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

Модель прогнозування курсу криптовалют на основі нейронних мереж
(назва роботи)

Спеціальність (спеціалізація) 122 Комп'ютерні науки

Освітня програма Комп'ютерні системи та мережі

Виконавець _____ Валерія МАНЮК
(підпис) *(імя, прізвище)*

Науковий керівник _____ Вікторія СТРІЛЕЦЬ
(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 ММАД
_____ Володимир СТРУКОВ

«____» _____2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувача першого (бакалаврського) рівня вищої освіти

Манюк Валерії Павлівни
(прізвище, ім'я, по батькові здобувача)

Спеціальність: 122 «Комп'ютерні науки»
Освітня програма: «Комп'ютерні системи та мережі»

1. Тема роботи «**Модель прогнозування курсу криптовалют на основі нейронних мереж»**
керівник роботи **Стрілець Вікторія Євгенівна,** доцент кафедри комп'ютерних систем та робототехніки, кандидат технічних наук,
затверджені наказом по університету від "29" січня 2026 року № 4101-5/225

2. Строк подання студентом роботи 31.05.2026 р.

3. Вихідні дані до роботи: історичні дані про ціну криптовалюти (біткойна).

4. Перелік питань, які потрібно розробити:
   1. Проаналізувати сучасні та традиційні підходи до прогнозування фінансових часових рядів.
   2. Формалізувати задачу прогнозування: математична постановка, обмеження, період прогнозу, вимоги до якості.
   3. Підготувати історичні фінансові дані (про ціну криптовалюти).
   4. Спроєктувати дві нейромережеві моделі, визначити протокол валідації у часі й метрики для їх оцінки.
   5. Реалізувати й навчити моделі, забезпечивши відтворюваність експериментів.

6. Провести тестування та порівняльний аналіз результатів і помилок; сформулювати висновки та напрями подальшого вдосконалення.

5. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Літературний огляд з проблематики прогнозування фінансових часових рядів	10.02.26
2	Аналіз підходів до прогнозування часових рядів	01.03.26
3	Збір і підготовка даних (OHLCV, 2018–2025)	15.03.26
4	Проєктування та налаштування багатошарової мережі прямого поширення	01.04.26
5	Розробка та налаштування рекурентної мережі LSTM	15.04.26
6	Тестування моделей і первинний аналіз результатів	01.05.26
7	Порівняльний аналіз ефективності та стабільності; рекомендації щодо поліпшення	15.05.26
8	Підготовка та оформлення розділів кваліфікаційної роботи	25.05.26
9	Оформлення пояснювальної записки та узагальнення результатів	30.05.26
10	Подання кваліфікаційної роботи керівнику та рецензенту	31.05.2026

5. Дата видачі завдання 01.02.2026 р.

Здобувач вищої освіти _____ Валерія Манюк
підпис — ініціали, прізвище

Керівник роботи _____ Вікторія Стрілець
підпис — ініціали, прізвище

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи складається зі вступу, трьох розділів, висновків, списку використаних джерел та трьох додатків. Загальний обсяг роботи складає 59 сторінок, із яких 48 сторінок основної частини з 9 рисунками, 1 таблицями, 35 найменуваннями списку використаних джерел та одним додатком.

Метою дослідження є підвищення ефективності прогнозування курсу криптовалют та забезпечення надійності прийняття торгових рішень шляхом створення та дослідження нейромережевих моделей на основі архітектур глибокого навчання.

Об'єкт дослідження – процес формування вартості криптовалютних активів та динаміка фінансових часових рядів в умовах волатильності ринку.

Предмет дослідження – математичні моделі, методи машинного і глибокого навчання у задачах прогнозування часових рядів, алгоритми попередньої обробки даних та метрики оцінки якості регресійних моделей.

Проблема, яка вирішується в роботі, полягає у забезпеченні високої точності короткострокового прогнозування (таймфрейми 15m–1h) в умовах високої волатильності та нестаціонарності ринку шляхом використання моделей глибокого навчання.

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

Ключові слова: *ПРОГНОЗУВАННЯ КУРСУ, КРИПТОВАЛЮТА, BITCOIN, ГЛИБОКЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, LSTM, MLP, ЧАСОВІ РЯДИ, АЛГОРИТМІЧНА ТОРГІВЛЯ.*

ABSTRACT

The explanatory note to the qualification work consists of an introduction, three chapters, conclusions, a list of references, and three appendices. The total volume of the work is 59 pages, of which 48 pages are the main part, including 9 figures, 1 table, 35 items in the list of references, and one appendice.

The purpose of the research is to improve the efficiency of cryptocurrency rate forecasting and ensure the reliability of trading decision-making through the creation and investigation of neural network models based on deep learning architectures.

The object of research is the process of value formation of cryptocurrency assets and the dynamics of financial time series under conditions of market volatility.

The subject of research is mathematical models, machine and deep learning methods in time series forecasting tasks, data preprocessing algorithms, and quality assessment metrics of regression models.

The problem solved in the work lies in ensuring high accuracy of short-term forecasting (15m–1h timeframes) under conditions of high market volatility and non-stationarity through the use of logarithmic returns and memory mechanisms.

The scope of application includes Financial Technologies (FinTech), the development of algorithmic trading systems, digital asset management, and automated risk management.

Keywords: RATE PREDICTION, CRYPTOCURRENCY, BITCOIN, DEEP LEARNING, NEURAL NETWORKS, LSTM, MLP, TIME SERIES, ALGORITHMIC TRADING.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ 7
ВСТУП 8
1 АНАЛІЗ ЗАДАЧІ ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ І ПІДХОДІВ ДО ЇЇ ВИРІШЕННЯ 11
1.1 Формалізація задачі прогнозування курсу криптовалют 11
1.2 Класичні підходи прогнозування часових рядів 13
1.3 Методи машинного навчання для прогнозування часових рядів 15
1.4 Задачі і методи попередньої обробки часових рядів 19
Висновки за розділом 1 24
2 ПРОЄКТУВАННЯ НЕЙРОМЕРЕЖЕВИХ МОДЕЛЕЙ ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ 26
2.1 Формалізація процесу побудови моделей прогнозування 26
2.2 Архітектура нейромережевих моделей прогнозування 29
2.3 Програмна реалізація та інструментальні засоби 35
Висновки до розділу 2 37
3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ МОДЕЛЕЙ ПРОГНОЗУВАННЯ 39
3.1 Програмна реалізація модулів попередньої обробки даних 39
3.2 Методика проведення експериментів та метрики оцінки якості 41
3.3 Результати експериментального моделювання 43
3.4 Обговорення результатів та порівняльний аналіз ефективності моделей 47
Висновки до розділу 3 50
ВИСНОВКИ 52
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 54
ДОДАТОК А. Лістинг вихідного коду модуля нейромережевого прогнозування 59

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

API – Application Programming Interface (Інтерфейс прикладного програмування)

BTC – Bitcoin (Криптовалюта Біткоїн)

CSV – Comma-Separated Values (Текстовий формат зберігання табличних даних)

CUDA – Compute Unified Device Architecture (Архітектура апаратних обчислень NVIDIA)

DL – Deep Learning (Глибоке навчання)

GPU – Graphics Processing Unit (Графічний процесор)

HFT – High-Frequency Trading (Високочастотна алгоритмічна торгівля)

LSTM – Long Short-Term Memory (Довга короткострокова пам’ять – тип рекурентної нейромережі)

MAE – Mean Absolute Error (Середня абсолютна помилка)

MAPE – Mean Absolute Percentage Error (Середня абсолютна помилка у відсотках)

MLP – Multilayer Perceptron (Багатошаровий персептрон)

MSE – Mean Squared Error (Середня квадратична помилка)

OHLCV – Open, High, Low, Close, Volume (Біржовий формат даних: Відкриття, Макс., Мін., Закриття, Об’єм)

ReLU – Rectified Linear Unit (Функція активації «випрямляч»)

RMSE – Root Mean Squared Error (Корінь із середньоквадратичної помилки)

RNN – Recurrent Neural Network (Рекурентна нейронна мережа)

SGD – Stochastic Gradient Descent (Стохастичний градієнтний спуск)

USD – United States Dollar (Долар США)

ВСТУП

У зв’язку зі стрімкою цифровізацією глобальної економіки та зростанням капіталізації ринку віртуальних активів, все більш актуальною постає проблема підвищення ефективності прийняття рішень у сфері алгоритмічної торгівлі та управління фінансовими ризиками. Сучасний криптовалютний ринок є складною стохастичною системою, що характеризується високою волатильністю, нелінійною динамікою та чутливістю до інформаційного шуму. В умовах жорсткої конкуренції та переходу до високочастотних торгових стратегій (HFT) ключовим фактором рентабельності стає здатність торгової системи не просто реагувати на зміну ціни постфактум, а прогнозувати ринкові рухи та діяти на випередження, забезпечуючи мінімізацію фінансових втрат.

Актуальність роботи. Серед прикладних областей, де зазначена проблема є вкрай гострою, слід віднести короткострокове прогнозування курсу Bitcoin (BTC). Специфіка цієї задачі полягає у необхідності виявлення прихованих закономірностей у масивах "великих даних" (Big Data), які мають нестаціонарну природу. Технологічний процес формування ціни залежить від множини факторів: історичних патернів, обсягів торгів та психології учасників ринку. Традиційні методи технічного аналізу, які використовуються у більшості торгових терміналів (ковзні середні, осцилятори), демонструють низьку ефективність в умовах хаотичного ринку. Вони працюють із часовим запізненням, що призводить до запізнення сигналів та збиткових угод. Водночас класичні економетричні моделі (ARIMA) часто не здатні охопити складні нелінійні залежності. Отже, задача розробки автоматизованої системи, яка поєднує точність методів глибокого навчання (Deep Learning) для прогнозування часових рядів із сучасними підходами до попередньої обробки даних, є вкрай актуальною науково-прикладною проблемою.

Метою дослідження є підвищення ефективності прогнозування курсу криптовалют та забезпечення надійності прийняття торгових рішень шляхом

створення та дослідження нейромережевих моделей на основі архітектур глибокого навчання.

Об’єкт дослідження – процес формування вартості криптовалютних активів та динаміка фінансових часових рядів в умовах волатильності ринку.

Предмет дослідження – математичні моделі, методи машинного і глибокого навчання у задачах прогнозування часових рядів, алгоритми попередньої обробки даних та метрики оцінки якості регресійних моделей.

Методи дослідження: методи системного аналізу (для формалізації задачі прогнозування), методи попередньої обробки даних (логарифмування та нормалізація для забезпечення стаціонарності рядів), методи глибокого навчання (архітектури Stacked LSTM та MLP для регресійного аналізу), методи математичної оптимізації (алгоритм Adam для навчання мереж), методи статистики (розрахунок метрик MAE, RMSE, MAPE) та методи імітаційного моделювання (стратегія Walk-Forward Validation для оцінки ефективності на історичних даних).

Завдання дослідження:

1. Виконати аналіз особливостей динаміки криптовалютного ринку та існуючих методів прогнозування, обґрунтувати переваги переходу від класичного статистичного аналізу до методів штучного інтелекту (Data-Driven Approach).

2. Розробити методику попередньої обробки даних, що включає трансформацію абсолютних цін у логарифмічні прибутковісті (LogReturns) та формування навчальних вибірок методом ковзного вікна (Sliding Window).

3. Спроєктувати архітектуру моделей прогнозування на основі глибоких нейронних мереж, і провести порівняльний аналіз моделей MLP та Stacked LSTM з використанням механізмів регуляризації (Dropout) та ранньої зупинки (Early Stopping).

4. Виконати програмну реалізацію моделей мовою Python та провести експериментальне дослідження їх ефективності на різних часових інтервалах

(від 15 хвилин до 1 дня), оцінивши точність прогнозу та наявність фазових запізнень.

Практичне значення одержаних результатів полягає у розробці програмного модуля, готового до інтеграції в автоматизовані торгові системи, який забезпечує високу точність прогнозування (MAPE < 0.2% на 15-хвилинному таймфреймі) та дозволяє трейдерам і аналітикам отримувати обґрунтовані сигнали про майбутній напрямок руху ціни, що підтверджено результатами тестування на реальних даних за 2024–2025 роки.

1 АНАЛІЗ ЗАДАЧІ ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ І ПІДХОДІВ ДО ЇЇ ВИРІШЕННЯ

1.1 Формалізація задачі прогнозування курсу криптовалют

Задача прогнозування курсу криптовалют з точки зору системного аналізу та теорії управління належить до класу задач ідентифікації стану динамічної системи в умовах невизначеності. Об’єктом дослідження виступає фінансовий часовий ряд, який представляє собою впорядковану в часі послідовність спостережень за вартістю активу. Згідно з фундаментальними положеннями теорії прогнозування, описаними в роботах [1] та [8], такий ряд можна формалізувати як дискретну функцію часу, де кожному моменту відповідає певне значення ціни закриття торгів, обсягу торгів або інших ринкових показників. Головною особливістю криптовалютних рядів, на відміну від класичних фінансових інструментів, є їхня висока волатильність, нелінійність та стохастична природа, що суттєво ускладнює процес побудови адекватних математичних моделей [6].

Математична постановка задачі прогнозування полягає у знаходженні такої функціональної залежності, яка дозволила б з мінімальною похибкою оцінити майбутнє значення часового ряду на основі його попередніх значень. Нехай маємо множину історичних даних, що описують динаміку курсу біткоїна за певний період. Позначимо через $P(t)$ значення ціни активу в момент часу t. Тоді вектор вхідних даних для моделі прогнозування можна представити як скінченну послідовність попередніх значень глибиною n, тобто

$$X(t) = \{P(t), P(t-1), ..., P(t-n+1)\}. \quad (1.1)$$

Задача полягає у побудові апроксимуючої функції F, яка відображає вхідний вектор у прогнозоване значення на k кроків уперед.

У контексті даної роботи розглядається короткострокове прогнозування на один крок уперед, що є типовим для задач алгоритмічної торгівлі та відповідає підходам, викладеним у джерелах [19] та [29].

Критерієм якості побудованої моделі виступає мінімізація функції втрат, яка характеризує відхилення прогнозованих значень від реальних. Найчастіше в якості такої функції використовують середньоквадратичну помилку або середню абсолютну помилку, оскільки вони є диференційованими та дозволяють застосовувати градієнтні методи оптимізації при навчанні нейронних мереж [12]. Важливим аспектом формалізації є врахування статистичних властивостей досліджуваних даних. Як зазначається в [15] та [32], часові ряди криптовалют є нестаціонарними, тобто їхні основні ймовірнісні характеристики, такі як математичне сподівання та дисперсія, змінюються з часом. Це накладає суттєві обмеження на використання класичних економетричних методів і вимагає попередньої обробки даних для приведення їх до стаціонарного вигляду, наприклад, шляхом переходу від абсолютних цін до логарифмічної прибутковості [23].

Окрім внутрішньої динаміки цін, на курс криптовалют впливає значна кількість зовнішніх чинників, таких як регуляторні новини, макроекономічні показники та психологічний стан учасників ринку. Однак у межах даної кваліфікаційної роботи задача формалізується в рамках гіпотези ефективного ринку в її слабкій формі, яка припускає, що вся суттєва інформація вже відображена в історичних цінах активу. Такий підхід дозволяє обмежитися використанням виключно історичних ринкових даних (ціни Open, High, Low, Close та об’єм Volume) як вхідних параметрів моделі, що узгоджується з методологією технічного аналізу та підходами машинного навчання, описаними в [5] та [28].

Специфіка об’єкта прогнозування також передбачає наявність шумів та викидів у вхідних даних, які можуть бути викликані технічними збоями на біржах або спекулятивними маніпуляціями. Тому математична модель повинна володіти властивістю робастності, тобто стійкості до аномальних

значень. Використання глибоких нейронних мереж, зокрема рекурентних архітектур LSTM, є обґрунтованим саме з точки зору їхньої здатності апроксимувати складні нелінійні залежності в зашумлених даних та враховувати довгострокові часові кореляції, які неможливо описати простими лінійними рівняннями [9, 25]. Таким чином, формалізована задача зводиться до задачі регресійного аналізу часових рядів засобами навчання з учителем, де метою є мінімізація емпіричного ризику на тестовій вибірці даних [10].

1.2 Класичні підходи прогнозування часових рядів

Історично першими та найбільш дослідженими методами прогнозування часових рядів є статистичні моделі, які базуються на припущенні, що майбутні значення процесу є лінійною функцією його попередніх значень та випадкових похибок. Фундаментальні основи цього підходу викладені в класичних працях з теорії ймовірностей та математичної статистики, де задача прогнозування розглядається як задача екстраполяції детермінованої компоненти ряду [1]. Найпростішим класом таких методів є *моделі ковзного середнього* (MA – Moving Average) та авторегресії (AR – AutoRegression). Модель авторегресії порядку p описує поточне значення ряду як зважену суму p попередніх значень, тоді як модель ковзного середнього порядку q фокусується на моделюванні похибки як лінійної комбінації q попередніх значень білого шуму. Комбінація цих двох підходів призвела до появи моделей ARMA, які довгий час залишалися стандартом при аналізі стаціонарних процесів [8].

Однак більшість реальних фінансових рядів, зокрема і котирування криптовалют, не є стаціонарними, тобто їхні статистичні характеристики змінюються з часом. Для роботи з такими даними була розроблена методологія Бокса-Дженкінса, відома як модель *ARIMA* (AutoRegressive Integrated Moving Average). Ключовою відмінністю ARIMA є інтегральна компонента, яка дозволяє привести нестаціонарний ряд до стаціонарного вигляду шляхом взяття різниць відповідного порядку [35]. Цей підхід довів свою ефективність у короткостроковому прогнозуванні традиційних активів зі стабільною

волатильністю, проте він має суттєві обмеження при застосуванні до високодинамічних ринків. Як зазначається у дослідженні [31], моделі класу ARIMA здатні якісно описувати лише лінійні залежності у даних, ігноруючи складні нелінійні патерни, що є характерними для криптовалют.

Окремим напрямом класичного прогнозування є *моделювання волатильності*, оскільки для фінансових ринків оцінка ризику часто є не менш важливою, ніж оцінка напрямку руху ціни. Для цього застосовуються *моделі умовної гетероскедастичності*, зокрема ARCH (AutoRegressive Conditional Heteroskedasticity) та її узагальнення GARCH [6]. Ці моделі відмовляються від припущення про сталість дисперсії похибки прогнозу, дозволяючи моделювати кластеризацію волатильності – явище, коли періоди значних цінових коливань змінюються періодами відносного спокою. У роботі [15] показано, що хоча GARCH-моделі краще справляються з оцінкою ризиків криптовалютних активів порівняно з простими лінійними моделями, вони все ще поступаються методам штучного інтелекту в точності точкового прогнозування ціни.

Головним недоліком класичних статистичних підходів у контексті даної роботи є їхня жорстка параметрична структура та необхідність виконання суворих статистичних припущень щодо розподілу даних, які на практиці часто порушуються. Криптовалютний ринок характеризується наявністю "важких хвостів" у розподілі прибутковості, мультифрактальністю та хаотичною поведінкою, що робить лінійні моделі недостатньо гнучкими для адекватного відображення ринкової динаміки [34]. Крім того, класичні методи погано масштабуються при додаванні екзогенних змінних (наприклад, обсягу торгів або технічних індикаторів) і не здатні автоматично виділяти приховані ознаки з сирих даних. Саме ці обмеження зумовили перехід дослідників до використання методів машинного навчання, які не вимагають апріорних знань про закон розподілу даних та здатні апроксимувати складні нелінійні функції довільної точності [5].

1.3 Методи машинного навчання для прогнозування часових рядів

Розвиток обчислювальних потужностей та накопичення великих обсягів історичних даних сприяли активному впровадженню методів машинного навчання у сферу фінансового аналізу. На відміну від класичних статистичних підходів, розглянутих у попередньому підрозділі, методи машинного навчання, зокрема штучні нейронні мережі, не вимагають апріорних припущень щодо стаціонарності ряду або нормального закону розподілу його значень. Як зазначається у фундаментальній праці [3], основною перевагою цих методів є здатність автоматично виявляти складні нелінійні залежності та приховані патерни у багатовимірних просторах ознак, що є критично важливим для прогнозування волатильних активів, таких як криптовалюти. У сучасному науковому дискурсі методи машинного навчання для задач прогнозування часових рядів умовно поділяють на дві великі групи: ансамблеві методи на основі дерев рішень та нейромережеві моделі.

До першої групи належать такі алгоритми, як *Random Forest* та *градієнтний бустинг* (XGBoost, LightGBM). Згідно з [20] ці методи демонструють високу ефективність у задачах класифікації та регресії завдяки здатності мінімізувати зміщення та дисперсію прогнозу шляхом побудови композиції слабких базових алгоритмів. Однак їхнім суттєвим недоліком при роботі з часовими рядами є відсутність механізму збереження часової послідовності подій: вхідні дані подаються у вигляді фіксованого набору ознак, і модель не враховує динаміку змін у часі як неперервний процес. Це обмеження стимулювало перехід дослідників до використання штучних нейронних мереж, які здатні апроксимувати будь-яку неперервну функцію з довільною точністю, що підтверджується теоремою про універсальну апроксимацію [16].

Базовою архітектурою глибокого навчання є *багатошаровий персептрон* (MLP – Multilayer Perceptron). Ця модель складається з вхідного шару, одного або кількох прихованих шарів та вихідного шару (рис. 1.1). Кожен нейрон у шарі пов’язаний з усіма нейронами наступного шару через

вагові коефіцієнти, які налаштовуються в процесі навчання за алгоритмом зворотного поширення помилки [10]. Використання нелінійних функцій активації, таких як ReLU або Sigmoid, дозволяє MLP моделювати складні криволінійні границі рішень.



Рисунок 1.1 – Концептуальна структура багатошарового персептрона

Як видно з рисунка 1.1, архітектура прямого поширення передбачає суворо односпрямований рух даних від вхідного вектору до вихідного прогнозу. Відсутність петель зворотного зв'язку означає, що мережа не формує внутрішнього стану або пам'яті. Це обумовлює її головний недолік при роботі з фінансовими часовими рядами: модель сприймає кожен вхідний вектор незалежно від попередніх, втрачаючи критично важливу інформацію про хронологічну історію процесу.

У роботах [5] та [11] показано, що MLP може ефективно використовуватися для короткострокового прогнозування фінансових індексів. Проте, як і у випадку з деревами рішень, стандартна архітектура MLP є мережею прямого поширення (Feed Forward Network), що означає відсутність зворотних зв’язків. Це призводить до того, що модель сприймає кожен вхідний вектор незалежно від попередніх, втрачаючи інформацію про історію процесу, що є критичним недоліком для аналізу послідовностей.

Для вирішення проблеми врахування часового контексту було розроблено клас *рекурентних нейронних мереж* (RNN – Recurrent Neural Networks). Ключовою особливістю RNN є наявність циклічних зв’язків, які дозволяють передавати інформацію про стан прихованого шару з попереднього кроку часу на поточний (рис. 1.2) [2]. Це створює аналог короткочасної пам’яті, що дозволяє мережі враховувати динаміку змін.

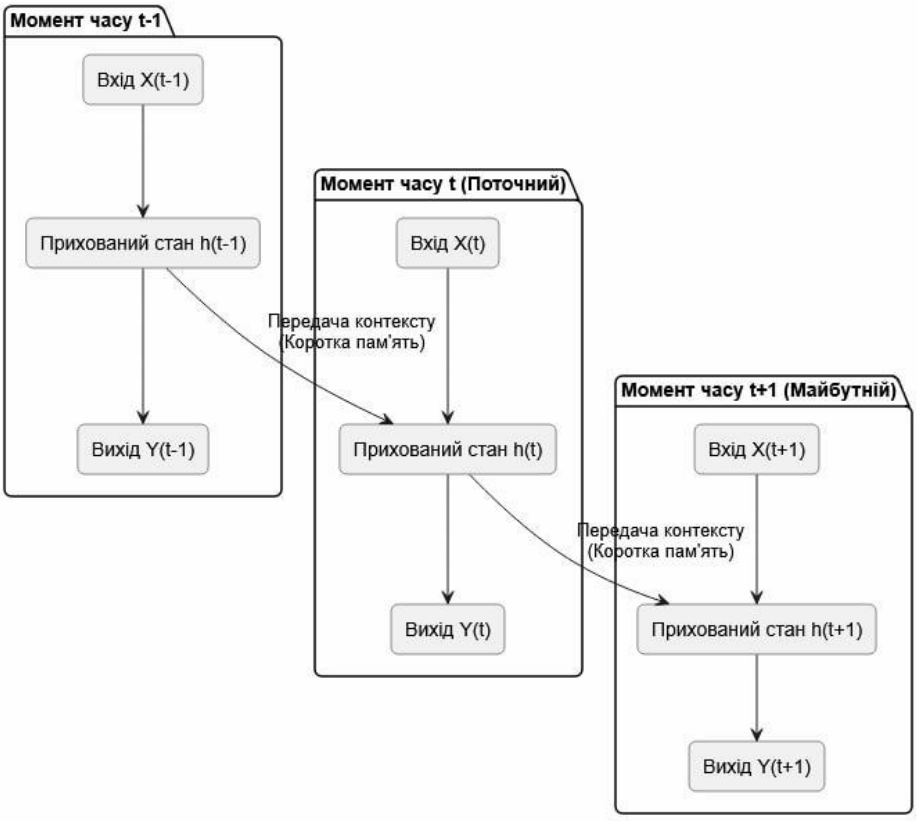


Рисунок 1.2 – Принцип розгортання рекурентної нейронної мережі (RNN) у часі

На рисунку 1.2 продемонстровано механізм функціонування рекурентних зв'язків. У розгорнутому вигляді мережа представляє собою ланцюжок ідентичних модулів, де кожен наступний стан мережі залежить не лише від поточного вхідного значення, але й від прихованого стану h, сформованого на попередньому кроці. Завдяки такій топології модель отримує здатність аналізувати не просто статичні "зрізи" ринку, а динамічні послідовності подій.

Однак, як описано в [24], класичні RNN страждають від проблеми зникаючого або вибухаючого градієнта при навчанні на довгих послідовностях. Це унеможливлює виявлення довгострокових залежностей, коли важлива подія (наприклад, різка зміна тренду) віддалена від моменту прогнозу значним проміжком часу.

Еволюційним розвитком ідеї рекурентності стала *архітектура довгої короткострокової пам'яті* (LSTM – Long Short-Term Memory), запропонована Хохрайтером та Шмідхубером [25]. Основна інновація LSTM полягає у введенні спеціальних блоків пам'яті та механізму гейтів (вентилів): вхідного гейта, гейта забування та вихідного гейта (рис. 1.3). Ці структури дозволяють мережі адаптивно вирішувати, яку інформацію слід зберегти, а яку – відкинути, що ефективно вирішує проблему зникаючого градієнта.

Як показано на схемі (рис. 1.3), наскрізний потік глобального стану пам'яті C відіграє ключову роль у збереженні довгострокових залежностей. Він проходить через весь ланцюжок рекурентних модулів, зазнаючи лише незначних лінійних взаємодій. Механізм гейтів працює як складна система фільтрації: гейт забування видаляє застарілий ринковий шум, вхідний гейт інтегрує нові актуальні патерни, а вихідний гейт формує фінальний сигнал, що робить архітектуру LSTM стійкою до інформаційних викидів та високої волатильності криптовалют.

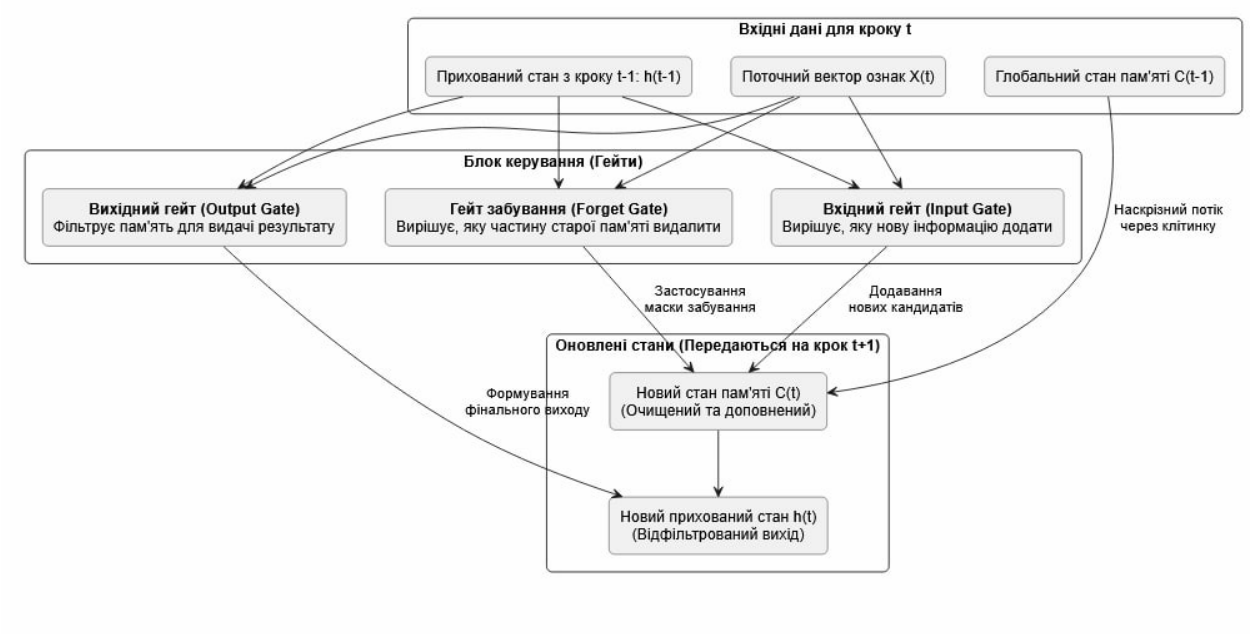


Рисунок 1.3 – Логічна взаємодія керуючих вентилів (гейтів) у комірці LSTM

Згідно з дослідженнями [9] та [23], саме LSTM-мережі на сьогодні є стандартом "де-факто" для прогнозування фінансових часових рядів, оскільки вони здатні одночасно аналізувати високочастотні коливання та довгострокові тренди.

Альтернативою LSTM є спрощена *архітектура GRU* (Gated Recurrent Unit) [21], яка має меншу кількість параметрів, проте, як показують порівняльні тести в [33], на складних та зашумлених даних, таких як курс біткоїна, повноцінна LSTM-архітектура часто забезпечує вищу точність та стабільність прогнозу. Тому в рамках даної роботи основний акцент робиться на порівнянні класичного персептрона та глибинної мережі LSTM як представників двох різних поколінь методів машинного навчання.

1.4 Задачі і методи попередньої обробки часових рядів

Якість прогнозування будь-якої моделі машинного навчання, незалежно від складності її архітектури, критично залежить від якості вхідних даних. У контексті фінансового прогнозування етап попередньої обробки даних (Data Preprocessing) часто є більш вагомим для кінцевого результату, ніж вибір

конкретного алгоритму оптимізації ваг нейронної мережі [32]. Сирі біржові дані, які надходять у форматі OHLCV (Open, High, Low, Close, Volume), містять значну кількість шумів, викидів та пропущених значень, що вимагає застосування комплексних процедур очищення та трансформації.

Проблема наявності пропущених значень (Missing Data) у криптовалютних часових рядах має свою специфіку. На відміну від традиційних фінансових ринків, таких як фондові біржі з чітко регламентованими торговими сесіями та вихідними днями, ринок віртуальних активів функціонує безперервно. Відповідно, пропуски в історичних даних найчастіше виникають внаслідок суто технічних факторів: збоїв у роботі програмних інтерфейсів (API) криптовалютних бірж, планового технічного обслуговування серверів, мережевих затримок або форс-мажорних зупинок торгів.

Для класичного статистичного аналізу втрата кількох відліків може бути некритичною, проте для глибоких нейронних мереж, і особливо для рекурентних архітектур типу LSTM, безперервність часової послідовності є фундаментальною вимогою. Просте видалення рядків із пропущеними значеннями є абсолютно неприпустимим підходом у задачах прогнозування часових рядів. Це призводить до руйнування хронологічної цілісності ковзного вікна, спотворення лагових залежностей та безповоротної втрати інформації про ринкову динаміку.

У зв'язку з цим виникає необхідність застосування методів імпутації, тобто штучного відновлення відсутніх даних. У межах даної роботи досліджено та систематизовано три основні підходи до вирішення цієї задачі:

- *Заповнення вперед* (Forward Fill). Цей метод полягає у заміщенні пропущеного фрагмента останнім відомим і фактично зафіксованим спостереженням. Такий підхід є найбільш фінансово та логічно обґрунтованим для алгоритмічних торгових систем, оскільки він імітує реальну поведінку ринку: якщо протягом певного періоду нових угод не відбулося і нові дані не

надійшли, вважається, що ціна активу залишається на рівні останньої підтвердженої ціни закриття.

- *Заповнення назад* (Backward Fill). За логікою цього методу пропуск заміщується найближчим наступним відомим значенням ряду. Проте цей підхід має вкрай обмежене застосування в реальних системах машинного навчання для трейдингу. Основна проблема полягає у тому, що він використовує інформацію з «майбутнього» для заповнення поточних прогалин, що неминуче призводить до ефекту витоку даних (Data Leakage) під час тестування моделей і формує хибно-оптимістичні метрики точності.
- *Лінійна інтерполяція* передбачає обчислення пропущеного значення як середнього зваженого між двома найближчими валідними точками – тією, що передує пропуску, і тією, що слідує за ним. Інтерполяція дозволяє отримати більш плавні цінові траєкторії, що добре підходить для згладжування макроекономічних показників. Однак на високочастотних криптовалютних даних цей метод може штучно занижувати рівень локальної волатильності, маскуючи різкі ринкові рухи.

Першочерговою проблемою при роботі з фінансовими часовими рядами є їхня *нестаціонарність*. Як показано у дослідженнях [15] та [27], ціна криптовалюти (наприклад, Bitcoin) може змінюватися на порядки протягом кількох років (від 3 000 до 70 000 доларів США). Навчання нейронної мережі на абсолютних значеннях цін призводить до проблеми зміщення розподілу: тестова вибірка може містити значення, які виходять далеко за межі діапазону значень навчальної вибірки. У такій ситуації модель не здатна адекватно екстраполювати залежності, оскільки функції активації нейронів (наприклад, гіперболічний тангенс у LSTM) насичуються, а нормалізуючі коефіцієнти (обчислені, наприклад, методом Z-score) стають неактуальними.

Для вирішення цієї проблеми у сучасній економетриці та машинному навчанні загальноприйнятим стандартом є перехід від аналізу цін P_t до аналізу їх відносних змін – прибутковості (Returns). Найбільш ефективним методом стабілізації дисперсії ряду є логарифмування відношення поточного значення

ціни до попереднього [6]. Логарифмічна прибутковість R_t визначається як натуральний логарифм відношення цін:

$$R_t = \ln(P_t/P_{t-1}). \quad (1.2)$$

Ця трансформація має низку математичних переваг: вона робить розподіл значень близьким до нормального, забезпечує адитивність прибутковості на різних часових інтервалах та усуває ефект масштабу, дозволяючи моделі однаково ефективно працювати як при низьких, так і при високих цінах активу [7].

Наступним важливим етапом є *нормалізація* (шкалювання) ознак.

Оскільки вхідний простір ознак є суттєво гетерогенним, безпосереднє подання «сирих» даних на вхід нейронної мережі робить її навчання вкрай неефективним. Це пояснюється тим, що параметри мають абсолютно різні одиниці виміру та масштаби: абсолютна ціна біткоїна може становити десятки тисяч доларів, обсяг торгів обчислюється мільйонами одиниць, а популярні осцилятори коливаються у вузьких межах. Оскільки швидкість оновлення ваг у градієнтному спуску прямо залежить від абсолютного розміру ознаки, великомасштабні змінні починають повністю домінувати над дрібномасштабними, нівелюючи їхній вплив на кінцевий прогноз.

Для усунення цього ефекту та забезпечення стабільної збіжності оптимізаційних алгоритмів необхідно застосовувати процедуру шкалювання. У теорії машинного навчання виділяють три базові методи такого перетворення:

- *Лінійна нормалізація* (Min-Max Scaling) здійснює пропорційне стиснення всіх вхідних даних у чітко фіксований діапазон, найчастіше від нуля до одиниці. Це досягається шляхом оцінки відстані кожного елемента від мінімального значення у вибірці відносно загального розмаху (від мінімуму до максимуму). Головною перевагою підходу є суворе збереження нульових значень, що особливо корисно для обробки обсягів торгів. Однак алгоритм є

вкрай вразливим до наявності екстремальних аномалій (викидів). Один різкий ціновий спад або пік стисне всю основну масу робочих даних у настільки вузький діапазон, що нейромережа втратить здатність розрізняти дрібні цінові коливання.

- *Стандартизація* (Z-Score Normalization) на відміну від попереднього методу не обмежує дані жорсткими числовими межами. Її мета – змістити часовий ряд таким чином, щоб його середнє арифметичне значення дорівнювало нулю, а стандартне відхилення – одиниці. Цей підхід є значно стійкішим до поодиноких ринкових викидів. Оскільки більшість функцій активації глибоких нейромереж, зокрема рекурентних, працюють найефективніше саме в околі нуля, просторова симетрія, яку забезпечує стандартизація, робить її базовим і найпоширенішим вибором для підготовки фінансових даних.
- *Робастне шкалювання* (Robust Scaling). Враховуючи специфіку криптовалютного ринку, що характеризується непередбачуваними спалахами волатильності та так званими «важкими хвостами» у розподілі прибутковості, класичні статистичні метрики можуть виявитися недостатньо надійними. Будь-який сильний ринковий шок суттєво викривляє класичне середнє значення та дисперсію вибірки. Робастне шкалювання вирішує цю проблему шляхом відмови від цих метрик на користь медіани (центрального значення відсортованого ряду) та інтерквартильного розмаху, який охоплює лише центральні п'ятдесят відсотків даних. Таким чином, усі екстремальні відхилення та шуми, що лежать на краях розподілу, повністю ігноруються при розрахунку масштабу. Це забезпечує максимальну стабільність і стійкість процесу навчання нейромережі навіть на найбільш зашумлених та хаотичних ділянках ринку.

З огляду на наведені характеристики та чутливість нейронних мереж до нестабільності градієнтів під час зворотного поширення помилки [18], у даній роботі як основний метод було обрано стандартизацію (Standard Scaling).

Такий підхід гарантовано приводить усі ознаки до єдиного масштабу, що суттєво прискорює збіжність алгоритму Adam [26].

Окрім трансформації цільової змінної, важливу роль відіграє формування простору ознак (Feature Engineering). Для підвищення інформативності моделі до вхідного вектору часто додають технічні індикатори, які агрегують інформацію про ринкові тренди та волатильність. До таких індикаторів відносять прості ковзні середні (SMA) для згладжування шуму, індекс відносної сили (RSI) для визначення зон перекупленості/перепроданості, а також історичну волатильність [28].

Комплексна попередня обробка, що включає логарифмування, нормалізацію та генерацію додаткових ознак, дозволяє перетворити хаотичний потік біржових котирувань на структурований набір даних, придатний для ефективного навчання глибоких нейронних мереж [22].

Висновки за розділом 1

У першому розділі кваліфікаційної роботи проведено комплексний аналіз задачі прогнозування курсу криптовалют, яку сформульовано як задачу обробки нелінійних нестаціонарних часових рядів. На основі огляду фахової літератури [6, 14] встановлено, що специфіка криптовалютного ринку, яка полягає у наявності важких хвостів розподілу та кластеризації волатильності, робить класичні економетричні методи, такі як ARIMA, недостатньо ефективними для отримання високоточних прогнозів. Доведено, що лінійні моделі не здатні адекватно описувати складну динаміку активу при різких змінах ринкової кон’юнктури [31, 35].

У ході порівняльного аналізу сучасних методів машинного навчання обґрунтовано доцільність використання апарату штучних нейронних мереж. Визначено, що хоча багатошарові персептрони (MLP) є потужними універсальними апроксиматорами, вони мають суттєвий недолік у вигляді відсутності механізму врахування часової послідовності подій [11]. У зв’язку з цим, як найбільш перспективну архітектуру для подальшого дослідження

обрано рекурентні нейронні мережі типу LSTM. Завдяки наявності блоків пам’яті та системи гейтів, LSTM здатні ефективно вирішувати проблему зникаючого градієнта та виявляти довгострокові залежності в історичних даних, що є критично важливим для фінансового прогнозування [9, 23].

У розділі описана стратегія попередньої обробки даних для задачі прогнозування курсу криптовалют. Показано, що безпосереднє використання абсолютних цін активу призводить до проблеми нестаціонарності, що унеможливлює коректне навчання моделей на тривалих історичних інтервалах. Тому прийнято рішення про перехід до прогнозування логарифмічної прибутковості (Log Returns), що дозволяє стабілізувати дисперсію ряду та забезпечити інваріантність моделі до масштабу цін [15, 32].

Сформульовані в розділі теоретичні положення та вимоги до вхідних даних створюють необхідну базу для переходу до другого розділу роботи, присвяченого безпосередньому проєктуванню архітектури нейромережевих моделей та формалізації процесу їх навчання.

2 ПРОЄКТУВАННЯ НЕЙРОМЕРЕЖЕВИХ МОДЕЛЕЙ ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ

2.1 Формалізація процесу побудови моделей прогнозування

Процес розробки та валідації нейромережевих моделей для прогнозування курсу криптовалют є складною багатокроковою процедурою, яка вимагає чіткої структуризації та дотримання принципів системного аналізу. Згідно з методологією CRISP-DM (Cross-Industry Standard Process for Data Mining), адаптованою для фінансових часових рядів, життєвий цикл побудови моделі розглядається як ітеративний процес, що включає збір даних, попередню обробку, конструювання ознак, проєктування архітектури, навчання, валідацію та тестування [5].

У роботі формалізований процес побудови моделей представлено у вигляді алгоритмічної послідовності дій (рис. 2.1), де вихідні дані кожного попереднього етапу є вхідними для наступного, а зворотні зв’язки дозволяють коригувати гіперпараметри системи для досягнення заданої точності.

Першим етапом є отримання та первинний аналіз даних. Джерелом інформації виступають історичні дані біржових котирувань у форматі OHLCV, які включають ціни відкриття, максимуму, мінімуму, закриття та обсяг торгів. На цьому етапі здійснюється перевірка цілісності масивів, виявлення пропущених значень та аномальних викидів, які можуть негативно вплинути на збіжність нейронної мережі. Для забезпечення репрезентативності вибірки та здатності моделі до узагальнення використовується широкий часовий діапазон з 2018 по 2025 роки, що охоплює різні фази ринкового циклу криптовалют, включаючи періоди стрімкого зростання та корекції [32].

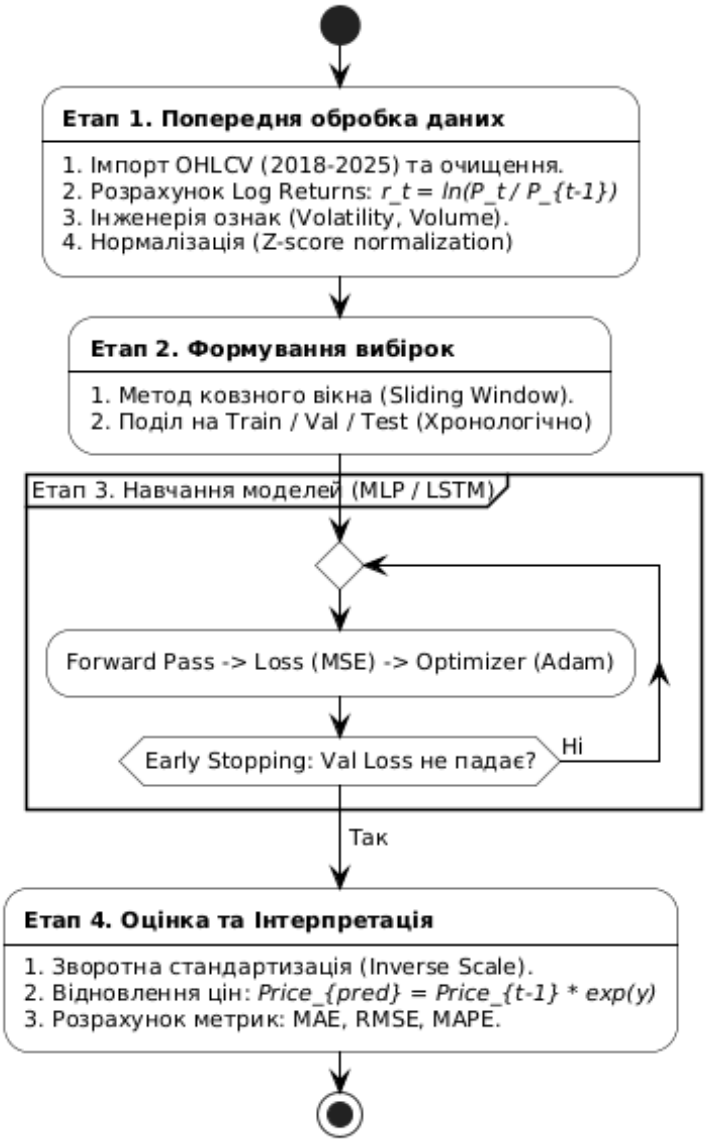


Рисунок 2.1 – Блок-схема алгоритму побудови та валідації моделей прогнозування

На другому етапі відбувається трансформація та інженерія ознак. Відповідно до висновків, зроблених у першому розділі, критично важливим кроком є перетворення нестаціонарного ряду абсолютних цін у стаціонарний ряд логарифмічних прибутковостей. Це дозволяє стабілізувати дисперсію та усунути проблему зміщення розподілу даних. Математично ця трансформація для моменту часу t описується наступним відношенням:

$$R_t = \ln\left(\frac{P_t}{P_{t-1}}\right), \tag{2.1}$$

де P_t – ціна закриття в момент часу t,

P_{t-1} – ціна в попередній момент.

Додатково до вектору вхідних ознак додаються розрахункові індикатори, зокрема історична волатильність та обсяг торгів, що дозволяє моделі ідентифікувати періоди підвищеного ризику [15]. Після формування простору ознак виконується процедура стандартизації даних (Z-score normalization), яка приводить усі змінні до єдиного діапазону з нульовим математичним сподіванням та одиничною дисперсією [22].

Третій етап полягає у формуванні навчальних вибірок методом ковзного вікна (Sliding Window). Часовий ряд перетворюється на набір послідовностей фіксованої довжини, де кожному вхідному вектору $X(t)$, що містить історію поведінки активу за n кроків, ставиться у відповідність цільове значення прибутковості на наступному кроці. Важливим аспектом проєктування є поділ генеральної сукупності даних на три незалежні підмножини: навчальну, валідаційну та тестову. У роботі використано хронологічний принцип поділу, що імітує реальні умови прогнозування, коли модель навчається виключно на даних минулого і тестується на даних майбутнього, які не брали участі в процесі оптимізації параметрів [1].

Четвертий етап охоплює побудову та навчання моделей MLP та Stacked LSTM. Формалізація процесу навчання зводиться до задачі мінімізації цільової функції втрат у багатовимірному просторі вагових коефіцієнтів мережі. В якості критерію оптимізації обрано середньоквадратичну помилку (MSE), яка визначається формулою:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{2.2}$$

де N – розмір вибірки (батчу),

y_i – реальне значення прибутковості,

$\hat{y}_i$ – прогнозоване значення,

θ – вектор параметрів моделі.

Мінімізація функції $L(\theta)$ здійснюється за допомогою адаптивного алгоритму оптимізації Adam. Для запобігання перенавчанню впроваджено механізм ранньої зупинки (Early Stopping), який припиняє процес навчання, якщо помилка на валідаційній вибірці не зменшується протягом заданої кількості епох, що гарантує збереження узагальнюючої здатності моделі [16].

Фінальним етапом є оцінка якості та інтерпретація результатів. Ефективність розроблених моделей перевіряється на відкладеній тестовій вибірці за допомогою метрик середньої абсолютної помилки (MAE) та середньоквадратичної помилки (RMSE). Критично важливою частиною цього етапу є процедура зворотної трансформації прогнозованих значень з простору логарифмічних прибутковостей у простір абсолютних цін за формулою

$$P_{forecast} = P_{last} \cdot e^{R_{pred}}. \quad (2.3)$$

Це дозволяє візуалізувати результати моделювання, порівняти їх з реальною ринковою динамікою та оцінити практичну придатність розроблених архітектур для вирішення прикладної задачі прогнозування курсу [13].

2.2 Архітектура нейромережевих моделей прогнозування

У дослідженні для вирішення задачі короткострокового прогнозування курсу криптовалюти спроєктовано та програмно реалізовано дві різні архітектури нейронних мереж: багатошаровий персептрон (MLP) та глибока рекурентна мережа довгої короткострокової пам’яті (Stacked LSTM). Вибір саме цих архітектур обумовлений необхідністю проведення порівняльного аналізу ефективності класичних методів апроксимації функцій та спеціалізованих методів обробки послідовностей у задачах фінансового

моделювання. Обидві моделі реалізовано з використанням бібліотеки глибокого навчання PyTorch, що забезпечує гнучкість у налаштуванні обчислювальних графів та ефективне використання апаратних ресурсів.

Першою базовою моделлю є *MLP Regressor* (рис. 2.2), який являє собою нейронну мережу прямого поширення.

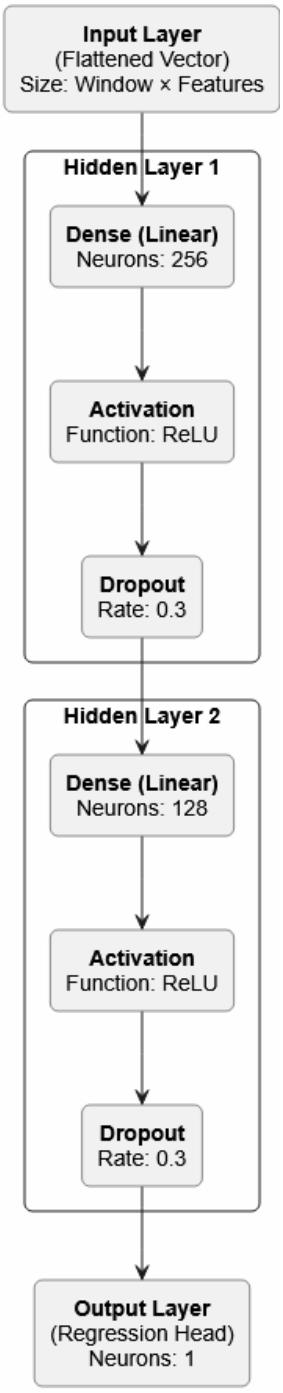


Рисунок 2.2 – Структурна схема моделі MLP

Архітектура моделі складається з вхідного шару, розмірність якого визначається добутком довжини ковзного вікна на кількість ознак, двох прихованих повнозв’язних шарів та вихідного шару. Математично функціонування кожного прихованого шару описується лінійною трансформацією вхідного вектора з подальшим застосуванням нелінійної функції активації. Для забезпечення здатності мережі відтворювати складні нелінійні залежності у роботі використано функцію активації ReLU (Rectified Linear Unit), яка визначається формулою:

$$f(x) = \max(0, x). \tag{2.4}$$

Перевагою ReLU порівняно з класичними сигмоїдальними функціями є вирішення проблеми згасання градієнта у глибоких мережах та прискорення обчислень. Для запобігання перенавчанню моделі, враховуючи стохастичну природу фінансових даних, після кожного прихованого шару застосовано метод регуляризації Dropout. Цей метод полягає у випадковому “вимкненні” певної частки нейронів під час тренування, що змушує мережу формувати більш стійкі та розподілені ознаки.

Другою, основною моделлю дослідження, є *Stacked LSTM Regressor* (рис. 2.3). На відміну від MLP, ця архітектура розроблена спеціально для роботи з часовими рядами і враховує хронологічну послідовність даних. У роботі використано архітектуру ”Stacked” (стекову), яка передбачає послідовне з’єднання кількох шарів LSTM. Це дозволяє моделі формувати ієрархічне представлення даних: перший шар виділяє низькорівневі часові патерни, а другий шар інтегрує їх у більш абстрактні високорівневі залежності. Ключовим елементом архітектури є комірка пам’яті LSTM, яка керує потоком інформації за допомогою механізму гейтів.

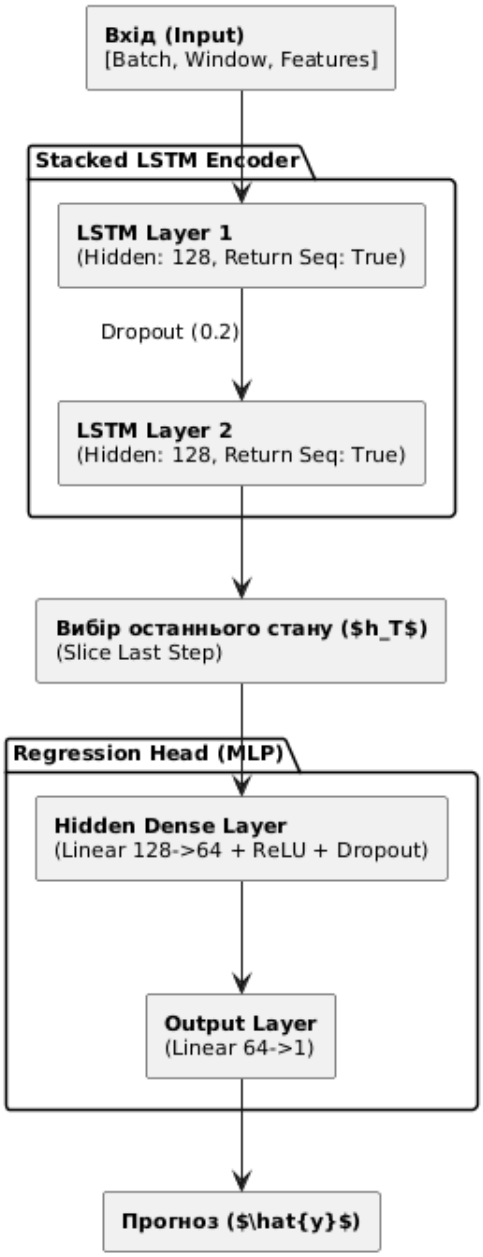


Рисунок 2.3 – Структурна схема двошарової рекурентної мережі

Математична модель одного кроку роботи LSTM-комірки описується системою рівнянь. Спочатку обчислюється стан гейта забування (forget gate), який визначає, яку частину інформації з попереднього стану клітинки C_{t-1} слід зберегти:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f). \quad (2.5)$$

Далі обчислюються вхідний гейт (input gate) та вектор кандидатів у новий стан клітинки, що відповідають за додавання нової інформації до пам’яті:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{2.6}$$

$$\tilde{C_t} = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C). \tag{2.7}$$

На основі отриманих значень відбувається оновлення глобального стану клітинки C_t, яке є сумою збереженої старої інформації та нової зваженої інформації:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C_t}. \tag{2.8}$$

Нарешті, обчислюється вихідний гейт (output gate) та поточний прихований стан h_t, який передається на наступний крок часу або на наступний шар мережі:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{2.9}$$

$$h_t = o_t \cdot \tanh(C_t). \tag{2.10}$$

У наведених формулах (2.6), (2.7), (2.8) і (2.9) σ – сигмоїдальна функція активації, $\tanh$ – гіперболічний тангенс, W та b – вагові коефіцієнти та зміщення. У спроєктованій архітектурі Stacked LSTM вихідний прихований стан останнього шару передається у спеціалізований вихідний блок (Head), який трансформує простір ознак у скалярне значення прогнозу.

Конкретна конфігурація гіперпараметрів для розроблених моделей була обрана емпіричним шляхом на основі серії експериментів. Для моделі MLP реалізовано топологію з двома прихованими шарами розмірністю 256 та 128 нейронів відповідно. Між шарами встановлено прошарки Dropout з

коефіцієнтом ймовірності 0.3, що забезпечує агресивну регуляризацію для запобігання запам’ятовуванню шуму.

Важливим етапом проєктування архітектури є вибір оптимального розміру ковзного вікна (Look-back window), який визначає «глибину пам’яті» моделі. Занадто вузьке вікно призводить до того, що мережа не має достатньо контексту для виявлення закономірностей і починає реагувати на випадковий шум як на зміну тренду. Натомість занадто широке вікно не лише експоненційно збільшує час навчання через зростання обчислювальної складності, але й спричиняє ефект «розмиття» актуальної інформації застарілими даними. У ході серії експериментів було встановлено, що оптимальний розмір вікна повинен корелювати з обраним таймфреймом: для внутрішньоденної торгівлі (15 хвилин) достатньо вікна, що охоплює кілька годин ринкової активності, тоді як для денних прогнозів модель потребує даних щонайменше за кілька попередніх тижнів.

Не менш критичним параметром є розмір пакету даних (Batch Size). Обробка даних малими пакетами дозволяє моделі швидше оновлювати ваги та уникати потрапляння в локальні мінімуми функції втрат завдяки високому рівню стохастичності. Проте фінансові дані є високошумними, і малий розмір пакету робить процес навчання нестабільним – градієнтні кроки стають занадто хаотичними. Використання великих пакетів забезпечує більш точну та усереднену оцінку напрямку антиградієнта, а також дозволяє максимально ефективно утилізувати обчислювальні потужності графічного процесора (GPU). Саме тому архітектура була налаштована на використання збалансованого розміру пакету, який гарантує плавну збіжність помилки.

Архітектура Stacked LSTM реалізована з використанням двох послідовних рекурентних шарів (num_layers=2) із розмірністю прихованого стану 128 нейронів. Для покращення збіжності та уникнення перенавчання між LSTM-шарами використовується Dropout 0.2. Вихідний сигнал з останнього шару LSTM подається на повнозв’язний блок (Head), який складається з лінійного шару (128 → 64 нейрони), функції активації ReLU,

шару Dropout (0.2) та фінального нейрона, що генерує прогноз логарифмічної прибутковості. Така складна структура “голови” мережі дозволяє додатково обробити виділені часові ознаки перед формуванням кінцевого результату.

2.3 Програмна реалізація та інструментальні засоби

Практична реалізація розроблених нейромережевих моделей та алгоритмів прогнозування виконана з використанням високорівневої мови програмування Python. Вибір даного інструментарію обумовлений його домінуючою позицією у сфері науки про дані (Data Science) та наявністю розвиненої екосистеми бібліотек для машинного навчання, статистичного аналізу та візуалізації результатів. Розроблене програмне забезпечення функціонує як модульна система, що забезпечує повний цикл обробки даних: від завантаження історичних котирувань до генерації сигналів та оцінки точності прогнозу.

Основним інструментом для побудови та навчання нейронних мереж обрано бібліотеку глибокого навчання PyTorch. На відміну від статичних графів обчислень (як у TensorFlow версій 1.x), PyTorch використовує динамічний обчислювальний граф (Dynamic Computational Graph), що дозволяє гнучко змінювати архітектуру мережі "на льоту" та значно спрощує процес налагодження рекурентних моделей (LSTM). Для роботи з тензорними операціями та матричними обчисленнями використано бібліотеку NumPy, а для маніпуляцій з часовими рядами та табличними даними у форматі CSV – бібліотеку Pandas.

Архітектура програмного рішення побудована на об'єктно-орієнтованому підході. Базовим класом для моделей є nn.Module бібліотеки PyTorch, від якого успадковані реалізовані класи MLPRegressor та LSTMRegressor. Процес підготовки даних реалізовано через стандартні механізми PyTorch TensorDataset та DataLoader, що дозволяє ефективно керувати пам'яттю та виконувати пакетну (batch) обробку даних. У програмі

використано розмір пакету (BATCH_SIZE) 512 зразків, що забезпечує баланс між швидкістю навчання та стабільністю оцінки градієнта.

Важливим для процесу навчання є вибір алгоритму оптимізації. У програмному коді застосовано алгоритм стохастичної оптимізації Adam (Adaptive Moment Estimation), який автоматично адаптує крок навчання для кожного параметру мережі окремо. Для підвищення ефективності збіжності реалізовано планувальник швидкості навчання ReduceLROnPlateau. Цей механізм автоматично зменшує Learning Rate (початкове значення 10−3) у два рази, якщо помилка на валідаційній вибірці перестає знижуватися, що дозволяє моделі більш точно знаходити локальний мінімум функції втрат.

Для запобігання перенавчанню (overfitting) та економії обчислювальних ресурсів розроблено та імплементовано клас EarlyStopper. Він реалізує стратегію ранньої зупинки: процес навчання переривається, якщо метрика якості на валідаційній множині не покращується протягом визначеної кількості епох (PATIENCE = 12). Це гарантує, що у фінальну експлуатацію буде взято модель з найкращою здатністю до узагальнення, а не модель, яка просто запам'ятала навчальну вибірку.

Для оцінки якості моделей та трансформації даних використано бібліотеку Scikit-learn. Зокрема, клас StandardScaler забезпечує нормалізацію вхідних ознак (z-score scaling), а функції mean_squared_error та mean_absolute_error використовуються для розрахунку фінальних метрик. Програмне забезпечення також підтримує апаратне прискорення обчислень за технологією CUDA (Compute Unified Device Architecture). Код автоматично визначає наявність графічного процесора (GPU) NVIDIA та переносить обчислення тензорних операцій з центрального процесора (CPU) на відеокарту, що дозволяє скоротити час навчання глибоких моделей у десятки разів.

Візуалізація результатів реалізована за допомогою бібліотеки Matplotlib. Програма автоматично генерує графіки динаміки функції втрат (Learning Curves) та графіки порівняння прогнозованих цін з реальними (Predicted vs

Actual), що дозволяє проводити якісний візуальний аналіз адекватності роботи моделей на різних часових інтервалах.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи здійснено теоретичне обґрунтування та практичну реалізацію методології побудови інтелектуальних систем для прогнозування курсу криптовалют. На основі проведеного аналізу предметної області було формалізовано повний життєвий цикл розробки моделей, починаючи від попередньої обробки "сирих" біржових даних і закінчуючи процедурами валідації результатів.

Ключовим досягненням етапу проєктування стало вирішення проблеми нестаціонарності фінансових часових рядів. Обґрунтовано та впроваджено підхід, що базується на переході від абсолютних цінових значень до логарифмічних прибутковостей (LogReturns). Це дозволило стабілізувати дисперсію даних та забезпечити коректні умови для навчання градієнтних методів. Використання методу ковзного вікна (Sliding Window) дозволило трансформувати задачу часового прогнозування у формат навчання з учителем, придатний для обробки нейронними мережами.

В ході дослідження було спроєктовано архітектури двох нейромережевих моделей: багатошарового персептрона (MLP) та рекурентної мережі довгої короткострокової пам'яті (Stacked LSTM). Для моделі LSTM було обрано та реалізовано двошарову структуру з розмірністю прихованого стану 128 нейронів та спеціалізованим вихідним блоком. Такий підхід дозволяє моделі ефективно виявляти як короткострокові флуктуації, так і довгострокові тренди ринку, використовуючи механізм гейтів для фільтрації інформаційного шуму.

Визначено параметри процесу навчання моделей, який включає використання адаптивного оптимізатора Adam, динамічне регулювання швидкості навчання (Learning Rate Scheduler) та методи регуляризації (Dropout). Впровадження механізму ранньої зупинки (Early Stopping)

дозволило автоматизувати процес пошуку оптимальної кількості епох навчання та запобігти перенавчанню моделей на обмежених вибірках.

Програмна реалізація системи виконана мовою Python з використанням бібліотеки PyTorch, що забезпечує високу продуктивність обчислень завдяки підтримці апаратного прискорення на GPU. Створене програмне забезпечення є гнучким інструментом, що дозволяє проводити серії експериментів на різних часових інтервалах (від 15 хвилин до 1 дня) та оперативно оцінювати якість прогнозів. Отримані на цьому етапі інструментальні засоби та моделі є повністю готовими до проведення експериментальних досліджень, результати яких будуть викладені у третьому розділі роботи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ МОДЕЛЕЙ ПРОГНОЗУВАННЯ

3.1 Програмна реалізація попередньої обробки даних

Важливим компонентом розробленої моделі є етап попередньої обробки даних, результат якого безпосередньо впливає на стабільність навчання та узагальнюючу здатність нейромережевих моделей. Оскільки сирі біржові котирування, що надходять із зовнішніх джерел, можуть мати структурні розбіжності, програмне рішення реалізує повний цикл очищення, фільтрації та трансформації вхідних потоків інформації.

Першим етапом роботи є завантаження та уніфікація часових рядів. Програма здійснює послідовне зчитування історичних файлів для кожного з досліджуваних часових інтервалів. Автоматично виконується пошук колонки з часовими мітками за ключовими словами та здійснює її обов'язкове перетворення у стандартизований формат дат із прив'язкою до єдиного міжнародного часового поясу UTC. На цьому ж етапі відбувається хронологічне сортування всього масиву даних та видалення тих записів, які мають пошкоджені або відсутні часові мітки, що гарантує безперервність подальшого аналізу. Програма також виконує автоматичне перейменування робочих колонок до загальноприйнятого в індустрії стандарту OHLCV та перевіряє наявність інформації про обсяги торгів. У разі повної відсутності даних про обсяги на певних ділянках, програмний модуль ініціалізує цю змінну нульовими значеннями для збереження цілісності простору ознак.

Другим, критично важливим етапом препроцесингу є масштабування ознак за допомогою методів стандартизації. Оскільки вхідні дані мають суттєво різні числовий діапазони, програмна реалізація передбачає використання StandardScaler для приведення розподілу кожної змінної до нульового математичного сподівання та одиничної дисперсії. При цьому в алгоритмі реалізовано суворе архітектурне обмеження для запобігання ефекту витоку даних з майбутнього в минуле (Data Leakage). Обчислення параметрів

стандартизації виконується виключно на основі тренувальної вибірки. Валідаційний та тестовий набори даних трансформуються на основі вже отриманих параметрів за тренувальною вибіркою. Такий підхід гарантує, що моделі під час тестування працюють у реалістичних ринкових умовах, не маючи жодного уявлення про статистичні характеристики майбутніх періодів.

Третім етапом є алгоритмічне формування навчальних та тестових зразків за методом ковзного вікна. Програмна функція трансформації перетворює лінійні табличні дані у набори послідовностей фіксованої довжини. Довжина цього вікна динамічно адаптується під кожен таймфрейм відповідно до визначених констант. Програма ітеративно проходить по всьому масиву даних, формуючи вхідні матриці, що містять ретроспективну історію поведінки активу, та зіставляє їм відповідні цільові значення ціни закриття на наступному кроці, враховуючи заданий одиничний горизонт прогнозування.

Фінальний етап підготовки даних враховує специфіку кожної з двох досліджуваних архітектур. Оскільки рекурентна мережа LSTM за своєю природою очікує на вхід тривимірні тензори, що зберігають часову послідовність, сформовані ковзні вікна передаються в неї без змін. Водночас багатошаровий персептрон MLP є мережею прямого поширення і не здатний працювати з багатовимірними часовими структурами. Для вирішення цієї проблеми в програмному коді реалізовано етап векторизації (випрямлення) даних: перед подачею на вхід MLP тривимірні матриці автоматично переформатовуються у двовимірні плоскі вектори, де довжина вікна множиться на кількість ознак. Оброблена таким чином інформація упаковується в спеціалізовані об'єкти TensorDataset та передається в DataLoader для пакетного навчання з розміром батчу в 512 зразків, що забезпечує максимальну швидкість обчислень при використанні апаратного прискорення CUDA.

3.2 Методика проведення експериментів та метрики оцінки якості

Експериментальна частина кваліфікаційної роботи спрямована на емпіричну перевірку розроблених нейромережевих архітектур та порівняльний аналіз їхньої здатності прогнозувати динаміку курсу біткоїна (BTC/USD). Основною метою експерименту є не лише отримання мінімальної помилки прогнозу, але й визначення залежності якості моделювання від обраного часового інтервалу (Timeframe) та типу архітектури (MLP проти LSTM).

Методика проведення експериментів базується на принципі “Walk-Forward Validation” (перевірка з просуванням у часі), що імітує реальні умови алгоритмічної торгівлі. У якості емпіричної бази дослідження використано набір даних «Bitcoin Historical Datasets 2018–2025», отриманий з відкритого репозиторію Kaggle. Загальний масив даних охоплює період з січня 2018 року по початок 2025 року та був розділений на три незалежні підмножини з чіткою хронологічною послідовністю:

1. *Навчальна вибірка* (Training Set): дані до 31 грудня 2022 року. Використовується для оптимізації вагових коефіцієнтів моделей (backpropagation).

2. *Валідаційна вибірка* (Validation Set): період з 01 січня 2023 по 31 грудня 2023 року. Використовується для налаштування гіперпараметрів, вибору найкращої архітектури та роботи механізму Early Stopping.

3. *Тестова вибірка* (Test Set): дані з 01 січня 2024 року по теперішній час (2025 рік). Це “невидимі” для моделі дані, на яких здійснюється фінальна оцінка якості. Такий підхід гарантує об’єктивність результатів, оскільки модель тестується на ринкових умовах, які не були відомі під час її навчання.

Дослідження проводилося на чотирьох різних таймфреймах: 15 хвилин, 1 година, 4 години та 1 день. Для кожного таймфрейму було встановлено специфічний розмір вхідного вікна (Look-back window), який визначався емпірично (від 32 до 128 кроків) для забезпечення балансу між глибиною історичної пам’яті та актуальністю ринкових патернів.

Критично важливою особливістю методики оцінки є *зворотна трансформація прогнозу*. Оскільки моделі навчаються на нормалізованих логарифмічних прибутковостях ($LogReturns$), прямий вихід нейронної мережі є важким для економічної інтерпретації. Тому перед розрахунком метрик якості виконується процедура реконструкції абсолютної ціни за формулою:

$$P_{pred}(t+1) = P_{actual}(t) \cdot \exp(y_{pred}), \tag{3.1}$$

де $P_{actual}(t)$ – реальна ціна закриття на попередньому кроці, y_{pred} – прогнозована логарифмічна прибутковість.

Саме на основі відновлених цін (P_{pred}) та реальних ринкових цін (P_{true}) розраховуються метрики ефективності.

Для комплексної оцінки точності моделей використано три статистичні метрики:

1. Середня абсолютна помилка (Mean Absolute Error, MAE) показує, на скільки доларів в середньому помиляється модель:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|. \tag{3.2}$$

Ця метрика є лінійною та легкою для інтерпретації.

2. Корінь із середньоквадратичної помилки (Root Mean Squared Error, RMSE):

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}. \tag{3.3}$$

Ця метрика сильніше штрафує модель за великі відхилення (викиди). У контексті фінансових ринків RMSE є критично важливим показником,

оскільки велика помилка може призвести до значних фінансових втрат (наприклад, під час різких цінових стрибків).

3. Середня абсолютна помилка у відсотках (Mean Absolute Percentage Error, MAPE):

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i-\hat{y}_i}{y_i}\right| \cdot 100\%. \quad (3.4)$$

Оскільки вартість біткоїна змінюється в широкому діапазоні, абсолютні помилки в доларах можуть бути важкими для порівняння на різних проміжках часу. MAPE дозволяє оцінити відносну точність моделі у відсотках, що робить її універсальною метрикою для порівняння ефективності.

У формулах (3.1) – (3.4) y_i – реальна ціна активу, $\hat{y}_i$ – прогнозована ціна, N – кількість спостережень у тестовій вибірці.

Стратегія аналізу передбачає поетапне порівняння: спочатку зіставляються результати роботи MLP та LSTM для кожного окремого таймфрейму, після чого проводиться крос-аналіз впливу частоти даних на точність прогнозування. Найкращою вважається модель, яка демонструє найменші значення MAPE та RMSE на тестовій вибірці, при цьому зберігаючи стабільність прогнозу без значних запізнень (lag effect).

3.3 Результати експериментального моделювання

Для проведення порівняльного аналізу ефективності розроблених нейромережевих архітектур було виконано серію експериментів на тестовій вибірці, що охоплює період з 1 січня 2024 року по поточний момент. Тестування здійснювалося для чотирьох часових інтервалів: 15 хвилин, 1 година, 4 години та 1 день. Для кожного сценарію було розраховано метрики MAE (в доларах США), RMSE та MAPE (у відсотках). Отримані кількісні показники точності прогнозування для моделей MLP та Stacked LSTM зведено в таблицю 3.1.

Детальний аналіз отриманих результатів дозволяє виявити чітку залежність між довжиною горизонту прогнозування та величиною похибки. На мінімальному досліджуваному інтервалі в 15 хвилин спостерігається найвища точність прогнозу. MAE становить близько 137 доларів, що при середній вартості активу в досліджуваному періоді еквівалентно похибці менше ніж 0.18%. На цьому таймфреймі різниця між архітектурами є статистично незначущою: LSTM випереджає MLP лише на соті частки відсотка. Це пояснюється інерційністю ринку на макрорівні, де ціна за короткий проміжок часу не встигає зазнати суттєвих змін, тому навіть прості моделі прямого поширення здатні забезпечити високу точність апроксимації.

Таблиця 3.1 - Порівняльна характеристика точності прогнозування моделей

Часовий інтервал	**Модель**	**MAE,**	**RMSE, $**	**MAPE, %**
15 хвилин (15m)	MLP	137.04	210.66	0.1709
	LSTM	**137.01**	**210.56**	**0.1708**
1 година (1h)	MLP	272.49	419.49	0.3394
	LSTM	**271.15**	**418.86**	**0.3378**
4 години (4h)	MLP	553.01	833.16	0.6895
	LSTM	**550.35**	**831.36**	**0.6858**
1 день (1d)	MLP	1463.24	2021.42	1.8310
	LSTM	**1427.97**	**1993.19**	**1.7929**

При збільшенні часового інтервалу до 1 години та 4 годин похибка закономірно зростає. Для годинного інтервалу MAPE становить близько 0.34%, а для 4-годинного зростає до 0.69%. Абсолютна помилка на 4-годинному таймфреймі сягає 550 доларів. Саме на цих інтервалах починає проявлятися перевага рекурентної архітектури. LSTM демонструє менші значення як MAE, так і RMSE, що свідчить о її кращій здатності фільтрувати

ринковий шум. Різниця в середньоквадратичній помилці (RMSE) на користь LSTM вказує на те, що ця модель робить менше грубих помилок під час локальних сплесків волатильності.

Найбільш показовими є результати для денного інтервалу, де складність прогнозування є максимальною через вплив фундаментальних факторів та новинного фону. Похибка MAPE зростає до рівня 1.8%, а середнє відхилення прогнозу від факту становить понад 1400 доларів. На цьому горизонті архітектурна перевага Stacked LSTM стає очевидною: модель знижує абсолютну помилку на 35 доларів порівняно з MLP, а середньоквадратичну – майже на 30 доларів. Це підтверджує гіпотезу про те, що механізм комірок пам'яті дозволяє LSTM ефективніше враховувати довгостроковий контекст цінових рухів, який втрачається у звичайних мережах прямого поширення через відсутність зворотних зв'язків.

Візуальний аналіз графіків прогнозування, представлених на рисунках 3.1 та 3.2, дозволяє оцінити характер поведінки моделей.

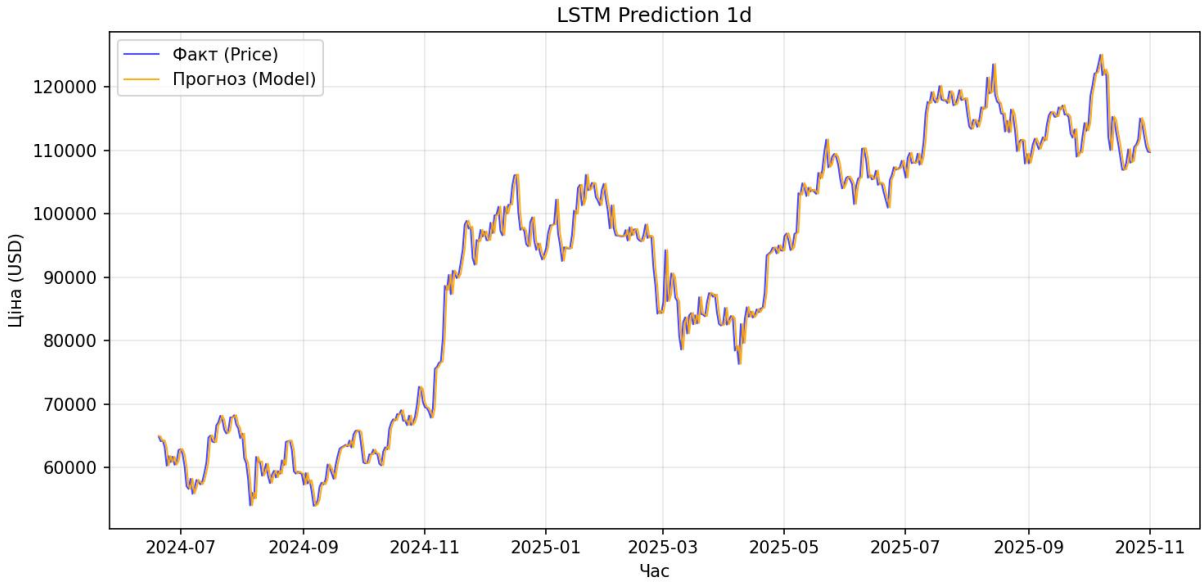


Рисунок 3.1 – Результат прогнозування моделі LSTM

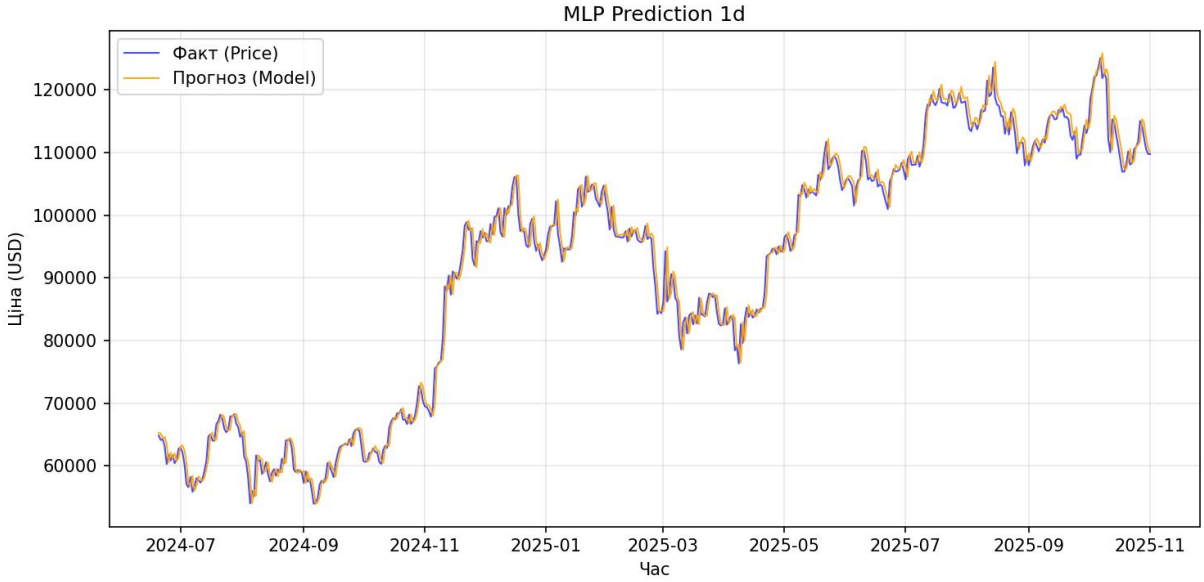


Рисунок 3.2 – Результат прогнозування моделі MLP

На графіках видно, що обидві моделі коректно відтворюють загальний напрямок тренду. Однак модель MLP характеризується ефектом запізнення реакції на різкі розвороти ринку. Це проявляється у зміщенні кривої прогнозу вправо відносно реальної ціни. У той же час, крива прогнозу LSTM є більш згладженою та краще адаптується до зміни локальних трендів, що підтверджується меншим значенням метрики RMSE.

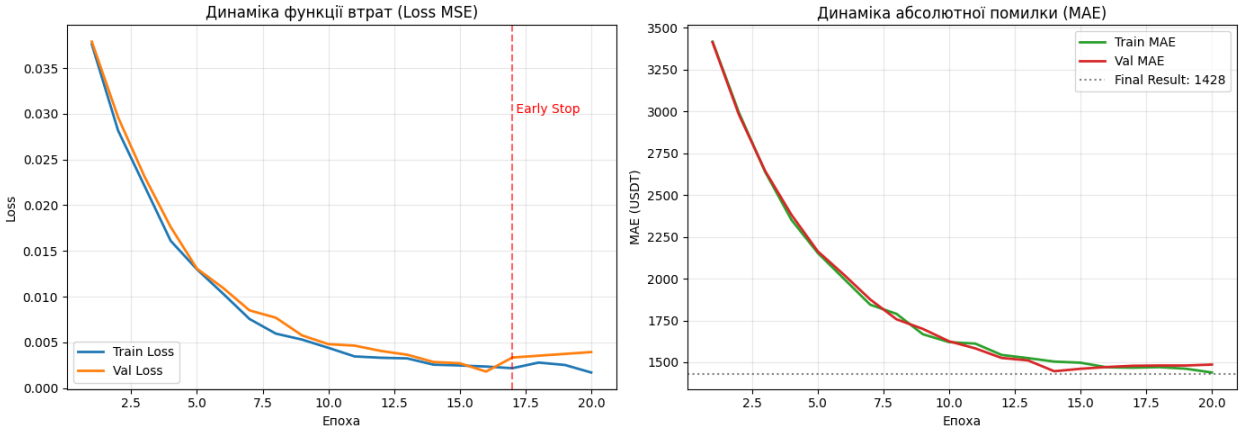


Рисунок 3.3 – Динаміка навчання Stacked LSTM: функція втрат (Loss) та реальна помилка (MAE) із точкою ранньої зупинки

Характер кривих свідчить про стабільність та коректність процесу навчання. Спостерігається стрімке експоненційне зниження помилки протягом перших 10 епох, що вказує на активне засвоєння моделлю закономірностей даних. Важливо зазначити, що криві валідації (Val) щільно слідують за кривими навчання (Train), не демонструючи різкого розходження, що підтверджує відсутність ефекту перенавчання (Overfitting). Механізм ранньої зупинки (Early Stopping) автоматично перервав навчання на 16-й епосі, коли помилка на валідаційній вибірці перестала покращуватися. Це підтверджує правильність вибору гіперпараметрів та ефективність використання шарів регуляризації Dropout.

Таким чином, експериментально підтверджено, що для короткострокового прогнозування (до 1 години) моделі MLP та LSTM є взаємозамінними, проте для середньострокових прогнозів (від 4 годин до 1 доби) використання глибоких рекурентних мереж є більш доцільним з точки зору мінімізації фінансових ризиків.

3.4 Обговорення результатів та порівняльний аналіз ефективності моделей

На основі отриманих експериментальних даних (п. 3.2) проведено комплексний аналіз, метою якого є інтерпретація поведінки досліджуваних нейромережевих архітектур та визначення меж їх практичної застосовності. Результати дослідження виявили низку стійких закономірностей, які потребують детального обговорення в контексті теорії часових рядів та алгоритмічної торгівлі.

Вплив часового масштабу (Timeframe) на точність прогнозування. Ключовим спостереженням є обернена залежність між часовим інтервалом та точністю моделі. Мінімальна помилка MAPE (0.17%) на 15-хвилинному таймфреймі пояснюється високим рівнем автокореляції цінових відліків. На таких коротких проміжках ринок має значну інерцію: ціна в момент часу t+1 з високою ймовірністю буде близькою до ціни в момент t. У цих умовах задача

прогнозування значною мірою зводиться до апроксимації локального тренду, з чим успішно справляються навіть прості архітектури.

Натомість, різке зростання помилки на денному інтервалі (MAPE ~1.8%) свідчить про зміну природи даних і вимагає окремого обговорення в контексті ринкового ціноутворення. Моделі, розроблені в межах даного дослідження, базуються виключно на історичних числових даних – цінах та обсягах торгів. Це класичний Data-Driven підхід, який є основою технічного аналізу. Проте рух курсу криптовалют на макрорівні визначається не стільки технічними патернами, скільки фундаментальними факторами та інформаційним фоном.

На денному інтервалі ціна біткоїна стає вкрай чутливою до зовнішніх екзогенних шоків. До них належать рішення регуляторних органів (наприклад, заборона майнінгу в певних юрисдикціях або судові позови проти великих бірж), публікації макроекономічної статистики (зміна ключової процентної ставки, рівень інфляції в США), а також заяви лідерів думок та інституційних інвесторів. Технічні індикатори та історичні ціни фіксують ці події постфактум, коли зміна курсу вже відбулася.

Нейронна мережа не має доступу до новинних стрічок або соціальних мереж, тому не може передбачити імпульсний рух, викликаний несподіваною публікацією. Відповідно, при збільшенні горизонту прогнозування до однієї доби зростає питома вага фундаментального впливу, який модель не здатна обробити. Це частково пояснює явище фазового запізнення: модель фіксує зміну динаміки лише після того, як новина була відображена в ціні активу. Цей висновок підтверджує тезу про те, що для довгострокового прогнозування суто технічного аналізу за допомогою нейромереж недостатньо. Це відкриває перспективний напрямок для подальших досліджень – створення мультимодальних архітектур, де один потік даних обробляє числові котирування через LSTM, а інший – здійснює семантичний аналіз текстових новин (Sentiment Analysis) за допомогою трансформерних моделей, поєднуючи технічний та фундаментальний аналізи в єдиній системі.

Порівняльний аналіз архітектур MLP проти LSTM. Експеримент показав, що перевага складних рекурентних мереж (LSTM) над простішими персептронами (MLP) є нелінійною:

1. Зона паритету (15m – 1h): на малих таймфреймах різниця в точності між MLP та LSTM є статистично незначущою (менше 0.01% за MAPE). Це свідчить про те, що для високочастотних даних, де переважає короткострокова залежність, обчислювальна складність LSTM є надлишковою. Використання MLP у цьому сегменті є більш виправданим з точки зору ресурсоємності та швидкості навчання.

2. Зона переваги LSTM (4h – 1d): зі збільшенням горизонту прогнозування LSTM починає демонструвати свою ефективність. Зменшення абсолютної помилки на 35 доларів на денному графіку підтверджує, що механізм гейтів (forget gate, input gate) дозволяє моделі утримувати в пам’яті довгострокові патерни волатильності, які MLP “забуває” через обмежену розмірність вхідного вікна.

Проблема фазового зсуву (Lag Effect). Візуальний аналіз графіків прогнозування (рис. 3.1–3.2) виявив характерну для методів машинного навчання проблему – фазовий зсув прогнозу. Моделі схильні прогнозувати різку зміну ціни із запізненням на один крок (t+1). Це пояснюється тим, що функція втрат (MSE) карає модель за відхилення від середнього значення, тому нейромережа обирає консервативну стратегію: “прогноз на завтра дорівнює ціні сьогодні плюс невелика поправка на тренд”.

Модель LSTM демонструє менший лаг порівняно з MLP завдяки здатності аналізувати послідовність змін (динаміку), а не лише статичні значення у вікні. Це робить її більш придатною для створення торгових стратегій, де своєчасність входу в позицію є критичною.

Економічна інтерпретація та практичні рекомендації. З точки зору практичного застосування результати можна інтерпретувати наступним чином:

1. Для скальпінгу та HFT (15m): низька похибка прогнозу (0.17%) теоретично дозволяє отримувати прибуток. Однак, слід враховувати торгові комісії біржі та спред (різницю між ціною купівлі та продажу). Якщо сумарні витрати на угоду перевищують 0.2%, економічна доцільність використання моделі на цьому таймфреймі є сумнівною, незважаючи на високу математичну точність.

2. Для свінг-трейдингу (4h – 1d): похибка в 1.8% є суттєвою, що створює високі ризики. Використання моделі на цих інтервалах рекомендовано не як генератор прямих сигналів “купити/продати”, а як допоміжний інструмент для визначення загального тренду (фільтр напрямку торгівлі) у поєднанні з класичними індикаторами технічного аналізу.

Таким чином, проведене дослідження доводить, що вибір архітектури нейронної мережі має диктуватися специфікою задачі: для швидких внутрішньоденних операцій достатньо оптимізованого MLP, тоді як для середньострокового позиціонування необхідна глибина аналізу, яку забезпечує Stacked LSTM.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи проведено комплексне експериментальне дослідження розроблених нейромережевих моделей прогнозування курсу біткоїна на тестовій вибірці, що охоплює період з 2024 по 2025 роки. Емпірична перевірка здійснювалася за методологією Walk-Forward Validation на чотирьох часових інтервалах: 15 хвилин, 1 година, 4 години та 1 день.

За результатами проведених експериментів та аналізу отриманих даних можна зробити певні висновки.

1. Ефективність прогнозування залежить від горизонту планування. Встановлено чітку обернену кореляцію між довжиною часового інтервалу та точністю моделі. Найкращі показники досягнуто на 15-хвилинному таймфреймі, де середня абсолютна похибка у відсотках (MAPE) склала 0.17%,

що свідчить про високу передбачуваність високочастотних коливань на основі історичних даних. Зі збільшенням горизонту до 1 дня похибка зростає до 1.8%, що обумовлено впливом фундаментальних факторів, які виходять за межі суто технічного аналізу.

2. Перевага архітектури LSTM є ситуативною. Доведено, що на коротких проміжках часу (до 1 години) проста модель багатошарового персептрона (MLP) демонструє результати, співставні зі складною рекурентною мережею. Це дозволяє рекомендувати використання MLP для задач високочастотної торгівлі з метою економії обчислювальних ресурсів. Однак на середньострокових інтервалах (4 години – 1 день) архітектура Stacked LSTM підтвердила свою перевагу, знизивши абсолютну помилку прогнозу (RMSE) та продемонструвавши кращу стійкість до ринкового шуму.

3. Виявлено проблему фазового запізнення. Графічний аналіз результатів показав, що обидві моделі схильні прогнозувати зміну ціни з певним часовим лагом. Втім, рекурентна мережа LSTM, завдяки механізму довгострокової пам'яті, демонструє більш згладжену криву прогнозу та швидшу адаптацію до зміни тренду порівняно з інерційною моделлю MLP.

4. Практична цінність моделей. Розраховані метрики точності підтверджують потенційну прибутковість використання розроблених алгоритмів. Модель для 15-хвилинного таймфрейму є придатною для побудови автоматизованих торгових стратегій за умови низьких транзакційних витрат. Моделі для денних інтервалів рекомендується використовувати як допоміжний інструмент для визначення глобального вектору ринку та фільтрації хибних сигналів інших індикаторів.

Узагальнюючи результати, можна стверджувати, що мета роботи досягнута: розроблена система на базі Stacked LSTM здатна прогнозувати динаміку криптовалютних активів з точністю, достатньою для прийняття обґрунтованих інвестиційних рішень, перевершуючи базові нейромережеві рішення на складних, високоволатильних ділянках ринку.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальну задачу прогнозування курсу криптовалюти (Bitcoin) з використанням методів глибокого навчання. На основі проведеного теоретичного аналізу, математичного моделювання та серії обчислювальних експериментів отримано наступні результати:

1. Обґрунтовано методологію обробки фінансових даних. Встановлено, що використання абсолютних цінових значень для навчання нейронних мереж є неефективним через нестаціонарність фінансових часових рядів. У роботі реалізовано підхід, що базується на трансформації даних у простір логарифмічних прибутковостей (LogReturns). Це дозволило стабілізувати дисперсію, привести розподіл даних до нормального вигляду та забезпечити коректну збіжність градієнтних методів оптимізації.

2. Розроблено та програмно реалізовано нейромережеві моделі. Спроєктовано дві архітектури: базовий багатошаровий персептрон (MLP) та глибоку рекурентну мережу (Stacked LSTM) з двома послідовними шарами пам'яті. Для підвищення узагальнюючої здатності моделей впроваджено комплекс методів регуляризації, що включає шари Dropout (із коефіцієнтом 0.2–0.3), адаптивний оптимізатор Adam та механізм ранньої зупинки навчання (Early Stopping). Програмна реалізація виконана мовою Python з використанням бібліотеки PyTorch та підтримує апаратне прискорення обчислень (CUDA).

3. Визначено залежність точності прогнозу від часового масштабу. Експериментально доведено наявність оберненої кореляції між горизонтом прогнозування та точністю моделі. Найкращі результати отримано на 15-хвилинному таймфреймі, де середня абсолютна помилка у відсотках (MAPE) склала 0.17%. Зі збільшенням інтервалу до 1 доби похибка зростає до 1.8%, що пояснюється впливом стохастичних макроекономічних факторів, які не можуть бути враховані виключно методами технічного аналізу.

4. Проведено порівняльний аналіз архітектур. Встановлено, що на коротких часових інтервалах (до 1 години) ефективність простої моделі MLP та складної моделі LSTM є статистично тотожною. Це робить доцільним використання простіших моделей для високочастотної торгівлі з метою економії ресурсів. Однак на середньострокових інтервалах (4 години – 1 день) архітектура Stacked LSTM продемонструвала беззаперечну перевагу, знизивши середньоквадратичну помилку (RMSE) та продемонструвавши кращу здатність до ідентифікації трендів в умовах високої волатильності.

5. Виявлено практичні обмеження та сформовано рекомендації. Аналіз графіків "прогноз-факт" виявив наявність ефекту фазового запізнення (lag effect), який є менш вираженим у моделі LSTM. З огляду на це, практичне використання розробленої системи рекомендується у двох форматах: як генератор сигналів для алгоритмічної торгівлі на малих таймфреймах (15m) та як інструмент підтвердження тренду (фільтр) для позиційної торгівлі на денних графіках.

Розроблена інформаційна система довела свою працездатність та ефективність. Отримані показники точності (MAPE < 2% на денному інтервалі та < 0.2% на 15-хвилинному) відповідають сучасним вимогам до систем алгоритмічного трейдингу. Результати дослідження можуть бути використані для створення торгових ботів, систем управління ризиками та як база для подальших наукових розвідок у напрямку гібридних нейромережевих моделей (CNN-LSTM) або навчання з підкріпленням (Reinforcement Learning).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Бідюк П. І., Романюк В. В. Методи прогнозування часових рядів : навч. посіб. Київ : НТУУ «КПІ», 2010. 124 с. URL: (https://ela.kpi.ua/handle/123456789/1056) (дата звернення: 12.04.2026).

2. Бородіна О. А., Білощицький А. О. Використання рекурентних нейронних мереж для прогнозування фінансових часових рядів. *Інформаційні технології та комп'ютерна інженерія*. 2020. № 2 (48). С. 15–22. URL: (https://doi.org/10.31649/1999-9941-2020-48-2-15-22) (дата звернення: 25.02.2026).

3. Глибовець М. М. Основи штучного інтелекту : підручник. Київ : Вид. дім «КМ Академія», 2019. 430 с.

4. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ : ДП «УкрНДНЦ», 2016. 17 с. URL: (http://online.budstandart.com/ua/catalog/doc-page?id_doc=61869) (дата звернення: 04.03.2026).

5. Ковальчук А. М., Савченко В. М. Аналіз методів машинного навчання для прогнозування курсу криптовалют. *Вісник Хмельницького національного університету*. 2021. № 3. С. 88–94. URL: (http://journals.khnu.km.ua/vestnik/) (дата звернення: 17.03.2026).

6. Кривонос Ю. Г., Романюк В. В., Ситник В. М. Інтелектуальний аналіз даних у фінансовій сфері : монографія. Київ : Наукова думка, 2017. 320 с.

7. Литвин В. В., Висоцька В. А. Інтелектуальні системи підтримки прийняття рішень : навч. посіб. Львів : Вид-во Львівської політехніки, 2019. 364 с. URL: (https://vlp.com.ua/node/16789) (дата звернення: 14.11.2025).

8. Любчик Л. М. Теорія систем та математичне моделювання : підручник. Харків : НТУ «ХПІ», 2018. 412 с.

9. Мельник А. О., Кіт Г. В. Дослідження ефективності застосування нейронних мереж LSTM для прогнозування вартості активів. *Системні дослідження та інформаційні технології*. 2022. № 1. С. 45–56. URL:

(https://doi.org/10.20535/SRIT.2308-8893.2022.1.04) (дата звернення: 03.01.2026).

10. Пасічник В. В., Щербина Ю. М. Машинне навчання : підручник. Львів : Магнолія 2006, 2019. 400 с.

11. Субботін С. О. Нейронні мережі: теорія та практика : навч. посіб. Житомир : Вид-во ЖДУ ім. І. Франка, 2020. 256 с. URL: (http://eprints.zu.edu.ua/30867/) (дата звернення: 19.04.2026).

12. Ткачук А. М. Методи оптимізації в системах управління : конспект лекцій. Харків : НТУ «ХПІ», 2019. 145 с.

13. Черногор Н. А., Рубан І. В. Порівняльний аналіз архітектур нейронних мереж для задач прогнозування. *Сучасні інформаційні системи*. 2018. Т. 2, № 3. С. 58–63. URL: (http://www.kpi.kharkov.ua/archive/MicroCAD/2018/S15/) (дата звернення: 07.01.2026).

14. Шевченко В. Л. Технології Blockchain та Bitcoin: основи та практичне застосування : навч. посіб. Київ : КПІ ім. Ігоря Сікорського, 2018. 190 с.

15. Яремчук С. В. Моделювання волатильності криптовалют за допомогою глибокого навчання. *Економіка та управління*. 2023. № 4. С. 112–120.

16. Aggarwal C. C. Neural Networks and Deep Learning: A Textbook. Cham : Springer, 2018. 497 p. URL: (https://link.springer.com/book/10.1007/978-3-319-94463-0) (дата звернення: 19.12.2025).

17. Bao W., Yue J., Rao Y. A deep learning framework for financial time series using stacked autoencoders and LSTM. *PLoS ONE*. 2017. Vol. 12, No. 7. URL: (https://doi.org/10.1371/journal.pone.0180944) (дата звернення: 18.10.2025).

18. Bergstra J., Bengio Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*. 2012. Vol. 13. P. 281–305. URL:

(https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf) (дата звернення: 28.03.2026).

19. Brownlee J. Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs. Melbourne : Machine Learning Mastery, 2018. 230 p. URL: (https://machinelearningmastery.com/deep-learning-for-time-series-forecasting/) (дата звернення: 13.05.2026).

20. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, 2016. P. 785–794. URL: (https://doi.org/10.1145/2939672.2939785) (дата звернення: 25.02.2026).

21. Cho K. 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://aclanthology.org/D14-1179/) (дата звернення: 17.03.2026).

22. Chollet F. Deep Learning with Python. Shelter Island : Manning Publications, 2021. 504 p. URL: (https://www.manning.com/books/deep-learning-with-python-second-edition) (дата звернення: 25.02.2026).

23. Fischer T., Krauss C. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*. 2018. Vol. 270, Iss. 2. P. 654–669. URL: (https://doi.org/10.1016/j.ejor.2017.11.054) (дата звернення: 17.03.2026).

24. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. 800 p. URL: (https://www.deeplearningbook.org/) (дата звернення: 13.05.2026).

25. Hochreiter S., Schmidhuber J. Long Short-Term Memory. *Neural Computation*. 1997. Vol. 9, Iss. 8. P. 1735–1780. URL: (https://doi.org/10.1162/neco.1997.9.8.1735) (дата звернення: 12.04.2026).

26. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*. San Diego, 2015. URL: (https://arxiv.org/abs/1412.6980) (дата звернення: 12.04.2026).

27. Lahmiri S., Bekiros S. Cryptocurrency forecasting with deep learning chaotic neural networks. *Chaos, Solitons & Fractals*. 2019. Vol. 118. P. 35–40. URL: (https://doi.org/10.1016/j.chaos.2018.11.014) (дата звернення: 17.03.2026).

28. McNally S., Roche J., Caton S. Predicting the Price of Bitcoin Using Machine Learning. *26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*. Cambridge, 2018. P. 339–343. URL: (https://doi.org/10.1109/PDP2018.2018.00060) (дата звернення: 12.04.2026).

29. Nelson D. M., Pereira A. C., de Oliveira R. A. Stock market's price movement prediction with LSTM neural networks. *International Joint Conference on Neural Networks (IJCNN)*. Anchorage, 2017. P. 1419–1426. URL: (https://doi.org/10.1109/IJCNN.2017.7966019) (дата звернення: 14.02.2026).

30. Patterson J., Gibson A. Deep Learning: A Practitioner's Approach. Sebastopol : O'Reilly Media, 2017. 532 p. URL: (https://www.oreilly.com/library/view/deep-learning-a/9781491924570/) (дата звернення: 20.01.2026).

31. Siami-Namini S., Tavakoli N., Namin A. S. A Comparison of ARIMA and LSTM in Forecasting Time Series. *17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. Orlando, 2018. P. 1394–1401. URL: (https://doi.org/10.1109/ICMLA.2018.00227) (дата звернення: 08.10.2025).

32. Smuts N. Data Mining for Cryptocurrency Valuation. *International Conference on Data Mining (ICDM)*. Singapore, 2019. P. 120–128. URL: (https://arxiv.org/abs/1905.09999) (дата звернення: 17.03.2026).

33. Tandon S. et al. Bitcoin Price Prediction using LSTM and GRU. *International Journal of Engineering Research & Technology (IJERT)*. 2021. Vol. 10, Iss. 6. P. 250–255. URL: (https://www.ijert.org/bitcoin-price-prediction-using-lstm-and-gru) (дата звернення: 19.03.2026).

34. Yamak P. T., Yifan L., Gadosey P. K. A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting. *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*. Sanya, 2019. P. 49–55. URL: (https://doi.org/10.1145/3377713.3377722) (дата звернення: 21.02.2026).

35. Zhang G. P. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*. 2003. Vol. 50. P. 159–175. URL: (https://doi.org/10.1016/S0925-2312(01)00702-0) (дата звернення: 05.12.2025).

ДОДАТОК А. Лістинг вихідного коду модуля нейромережевого прогнозування

Лістинг А.1

```
# -*- coding: utf-8 -*-
import numpy as np
import pandas as pd
import torch
import torch.nn as nn
from torch.utils.data import TensorDataset, DataLoader
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error

# === 1. ПІДГОТОВКА ДАНИХ (ВІКОННИЙ МЕТОД) ===
def create_sequences(X_scaled, y_scaled, window, horizon=1):
    """Формує послідовності для навчання часових рядів."""
    xs, ys = [], []
    for end in range(window - 1, len(X_scaled) - horizon):
        start = end - (window - 1)
        xs.append(X_scaled[start:end + 1, :])
        ys.append(y_scaled[end + horizon, 0])
    return np.asarray(xs, dtype=np.float32), np.asarray(ys,
dtype=np.float32).reshape(-1, 1)

# === 2. АРХІТЕКТУРИ НЕЙРОННИХ МЕРЕЖ ===
class MLPRegressor(nn.Module):
    """Багатошаровий Перцептрон (MLP)."""
    def __init__(self, in_features):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(in_features, 128),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(64, 1),
        )
    def forward(self, x):
        return self.net(x)

class LSTMRegressor(nn.Module):
    """Рекурентна нейронна мережа (LSTM)."""
    def __init__(self, in_features, hidden=64):
        super().__init__()
        self.lstm = nn.LSTM(input_size=in_features, hidden_size=hidden,
num_layers=1, batch_first=True)
```

Продовження лістингу А.1

```
        self.head = nn.Sequential(
            nn.Dropout(0.2),
            nn.Linear(hidden, 64),
            nn.ReLU(),
            nn.Linear(64, 1),
        )
    def forward(self, x):
        out, _ = self.lstm(x)
        last = out[:, -1, :] # Беремо вихід з останнього часового кроку
        return self.head(last)

# === 3. ЦИКЛ НАВЧАННЯ МОДЕЛІ ===
def train_model(model, train_loader, criterion, optimizer, epochs=50,
device='cpu'):
    """Базовий цикл навчання PyTorch."""
    model = model.to(device)
    for ep in range(epochs):
        model.train()
        train_loss = 0.0
        for xb, yb in train_loader:
            xb, yb = xb.to(device), yb.to(device)
            optimizer.zero_grad()
            pred = model(xb)
            loss = criterion(pred, yb)
            loss.backward()
            optimizer.step()
            train_loss += loss.item() * len(xb)
    return model

# === 4. ОСНОВНИЙ БЛОК ВИКОНАННЯ (ПРИКЛАД ДЛЯ ОДНОГО ТАЙМФРЕЙМУ) ===
if __name__ == "__main__":
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    window = 64

    # Завантаження та нормалізація (імітація для прикладу)
    # df = pd.read_csv("btc_data.csv")
    # X_all = df[["open", "high", "low", "close", "volume"]].values
    # y_all = df[["close"]].values
    X_scaled = StandardScaler().fit_transform(np.random.rand(1000,
5)).astype(np.float32)
    y_scaled = StandardScaler().fit_transform(np.random.rand(1000,
1)).astype(np.float32)

    # Створення послідовностей
    X_seq, y_seq = create_sequences(X_scaled, y_scaled, window=window)

    # Розбиття на train/test (80/20)
```

Закінчення лістингу А.1

```
    split_idx = int(len(X_seq) * 0.8)
    X_train, X_test = X_seq[:split_idx], X_seq[split_idx:]
    y_train, y_test = y_seq[:split_idx], y_seq[split_idx:]

    # Підготовка даних для LSTM: (batch, window, features)
    dl_train_lstm = DataLoader(TensorDataset(torch.tensor(X_train),
torch.tensor(y_train)), batch_size=512, shuffle=True)
    dl_test_lstm  = DataLoader(TensorDataset(torch.tensor(X_test),
torch.tensor(y_test)), batch_size=512)

    # Навчання LSTM
    lstm_model = LSTMRegressor(in_features=5).to(device)
    optimizer = torch.optim.Adam(lstm_model.parameters(), lr=1e-3)
    train_model(lstm_model, dl_train_lstm, nn.MSELoss(), optimizer,
epochs=50, device=device)

    # Оцінка моделі
    lstm_model.eval()
    with torch.no_grad():
        preds = lstm_model(torch.tensor(X_test).to(device)).cpu().numpy()

    print(f"MAE: {mean_absolute_error(y_test, preds):.4f}")
    print(f"RMSE: {np.sqrt(mean_squared_error(y_test, preds)):.4f}")
```