресурс] – Режим доступу: https://buklib.net/books/27201/?utm_source=chatgpt.com (Дата звернення 01.11.2025)
2. UANEWS – [Електронний ресурс] – Режим доступу: https://ua.news/ua/ (Дата звернення 01.11.2025)
3. Кобилін, О., Вечірська, І., Кравченко, О. (2024). Порівняння нейронних мереж типу RNN та LSTM. (Дата звернення 02.11.2025)
4. Huggingface – [Електронний ресурс] – Режим доступу: https://huggingface.co/papers/1712.07628?utm_source=chatgpt.com (Дата звернення 02.11.2025)
5. Chrome Developers Documentation – [Електронний ресурс] – Режим доступу: https://developer.chrome.com/docs/extensions/ (Дата звернення 05.11.2025)
6. MDN Web Docs – WebExtensions API – [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions (Дата звернення 05.11.2025)
7. Mozilla Manifest V3 Migration Guide – [Електронний ресурс] – Режим доступу: https://extensionworkshop.com/documentation/develop/manifest-v3-migration-guide/ (Дата звернення 05.11.2025)
8. W3C Web APIs and DOM Standards – [Електронний ресурс] – Режим доступу: https://www.w3.org/TR/?tag=api (Дата звернення 07.11.2025)
9. GitHub – [Електронний ресурс] – Режим доступу: https://github.com/ (Дата звернення 07.11.2025)
10. Udemy: Build Chrome Extensions from Scratch – [Електронний ресурс] – Режим доступу: https://www.udemy.com/course/chrome-extension/ (Дата звернення 08.11.2025)
11. StackOverflow – Browser Extension Architecture Discussions – [Електронний ресурс] – Режим доступу:

https://stackoverflow.com/questions/tagged/browser-extension (Дата звернення 09.11.2025)
12. Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning. (Дата звернення: 15.11.2025)