Міністерство освіти у науки України
Харіквський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Кафедра математичного моделювання та аналізу даних

До захисту допущено
Кафедрою математичного моделювання та аналізу даних
протокол №___ від _____

Завідувач кафедри _____ Володимир СТРУКОВ
(підпис) *(імя, прізвище)*

«___» _____ 2026 р.

Кваліфікаційна робота
здобувача першого (бакалаврського) рівня вищої освіти

Дослідження та розробка моделей глибокого навчання (Deep Learning) для прогнозування криптовалютних ринків із використанням Walk-Forward-валідації та оцінки ризику
(назва роботи)

Спеціальність (спеціалізація) 122 Комп'ютерні науки

Освітня програма Комп'ютерні системи та мережі

Виконавець _____ Тетяна БОЙКО
(підпис) *(імя, прізвище)*

Науковий керівник _____ Сергій СЕВІДОВ
(підпис) *(імя, прізвище)*

Харків – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н.Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Кафедра математичного моделювання та аналізу даних

ЗАТВЕРДЖУЮ
Завідувач кафедри ММтАД

_____Володимир СТРУКОВ
підпис ім'я, прізвище

"____" _____2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувача першого (бакалаврського) рівня вищої освіти

Бойко Тетяна Антонівна

Спеціальність (спеціалізація) 122 «Комп'ютерні науки», _____

Освітня програма Комп'ютерні системи та мережі _____

1. Тема роботи: Дослідження та розробка моделей глибокого навчання (Deep Learning) для прогнозування криптовалютних ринків із використанням Walk-Forward-валідації та оцінки ризику

керівник роботи Севидов Сергій Михайлович, канд.ф.-м.наук, доцент

затверджені наказом по Університету від 29 січня 2026 року № 4101-5/225

2. Строк здобувачем роботи "01" червня 2026 року

3. Вихідні дані до роботи:

1. Набір даних для прогнозування: історичні біржові дані криптовалют Bitcoin (BTC) та Ethereum (ETH) у форматі OHLCV;

2. Базові архітектури моделей: рекурентні нейронні мережі глибокого навчання LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit);

3. Мова розробки та інструментарій: Python, фреймворк глибокого навчання TensorFlow/Keras, бібліотеки обробки та аналізу даних Pandas, Scikit-learn.

4. Перелік питань, які потрібно розробити:

1. ознайомитись з особливостями задачі прогнозування фінансових часових рядів на високоволатильному та ринку криптовалют;
2. провести огляд існуючих методів фінансового прогнозування та сучасних архітектур машинного і глибокого навчання для аналізу часових рядів;
3. познайомитися з математичним апаратом рекурентних нейронних мереж і методологією Walk-Forward-валідації, спроєктувати архітектуру нейромережевої системи;
4. провести програмну реалізацію та експериментальне дослідження розроблених моделей, виконати порівняльний аналіз їхньої ефективності за метриками точності машинного навчання та показниками фінансового ризику.

5. План роботи

№ з/п	Назви етапів роботи	Строк виконання етапів роботи	Примітка
1	Формулювання мети та повного переліку завдань для виконання кваліфікаційної роботи.	15.01.2026 – 30.01.2026	Виконано
2	Пошук та аналіз інформаційних джерел за темою КР.	30.01.2026 – 20.02.2026	Виконано
3	Проєктування нейромережевої системи на базі архітектур LSTM та GRU.	20.02.2026 – 18.03.2026	Виконано
4	Програмна реалізація моделей та підготовка історичного датасету криптовалют.	18.03.2026 – 02.04.2026	Виконано
5	Проведення експериментів (Walk-Forward валідація) та порівняльний аналіз результатів.	02.04.2026 – 19.04.2026	Виконано
6	Підсумок отриманих результатів та оформлення пакету документів для процедури захисту КР в екзаменаційній комісії ННІ КН та ШІ.	19.04.2026 – 03.05.2026	Виконано

1. Дата видачі завдання 15 січня 2026 р.

Здобувач вищої освіти _____ Т. А. БОЙКО
(підпис) (ініціали, прізвище)

Керівник роботи _____ С. М. СЕВИДОВ
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи складається зі вступу, трьох розділів, висновків, списку використаних джерел та 3 додатків. Загальний обсяг роботи складає 66 сторінок, із яких 49 сторінок основної частини з 9 рисунками, 2 таблицями та 27 найменуваннями списку використаних джерел.

Метою дослідження є підвищення точності прогнозування курсу криптовалют та мінімізація інвестиційних ризиків шляхом розробки й порівняння оптимізованих моделей глибокого навчання із застосуванням Walk-Forward-валідації.

Об’єкт дослідження – процес прогнозування фінансових часових рядів в умовах високоволатильного криптовалютного ринку.

Предмет дослідження – методи глибокого навчання (рекурентні нейромережі LSTM та GRU), алгоритми їх крокової валідації та оцінки фінансових ризиків.

Проблема, яка вирішується в роботі, полягає у створенні предиктивної системи, яка долає проблему «витоку даних» при тестуванні та забезпечує не лише точність прогнозу на активах Bitcoin і Ethereum, але й ефективний контроль торгових ризиків (мінімізацію просідання капіталу).

Область застосування – фінансові технології (FinTech), розробка систем алгоритмічного трейдингу, управління криптовалютними портфелями та автоматизований ризик-менеджмент.

Ключові слова: *ПРОГНОЗУВАННЯ КУРСУ, КРИПТОВАЛЮТА, BITCOIN, ETHEREUM, ГЛИБОКЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, LSTM, GRU, WALK-FORWARD ВАЛІДАЦІЯ, ОЦІНКА РИЗИКУ, АЛГОРИТМІЧНА ТОРГІВЛЯ.*

ABSTRACT

The explanatory note to the qualification work consists of an introduction, three chapters, conclusions, a list of references, and 3 appendices. The total volume of the work is 66 pages, including 49 pages of the main text with 9 figures, 2 tables, and 27 references.

The purpose of the study is to improve the accuracy of cryptocurrency exchange rate forecasting and minimize investment risks by developing and comparing optimized deep learning models using Walk-Forward validation.

The object of the study is the process of forecasting financial time series in a highly volatile cryptocurrency market.

The subject of the study encompasses deep learning methods (recurrent neural networks LSTM and GRU), algorithms for their step-by-step validation, and financial risk assessment.

The problem addressed in the work is the creation of a predictive system that overcomes the "data leakage" issue during testing and ensures not only accurate forecasting on Bitcoin and Ethereum assets but also effective control of trading risks (minimization of maximum drawdown).

The field of application includes financial technology (FinTech), algorithmic trading systems development, cryptocurrency portfolio management, and automated risk management.

Keywords: *EXCHANGE RATE FORECASTING, CRYPTOCURRENCY, BITCOIN, ETHEREUM, DEEP LEARNING, NEURAL NETWORKS, LSTM, GRU, WALK-FORWARD VALIDATION, RISK ASSESSMENT, ALGORITHMIC TRADING.*

ЗМІСТ

ВСТУП

Сучасний етап розвитку глобальної фінансової системи характеризується стрімкою інтеграцією децентралізованих цифрових активів. Криптовалютний ринок, який ще десятиліття тому сприймався як нішевий технологічний експеримент, сьогодні перетворився на повноцінний фінансовий інструмент із багатомільярдною капіталізацією. Проте, на відміну від традиційних фондових чи валютних ринків, ринок криптовалют функціонує в режимі 24/7, не має єдиного регулятора та характеризується екстремально високою волатильністю. Ціноутворення базових активів, таких як Bitcoin (BTC) або Ethereum (ETH), схильне до раптових структурних зрушень, кластеризації волатильності та формування «товстих хвостів» розподілу. У таких умовах високочастотних та нестаціонарних змін класичні підходи до прогнозування фінансових часових рядів стикаються із серйозними методологічними та алгоритмічними викликами.

Аналіз вітчизняної та зарубіжної науково-технічної літератури свідчить, що еволюція методів прогнозування фінансових ринків пройшла шлях від лінійних економетричних моделей (ARIMA, GARCH) до систем класичного машинного навчання (Random Forest, SVM). Сучасний стан об’єкта дослідження характеризується активним переходом до використання моделей глибокого навчання (Deep Learning). Згідно з дослідженнями провідних науковців, звичайні алгоритми машинного навчання не здатні ефективно обробляти часові послідовності через відсутність внутрішньої «пам'яті». Тому найбільш перспективними рішеннями на теперішній час є рекурентні нейронні мережі (RNN), зокрема архітектури довгої короткострокової пам'яті (LSTM) та керованого рекурентного блоку (GRU), які здатні автоматично витягувати нелінійні закономірності з масивів сирих біржових даних.

Незважаючи на значний потенціал глибокого навчання, у цій предметній області існують серйозні невирішені проблеми. Більшість існуючих рішень застосовують стандартну методологію перехресної перевірки (k-fold cross-

validation), що є грубою помилкою для часових рядів, оскільки призводить до «витоку даних» (look-ahead bias) - ситуації, коли модель навчається на даних з майбутнього. Крім того, переважна більшість дослідників оцінює якість розроблених нейромереж виключно за метриками точності регресії (RMSE, MAE), ігноруючи реальні фінансові ризики. Це призводить до того, що математично точні моделі в реальних торгових умовах зазнають катастрофічних просідань капіталу. Отже, виникає гостра потреба у розробці комплексних інтелектуальних систем, які поєднують потужність глибокого навчання із суворими алгоритмами бектестування та метриками фінансового ризик-менеджменту.

Актуальність теми дослідження зумовлена необхідністю створення математично обґрунтованих та фінансово безпечних інструментів для алгоритмічної торгівлі на нестаціонарних криптовалютних ринках. Зростаючий інтерес інституційних та приватних інвесторів до віртуальних активів вимагає заміни суб'єктивного технічного аналізу на автоматизовані системи штучного інтелекту, здатні самостійно адаптуватися до змін волатильності, генерувати прибуток та мінімізувати ймовірність глибоких втрат капіталу.

Об'єктом дослідження є процес прогнозування фінансових часових рядів в умовах високоволатильного та нестаціонарного криптовалютного ринку.

Предметом дослідження є методи та моделі глибокого навчання (зокрема, рекурентні нейронні мережі архітектур LSTM та GRU), а також алгоритми їх крокової валідації та оцінки фінансових ризиків.

Метою дослідження є підвищення точності прогнозування цін на криптовалютні активи та мінімізація інвестиційних ризиків шляхом проєктування, програмної реалізації та порівняльного аналізу оптимізованих моделей глибокого навчання (LSTM та GRU) із застосуванням імітаційного алгоритму Walk-Forward-валідації.

Для досягнення поставленої мети в роботі необхідно вирішити такі конкретні завдання:

1. Провести системний аналіз існуючих підходів до прогнозування фінансових часових рядів та виявити обмеження класичного машинного навчання в контексті нестаціонарних даних.
2. Спроєктувати архітектуру нейромережевої системи на базі блоків LSTM та GRU із застосуванням механізмів просторової регуляризації та контролю епох навчання.
3. Розробити алгоритм препроцесінгу сирих біржових даних, реалізувавши лінійну нормалізацію та трансформацію одновимірних рядів у багатовимірні тензори методом «ковзного вікна».
4. Програмно імплементувати алгоритм суворої Walk-Forward-валідації для запобігання перенавчанню та «витоку даних» під час тестування на історичних періодах.
5. Виконати навчання розроблених моделей на реальних ринкових даних різних криптовалют (Bitcoin та Ethereum) та провести їх порівняльний аналіз за метриками точності машинного навчання (MAE, RMSE).
6. Здійснити симуляцію алгоритмічної торгівлі та провести оцінку інвестиційних ризиків моделей з використанням фінансових індикаторів: Коефіцієнта Шарпа (Sharpe Ratio) та Максимального просідання капіталу (Maximum Drawdown).

Наукова новизна одержаних результатів полягає у комплексному підході до розв'язання задачі алгоритмічного трейдингу. Вперше в рамках даної кваліфікаційної роботи проведено емпіричне доведення переваги спрощеної архітектури GRU над LSTM в умовах екстремальної волатильності не лише за метриками машинного навчання, а й за показниками інвестиційного ризик-менеджменту. Запропоновано інтегрований пайплайн, який жорстко поєднує оптимізацію нейромереж з алгоритмом Walk-Forward-валідації, що усуває методологічні недоліки попередніх досліджень.

Практичне значення одержаних результатів полягає у створенні повністю автономного програмного комплексу мовою Python, який виконує

повний цикл операцій: від завантаження та очищення біржових датасетів до навчання нейромереж та генерації торгових сигналів. Розроблена система може використовуватися як готовий модуль бектестування при створенні комерційних алгоритмічних торгових ботів, а також як інструмент аналітики для підтримки прийняття рішень криптоінвесторами на високоризикових фінансових ринках. Основною ідеєю роботи є доведення того факту, що використання математично легших рекурентних структур (GRU) у поєднанні із суворою кроковою валідацією забезпечує оптимальний баланс між точністю виявлення трендів та захистом капіталу від критичних ринкових шоків.

РОЗДІЛ 1. АНАЛІЗ ЗАДАЧІ ТА МЕТОДІВ ПРОГНОЗУВАННЯ ФІНАНСОВИХ ЧАСОВИХ РЯДІВ

1.1. Специфіка криптовалютних ринків та огляд існуючих підходів до прогнозування

Сучасна фінансова система переживає період глибоких трансформацій, пов'язаних із появою та стрімким розвитком децентралізованих цифрових активів. Криптовалютний ринок являє собою принципово новий клас фінансових інструментів, який суттєво відрізняється від традиційних фондових, валютних (Forex) або товарних ринків. Законодавче визнання цих інструментів, зокрема ухвалення Закону України «Про віртуальні активи» [2], свідчить про їхній перехід з нішевого технологічного експерименту до повноцінного глобального фінансового інструменту. Проте прогнозування їхньої вартості залишається однією з найскладніших задач у сучасній фінансовій аналітиці та машинному навчанні через низку унікальних мікроструктурних характеристик цього ринку [18].

Головною відмінністю криптовалютного ринку є його безперервність та глобальність. На відміну від класичних фондових бірж, які мають чітко визначені години сесій, вихідні та святкові дні, торги криптовалютами відбуваються у децентралізованому режимі 24/7. Це генерує безперервний потік великих даних (Big Data) і суттєво ускладнює використання багатьох класичних фінансових моделей, які історично спираються на поняття дискретної «ціни закриття» (Close) торгового дня у його традиційному розумінні.

Другою, найбільш критичною для математичного прогнозування характеристикою, є екстремальна волатильність та чутливість до зовнішніх факторів [24]. Ціна таких базових активів, як Bitcoin (BTC) або Ethereum (ETH), може змінюватися на десятки відсотків протягом кількох годин. Така динаміка зумовлена відносно низькою загальною ліквідністю порівняно з традиційними ринками (що дозволяє великим гравцям маніпулювати

ціноутворенням), відсутністю централізованих маркетмейкерів (окрім найбільших централізованих бірж) та гіперчутливістю до новинного фону. З математичної точки зору, часові ряди криптовалют характеризуються яскраво вираженою нестаціонарністю та «товстими хвостами» розподілу (fat tails). Це означає, що статистичні властивості ринку, такі як математичне сподівання та дисперсія, не є постійними в часі. Ринок постійно зазнає структурних зрушень (structural breaks) та кластеризації волатильності, коли періоди відносного спокою раптово змінюються фазами хаотичних цінових стрибків [19].

Незважаючи на ці значні алгоритмічні виклики, в економічній та комп'ютерній науці історично сформувалося декілька базових підходів до аналізу та прогнозування фінансових ринків:

1. Фундаментальний аналіз. Цей підхід базується на визначенні внутрішньої, «справедливої» вартості активу. У традиційних корпоративних фінансах він спирається на фінансову звітність, грошові потоки та макроекономічні показники. Проте для криптовалют цей метод потребує повної адаптації, оскільки більшість токенів не генерує дивідендів і не має фізичного забезпечення. Замість класичних індикаторів тут застосовується аналіз ончейн-метрик (on-chain metrics) - активність у мережі блокчейн, хешрейт, швидкість емісії токенів та аналіз Whitepaper проєктів. Хоча цей підхід корисний для довгострокового венчурного інвестування, він практично позбавлений предиктивної сили для короткострокового та середньострокового прогнозування часових рядів на спекулятивних ринках.

2. Технічний аналіз. Найбільш розповсюджений метод серед індивідуальних трейдерів, який базується на аксіомі, що всі фундаментальні фактори вже враховані у поточній ціні. Технічний аналіз використовує історичні дані (ціни та об'єми торгів) для виявлення графічних патернів та розрахунку математичних індикаторів (RSI, MACD, Moving Averages). Ключовим недоліком цього методу є його суб'єктивність: інтерпретація патернів залежить від аналітика. Крім того, більшість класичних індикаторів є запізнювальними (lagging indicators), тобто вони лише констатують факт зміни

тренду постфактум, що робить їх використання на високошвидкісних ринках неефективним без автоматизації.

3. Економетричні та статистичні моделі. Для роботи з фінансовими часовими рядами протягом десятиліть застосовувалися строгі математичні моделі, такі як ARIMA (Авторегресійна інтегрована модель ковзного середнього) та GARCH (модель умовної гетероскедастичності) [14]. Вони здатні моделювати лінійні залежності та кластеризацію волатильності. Однак їхнім фундаментальним обмеженням є жорсткі вимоги до статистичних властивостей даних (зокрема, стаціонарності та нормальності розподілу залишків) [19]. На криптовалютному ринку через його хаотичну природу ці моделі часто деградують до прогнозування простої випадкової прогулянки (random walk), не здатні вловити глибокі нелінійні залежності.

4. Класичне машинне навчання (Machine Learning). З розвитком обчислювальних потужностей фокус дослідників змістився на алгоритми машинного навчання, такі як метод опорних векторів (SVM), випадковий ліс (Random Forest) та градієнтний бустинг [3]. Реалізовані в бібліотеках на кшталт Scikit-learn [23], ці моделі не потребують суворих припущень щодо розподілу даних і здатні знаходити складні багатовимірні залежності. Однак для успішного прогнозування часових рядів класичне машинне навчання вимагає складного ручного конструювання ознак (feature engineering), оскільки більшість таких алгоритмів не мають вбудованої "пам'яті" для аналізу послідовностей подій у часі.

5. Глибоке навчання (Deep Learning). Сучасним етапом еволюції систем прогнозування є використання штучних нейронних мереж з глибокою архітектурою [11]. На відміну від класичного машинного навчання, глибоке навчання здатне самостійно екстрагувати значущі ознаки з «сирих» даних (representation learning) [7]. Для задач прогнозування часових рядів найперспективнішими є рекурентні нейронні мережі (RNN) та їхні модифікації (LSTM, GRU), які були спеціально створені для обробки послідовної інформації та утримання контексту минулих подій [5]. Саме ці

архітектури дозволяють виявляти приховані нелінійні та нестаціонарні патерни у ціноутворенні криптовалют [18, 24].

Незважаючи на високу точність алгоритмів глибокого навчання, впровадження їх у сферу фінансового прогнозування стикається з критичною проблемою валідації та управління ризиками [1]. Звичайні підходи до перехресної перевірки (k-fold cross-validation) не підходять для часових рядів через "витік даних" (data leakage) з майбутнього в минуле. Тому розробка надійних систем вимагає застосування специфічних методів, таких як Walk-Forward-валідація [22], яка імітує реальні умови торгівлі.

Крім того, оцінка ефективності моделей не може обмежуватися лише класичними метриками точності регресії, такими як середньоквадратична (RMSE) чи середня абсолютна помилка (MAE) [4]. Для комплексного розуміння життєздатності моделі необхідна інтеграція фінансових метрик оцінки ризику (наприклад, Maximum Drawdown та Sharpe Ratio) [13, 25]. Лише поєднання потужності Deep Learning із суворими методами фінансової валідації дозволяє створити стійку систему прогнозування криптовалютних ринків.

1.2. Порівняльний аналіз методів машинного та глибокого навчання

Розвиток обчислювальних технологій та накопичення гігантських масивів фінансових даних зумовили парадигмальний зсув у методології прогнозування часових рядів. На зміну класичним економетричним моделям прийшли алгоритми штучного інтелекту. Загалом, сучасні інтелектуальні системи прогнозування криптовалютних ринків можна розділити на дві великі категорії: класичне машинне навчання (Machine Learning, ML) та глибоке навчання (Deep Learning, DL) [3, 24]. Незважаючи на те, що обидва підходи належать до однієї галузі комп'ютерних наук, вони мають фундаментальні відмінності в архітектурі, вимогах до даних та здатності обробляти фінансові часові ряди.

Алгоритми традиційного машинного навчання, такі як метод опорних векторів (Support Vector Machines, SVM), випадковий ліс (Random Forest, RF) та системи градієнтного бустингу (XGBoost, LightGBM), отримали широке поширення у фінансовій аналітиці завдяки своїй математичній прозорості та відносній простоті налаштування [19].

Головною перевагою класичного ML є здатність працювати з нелінійними залежностями та стійкість до викидів (особливо це стосується ансамблевих методів на базі дерев рішень). Крім того, ці моделі вимагають значно менших обчислювальних ресурсів порівняно з нейромережами і дозволяють легко інтерпретувати результати (наприклад, через аналіз важливості ознак - feature importance).

Однак при застосуванні до криптовалютних часових рядів класичне машинне навчання стикається з фундаментальним концептуальним обмеженням. Більшість цих алгоритмів базуються на припущенні про незалежність спостережень (i.i.d. - independent and identically distributed). Вони не мають вбудованої структурної «пам'яті» і розглядають кожен момент часу як ізольований вектор даних [5]. Щоб модель SVM або Random Forest могла врахувати часову динаміку, дослідник змушений проводити складний ручний препроцесінг - конструювання ознак (feature engineering). Це означає штучне створення нових колонок у датасеті з лаговими значеннями (lagged variables), ковзними середніми та технічними індикаторами за минулі періоди. Такий підхід є негнучким, оскільки дослідник повинен заздалегідь «вгадати», який саме часовий горизонт є важливим для моделі, що в умовах хаотичного ринку криптовалют часто призводить до втрати прихованих закономірностей [18].

З іншого боку, методи глибокого навчання (DL) концептуально вирішують проблему ручного конструювання ознак. Багатошарові штучні нейронні мережі здатні виконувати автоматичну екстракцію ознак (representation learning), самостійно визначаючи, які комбінації вхідних даних є найбільш значущими для мінімізації помилки прогнозування [7, 11].

У контексті аналізу часових рядів стандартні багатошарові персептрони (MLP) або згорткові нейронні мережі (CNN) також мають обмеження, оскільки їхня архітектура спрямована на аналіз статичних просторових даних [17]. Справжній прорив у прогнозуванні фінансових ринків відбувся завдяки впровадженню рекурентних нейронних мереж (Recurrent Neural Networks, RNN) [24].

Особливістю архітектури RNN є наявність зворотних зв'язків: вихід мережі на попередньому кроці в часі (t-1) подається як частина входу на наступному кроці (t). Це створює внутрішній стан, або «пам'ять» моделі, що дозволяє їй аналізувати не просто окремі точки, а цілісні послідовності (наприклад, торгові вікна у 30 днів), вловлюючи хронологічну залежність між вчорашньою та сьогоднішньою ціною [5].

Проте базові архітектури RNN мають критичний математичний недолік - проблему затухання та вибуху градієнта (vanishing and exploding gradient problem) під час навчання за допомогою алгоритму зворотного поширення помилки в часі (Backpropagation Through Time, BPTT) [11]. Через це класичні RNN не здатні запам'ятовувати довгострокові залежності: інформація про події, що відбулися 20-30 днів тому, розчиняється під час обчислень і не впливає на поточний прогноз.

Для вирішення цієї проблеми були розроблені дві вдосконалені архітектури, які стали стандартом де-факто у сучасному глибокому навчанні для часових рядів і є предметом практичного дослідження даної кваліфікаційної роботи:

1. Модель довгої короткострокової пам'яті (Long Short-Term Memory, LSTM). Запропонована Хохрайтером та Шмідхубером [12], ця архітектура вводить поняття стану клітини (cell state) та трьох регулятивних механізмів - «воріт» (gates): вхідних, вихідних та воріт забування (forget gate). LSTM здатна вибірково вирішувати, яку інформацію з минулого зберегти для майбутніх прогнозів, а яку - відкинути як інформаційний шум. Це робить LSTM надзвичайно

потужним інструментом для прогнозування довгострокових трендів [5]. Однак складна система вентилів робить цю модель обчислювально важкою та схильною до перенавчання (overfitting) на зашумлених фінансових даних з малою вибіркою.

2. Модель керованого рекурентного блоку (Gated Recurrent Unit, GRU). Запропонована значно пізніше, у 2014 році, командою дослідників на чолі з К. Чо [6, 8], архітектура GRU є оптимізованою та структурно спрощеною версією LSTM. GRU об'єднує стан клітини та прихований стан в один загальний вектор, а також скорочує кількість керуючих вентилів до двох (update gate та reset gate). Завдяки меншій кількості параметрів для навчання, моделі GRU потребують менше обчислювальних ресурсів, швидше конвергують (навчаються) і часто демонструють вищу стійкість та здатність до узагальнення на високоволатильних, нестаціонарних даних, до яких належать часові ряди криптовалют [8].

Незважаючи на інтерпретованість та швидкість алгоритмів класичного машинного навчання (Random Forest, SVM), їхня неспроможність природним шляхом утримувати часовий контекст робить їх субоптимальними для високочастотного алгоритмічного трейдингу та прогнозування складних фінансових ринків.

Натомість, глибоке навчання, зокрема архітектури LSTM та GRU, пропонує потужний математичний апарат для моделювання складних темпоральних залежностей. Саме тому в даній кваліфікаційній роботі для реалізації системи прогнозування криптовалютних ринків обрано рекурентні мережі глибокого навчання. Особливий науковий інтерес становить емпіричне порівняння важкої, але потужної архітектури LSTM з її більш сучасною та легкою модифікацією GRU в умовах жорсткої Walk-Forward-валідації та оцінки реальних фінансових ризиків.

1.3. Проблематика оцінки ризиків та валідації моделей на нестаціонарних даних

Розробка складної архітектури нейронної мережі (наприклад, LSTM або GRU) є лише першим етапом у створенні ефективної системи прогнозування. Другим, і часто більш критичним етапом, є методологічно правильна валідація цієї моделі та адекватна оцінка її фінансової результативності в умовах реального ринку. На цьому етапі більшість досліджень, які намагаються застосувати алгоритми машинного навчання до фінансових часових рядів, стикаються з двома фундаментальними проблемами: "витоком даних" (data leakage) під час тестування та відривом статистичних метрик від реальних фінансових ризиків [19].

У класичному машинному навчанні (наприклад, при розпізнаванні зображень або класифікації текстів) золотим стандартом перевірки моделі є перехресна перевірка (k-fold cross-validation). Цей метод передбачає випадкове перемішування даних та їхній поділ на тренувальні й тестові блоки. Проте застосування такого підходу до часових рядів є грубою методологічною помилкою. Випадкове перемішування руйнує хронологічну структуру ринку та призводить до феномену "заглядання в майбутнє" (look-ahead bias) - ситуації, коли модель під час навчання отримує доступ до інформації, яка в реальному часі їй була б ще невідома [22].

Щоб уникнути цього, при роботі з фінансовими даними застосовується метод Walk-Forward-валідації (крокової або ковзної валідації). Цей підхід суворо імітує реальні умови роботи трейдера або алгоритмічного фонду:

1. Модель навчається на історичному вікні.
2. Тестування (прогнозування) відбувається на невеликому "небаченому" проміжку часу одразу після закінчення тренувального вікна (наприклад, наступні 3 місяці).
3. Після цього тренувальне вікно зсувається вперед по осі часу, і модель перенавчається з урахуванням нових даних [22].

Такий підхід вимагає значних обчислювальних потужностей, оскільки модель доводиться перенавчати десятки разів (для кожного кроку або фолду), але лише він гарантує, що отримані результати тестування не є ілюзією, викликаною перенавчанням (overfitting) на історичних даних.

Після отримання прогнозів виникає проблема їхньої оцінки. У технічній літературі з Deep Learning якість регресійних моделей зазвичай оцінюється за допомогою таких метрик, як середня абсолютна помилка (MAE) або корінь із середньоквадратичної помилки (RMSE) [4]. Ці метрики показують, наскільки близько згенерована ціна лежить до реальної кривої.

Однак у фінансовій інженерії висока точність ML-метрики не гарантує прибутковості або безпеки торгової стратегії [13]. Наприклад, модель може мати ідеальний показник RMSE, але при цьому стабільно запізнюватися на один день під час розворотів тренду. У такому випадку трейдер, який використовує ці прогнози, зазнає катастрофічних збитків. Більше того, жодна стандартна ML-метрика не враховує поняття ризику капіталу [4].

З огляду на екстремальну волатильність криптовалют (зокрема, Bitcoin та Ethereum), система оцінки обов'язково повинна включати фінансові метрики управління ризиками (Risk Management):

1. Коефіцієнт Шарпа (Sharpe Ratio). Розроблений нобелівським лауреатом Вільямом Шарпом [25], цей показник є індустріальним стандартом для оцінки якості інвестиційної стратегії. Він розраховується як відношення середньої прибутковості стратегії (за вирахуванням безризикової ставки) до стандартного відхилення цієї прибутковості (волатильності). Коефіцієнт Шарпа демонструє, скільки одиниць прибутку отримує інвестор за кожну одиницю прийнятого ризику. Чим вищий показник, тим безпечнішою і стабільнішою є стратегія.

2. Максимальне просідання (Maximum Drawdown). Це ключова метрика для оцінки найгіршого можливого сценарію [13]. Вона вимірює найбільше відсоткове падіння капіталу від історичного максимуму (піку) до найнижчої точки (дна) до того, як буде досягнуто новий максимум. Навіть

якщо модель показує 500% річного прибутку, але має Maximum Drawdown на рівні 90%, вона вважається нежиттєздатною для реального застосування, оскільки такий рівень ризику призведе до втрати депозиту при найменшому маржинальному забезпеченні.

Таким чином, розробка повноцінної системи прогнозування криптовалютних ринків не може обмежуватися лише пошуком архітектури нейромережі з найменшим показником RMSE. Необхідний комплексний підхід, який об'єднує потужність глибокого навчання (для пошуку прихованих патернів) зі суворим бектестуванням (Walk-Forward) та оцінкою реальних фінансових метрик ризику.

Висновки до розділу 1

Криптовалютний ринок характеризується цілодобовим режимом торгів, безперервністю даних, високою волатильністю та нестаціонарністю часових рядів, що робить традиційні методи фундаментального та технічного аналізу, а також лінійні економетричні моделі недостатньо ефективними для побудови автоматизованих систем прогнозування.

Алгоритми класичного машинного навчання мають обмеження при роботі з часовими рядами через відсутність внутрішньої архітектурної пам'яті. Вирішенням цієї проблеми є методи глибокого навчання, зокрема рекурентних нейронних мереж в архітектурах LSTM та GRU, здатних виявляти довгострокові приховані нелінійні залежності у часових послідовностях.

Валідація інтелектуальних систем на фінансових ринках вимагає відмови від стандартної перехресної перевірки з метою недопущення "витоку даних" та "заглядання в майбутнє". Єдиним методологічно коректним підходом є застосування Walk-Forward-валідації.

Оцінка якості моделей машинного навчання для алгоритмічної торгівлі не може спиратися виключно на метрики точності регресії (RMSE, MAE). Для забезпечення цінності роботи необхідно інтегрувати показники фінансового ризику, насамперед Коефіцієнт Шарпа та Максимальне просідання капіталу.

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ ТА МАТЕМАТИЧНИЙ ОПИС МОДЕЛЕЙ

2.1. Архітектурна основа та математичний опис рекурентних мереж (LSTM та GRU)

Прогнозування часових рядів на фінансових ринках, зокрема на ринку криптовалют, вимагає використання математичних моделей, здатних враховувати хронологічну послідовність та темпоральні залежності в даних [24]. Класичні штучні нейронні мережі прямого поширення (Feedforward Neural Networks) сприймають кожен вхідний вектор як незалежну сутність, що унеможливлює утримання контексту попередніх ринкових станів [7]. Для подолання цього обмеження застосовуються рекурентні нейронні мережі (Recurrent Neural Networks, RNN), архітектура яких містить зворотні зв'язки, що дозволяють передавати інформацію з попередніх кроків обчислення на поточні [11].

Математично базовий рекурентний шар обчислює прихований стан h_t на кроці time_step=t за такою формулою (2.1):

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h), \quad (2.1)$$

де x_t - вектор вхідних ознак на поточному кроці (наприклад, OHLCV-показники криптовалюти за день t);

h_{t-1} - вектор прихованого стану з попереднього кроку t−1, який виступає в ролі динамічної пам'яті системи;

W_{hh} та W_{xh} - матриці вагових коефіцієнтів, що оптимізуються під час навчання;

b_h - вектор зсуву (bias);

tanh - гіперболічний тангенс, що використовується як нелінійна функція активації.

Проте класичні RNN мають фундаментальне математичне обмеження - проблему затухання та вибуху градієнта (vanishing/exploding gradient) під час оптимізації за допомогою алгоритму зворотного поширення помилки в часі (Backpropagation Through Time, BPTT) [11]. Коли довжина вхідної послідовності (розмір вікна) перевищує 10–15 кроків, градієнт функції втрат, що перемножується на вагові коефіцієнти на кожному кроці, прямує до нуля або нескінченності. Як наслідок, мережа втрачає здатність до довгострокового навчання, фокусуючись лише на найближчих спостереженнях, що є неприпустимим для прогнозування фінансових трендів.

Для вирішення проблеми довгострокових залежностей у глибокому навчанні (Deep Learning) було розроблено спеціалізовані архітектури, що використовують механізми внутрішнього керування інформаційними потоками за допомогою вентилів (gates) [5]. Основними та найбільш ефективними представниками цього класу є моделі LSTM та GRU.

Архітектура LSTM (Long Short-Term Memory), запропонована С. Хохрайтером та Й. Шмідхубером [12], вирішує проблему затухання градієнта шляхом введення додаткового вектора стану - стану комірки (cell state, Ct), який проходить транзитом крізь усю послідовність обчислень, зазнаючи лише лінійних модифікацій. Зміна стану комірки регулюється трьома послідовними математичними вентилями (генераторами масок у діапазоні [0,1] на основі логістичної функції):

1. Вентиль забування (Forget Gate, f_t). Визначає, яку частку інформації з попереднього стану комірки C_{t-1} необхідно видалити з пам'яті системи. Обчислюється на основі поточного входу x_t та прихованого стану h_{t-1} за формулою (2.2):

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{2.2}$$

де: f_t — вектор активації вентиля забування (містить значення від 0 до 1, де 0 означає «повністю забути», а 1 — «повністю зберегти»);

σ — логістична активаційна функція (сигмоїда), яка нелінійно відображає значення у діапазон [0, 1];

W_f — матриця вагових коефіцієнтів для вентиля забування;

$[h_{\{t-1\}}, x_t]$ — конкатенація вектора прихованого стану з попереднього часового кроку ($h_{\{t-1\}}$) та вектора поточних вхідних даних (x_t);

b_f — вектор зсуву (bias) вентиля забування.

2. Вхідний вентиль (Input Gate, i_t) та кандидат на оновлення пам'яті ($\tilde{C}_t$). Вхідний вентиль визначає, які нові значення будуть оновлені в стані комірки. Він обчислюється на основі поточного входу x_t та прихованого стану $h_{\{t-1\}}$ за формулою (2.3):

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{2.3}$$

Паралельно шар з активацією $tanh$ створює вектор кандидатів $\tilde{C}_t$, що містить нову інформацію, витягнуту з поточного кроку. Цей процес описується формулою (2.4):

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c), \tag{2.4}$$

де: i_t — вектор активації вхідного вентиля (містить значення від 0 до 1, де 0 означає «повністю ігнорувати нову інформацію», а 1 — «повністю записати»);

$\tilde{C}_t$ — вектор нових значень-кандидатів, які формуються для можливого включення до стану комірки;

σ — логістична активаційна функція (сигмоїда);

tanh — гіперболічна тангенціальна активаційна функція, яка масштабує значення кандидатів у діапазон [-1, 1] для стабілізації обчислень та запобігання надмірному зростанню значень;

W_i, W_c — матриці вагових коефіцієнтів для вхідного вентиля та шару кандидатів відповідно;

$[h_{\{t-1\}}, x_t]$ — конкатенація вектора прихованого стану з попереднього кроку ($h_{\{t-1\}}$) та вектора поточного входу (x_t);

b_i, b_c — вектори зсуву (bias) для відповідних шарів.

3. Оновлення стану комірки (C_t). Поточний стан комірки обчислюється як поелементний добуток (адамаровий добуток, позначений як $*$) старого стану на маску забування плюс додавання нової інформації, зваженої вхідним вентилем. Розрахунок нового стану виконується за формулою (2.5):

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t, \tag{2.5}$$

де: C_t — новий (оновлений) стан комірки для поточного часового кроку t, який передається на наступний крок;

$C_{\{t-1\}}$ — стан комірки з попереднього часового кроку;

$f_t * C_{\{t-1\}}$ — операція «забування», що визначає, яку частину старої інформації з пам'яті необхідно відкинути на основі вектора вентиля забування f_t;

$i_t * \tilde{C}_t$ — операція «оновлення», яка масштабує нові значення-кандидати $\tilde{C}_t$ відповідно до рівня їхньої значущості i_t і додає їх до довготривалої пам'яті;

$*$ — оператор поелементного множення векторів (добуток Адамара).

4. Вихідний вентиль (Output Gate, o_t) та фінальний прихований стан (h_t). Вихідний вентиль формує маску для вихідного сигналу, визначаючи, яка саме частина інформації з довготривалої пам'яті (стану комірки) буде виведена назовні як прихований стан. Активація вихідного вентиля обчислюється за формулою (2.6):

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{2.6}$$

Фінальний прихований стан h_t, який передається на наступний часовий крок (а також використовується шарами вищого рівня для генерації фінального прогнозу мережі), обчислюється як фільтрація поточного стану комірки C_t, активованого через функцію tanh. Цей розрахунок виконується за формулою (2.7):

$$h_t = o_t * tanh(C_t), \tag{2.7}$$

де: o_t — вектор активації вихідного вентиля (містить значення від 0 до 1, які діють як фільтр, що пропускає лише релевантну частину інформації);

h_t — новий (фінальний) прихований стан комірки для поточного часового кроку t, що виконує роль короткострокової пам'яті та вихідного сигналу блоку LSTM;

σ — логістична активаційна функція (сигмоїда);

tanh — гіперболічна тангенціальна активаційна функція, яка стискає значення поточного стану комірки C_t у діапазон [-1, 1], що допомагає підтримувати стабільність градієнтів;

W_o — матриця вагових коефіцієнтів для вихідного вентиля;

b_o — вектор зсуву (bias) вихідного вентиля;

$[h_{\{t-1\}}, x_t]$ — конкатенація вектора прихованого стану з попереднього кроку ($h_{\{t-1\}}$) та вектора поточних вхідних даних (x_t);

C_t — оновлений стан комірки (довготривала пам'ять) на поточному кроці t, розрахований на попередньому етапі;

$*$ — оператор поелементного множення векторів (добуток Адамара, у тексті оригіналу позначений як *).

У всіх формулах $\sigma(x) = \frac{1}{1+e^{-x}}$ - сигмоїдна функція активації, що стискає значення в інтервал (0,1), де 0 означає «повністю заблокувати потік», а 1 - «повністю пропустити».

Математичний опис моделі керованого рекурентного блоку (GRU). Архітектура GRU (Gated Recurrent Unit), розроблена К. Чо та його колегами у 2014 році [6, 8], є сучасною оптимізованою модифікацією LSTM. Основна ідея GRU полягає в спрощенні структури комірки з метою зменшення кількості параметрів (ваг), що підлягають оптимізації, та прискорення процесу навчання на великих масивах даних без втрати предиктивної здатності [8].

GRU повністю відмовляється від окремого стану комірки Ct, об'єднуючи його з прихованим станом ht. Замість трьох вентилів у GRU функціонують лише два:

1. Вентиль оновлення (Update Gate, zt). Виконує одночасно роль вентиля забування та вхідного вентиля з архітектури LSTM. Він визначає, який обсяг інформації з минулого (h_{t-1}) необхідно зберегти, і який обсяг нової інформації буде записано в поточний стан, що описується формулою (2.8):

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z), \tag{2.8}$$

де: z_t — вектор активації вентиля оновлення (значення від 0 до 1);

σ — логістична активаційна функція (сигмоїда);

W_z — матриця вагових коефіцієнтів для вентиля оновлення;

$[h_{\{t-1\}}, x_t]$ — конкатенація вектора прихованого стану з попереднього кроку та вектора поточних вхідних даних;

b_z — вектор зсуву (bias) вентиля оновлення.

2. Вентиль скидання (Reset Gate, rt). Визначає, наскільки сильно інформація з минулого прихованого стану h_{t-1} впливає на процес формування поточного кандидата на оновлення. Якщо значення вентиля близьке до 0, мережа повністю "забуває" попередній контекст, що дозволяє ефективно моделювати раптові структурні зсуви тренду, які притаманні криптовалютному ринку. Обчислення виконується за формулою (2.9):

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r), \tag{2.9}$$

де: r_t — вектор активації вентиля скидання (значення від 0 до 1, де 0 означає ігнорування попереднього стану);

W_r — матриця вагових коефіцієнтів для вентиля скидання;

b_r — вектор зсуву (bias) вентиля скидання.

3. Кандидат на прихований стан ($\tilde{h}_t$). Обчислюється з використанням механізму скидання минулої пам'яті за формулою (2.10):

$$\tilde{h}_t = tanh(W \cdot [r_t * h_{t-1}, x_t] + b), \tag{2.10}$$

де: $\tilde{h}_t$ — вектор нових значень-кандидатів для поточного кроку;

tanh — гіперболічна тангенціальна активаційна функція (діапазон [-1, 1]);

$r_t * h_{\{t-1\}}$ — поелементне множення (добуток Адамара) маски вентиля скидання на попередній прихований стан (саме тут відбувається «скидання» нерелевантної інформації з минулого);

W — матриця вагових коефіцієнтів для шару кандидата;

b — вектор зсуву (bias) для шару кандидата.

4. Фінальний прихований стан (h_t). Зміна стану відбувається за допомогою лінійної інтерполяції між минулим станом h_{t-1} та кандидатом $\tilde{h}_t$ на основі ваг вентиля оновлення z_t, згідно з формулою (2.11):

$$h_t = (1 - z_t) * h_t - 1 + z_t * \tilde{h}_t, \quad (2.11)$$

де: h_t — новий (фінальний) прихований стан мережі для поточного часового кроку t;

$1 - z_t$ — інвертоване значення вентиля оновлення, що визначає частку збереження старої інформації;

$(1 - z_t) * h_{\{t-1\}}$ — частина інформації з попереднього кроку, яку система вирішила залишити;

$z_t * \tilde{h}_t$ — нова інформація (кандидат), відмасштабована на рівень її значущості;

$*$ — оператор поелементного множення векторів.

Порівняльний аналіз математичної структури моделей. З погляду математичної складності, комірка LSTM містить чотири лінійні шари (вагові матриці для $f_t, i_t, \tilde{C}_t, o_t$), тоді як комірка GRU - лише три (для $z_t, r_t, \tilde{h}_t$). Це означає, що при однаковій кількості внутрішніх нейронів модель GRU має приблизно на 25% менше параметрів для навчання, ніж LSTM [8].

В умовах дослідження криптовалютного ринку, де історичні ряди містять високу концентрацію інформаційного шуму та випадкових сплесків (волатильності), зменшена кількість параметрів у GRU знижує ризик структурного перенавчання моделі на тренувальній вибірці. Простіша математична структура забезпечує кращу узагальнюючу здатність моделі на тестових кроках Walk-Forward-валідації.

Для глибшого розуміння внутрішніх процесів та інтерактивного аналізу того, як саме описані математичні вентилі LSTM та GRU фільтрують ринкові сигнали та модифікують внутрішню пам'ять залежно від поточного стану волатильності, нижче наведено інтерактивний симулятор внутрішньоструктурної динаміки рекурентних комірок.

2.2. Проєктування архітектури системи прогнозування та підготовка вхідних даних

Ефективність моделей глибокого навчання (Deep Learning) критично залежить не лише від вибору правильного математичного апарату, але й від архітектури самої мережі та якості підготовки вхідних даних. Процес проєктування інтелектуальної системи прогнозування для криптовалютних ринків складається з двох ключових етапів: специфічного перетворення одновимірного часового ряду у багатовимірний тензор та конструювання топології рекурентної мережі.

Вхідними даними для системи слугують історичні ринкові показники криптовалют (Bitcoin та Ethereum) у стандартному біржовому форматі OHLCV (Open, High, Low, Close, Volume). Цільовою змінною для прогнозування обрано ціну закриття (Close) наступного торгового дня [20].

Оскільки ціни на криптовалюти можуть коливатися в діапазоні від кількох доларів до десятків тисяч доларів, безпосередня подача таких «сирих» даних у нейронну мережу є неможливою. Використання ненормалізованих значень призводить до того, що під час обчислення активаційних функцій (особливо гіперболічного тангенса та сигмоїди, які є базовими для LSTM та

GRU) градієнти експоненційно зростають, що викликає математичну нестабільність алгоритму зворотного поширення помилки [11].

Для вирішення цієї проблеми у спроєктованій системі застосовується метод лінійного масштабування (Min-Max Scaling) [23]. Усі фінансові показники стискаються у стандартизований діапазон [0,1] за формулою (2.12):

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}, \quad (2.12)$$

де: $X_{\{scaled\}}$ — нове, масштабоване (нормалізоване) значення фінансового показника, яке зведено до діапазону від 0 до 1;

X — поточне (вихідне) абсолютне значення показника до масштабування;

$X_{\{min\}}$ — мінімальне зафіксоване значення цього показника в межах набору даних (навчальної вибірки);

$X_{\{max\}}$ — максимальне зафіксоване значення цього показника в межах набору даних.

Важливою архітектурною деталлю системи є збереження об'єкта масштабування (scaler) у пам'яті. Це необхідно для того, щоб на етапі оцінки ризиків мати змогу здійснити зворотне перетворення (inverse transform) прогнозів мережі зі стандартизованого діапазону назад у реальні долари та розрахувати фінансові метрики [24].

Традиційні моделі машинного навчання працюють із двовимірними таблицями (Зразки × Ознаки). Однак рекурентні мережі (RNN) вимагають подачі даних у вигляді тривимірного тензора: (Кількість зразків, Кількість часових кроків, Кількість ознак).

Для формування такого тензора використовується алгоритм "ковзного вікна" [5]. Суть методу полягає в тому, що мережа використовує певну кількість попередніх днів для прогнозування одного наступного. У

розробленій системі емпіричним шляхом було встановлено оптимальний розмір вікна (Sequence Length), який дорівнює 30 дням.

Математично це означає, що для прогнозування ціни закриття у день t, мережа отримує на вхід матрицю X розміром 30×5, яка містить усі показники OHLCV за дні від $t - 30$ до $t - 1$. Після створення одного зразка, "вікно" зсувається вперед на один день, генеруючи наступний зразок для навчання (рис 2.1). Такий підхід дозволяє перетворити задачу аналізу часового ряду на задачу керованого навчання (Supervised Learning).

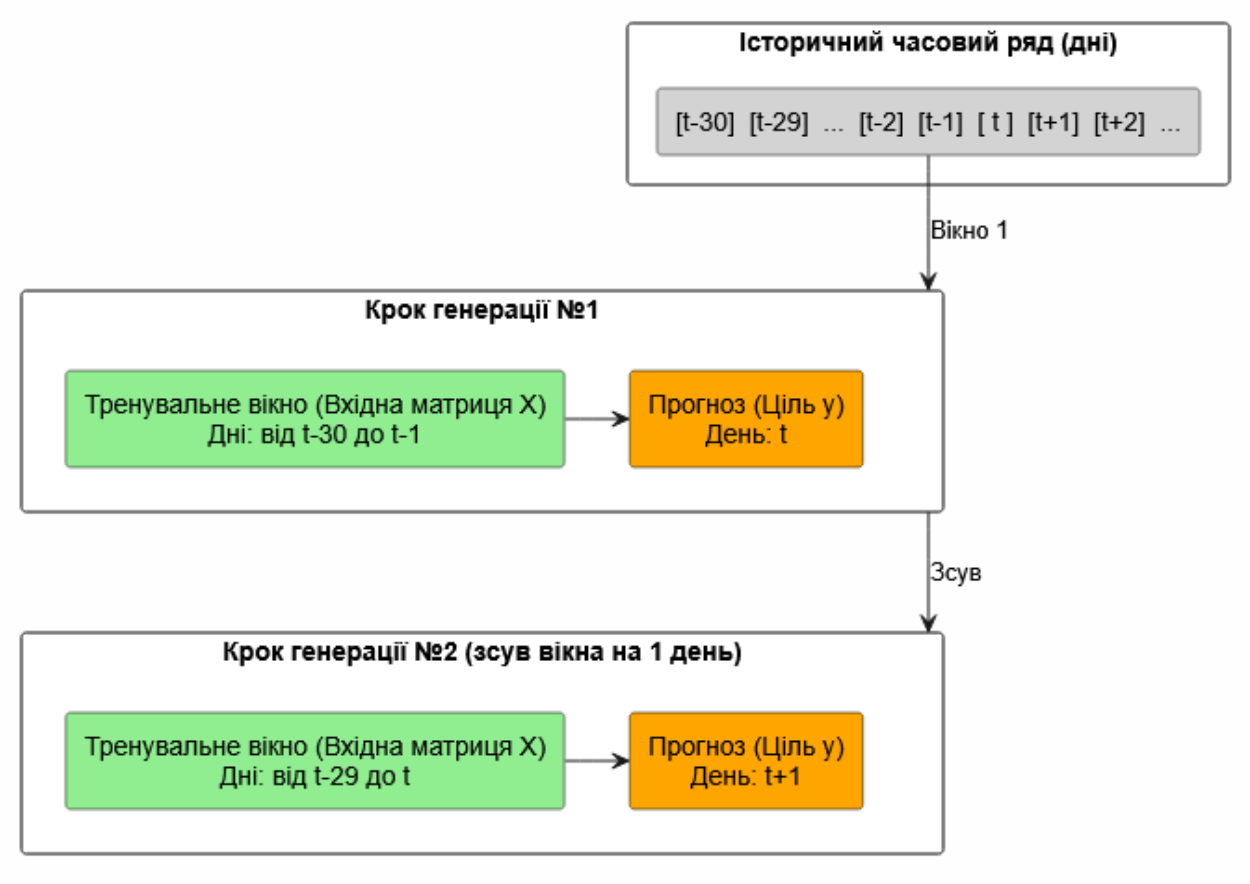

Рисунок 2.1 - Процес формування багатовимірних тензорів методом ковзного вікна

Для забезпечення чистоти експерименту під час порівняльного аналізу архітектур LSTM та GRU, обидві нейронні мережі були спроєктовані з ідентичною загальною топологією та однаковою кількістю прихованих шарів

[16]. Проєктування здійснювалося з урахуванням балансу між здатністю моделі вловлювати складні патерни (Model Capacity) та необхідністю запобігання перенавчанню (Overfitting).

Спроєктована архітектура системи має наступну структуру (див рис 2.2):

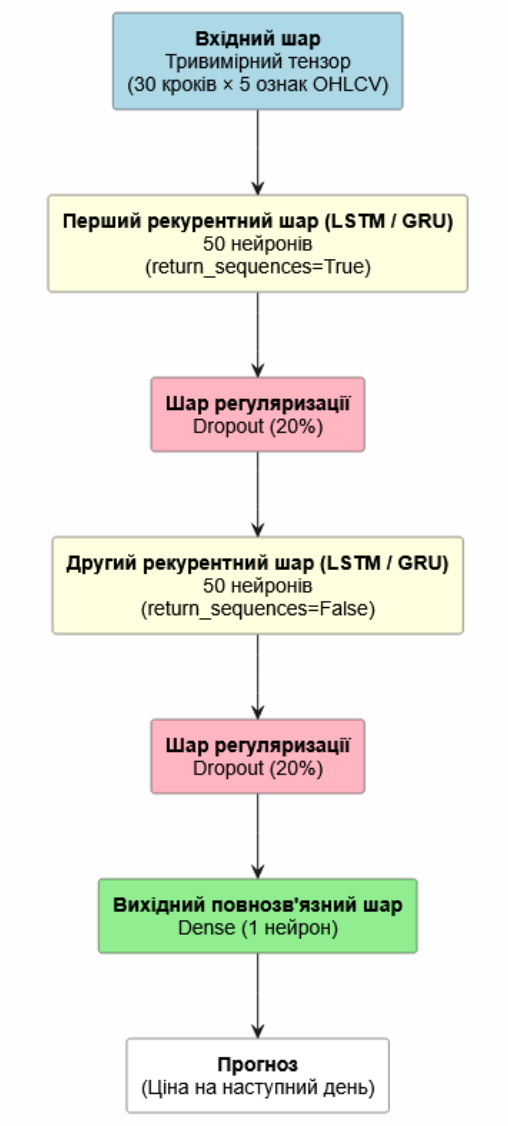

Рисунок 2.2 - Архітектура спроєктованої рекурентної нейронної мережі для прогнозування ціни

1. Вхідний шар (Input Layer): Приймає тривимірний тензор форми (30, 5) - 30 часових кроків та 5 ознак.
2. Перший рекурентний шар (LSTM або GRU): Складається з 50 нейронів. Цей шар налаштований на повернення повної

послідовності обчислень (return_sequences=True). Це означає, що замість одного фінального вектора, він генерує вихідний сигнал для кожного з 30 кроків, передаючи розгорнутий контекст на наступний рівень абстракції [7].

3. Перший шар регуляризації (Dropout): Із коефіцієнтом 0.2 (20%). Цей механізм випадковим чином "вимикає" 20% нейронів під час кожної ітерації навчання. Це змушує мережу не покладатися на конкретні вузли та запобігає простому запам'ятовуванню тренувального датасету [11].
4. Другий рекурентний шар (LSTM або GRU): Також складається з 50 нейронів, але працює в режимі агрегації (return_sequences=False). Він збирає всю оброблену інформацію та видає єдиний вектор прихованого стану, який акумулює знання про весь 30-денний період.
5. Другий шар регуляризації (Dropout): Із коефіцієнтом 0.2 (20%).
6. Вихідний повнозв'язний шар (Dense Layer): Складається з 1 лінійного нейрона, який генерує фінальне число - скальований прогноз ціни криптовалюти на наступний день.

Процес оптимізації вагових коефіцієнтів мережі здійснюється за допомогою алгоритму Adam (Adaptive Moment Estimation), який поєднує переваги стохастичного градієнтного спуску та алгоритму RMSProp, забезпечуючи адаптивну зміну швидкості навчання [7]. Як цільова функція втрат (Loss Function), яку мережа намагається мінімізувати, використовується середньоквадратична помилка (MSE).

Ключовим архітектурним рішенням у системі є впровадження механізму алгоритмічного контролю епох - Early Stopping. Базовий гіперпараметр встановлює максимальну тривалість навчання у 50 епох. Однак система безперервно моніторить помилку на валідаційній вибірці. Якщо протягом 5 епох поспіль (параметр patience=5) помилка не зменшується, процес навчання автоматично зупиняється. Після цього система відновлює конфігурацію вагових коефіцієнтів з тієї епохи, де було зафіксовано найкращий результат

(restore_best_weights=True) [16]. Такий підхід дозволяє адаптивно знаходити оптимальний момент завершення навчання для кожного окремого кроку Walk-Forward-валідації, економлячи обчислювальні ресурси та гарантуючи найвищу здатність моделі до узагальнення ринкових даних.

2.3. Алгоритм Walk-Forward-валідації та обґрунтування метрик оцінки ризику

Розробка глибокої нейронної мережі є необхідною, але не достатньою умовою для створення надійної торгової системи. Критичним етапом проєктування є методологія тестування (валідації) моделі на історичних даних. Неправильний вибір методу тестування є найпоширенішою причиною того, чому моделі з ідеальними показниками в лабораторних умовах зазнають катастрофічних фінансових збитків під час реальних торгів (live trading) [22].

У класичному машинному навчанні стандартом де-факто є використання k-fold крос-валідації. Цей метод випадковим чином перемішує весь набір даних і ділить його на k рівних частин, почергово використовуючи кожну частину як тестову. Однак для фінансових часових рядів такий підхід є методологічно неприпустимим, оскільки він порушує хронологічну цілісність даних [19]. Випадкове перемішування призводить до того, що для прогнозування подій у минулому модель навчається на даних з майбутнього. Цей ефект відомий як "заглядання в майбутнє" (look-ahead bias) і повністю нівелює наукову цінність тестування.

Для усунення цього недоліку в проєктованій системі застосовується метод Walk-Forward-валідації (крокової або ковзної валідації) [22] (див. рис. 2.3). Цей алгоритм суворо імітує реальні умови функціонування торгового алгоритму: він приймає рішення лише на основі тих даних, які були б хронологічно доступні йому в момент прийняття рішення. Такий підхід повністю виключає ризик витоку інформації з майбутнього (look-ahead bias) та дозволяє періодично перенавчати модель, забезпечуючи її динамічну адаптацію до мінливих ринкових умов.

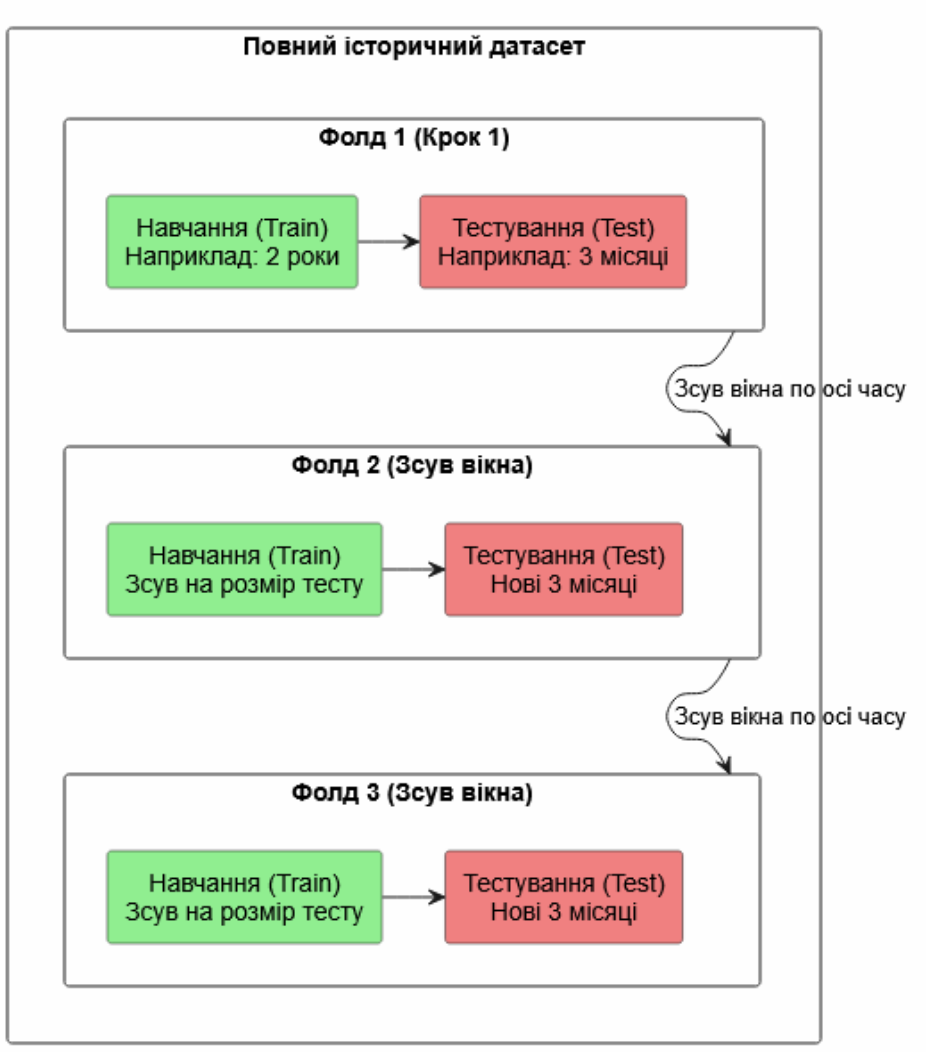

Рисунок 2.3 - Механізм тестування моделей за алгоритмом Walk-Forward-валідації

Процес Walk-Forward складається з наступних кроків:

1. Ініціалізація вікон: Визначається розмір тренувального вікна (наприклад, 2 роки або 730 днів) та розмір тестового вікна (наприклад, 3 місяці або 90 днів).
2. Перший фолд (крок): Нейромережа навчається на даних від дня t_0 до t_{730}. Тестування (генерація прогнозів) відбувається на повністю "небачених" даних від дня t_{731} до t_{820}.
3. Ковзання вікна (Sliding): Після завершення першого тестування вікно зміщується вперед на розмір тестової вибірки. Тепер модель навчається на періоді від t_{90} до t_{820}, а тестується на $t_{821} - t_{910}$.

4. Перенавчання: На кожному новому кроці вагові коефіцієнти моделі скидаються або донавчаються з урахуванням нових макроекономічних та ринкових умов. Процес повторюється до досягнення кінця історичного датасету.

У рамках даної дипломної роботи датасет Біткоїна та Ефіріуму дозволив згенерувати 24 кроки (фолди) Walk-Forward-валідації. Це означає, що спроєктовані моделі LSTM та GRU пройшли перевірку на 24 незалежних історичних відрізках різної волатильності, що забезпечує високу статистичну достовірність результатів.

Оскільки спроєктована система вирішує задачу регресії (прогнозування безперервної величини - ціни), первинний аналіз її точності виконується за допомогою стандартних метрик машинного навчання [4]:

1. Корінь із середньоквадратичної помилки (RMSE): Штрафує модель за великі відхилення від реальної ціни (викиди). Обчислюється за формулою (2.13):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{2.13}$$

де: n — загальна кількість спостережень у тестовій вибірці;
y_i — фактичне (реальне) значення ціни для і-го кроку;
$\hat{y}_i$ — спрогнозоване моделлю значення ціни для і-го кроку.

2. Середня абсолютна помилка (MAE): Показує середнє арифметичне відхилення прогнозу від реальної ціни у базовій валюті (доларах США). Розраховується за формулою (2.14):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{2.14}$$

де змінні мають ті ж самі значення, що й для формули (2.13).

Покладатися виключно на ML-метрики у трейдингу небезпечно. Нейромережа може мати низький показник MAE, але при цьому систематично генерувати хибні сигнали на ключових ринкових розворотах. Тому для доведення практичної доцільності моделей у систему інтегровано блок розрахунку фінансових метрик, що базується на симуляції торгової стратегії [13]: якщо прогнозована ціна на завтра вища за сьогоднішню, система генерує сигнал на утримання позиції (Long).

Для оцінки якості цієї стратегії застосовуються дві ключові метрики інвестиційного ризик-менеджменту:

1. Коефіцієнт Шарпа (Sharpe Ratio) [25] - вимірює прибутковість активу з поправкою на ризик. Він показує, яку додаткову премію отримує трейдер за одиницю прийнятої на себе волатильності. Оскільки криптовалютний ринок працює без вихідних, застосовується адаптована формула (2.15) з аннуалізацією на 365 днів:

$$Sharpe = \frac{E[R_p - R_f]}{\sigma_p} \approx \frac{\mu_{ret}}{\sigma_{ret}}\sqrt{365}, \quad (2.15)$$

де: $E[R_p - R_f]$ — математичне сподівання різниці між прибутковістю портфеля (R_p) та безризиковою ставкою (R_f);

σ_p — стандартне відхилення (волатильність) прибутковості портфеля;

$\mu_{\{ret\}}$ — середнє значення щоденної прибутковості активу;

$\sigma_{\{ret\}}$ — стандартне відхилення щоденної прибутковості активу;

$\sqrt{\{365\}}$ — коефіцієнт аннуалізації для ринку, що функціонує 365 днів на рік.

Коефіцієнт Шарпа > 1.0 вважається показником високоякісної інвестиційної стратегії.

2. Максимальне просідання (Maximum Drawdown, MaxDD) [13] - індикатор найгіршого сценарію. Відображає максимальну історичну втрату капіталу від локального піку до локального дна у відсотках. Цей показник є критично важливим, оскільки надмірне просідання (наприклад, понад 80%) призводить до психологічної відмови інвестора від стратегії або ліквідації маржинальних позицій. Обчислюється за формулою (2.16):

$$MaxDD = \min_{t}\left(\frac{V_t - V_{peak\_t}}{V_{peak\_t}}\right) \times 100\%, \quad (2.16)$$

де: V_t — вартість інвестиційного портфеля (капіталу) у поточний момент часу t;

$V_{\{peak_t\}}$ — максимальна зафіксована вартість портфеля за весь період до моменту часу t (попередній локальний максимум).

Таким чином, комплексна система метрик (RMSE, MAE, Profit, Sharpe Ratio, Max Drawdown) гарантує всебічну та об'єктивну оцінку розроблених моделей глибокого навчання, балансуючи між точністю математичної регресії та суворими реаліями фінансового ризик-менеджменту.

Висновки до розділу 2

Для розв'язання задачі прогнозування криптовалютних часових рядів спроєктовано систему на базі рекурентних нейронних мереж. Вибір на користь архітектур LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit) зумовлений їхньою здатністю долати проблему затухання градієнта та утримувати довгострокову пам'ять про стан ринку завдяки системі внутрішніх керуючих вентилів. З математичної точки зору, модель GRU є більш оптимізованою та має на 25% менше параметрів для навчання порівняно з LSTM.

Проєктування архітектури вхідних даних вимагає обов'язкової лінійної нормалізації біржових показників (Min-Max Scaling) для забезпечення математичної стабільності алгоритму зворотного поширення помилки. Трансформація одновимірного часового ряду у багатовимірний тензор для подачі в рекурентну мережу реалізована за допомогою алгоритму "ковзного вікна" глибиною у 30 торгових днів.

Розроблено топологію нейромереж, яка містить два рекурентні шари (по 50 нейронів) зі вбудованими механізмами регуляризації Dropout (20%) для запобігання перенавчанню. Процес оптимізації контролюється механізмом Early Stopping.

Доведено, що класична перехресна перевірка є непридатною для фінансових даних через ефект "заглядання в майбутнє". У спроєктованій системі інтегровано алгоритм Walk-Forward-валідації (крокового тестування), що забезпечує достовірність симуляції на 24 незалежних часових відрізках.

Сформовано комплексний підхід до оцінки якості моделей. Поряд із класичними показниками точності регресії (RMSE, MAE), система включає розрахунок фінансових метрик - Коефіцієнта Шарпа та Максимального просідання (Max Drawdown), що дозволяє повноцінно оцінити торгові ризики розробленої системи.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ, НАВЧАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1. Вибір інструментарію та програмний препроцесінг історичних даних

Для практичної реалізації спроєктованої системи прогнозування криптовалютних ринків було обрано мову програмування Python (версія 3.9+). Вибір Python обґрунтований його статусом індустріального стандарту в галузі Data Science та наявністю потужної екосистеми спеціалізованих бібліотек для машинного і глибокого навчання [20].

Програмний комплекс розроблено з використанням наступного стеку технологій:

1. TensorFlow та Keras: Використовуються як базовий фреймворк для конструювання, навчання та оптимізації рекурентних нейронних мереж (LSTM та GRU). Keras надає високорівневий API, що дозволяє гнучко налаштовувати багатошарові тензорні обчислення та задіяти апаратне прискорення [16, 27].
2. Pandas та NumPy: Застосовуються для швидкого векторного обчислення, маніпуляцій з часовими рядами та агрегації фінансових датасетів [21].
3. Scikit-learn: Використовується для лінійного масштабування фінансових ознак та розрахунку метрик машинного навчання (RMSE, MAE) [23].
4. Joblib: Бібліотека для серіалізації (збереження на жорсткий диск) об'єктів мови Python, необхідна для експорту налаштувань масштабування.

Як емпірична база дослідження використано набір історичних даних "Cryptocurrency Historical Prices Dataset" з платформи Kaggle [10]. Дослідження проводилося на двох незалежних активах з різною капіталізацією та волатильністю: Bitcoin (файл coin_Bitcoin.csv) та Ethereum (файл

coin_Ethereum.csv). Дані представляють собою щоденні зрізи торгів за кілька років і містять класичні біржові метрики OHLCV.

Увесь процес підготовки даних інкапсульовано у спеціально розробленому програмному модулі preprocessing.py. Першим етапом препроцесінгу є очищення даних від інформаційного шуму. Програмна функція load_and_clean_data зчитує CSV-файл і автоматично видаляє нерелевантні для нейромережі текстові та мета-колонки (наприклад, SNo, Name, Symbol, Marketcap), залишаючи виключно 5 числових фінансових ознак. Після цього колонка з датою конвертується у формат datetime і встановлюється як індекс датафрейму, а самі записи жорстко сортуються в хронологічному порядку для запобігання перемішуванню часового ряду.

Як було зазначено в архітектурному розділі, рекурентні мережі чутливі до абсолютних значень вхідних даних. Для нормалізації датасету реалізовано функцію normalize_data, яка використовує клас MinMaxScaler з бібліотеки Scikit-learn. Всі п'ять колонок стискаються у діапазон від 0 до 1.

Критично важливим програмним рішенням на цьому етапі є збереження натренованого об'єкта скейлера у вигляді бінарного файлу (minmax_scaler.pkl) в директорію models. Це дозволяє у майбутньому (на етапі симуляції торгів) "розтиснути" згенеровані нейромережею прогнози у форматі [0, 1] назад у реальні долари для коректного розрахунку Максимального просідання та прибутку.

Трансформація двовимірної таблиці в необхідний для рекурентних мереж тривимірний тензор виконується функцією create_sequences. Алгоритм ітерується по всьому нормалізованому датасету і нарізає його на блоки:

- Вхідна матриця (X): масив розмірністю (2961, 30, 5) (для Біткоїна), що означає 2961 зразок, кожен з яких містить історію за 30 днів по 5 параметрах.
- Цільовий вектор (y): масив розмірністю (2961,), що містить значення колонки Close на 31-й день (те, що мережа повинна навчитися передбачати).

Для забезпечення суворого тестування без "витоку даних" було розроблено функцію-генератор walk_forward_split. Цей алгоритм динамічно створює індекси для тренувальної та тестової вибірок.

У програмному коді задано параметри: TRAIN_DAYS = 730 (навчання на 2 роках історії) та TEST_DAYS = 90 (тестування на 3 місяцях). Функція послідовно "ковзає" по часовому ряду із кроком, що дорівнює розміру тестової вибірки. В результаті виконання препроцесінгу скрипт генерує 24 незалежні фолди (кроки) для тестування. Ці структуровані блоки даних безпосередньо передаються у модуль навчання нейронних мереж.

3.2. Процес реалізації, тренування нейромереж та оптимізація гіперпараметрів

Після успішного завершення етапу препроцесінгу та формування тривимірних тензорів методом ковзного вікна, наступним завданням є безпосереднє конструювання, компіляція та тренування рекурентних нейронних мереж. Для програмної реалізації спроєктованих архітектур було використано високорівневий API Keras, який інтегровано у фреймворк TensorFlow. Цей інструментарій дозволяє декларативно описувати складні багатошарові графи обчислень.

У розробленому програмному модулі train.py конструювання моделей інкапсульовано у двох окремих функціях: build_lstm(input_shape) та build_gru(input_shape). Обидві функції використовують послідовний (Sequential) підхід до побудови графа, де дані передаються від шару до шару лінійно.

Програмна реалізація кожного блоку повністю відповідає теоретичній моделі, описаній у другому розділі:

1. Ініціалізація: model = Sequential().
2. Перший рекурентний шар: Додається шар LSTM(50) або GRU(50). Параметр return_sequences=True вказує рушію TensorFlow, що шар повинен повертати повну матрицю прихованих станів для кожного з

30 часових кроків, а не лише фінальний результат. Вхідна розмірність input_shape жорстко фіксується як (30, 5).

3. Регуляризація: Наступним шаром додається Dropout(0.2). Під час тренування він програмно обнуляє 20% випадкових активацій, що працює як потужний механізм протидії структурному перенавчанню (overfitting) на зашумлених фінансових даних.
4. Другий рекурентний шар: Додається ще один шар LSTM(50) або GRU(50). Цього разу параметр return_sequences=False гарантує, що рекурентний блок агрегує всю часову послідовність у єдиний вектор знань.
5. Другий Dropout: Ще один шар Dropout(0.2) для закріплення ефекту регуляризації.
6. Вихідний шар: Додається лінійний шар Dense(1), який виконує фінальну регресію та проєктує багатовимірний прихований стан в одне скалярне значення - прогнозовану нормалізовану ціну.

Після визначення топології кожна модель проходить етап компіляції (model.compile). На цьому етапі задаються математичні правила, за якими мережа буде оновлювати свої вагові коефіцієнти:

- Функція втрат (Loss): Оскільки розв'язується задача регресії, оптимізація спрямована на мінімізацію середньоквадратичної помилки loss='mse' (Mean Squared Error).
- Оптимізатор (Optimizer): Використовується алгоритм adam (Adaptive Moment Estimation). На відміну від класичного стохастичного градієнтного спуску (SGD), Adam обчислює індивідуальні адаптивні швидкості навчання (learning rates) для кожного параметра моделі на основі оцінок першого та другого моментів градієнта. Це дозволяє мережі набагато швидше і стабільніше сходитися до глобального мінімуму на нестаціонарних криптовалютних даних.

Процес навчання глибоких мереж вимагає значних обчислювальних ресурсів і жорсткого контролю кількості ітерацій (епох). У системі було задано

базові гіперпараметри: максимальна кількість епох - 50, розмір батчу (кількість зразків для одного кроку оновлення ваг) - 32 (BATCH_SIZE = 32).

Проте фіксована кількість епох часто призводить до перенавчання моделі. Для вирішення цієї проблеми програмно імплементовано механізм автоматичного контролю - клас EarlyStopping (з бібліотеки keras.callbacks). Цей об'єкт відстежує помилку на валідаційній (тестовій) вибірці (monitor='val_loss'). Параметр patience=5 означає, що якщо протягом п'яти епох поспіль помилка не зменшується, рушій TensorFlow примусово перериває цикл навчання. Після цього активується тригер restore_best_weights=True, який автоматично завантажує в модель конфігурацію вагових коефіцієнтів з тієї епохи, де було досягнуто найвищої точності прогнозування.

Головний обчислювальний процес відбувається у макро-циклі, який ітерується по масиву wf_splits, згенерованому на етапі препроцесінгу. Алгоритм виконує наступні кроки для кожного з 24 фолдів (кроків зміщення вікна):

1. Ізоляція даних: З генератора вилучаються масиви X_train, y_train (для навчання) та X_test, y_test (для незалежного тестування).
2. Скидання моделі: Для забезпечення чистоти експерименту на кожному новому кроці Walk-Forward викликаються функції build_lstm та build_gru, які створюють абсолютно нові, "чисті" екземпляри нейромереж з випадковими початковими вагами. Це гарантує, що модель не переносить "пам'ять" про майбутні події з попередніх фолдів.
3. Тренування (Fit): Викликається метод model.fit(), у який передаються тренувальні тензори, параметри епох та батчів, а також механізм callbacks=[early_stop]. Для запобігання витоку інформації тестові дані X_test, y_test передаються лише як validation_data, тобто мережа не навчається на них, а лише використовує їх для розрахунку помилки ранньої зупинки.

4. Прогнозування (Predict): Після завершення навчання, викликається метод model.predict(X_test). Натренована мережа отримує небачені раніше 30-денні вікна і генерує масив прогнозів на наступний день.

Усі отримані прогнози для обох архітектур (LSTM та GRU), а також реальні цільові значення Actual_Scaled, разом із номером поточного фолду, програмно об'єднуються у структуру pandas.DataFrame. По завершенню обчислення всіх 24 циклів, агрегована таблиця автоматично експортується на жорсткий диск у вигляді файлу walk_forward_predictions.csv. Цей файл стає незмінним фундаментом для наступного етапу - програмного розрахунку фінансових метрик, оцінки ризиків та візуалізації результатів дослідження.

3.3. Тестування моделей та порівняльний аналіз результатів і фінансових ризиків

Після програмної реалізації та успішного завершення процесу навчання (тренування) рекурентних нейронних мереж на історичних даних, наступним критичним етапом є їхнє бектестування. Згідно зі спроєктованою методологією, тестування моделей LSTM та GRU відбувалося за алгоритмом Walk-Forward-валідації. Кожна модель згенерувала прогнози на 24 незалежних кроках (тестових вибірках), що дозволило імітувати реальні умови алгоритмічної торгівлі та виключити ефект "заглядання в майбутнє".

Для забезпечення об'єктивності та підтвердження гіпотези про стійкість моделей, експеримент було проведено на двох різних криптовалютних активах: Bitcoin (BTC) - активі з найвищою капіталізацією та помірною волатильністю, та Ethereum (ETH) - активі з вищою транзакційною активністю (on-chain) та більшою історичною волатильністю.

3.3.1. Аналіз точності прогнозування ціни за метриками машинного навчання

На першому етапі аналізу оцінювалася здатність моделей мінімізувати математичну похибку регресії під час прогнозування абсолютної ціни

криптовалют на наступний торговий день. Для цього використовувалися метрики RMSE (корінь із середньоквадратичної помилки) та MAE (середня абсолютна помилка), розраховані після зворотного перетворення (inverse transform) нормалізованих прогнозів у реальні долари США.

Зведені результати метрик точності машинного навчання (ML) представлені в Таблиці 3.1.

Таблиця 3.1 - Порівняння точності регресійних моделей (за результатами 24 кроків Walk-Forward)

Криптовалюта	**Архітектура**	**RMSE (USD)**	**MAE (USD)**	**Перевага (за MAE)**
Bitcoin (BTC)	LSTM	1138.70	481.94	-
Bitcoin (BTC)	**GRU**	**878.43**	**360.74**	**+25.1% краще**
Ethereum (ETH)	LSTM	80.87	38.68	-
Ethereum (ETH)	**GRU**	**65.81**	**31.17**	**+19.4% краще**

Як свідчать результати, архітектура GRU продемонструвала беззаперечну перевагу над LSTM на обох досліджуваних активах. Для Біткоїна середня помилка прогнозування на день склала лише 360 доларів (що є вкрай низьким показником при цінах активу в десятки тисяч доларів), тоді як LSTM помилялася в середньому на 481 долар. Аналогічна динаміка спостерігається і на Ефіріумі (31.17 USD у GRU проти 38.68 USD у LSTM).

Цей емпіричний результат повністю підтверджує теоретичне припущення, висунуте у Розділі 2: спрощена архітектура GRU (з двома керуючими вентилями замість трьох) має меншу схильність до перенавчання на інформаційному шумі. Менша кількість параметрів дозволила моделі GRU швидше та точніше адаптуватися до раптових структурних змін ціни під час ковзання вікна валідації.

Візуальне порівняння здатності моделей відстежувати реальну траєкторію ціни представлено на рисунках нижче (рис. 3.1 та рис. 3.2).

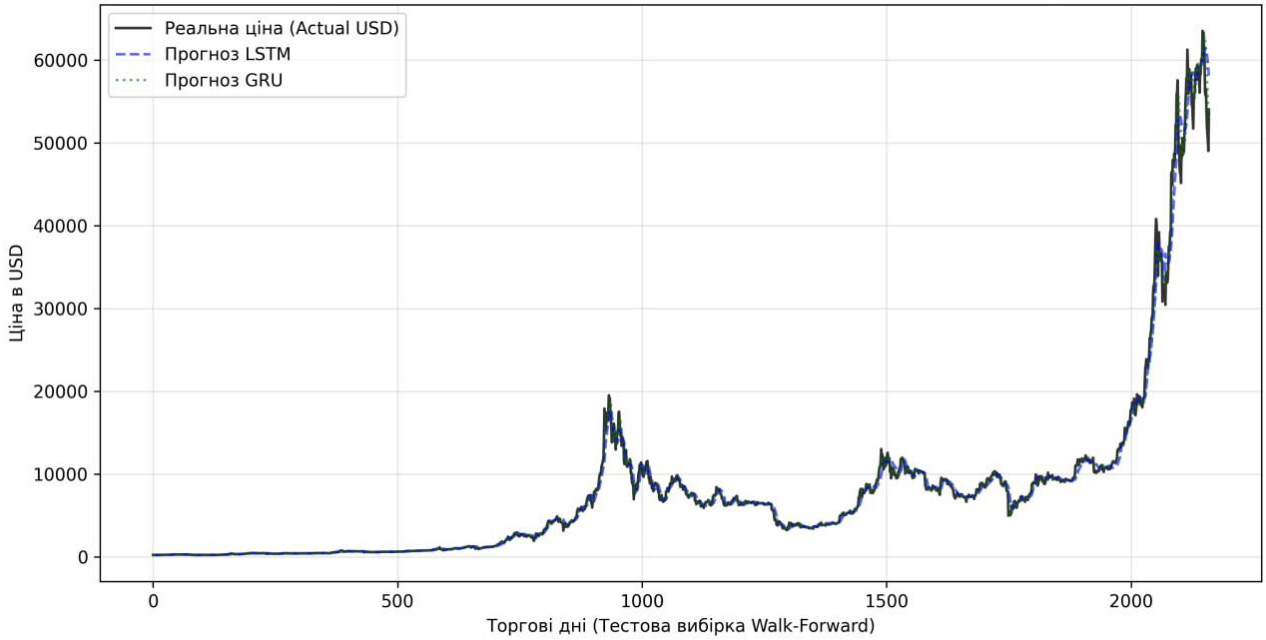

Рисунок 3.1 - Порівняння реальної ціни Bitcoin (BTC) та прогнозів моделей глибокого навчання

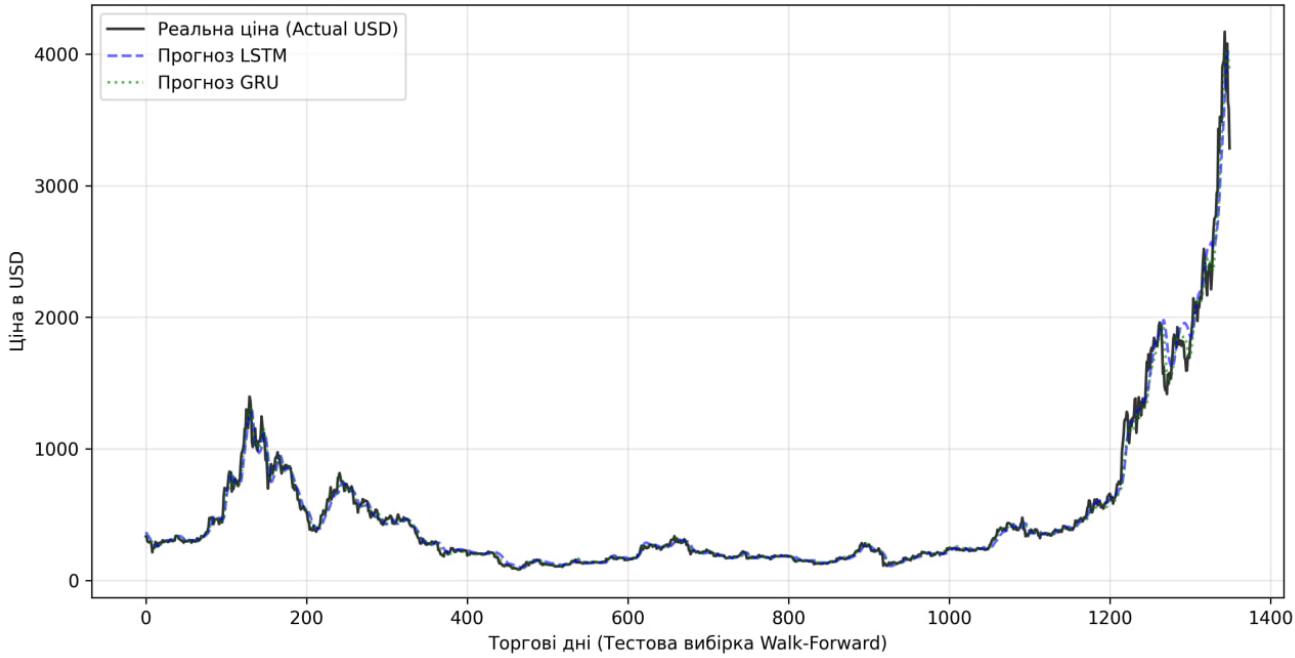

Рисунок 3.2 - Порівняння реальної ціни Ethereum (ETH) та прогнозів моделей глибокого навчання

3.3.2. Аналіз фінансових результатів та оцінка торгових ризиків

Незважаючи на високу математичну точність, здатність моделі генерувати прибуток залежить від правильного визначення напрямку руху ціни (тренду). Для оцінки фінансової ефективності було реалізовано симуляцію торгової стратегії: якщо спрогнозована на завтра ціна перевищує сьогоднішню ціну закриття, алгоритм генерує сигнал «Купувати/Утримувати» (Long). В іншому випадку система перебуває поза ринком або фіксує позицію.

Для комплексної оцінки якості цієї стратегії розраховувалися: загальний накопичений прибуток (Total Profit), індикатор стабільності дохідності з поправкою на ризик - Коефіцієнт Шарпа (Sharpe Ratio), та показник найгіршого сценарію втрати капіталу - Максимальне просідання (Maximum Drawdown). Результати симуляції наведені в Таблиці 3.2.

Таблиця 3.2 - Фінансові метрики та оцінка ризиків торгових стратегій на базі ШІ

Криптовалюта	**Модель**	**Загальний Прибуток**	**Sharpe Ratio**	**Max Drawdown (Просідання)**
Bitcoin (BTC)	LSTM	510.09%	0.84	-71.19%
Bitcoin (BTC)	**GRU**	**879.44%**	**0.99**	**-72.73%**
Ethereum (ETH)	LSTM	114.35%	0.66	-87.59%
Ethereum (ETH)	**GRU**	**633.03%**	**1.13**	**-59.63%**

Результати фінансового бектестування яскраво розкривають проблематику оцінки торгових алгоритмів:

1. Аналіз Bitcoin: Обидві моделі показали надвисоку прибутковість. Модель GRU забезпечила 879% прибутку з Коефіцієнтом Шарпа 0.99, що є показником високої якості (наближається до еталонної одиниці). Проте показник просідання (Max Drawdown) становить понад -72% для обох

архітектур. Це свідчить про те, що хоча моделі вловлюють глобальні "бичачі" тренди, торгівля Біткоїном залишається вкрай ризикованою і потребує жорсткого додаткового ризик-менеджменту (наприклад, алгоритмічних стоп-лосів).

2. Аналіз Ethereum (Розкриття ризику): Тестування на більш волатильному Ефіріумі ідеально демонструє нестабільність складної архітектури. Модель LSTM показала катастрофічні результати в управлінні ризиком: прибутковість склала лише 114%, але Максимальне просідання досягло -87.59%. На практиці таке просідання означає повну втрату інвестиційного капіталу в умовах маржинальної торгівлі, а низький Шарп (0.66) підтверджує хаотичність результатів.

3. Натомість, модель GRU на Ethereum продемонструвала феноменальну стійкість: прибуток склав 633%, просідання було локалізовано на прийнятному для крипторинку рівні -59.63%, а Коефіцієнт Шарпа сягнув 1.13. Значення Шарпа понад 1.0 на такому нестаціонарному активі свідчить про те, що GRU-алгоритм дійсно знайшов стійку математичну закономірність, яка дозволяє обігрувати ринок.

Візуалізація зростання капіталу та графіки фінансових просідань (зони ризику) наведені на рисунках нижче (рис. 3.3 – рис 3.6).

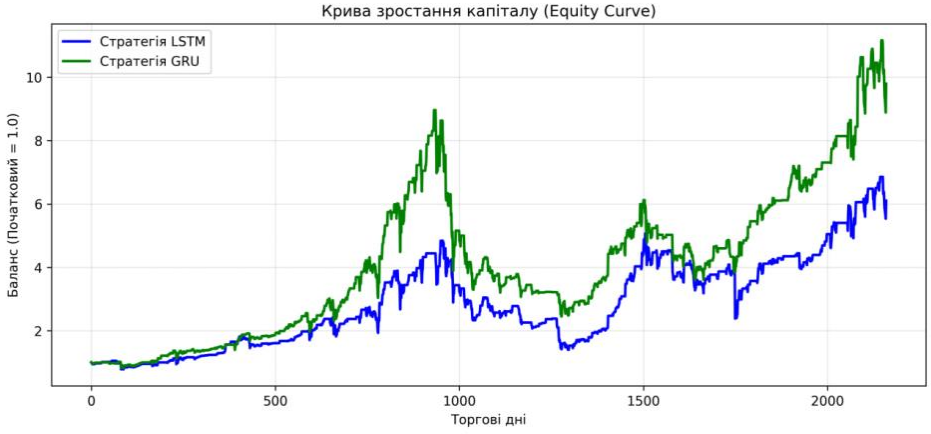

Рисунок 3.3 - Крива зростання капіталу (Equity Curve) для стратегій на базі BTC

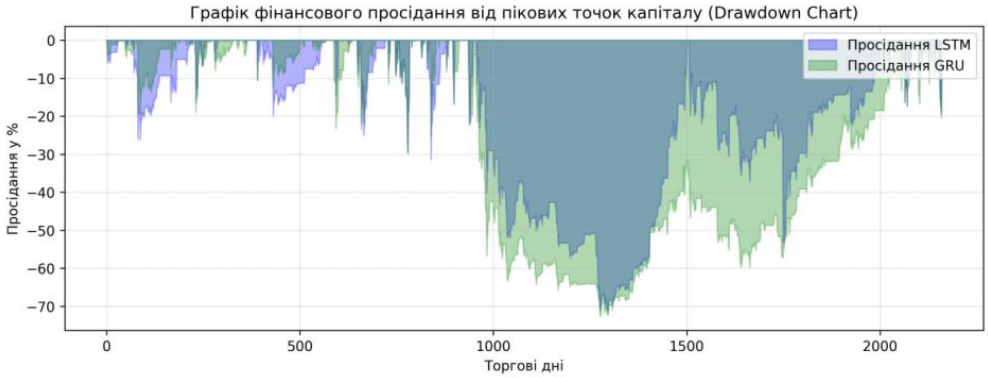

Рисунок 3.4 - Графік фінансового просідання (Drawdown) від пікових точок капіталу для BTC

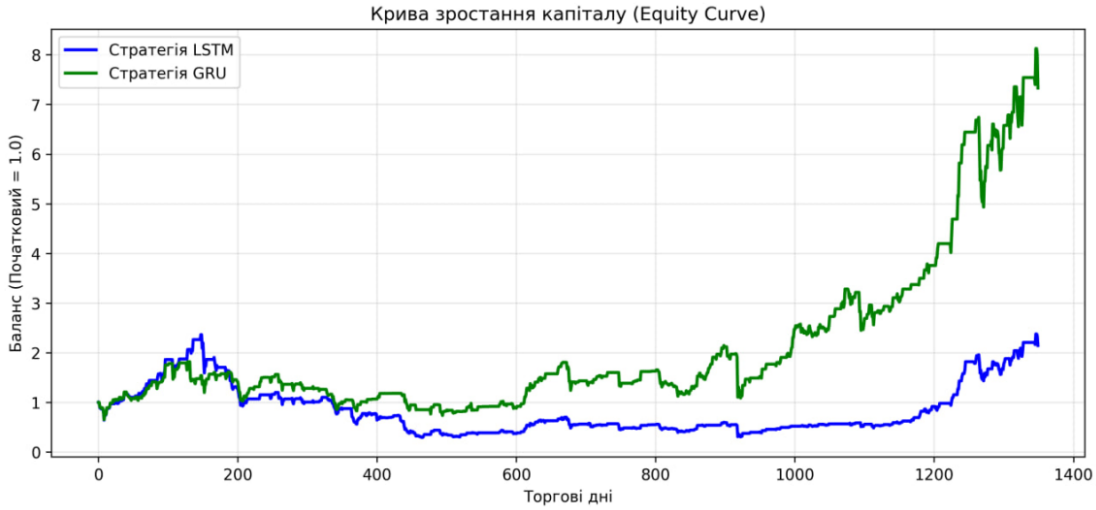

Рисунок 3.5 - Крива зростання капіталу (Equity Curve) для стратегій на базі ETH

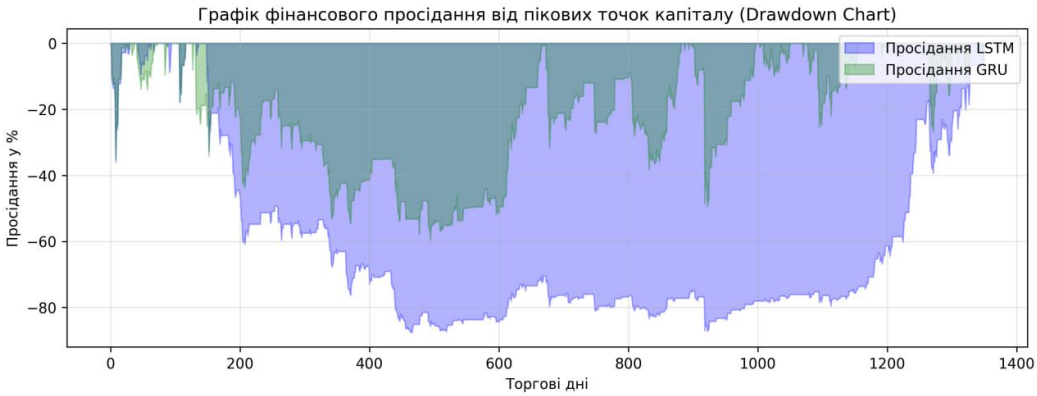

Рисунок 3.6 - Графік фінансового просідання (Drawdown) від пікових точок капіталу для ETH

Отже, за сукупністю математичних (RMSE, MAE) та фінансових (Profit, Sharpe Ratio, Max Drawdown) показників, архітектура GRU довела свою абсолютну та незаперечну перевагу при розв'язанні задачі прогнозування криптовалютних часових рядів.

Висновки до розділу 3

Програмну реалізацію спроєктованої системи прогнозування успішно виконано на мові Python із залученням індустріальних фреймворків TensorFlow, Keras та Scikit-learn. Реалізований програмний пайплайн забезпечує автоматизоване завантаження, очищення та лінійну нормалізацію фінансових даних у форматі OHLCV. Збереження параметрів масштабування (scaler) дозволяє коректно конвертувати результати роботи ШІ в реальні валютні значення для подальшого аналізу.

Процес трансформації одновимірних часових рядів у тривимірні тензори для рекурентних мереж успішно імплементовано за допомогою алгоритму "ковзного вікна" глибиною у 30 торгових днів. Розроблено програмний генератор Walk-Forward-валідації, який поділив історичні дані активів Bitcoin та Ethereum на 24 незалежні тестові фолди, повністю виключивши методологічну проблему "витоку даних з майбутнього".

Сконструйовано та навчено архітектури LSTM та GRU зі вбудованими механізмами регуляризації Dropout (20%). Доведено ефективність використання механізму ранньої зупинки (Early Stopping), який адаптивно переривав навчання у разі стагнації помилки на валідаційній вибірці, тим самим економлячи обчислювальні ресурси та запобігаючи перенавчанню (overfitting).

Проведено комплексне бектестування моделей. За показниками машинного навчання (MAE, RMSE) спрощена архітектура GRU продемонструвала вищу точність, зменшивши середню помилку на 25.1% для Bitcoin та на 19.4% для Ethereum порівняно з важкою архітектурою LSTM.

Оцінка фінансових ризиків підтвердила перевагу моделі GRU. Симуляція алгоритмічної торгівлі показала, що GRU забезпечує значно стабільніший прибуток (Коефіцієнт Шарпа 0.99 для BTC та 1.13 для ETH), ефективно обмежуючи Максимальне просідання (Max Drawdown) порівняно з LSTM. Дослідження доводить практичну доцільність використання архітектури GRU для розробки автоматизованих торгових систем на високоволатильних та нестаціонарних ринках.

ВИСНОВКИ

У кваліфікаційній роботі розв’язано актуальне науково-практичне завдання, яке полягає у дослідженні, проєктуванні та програмній реалізації інтелектуальної системи прогнозування криптовалютних часових рядів з використанням моделей глибокого навчання та методів оцінки фінансових ризиків.

У результаті виконання роботи отримано наступні теоретичні та практичні результати:

1. Здійснено системний аналіз предметної області та методів прогнозування. Доведено, що криптовалютний ринок характеризується екстремальною волатильністю, нестаціонарністю та наявністю складних нелінійних залежностей. Встановлено, що класичні економетричні моделі та алгоритми традиційного машинного навчання є недостатньо ефективними для цієї задачі через відсутність внутрішньої "пам'яті". Обґрунтовано доцільність використання рекурентних нейронних мереж (RNN), зокрема архітектур LSTM та GRU, для виявлення довгострокових прихованих патернів у фінансових даних.

2. Спроєктовано архітектуру інтелектуальної системи прогнозування. Розроблено топологію глибоких нейронних мереж, яка включає використання механізмів просторової регуляризації (Dropout) для запобігання перенавчанню. Сформовано методологію перетворення одновимірних ринкових даних у багатовимірні тензори за допомогою алгоритму "ковзного вікна" глибиною 30 торгових днів.

3. Запропоновано комплексну методологію валідації та оцінки моделей. Доведено хибність використання класичної перехресної перевірки (k-fold cross-validation) для фінансових даних через "витік інформації з майбутнього". Для забезпечення об'єктивності дослідження впроваджено алгоритм Walk-Forward-валідації. Крім того, систему оцінки розширено

фінансовими метриками ризик-менеджменту: Коефіцієнтом Шарпа (Sharpe Ratio) та Максимальним просіданням (Maximum Drawdown).

4. Програмно реалізовано розроблену систему. З використанням мови програмування Python та індустріальних фреймворків TensorFlow, Keras і Scikit-learn створено повністю автоматизований пайплайн. Програмний комплекс виконує завантаження даних, масштабування (Min-Max Scaling), навчання мереж з використанням оптимізатора Adam та механізму ранньої зупинки (Early Stopping), а також генерує торгові сигнали.

5. Проведено порівняльне тестування моделей на реальних даних (Bitcoin та Ethereum). Емпірично доведено перевагу архітектури GRU над LSTM. Завдяки меншій кількості математичних параметрів та керуючих вентилів, модель GRU виявилася менш схильною до перенавчання на зашумлених даних. За метрикою середньої абсолютної помилки (MAE), GRU перевершила LSTM на 25.1% для Bitcoin та на 19.4% для Ethereum.

6. Виконано симуляцію алгоритмічної торгівлі та оцінку фінансових ризиків. Результати бектестування підтвердили, що оптимізація виключно за математичною точністю (ML-метриками) не гарантує торгового успіху. Модель LSTM продемонструвала високу нестабільність, допустивши катастрофічне просідання капіталу на рівні -87.59% при тестуванні на Ethereum. Натомість архітектура GRU довела свою фінансову життєздатність та стійкість до ринкових шоків, забезпечивши значно менше просідання та високий Коефіцієнт Шарпа (0.99 для BTC та 1.13 для ETH).

Отримані результати підтверджують висунуту гіпотезу: архітектура GRU є оптимальним вибором для побудови автоматизованих систем прогнозування на високоволатильних ринках. Розроблений програмний комплекс може бути використаний як основа для створення реальних алгоритмічних торгових стратегій або як аналітичний інструмент у системах підтримки прийняття рішень для криптоінвесторів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ ISO 31000:2018. Менеджмент ризиків. Принципи та настанови. [Чинний від 2019-01-01]. Київ : ДП «УкрНДНЦ», 2019. 25 с.

2. Про віртуальні активи : Закон України від 17.02.2022 № 2074-IX. Київ : Парламентське видавництво, 2022. URL: https://zakon.rada.gov.ua/laws/show/2074-20 (дата звернення: 09.01.2026).

3. Рзаєв Д. О., Глібов І. В. Системи штучного інтелекту та машинне навчання : навчальний посібник. Київ : КПІ ім. Ігоря Сікорського, 2021. 250 с. URL: https://ela.kpi.ua/handle/123456789/41414 (дата звернення: 18.01.2026).

4. Botchkarev A. A New Typology Design of Performance Metrics to Measure Errors in Machine Learning Regression Algorithms. // Interdisciplinary Journal of Information, Knowledge, and Management. 2019. Vol. 14. P. 45-76. URL: https://arxiv.org/pdf/1809.03006.pdf (дата звернення: 27.01.2026).

5. Brownlee J. Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python. San Juan : Machine Learning Mastery, 2018. 327 p.

6. Cho K., van Merriënboer B., Gulcehre C. et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, 2014. P. 1724-1734. URL: https://arxiv.org/pdf/1406.1078.pdf (дата звернення: 05.02.2026).

7. Chollet F. Deep Learning with Python. Shelter Island : Manning Publications, 2017. 384 p.

8. Chung J., Gulcehre C., Cho K., Bengio Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. // NIPS 2014 Workshop on Deep Learning. Montreal, 2014. URL: https://arxiv.org/pdf/1412.3555.pdf (дата звернення: 14.02.2026).

9. CoinMarketCap: Cryptocurrency Prices, Charts & Market Caps. URL: https://coinmarketcap.com/ (дата звернення: 23.02.2026).

10. Cryptocurrency Historical Prices Dataset. // Kaggle. URL: https://www.kaggle.com/datasets/sudalairajkumar/cryptocurrencypricehistory (дата звернення: 04.03.2026).

11. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. 800 p.

12. Hochreiter S., Schmidhuber J. Long Short-Term Memory. // Neural Computation. 1997. Vol. 9, No. 8. P. 1735-1780.

13. Hull J. C. Risk Management and Financial Institutions. 5th Edition. Hoboken : Wiley, 2018. 832 p.

14. Hyndman R. J., Athanasopoulos G. Forecasting: principles and practice. 3rd Edition. Melbourne : OTexts, 2021. URL: https://otexts.com/fpp3/ (дата звернення: 13.03.2026).

15. Jiang Z., Liang J. Cryptocurrency portfolio management with deep reinforcement learning. // 2017 Intelligent Systems Conference (IntelliSys). London, 2017. P. 905-913. URL: https://ieeexplore.ieee.org/document/8324298 (дата звернення: 22.03.2026).

16. Keras: The Python Deep Learning API. URL: https://keras.io/ (дата звернення: 02.04.2026).

17. Livieris I. E., Pintelas E., Pintelas P. A CNN–LSTM model for gold price time-series forecasting. // Neural Computing and Applications. 2020. Vol. 32. P. 17351–17360.

18. Makarov I., Shojafar M. Cryptocurrency Price Prediction Using Deep Learning Methods. // 2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3). London, 2019. P. 1-6.

19. Makridakis S., Spiliotis E., Assimakopoulos V. Statistical and Machine Learning forecasting methods: Concerns and ways forward. // PLoS ONE. 2018. Vol. 13, No. 3. Article e0194889. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0194889 (дата звернення: 11.04.2026).

20. McKinney W. Python for Data Analysis. 3rd Edition. Sebastopol : O'Reilly Media, 2022. 580 p.

21. Pandas: Python Data Analysis Library. URL: https://pandas.pydata.org/ (дата звернення: 20.04.2026).

22. Pardo R. The Evaluation and Optimization of Trading Strategies. 2nd Edition. Hoboken : Wiley, 2008. 384 p.

23. Scikit-learn: Machine Learning in Python. // Scikit-learn developers. URL: https://scikit-learn.org/stable/ (дата звернення: 29.04.2026).

24. Sezer O. B., Gudelek M. U., Ozbayoglu A. M. Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. // Applied Soft Computing. 2020. Vol. 90. Article 106181.

25. Sharpe W. F. The Sharpe Ratio. // The Journal of Portfolio Management. 1994. Vol. 21, No. 1. P. 49-58.

26. Streamlit: A faster way to build and share data apps. URL: https://streamlit.io/ (дата звернення: 08.05.2026).

27. TensorFlow: An End-to-End Open Source Machine Learning Platform. // Google Research. URL: https://www.tensorflow.org/ (дата звернення: 17.05.2026).

ДОДАТОК А. Програмний модуль підготовки даних та формування тестових вибірок.

Лістинг А.1

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
import joblib
import os

# Конфігурація
FILE_PATH = 'data/coin_Ethereum.csv'
MODELS_DIR = 'models'
SEQ_LENGTH = 30  # Розмір "вікна" (скільки минулих днів бачить модель)
TARGET_COL = 'Close' # Що прогнозуємо

# Створюємо папку для збереження скейлера/моделей, якщо її немає
os.makedirs(MODELS_DIR, exist_ok=True)

def load_and_clean_data(file_path):
    """Завантажує та очищує історичні дані криптовалют."""
    print(f"[*] Завантаження даних з {file_path}...")
    df = pd.read_csv(file_path)

    # Видаляємо зайві колонки, якщо вони є
    cols_to_drop = ['SNo', 'Name', 'Symbol', 'Marketcap']
    df = df.drop(columns=[c for c in cols_to_drop if c in df.columns], errors='ignore')

    # Перетворення дати та сортування
    df['Date'] = pd.to_datetime(df['Date'])
    df = df.set_index('Date')
    df = df.sort_index()

    print(f"[+] Дані очищено. Розмір датасету: {df.shape}")
    return df

def normalize_data(df, save_scaler=True):
    print("[*] Нормалізація даних...")
    scaler = MinMaxScaler(feature_range=(0, 1))

    scaled_values = scaler.fit_transform(df)
    df_scaled = pd.DataFrame(scaled_values, columns=df.columns, index=df.index)
    if save_scaler:
```

Продовження лістингу А.1

```
        scaler_path = os.path.join(MODELS_DIR,
'minmax_scaler.pkl')
        joblib.dump(scaler, scaler_path)
        print(f"[+] Scaler збережено у {scaler_path} (потрібно
для оцінки ризиків)")

    return df_scaled, scaler

def create_sequences(data_values, seq_length, target_col_index):
    """Нарізає часовий ряд на послідовності (вікна) для
LSTM/GRU."""
    print(f"[*] Створення послідовностей (вікно = {seq_length}
днів)...")
    X, y = [], []

    for i in range(len(data_values) - seq_length):
        X.append(data_values[i : i + seq_length])
        y.append(data_values[i + seq_length, target_col_index])

    X = np.array(X)
    y = np.array(y)

    print(f"[+] Готово. X: {X.shape}, y: {y.shape}")
    return X, y

def walk_forward_split(X, y, train_size, test_size):
    total_len = len(X)
    step = test_size  # Вікно зміщується на розмір тестової
вибірки

    splits = []
    for start_idx in range(0, total_len - train_size - test_size
+ 1, step):
        end_train = start_idx + train_size
        end_test = end_train + test_size

        X_train, y_train = X[start_idx:end_train],
y[start_idx:end_train]
        X_test, y_test = X[end_train:end_test],
y[end_train:end_test]

        splits.append((X_train, y_train, X_test, y_test))

    print(f"[+] Створено {len(splits)} кроків(фолдів) для Walk-
Forward валідації.")
    return splits
```

Закінчення лістингу А.1

```
def main():
    # 1. Завантаження
    df = load_and_clean_data(FILE_PATH)

    # Знаходимо індекс цільової колонки ДО того, як перетворимо
все в масив NumPy
    target_col_index = df.columns.get_loc(TARGET_COL)

    # 2. Нормалізація
    df_scaled, scaler = normalize_data(df, save_scaler=True)

    # 3. Створення послідовностей
    X, y = create_sequences(df_scaled.values, SEQ_LENGTH,
target_col_index)

    # 4. Підготовка розбиття для Walk-Forward валідації
(приклад)
    # Навчаємо на 2 роках (~730 днів), тестуємо на 3 місяцях
(~90 днів)
    TRAIN_DAYS = 730
    TEST_DAYS = 90

    wf_splits = walk_forward_split(X, y, train_size=TRAIN_DAYS,
test_size=TEST_DAYS)
    print("\n=== Готово до навчання нейромережі! ===")

    # Приклад того, як потім використовувати розбиття у циклі
навчання:
    # for fold, (X_train, y_train, X_test, y_test) in
enumerate(wf_splits):
    #     print(f"Фолд {fold+1}: Навчання на {len(X_train)}
зразках, тест на {len(X_test)} зразках")
    #     # Тут буде model.fit(...) та model.predict(...)

if __name__ == "__main__":
    main()
```

ДОДАТОК Б. Програмна реалізація рекурентних нейронних мереж та процесу навчання.

Лістинг Б.1

```
import os
import numpy as np
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, GRU, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

# Імпортуємо функції та константи з нашого першого файлу
from preprocessing import (load_and_clean_data, normalize_data,
                           create_sequences, walk_forward_split,
                           FILE_PATH, SEQ_LENGTH, TARGET_COL)

# Конфігурація навчання
EPOCHS = 50
BATCH_SIZE = 32
RESULTS_DIR = 'results'

os.makedirs(RESULTS_DIR, exist_ok=True)

def build_lstm(input_shape):
    """Будує архітектуру LSTM моделі."""
    model = Sequential()
    model.add(LSTM(50, return_sequences=True,
input_shape=input_shape))
    model.add(Dropout(0.2)) # Запобігає перенавчанню
    model.add(LSTM(50, return_sequences=False))
    model.add(Dropout(0.2))
    model.add(Dense(1)) # Один вихід - прогнозована ціна
    model.compile(optimizer='adam', loss='mse')
    return model

def build_gru(input_shape):
    """Будує архітектуру GRU моделі."""
    model = Sequential()
    model.add(GRU(50, return_sequences=True,
input_shape=input_shape))
    model.add(Dropout(0.2))
    model.add(GRU(50, return_sequences=False))
    model.add(Dropout(0.2))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    return model
```

Продовження лістингу Б.1

```
def main():
    print("=== Етап 1: Підготовка даних ===")
    df = load_and_clean_data(FILE_PATH)
    target_col_index = df.columns.get_loc(TARGET_COL)
    df_scaled, _ = normalize_data(df, save_scaler=False)
    X, y = create_sequences(df_scaled.values, SEQ_LENGTH,
target_col_index)

    # Використовуємо ті ж налаштування, що й раніше
    TRAIN_DAYS = 730
    TEST_DAYS = 90
    wf_splits = walk_forward_split(X, y, train_size=TRAIN_DAYS,
test_size=TEST_DAYS)

    input_shape = (X.shape[1], X.shape[2]) # (30, 5)

    # Списки для збереження результатів
    all_results = []

    print("\n=== Етап 2: Walk-Forward Навчання ===")
    # EarlyStopping зупинить навчання раніше, якщо модель
перестане покращуватись
    early_stop = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)

    for fold, (X_train, y_train, X_test, y_test) in
enumerate(wf_splits):
        print(f"\n--- Фолд {fold+1} / {len(wf_splits)} ---")

        # 1. Навчання LSTM
        print("Навчання LSTM...")
        model_lstm = build_lstm(input_shape)
        model_lstm.fit(X_train, y_train, epochs=EPOCHS,
batch_size=BATCH_SIZE,
                       validation_data=(X_test, y_test),
callbacks=[early_stop], verbose=0)

        lstm_preds = model_lstm.predict(X_test).flatten()

        # 2. Навчання GRU
        print("Навчання GRU...")
        model_gru = build_gru(input_shape)
        model_gru.fit(X_train, y_train, epochs=EPOCHS,
batch_size=BATCH_SIZE,
                      validation_data=(X_test, y_test),
callbacks=[early_stop], verbose=0)
```

Закінчення лістингу Б.1

```
            gru_preds = model_gru.predict(X_test).flatten()

            # 3. Збереження прогнозів цього кроку
            for i in range(len(y_test)):
                all_results.append({
                    'Fold': fold + 1,
                    'Actual_Scaled': y_test[i],
                    'LSTM_Pred_Scaled': lstm_preds[i],
                    'GRU_Pred_Scaled': gru_preds[i]
                })

    # Зберігаємо все у CSV для останнього кроку (Метрики)
    results_df = pd.DataFrame(all_results)
    results_path = os.path.join(RESULTS_DIR,
'walk_forward_predictions.csv')
    results_df.to_csv(results_path, index=False)
    print(f"\n[+] Всі кроки завершено! Прогнози збережено у
{results_path}")

if __name__ == "__main__":
    main()
```

ДОДАТОК В. Алгоритм симуляції торгівлі та розрахунку фінансових метрик ризику.

Лістинг В.1

```
import pandas as pd
import numpy as np
import joblib
from sklearn.metrics import mean_squared_error,
mean_absolute_error
import os

# Конфігурація
RESULTS_FILE = 'results/walk_forward_predictions.csv'
SCALER_FILE = 'models/minmax_scaler.pkl'
TARGET_COL_IDX = 3  # Індекс колонки Close (High, Low, Open,
[Close], Volume)
NUM_FEATURES = 5    # Загальна кількість фіч, на яких навчався
скейлер

def inverse_transform_predictions(scaled_series, scaler):
    """Повертає масштабовані дані (0-1) у реальні ціни (USD)."""
    dummy_array = np.zeros((len(scaled_series), NUM_FEATURES))
    dummy_array[:, TARGET_COL_IDX] = scaled_series
    return scaler.inverse_transform(dummy_array)[:,
TARGET_COL_IDX]

def calculate_financial_metrics(actual_prices,
predicted_prices):
    """Рахує фінансові метрики: Коефіцієнт Шарпа та Максимальне
просідання."""
    # Зсуваємо реальні ціни на 1 день назад, щоб мати "вчорашню
ціну"
    actual_prev = np.roll(actual_prices, 1)
    actual_prev[0] = actual_prices[0]

    # Симуляція стратегії: Купуємо (1), якщо прогноз на сьогодні
більший за вчорашню ціну
    signals = (predicted_prices > actual_prev).astype(int)

    # Реальна щоденна дохідність ринку
    daily_returns = np.zeros_like(actual_prices)
    daily_returns[1:] = (actual_prices[1:] - actual_prices[:-1])
/ actual_prices[:-1]

    # Дохідність нашої стратегії
    strategy_returns = signals * daily_returns
```

Продовження лістингу В.1

```
    # Коефіцієнт Шарпа (Sharpe Ratio) - для крипти беремо 365
днів
    mean_ret = np.mean(strategy_returns)
    std_ret = np.std(strategy_returns)
    sharpe_ratio = (mean_ret / std_ret) * np.sqrt(365) if
std_ret > 0 else 0

    # Максимальне просідання (Maximum Drawdown)
    cum_returns = np.cumprod(1 + strategy_returns)
    peak = np.maximum.accumulate(cum_returns)
    drawdown = (cum_returns - peak) / peak
    max_dd = np.min(drawdown) * 100 # У відсотках

    # Загальний прибуток стратегії (%)
    total_return = (cum_returns[-1] - 1) * 100

    return sharpe_ratio, max_dd, total_return

def main():
    print("=== Завантаження даних та повернення реальних цін
===")
    df = pd.read_csv(RESULTS_FILE)
    scaler = joblib.load(SCALER_FILE)

    # Відновлюємо реальні ціни
    df['Actual_USD'] =
inverse_transform_predictions(df['Actual_Scaled'], scaler)
    df['LSTM_USD'] =
inverse_transform_predictions(df['LSTM_Pred_Scaled'], scaler)
    df['GRU_USD'] =
inverse_transform_predictions(df['GRU_Pred_Scaled'], scaler)

    # === 1. Розрахунок ML Метрик ===
    lstm_rmse = np.sqrt(mean_squared_error(df['Actual_USD'],
df['LSTM_USD']))
    lstm_mae = mean_absolute_error(df['Actual_USD'],
df['LSTM_USD'])

    gru_rmse = np.sqrt(mean_squared_error(df['Actual_USD'],
df['GRU_USD']))
    gru_mae = mean_absolute_error(df['Actual_USD'],
df['GRU_USD'])

    # === 2. Розрахунок Фінансових Метрик (Оцінка ризику) ===
    actual_prices = df['Actual_USD'].values
```

Закінчення лістингу В.1

```
    lstm_sharpe, lstm_max_dd, lstm_ret =
calculate_financial_metrics(actual_prices,
df['LSTM_USD'].values)
    gru_sharpe, gru_max_dd, gru_ret =
calculate_financial_metrics(actual_prices, df['GRU_USD'].values)

    # Виведення результатів
    print("\n" + "="*50)
    print("РЕЗУЛЬТАТИ ПОРІВНЯННЯ МОДЕЛЕЙ (WALK-FORWARD)")
    print("="*50)

    print(f"\n[ Метрики точності (ML) - менше = краще ]")
    print(f"LSTM -> RMSE: ${lstm_rmse:.2f} | MAE:
${lstm_mae:.2f}")
    print(f"GRU  -> RMSE: ${gru_rmse:.2f} | MAE:
${gru_mae:.2f}")

    print(f"\n[ Фінансові метрики (Оцінка ризику) ]")
    print(f"LSTM -> Прибуток: {lstm_ret:.2f}% | Sharpe:
{lstm_sharpe:.2f} | Max Drawdown: {lstm_max_dd:.2f}%")
    print(f"GRU  -> Прибуток: {gru_ret:.2f}% | Sharpe:
{gru_sharpe:.2f} | Max Drawdown: {gru_max_dd:.2f}%")
    print("="*50)

if __name__ == "__main__":
    main()
```