МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

Гончарук Крістіна Сергіївна

Програмний модуль класифікації станів фінансового ринку з використанням методів штучного інтелекту / Software Module for Classification of Financial Market States Using Artificial Intelligence

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Штучний інтелект

Кваліфікаційна робота

Виконала студентка групи КНШІ-41
К. С. Гончарук

Науковий керівник:
к.т.н., доц. В.С. Коваль

Кваліфікаційну роботу допущено до захисту
«___»_____ 2026 р.

Завідувач кафедри
_____ Н.В. Дзюбановська

Тернопіль – 2026

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ГОНЧАРУК Крістіні Сергіївні
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Програмний модуль класифікації станів фінансового ринку з використанням методів штучного інтелекту / Software Module for Classification of Financial Market States Using Artificial Intelligence

керівник роботи к.т.н., доцент, В.С. Коваль
затверджений наказом по університету від 01 грудня 2025 року № 894.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті та монографії з аналізу фінансових ринків і машинного навчання, технічна документація до бібліотек Python (Pandas, NumPy, Scikit-learn, SQLAlchemy), документація API криптовалютних бірж Binance та Bybit, документація PostgreSQL.
4. Основні питання, які потрібно розробити:
- проаналізувати особливості функціонування фінансових та криптовалютних ринків, сучасні підходи до класифікації ринкових станів і застосування методів штучного інтелекту у фінансовій аналітиці;
- спроєктувати архітектуру програмного модуля, структуру бази даних та підсистему збору й зберігання фінансових даних із зовнішніх API;
- розробити алгоритми попередньої обробки даних і формування простору ознак для автоматизованого виявлення ринкових структур, зон ліквідності та поведінкових патернів фінансового ринку;
- реалізувати та навчити модель машинного навчання для класифікації станів фінансового ринку, провести тестування програмного модуля та оцінити ефективність отриманих результатів.
5. Перелік графічного матеріалу у роботі:
- структурна схема розроблюваного програмного модуля;
- UML-діаграма прецедентів програмного модуля;
- схема загального алгоритму роботи програмного модуля;
- діаграма потоків даних системи класифікації;
- логічна структура бази даних.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Огляд літературних джерел відповідно до предметної області та складання плану роботи	до 01.01.2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02.2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 15.03.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05.2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06.2026 р.	

Студент _____ К.С. Гончарук
підпис

Керівник роботи _____к.т.н., доцент В.С. Коваль
підпис

АНОТАЦІЯ

Кваліфікаційна робота на тему «Програмний модуль класифікації станів фінансового ринку з використанням методів штучного інтелекту» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Штучний інтелект» написана обсягом у 71 сторінок і містить 15 ілюстрацій, 4 таблиці, 4 додатки та 35 використаних джерел.

Метою кваліфікаційної роботи є розробка та програмна реалізація програмного модуля для автоматизованої класифікації станів фінансового ринку на основі даних криптовалютних бірж із використанням методів штучного інтелекту та формування рекомендацій щодо прийняття торгових рішень.

Методами дослідження є методи системного аналізу, машинного навчання, математичної статистики, аналізу часових рядів, технології обробки фінансових даних та методи об'єктно-орієнтованого проєктування програмного забезпечення.

Внаслідок виконання роботи розроблено архітектуру програмного модуля для збору, обробки та аналізу фінансових даних із криптовалютних бірж. Реалізовано підсистему автоматизованого отримання ринкових даних через API, модуль формування простору ознак на основі показників мікроструктури ринку та алгоритмів Smart Money Concepts, а також підсистему класифікації станів ринку з використанням алгоритму градієнтного бустингу XGBoost. Для зберігання та обробки даних використано систему керування базами даних PostgreSQL. Проведено навчання, тестування та оцінювання моделі машинного навчання за допомогою сучасних метрик класифікації.

Розроблений програмний продукт є готовою до використання інформаційною системою для аналізу фінансових ринків, яка може застосовуватись трейдерами та фінансовими аналітиками для автоматизованого виявлення ринкових тенденцій, фаз накопичення, маніпуляцій та змін структури ринку з метою підтримки прийняття рішень.

Ключові слова: ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, ФІНАНСОВИЙ РИНОК, КРИПТОВАЛЮТА.

ANNOTATION

Qualification work on the topic «Software Module for Classification of Financial Market States Using Artificial Intelligence» for obtaining a bachelor's degree in specialty 122 «Computer Science» of the educational program «Artificial Intelligence» is written on 71 pages and contains 15 illustrations, 4 tables, 4 appendices, and 35 references.

The aim of the qualification work is to develop and implement a software module for automated classification of financial market states based on cryptocurrency exchange data using artificial intelligence methods and to generate recommendations for supporting trading decision-making.

Research methods include system analysis, machine learning methods, mathematical statistics, time series analysis, financial data processing technologies, and object-oriented software design.

As a result of the research, the architecture of a software module for collecting, processing, and analyzing financial data from cryptocurrency exchanges was developed. A subsystem for automated acquisition of market data via exchange APIs, a feature engineering module based on market microstructure indicators and Smart Money Concepts algorithms, as well as a market state classification subsystem using the XGBoost gradient boosting algorithm were implemented. PostgreSQL was used as the database management system for data storage and processing. The machine learning model was trained, tested, and evaluated using modern classification performance metrics.

The developed software product is a ready-to-use information system for financial market analysis that can be used by traders and financial analysts to automatically identify market trends, liquidity manipulations, and structural changes, improving the effectiveness and objectivity of trading decisions.

Keywords: ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, FINANCIAL MARKET, CRYPTOCURRENCY.

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку фінансових технологій криптовалютний ринок став надзвичайно динамічним та інформаційно насиченим середовищем. Щосекунди на торгових майданчиках генеруються величезні обсяги даних, які відображають не лише зміну вартості активів, а й поведінку великого інституційного капіталу. Традиційні підходи до аналізу, зокрема класичний візуальний технічний аналіз, часто виявляються неефективними через високий рівень інформаційного шуму та вразливість до людських емоцій (апофенії, страху втраченої вигоди). Основною проблемою для сучасного трейдера є складність об'єктивної та завчасної ідентифікації поточного стану ринкової структури. Відсутність надійних автоматизованих інструментів, які б поєднували аналіз ліквідності із сучасними алгоритмами штучного інтелекту, ускладнює своєчасне прийняття торгових рішень та підвищує ризики втрати капіталу.

Дана робота є актуальною у зв’язку з гострою необхідністю створення систем підтримки прийняття рішень, які здатні формалізувати мікроструктуру ціни та автоматично розпізнавати стадії маніпуляцій чи накопичення на висококонкурентному ринку криптовалют, мінімізуючи вплив емоційного фактору.

Мета роботи полягає у розробленні програмного модуля для автоматизованої класифікації станів фінансового ринку (зростаючий тренд, накопичення, злам структури, спадаючий тренд, маніпуляція) на основі даних криптовалютних бірж з використанням методів штучного інтелекту та передбачення доцільної стратегії поведінки трейдера.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Провести системний аналіз предметної області, специфіки функціонування криптовалютних бірж та методів визначення структурних елементів ринку.

2. Розробити архітектуру та підсистему автоматизованого збору історичних і поточних ринкових даних через API-конектори (наприклад, Binance, ByBIT або WhiteBIT).

3. Створити конвеєр формування простору ознак (Feature Engineering) для алгоритмічної ідентифікації пулів ліквідності, зон імбалансу та зламів структури.

4. Програмно реалізувати, навчити та протестувати алгоритми машинного навчання для високоточної класифікації поточного стану ринку.

5. Розробити програмну реалізацію фінансової стратегії, яка на основі класифікованого стану ринку генеруватиме відомості та рекомендації для прийняття рішень кінцевим користувачем.

Об’єктом дослідження є процес інтелектуального аналізу та класифікації структурних станів криптовалютного ринку.

Предметом дослідження є моделі та алгоритми машинного навчання, а також архітектурні рішення для створення систем підтримки прийняття торгових рішень.

Методи дослідження:

– системний аналіз та об'єктно-орієнтоване проєктування — для розробки архітектури програмного модуля;

– методи аналізу структури ринку — для математичного визначення та алгоритмічного маркування цільових класів станів ринку;

– методи машинного навчання, математичної статистики та обробки часових рядів — для навчання класифікаційних моделей (ансамблів дерев рішень) та оцінки їхньої ефективності.

Практичне значення одержаних результатів. Розроблений програмний продукт є готовим інструментом, оптимізованим для обробки реальних високочастотних даних криптовалютних бірж. Він може використовуватися трейдерами та фінансовими аналітиками для об'єктивної оцінки поточної фази доставки ціни, завчасного виявлення цінових маніпуляцій та формування ефективних торгових стратегій на базі сигналів штучного інтелекту

Основні результати кваліфікаційної роботи апробовано на IV Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих вчених «Інтелектуальні комп’ютерні системи та мережі» (м. Тернопіль, 20 травня 2026 р.), де вони були представлені та опубліковані у вигляді тез «Програмний модуль класифікації станів фінансового ринку з використанням методів штучного інтелекту» (Додаток А).

1 АНАЛІЗ СИСТЕМ З ІНТЕГРАЦІЄЮ ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Опис предметної області

Сучасні фінансові ринки є складними стохастичними системами, динаміка яких формується під впливом великої кількості нелінійних факторів. Особливий інтерес становить криптовалютний ринок, який функціонує цілодобово та характеризується високою волатильністю. За таких умов традиційні методи аналізу часових рядів часто втрачають прогностичну здатність, що обумовлює необхідність використання інтелектуальних систем підтримки прийняття рішень, здатних виявляти приховані закономірності та структурні стани ринку.

Сучасні дослідження [2, 3] показують, що методи машинного навчання, зокрема алгоритм XGBoost, забезпечують вищу ефективність прогнозування порівняно з класичними статистичними підходами. При цьому важливою умовою побудови таких систем є використання коректного хронологічного поділу даних, що дозволяє уникнути ефекту «зазирання в майбутнє» (look-ahead bias) та забезпечити достовірність результатів у реальних умовах торгівлі.

Аналіз таких масивів даних виступає нетривіальним завданням у контексті розроблення систем підтримки прийняття рішень. Історично поширеним підходом є застосування класичного технічного аналізу, що спирається на ретроспективні індикатори (ковзні середні, осцилятори). Проте, враховуючи специфіку сучасного криптовалютного ринку, де ціноутворення суттєво залежить від поведінки інституційного та великого алгоритмічного капіталу (маркетмейкерів або концепції «розумних грошей» — Smart Money), такі інструменти демонструють низьку прогностичну здатність [4, 5]. Великі учасники ринку часто ініціюють рух ціни з метою активації зон ліквідності дрібних учасників, що призводить до формування помилкових графічних патернів [6].

Відповідно, ключовим напрямом сучасної фінансової аналітики є перехід від пошуку стандартизованих візуальних патернів до формалізованого аналізу станів та структур ринку. У рамках цього підходу рух ціни розглядається як закономірний процес доставки активу між зонами накопиченої ліквідності. Для побудови

адекватних прогностичних моделей необхідно виокремити та математично описати базові стани криптовалютного ринку:

– зростаючий тренд (Uptrend): фаза ринку, за якої спостерігається послідовне формування вищих локальних максимумів (Higher High, HH) та вищих локальних мінімумів (Higher Low, HL), що свідчить про домінування покупців та зумовлює преференцію довгих (Long) позицій;

– спадаючий тренд (Downtrend): структурний стан, що описується послідовним оновленням нижчих мінімумів (Lower Low, LL) та нижчих максимумів (Lower High, LH), вказуючи на контроль ринку продавцями та обґрунтовуючи відкриття коротких (Short) позицій;

– накопичення / флет (Accumulation / Range): стан тимчасової рівноваги, під час якого ціна коливається у вузькому горизонтальному коридорі без вираженого тренду. Цей етап характеризується акумуляцією або дистрибуцією обсягів великим капіталом;

– злам структури (Break of Structure, BOS / Change of Character, CHoCH): критичний елемент цінового графіка, що фіксує зміну поточного ринкового контексту. BOS підтверджує валідність існуючого тренду, тоді як CHoCH є первинним індикатором потенційної зміни напрямку руху ціни;

– маніпуляція (Manipulation / Liquidity Sweep): аномальний ціновий імпульс, організований для поглинання скупчень стоп-ордерів (ліквідності) перед ініціацією основного трендового руху. Ідентифікація таких маніпуляцій є критичною для управління ризиками.

З математичної точки зору процес зміни фаз ринку доцільно моделювати як дискретний стохастичний процес із застосуванням апарату ланцюгів Маркова. Нехай еволюція ринкової структури описується випадковою величиною X_t у дискретний момент часу t, яка набуває значень із множини можливих станів:

$$X_t \in S = \{s_0, s_1, s_2, s_3, s_4\}, \quad (1.1)$$

де s_0 - накопичення;

s_1 - висхідний тренд;

s_2 - низхідний тренд;

s_3 - маніпуляція;

s_4 - злам структури.

Припускаючи виконання марковської властивості (відсутність післядії), ймовірність переходу системи у майбутній стан s_j залежить виключно від поточного стану s_i та не залежить від попередньої історії процесу:

$$P(X_{t+1} = s_j | X_t = s_i, X_{t-1} = s_{i-1}, \dots, X_0 = s_0) = P(X_{t+1} = s_j | X_t = s_i) = p_{ij}. \quad (1.2)$$

де X_t - випадкова величина, що описує стан ринкової структури у дискретний момент часу t;

s_i - поточний стан системи;

s_j - майбутній стан системи;

p_{ij} - ймовірність переходу системи зі стану s_i у стан s_j.

Поведінка такої системи повністю визначається початковим розподілом ймовірностей та матрицею ймовірностей переходів P розмірністю 5×5:

$$P = \begin{pmatrix} p_{00} & p_{01} & p_{02} & p_{03} & p_{04} \\ p_{10} & p_{11} & p_{12} & p_{13} & p_{14} \\ p_{20} & p_{21} & p_{22} & p_{23} & p_{24} \\ p_{30} & p_{31} & p_{32} & p_{33} & p_{34} \\ p_{40} & p_{41} & p_{42} & p_{43} & p_{44} \end{pmatrix}. \quad (1.3)$$

де P — матриця ймовірностей переходів;

p_{ij} - елементи матриці, що визначають ймовірності переходів між станами.

Для кожного рядка матриці виконується умова нормування:

$$\sum_{j=0}^{4} p_{ij} = 1, i \in \{0,1,2,3,4\}. \quad (1.4)$$

Наприклад, перехід p_{03} описує ймовірність того, що ринок зі стану тривалого накопичення перейде у фазу маніпуляції (Liquidity Sweep) для збору стоп-ордерів.

У свою чергу, перехід p_{34} формалізує ймовірність виникнення імпульсного зламу структури після здійсненої маніпуляції.

Завдання розроблюваної інтелектуальної системи полягає в оцінюванні та апроксимації ймовірностей переходів між фазами ринку на основі векторизованих вхідних ознак за допомогою алгоритмів машинного навчання. Це дозволяє формувати прогноз щодо подальшого розвитку ринкової структури та генерувати обґрунтовані рекомендації для користувача системи підтримки прийняття рішень.

Логіку формування ринкових структур та ключові точки їх зламу (підтвердження тренду та зміна характеру руху) схематично відображено на рисунку 1.1.

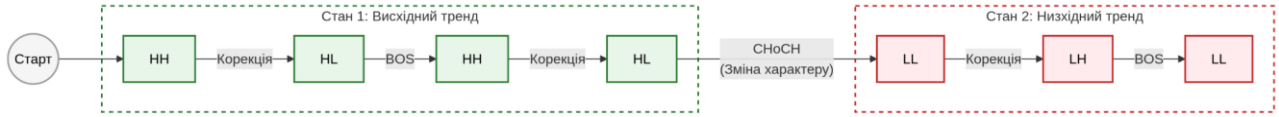

Рисунок 1.1 — Схематичне зображення висхідної та низхідної структури ринку з ідентифікацією точок BOS та CHoCH

Окрім загальних трендових рухів, ціноутворення на криптовалютному ринку підпорядковується алгоритмічним циклам накопичення та розподілу позицій великим капіталом. Послідовна зміна визначених раніше фаз накопичення, маніпуляції та подальшого розподілу формує концепцію AMD (Accumulation, Manipulation, Distribution), яку візуалізовано на рисунку 1.2.

Формалізація визначених структурних станів формує наукове підґрунтя для розроблення алгоритмічних торгових стратегій. Зважаючи на вразливість ручного аналізу до когнітивних упереджень (апофенія, емоційна нестабільність), перспективним напрямом є інтеграція методів штучного інтелекту.

Застосування алгоритмів машинного навчання дозволяє перейти від евристичного візуального аналізу до математичного моделювання [7]. Програмні комплекси здатні здійснювати безперервний збір історичних та поточних даних (через API-інтерфейси криптовалютних бірж) з їх подальшим автоматизованим обробленням. Концептуальну схему інформаційного конвеєра наведено на рисунку 1.3.

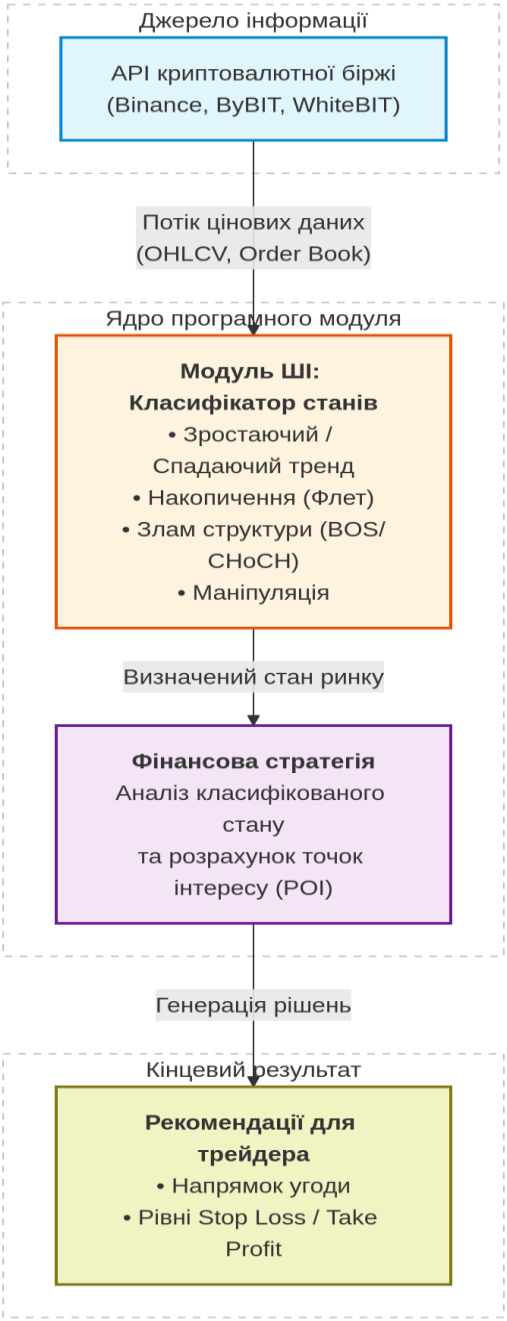

Рисунок 1.2 — Фазові стани фінансового ринку: накопичення, маніпуляція ціною та трендовий розподіл

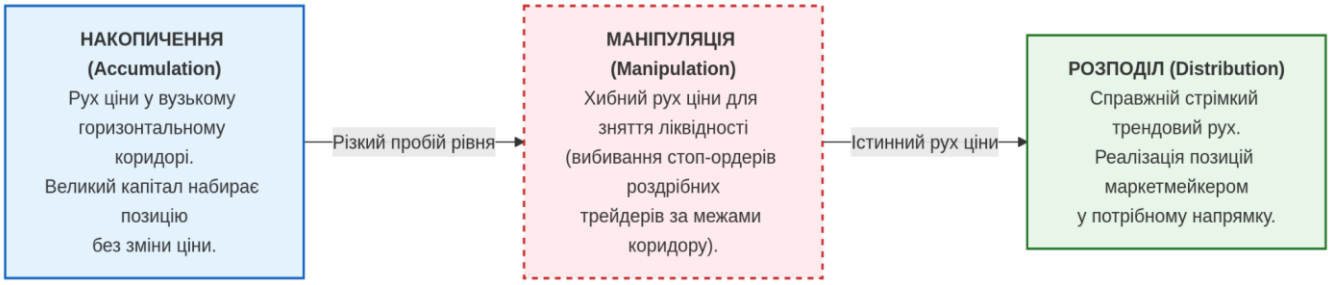

Рисунок 1.3 — Узагальнена архітектура процесу автоматизованої класифікації станів криптовалютного ринку

Як видно з рисунка 1.3, завдяки моделям штучного інтелекту забезпечується можливість класифікації поточного стану ринку (зокрема, розпізнавання фаз маніпуляції та збору ліквідності) в режимі реального часу. Синтез розпізнаних ринкових станів із визначеною фінансовою стратегією дозволяє генерувати об'єктивні рекомендації щодо поведінки трейдера (визначення точок входу, розміщення захисних стоп-заявок та рівнів фіксації прибутку), що мінімізує вплив суб'єктивного фактору та підвищує ефективність управління капіталом.

1.2 Огляд і аналіз існуючих аналогів

Для побудови автоматизованого програмного модуля класифікації станів криптовалютного ринку з використанням методів штучного інтелекту необхідно здійснити чітку алгоритмічну та математичну формалізацію візуальних графічних патернів. Концепції поведінки великого капіталу (Smart Money), які визначають мікроструктуру сучасних електронних торгів, повинні бути перетворені на строгі логічні та статистичні маркери. Саме ці показники формують простір ознак (Feature Space) для обробки алгоритмами машинного навчання [7].

Замість традиційних технічних індикаторів (наприклад, MACD чи RSI), які мають запізнілий характер і розраховуються як похідні від минулої ціни, сучасна алгоритмічна фінансова стратегія спирається на первинні показники цінової дії (Price Action) та ліквідності книги ордерів (Order Book). До основних показників поведінки фінансового ринку, необхідних для класифікації його станів та моделювання рішень трейдера, належать наступні групи алгоритмічних маркерів:

- показники структурної динаміки (цінові екстремуми);
- маркери порушення та підтвердження ринкової структури;
- показники ліквідності та маніпуляцій;
- показники зон інтересу (POI) та цінової неефективності.

Загальну класифікацію цих маркерів, які формують вхідний простір ознак для системи штучного інтелекту, наведено на рисунку 1.4.

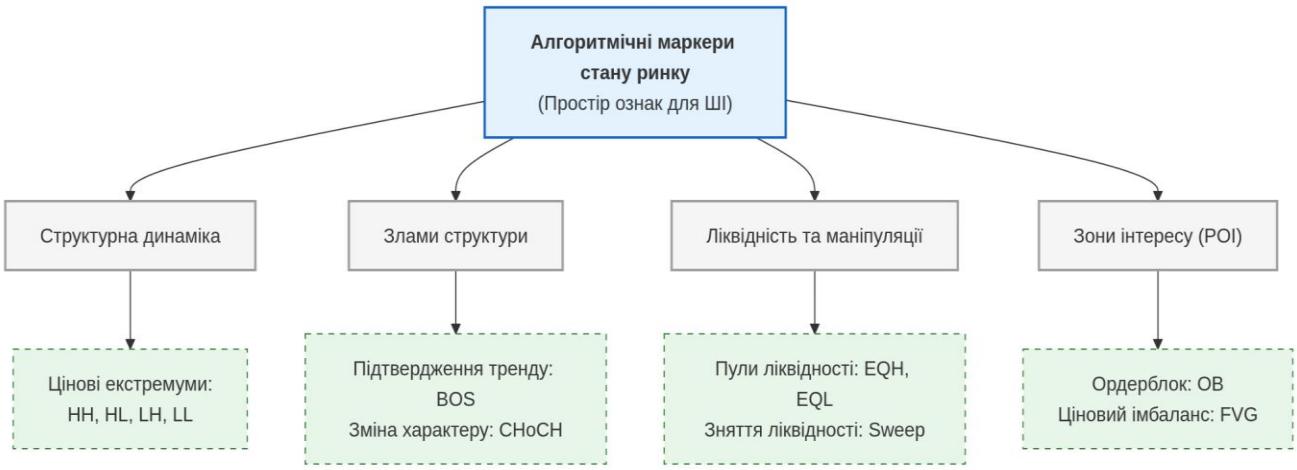

Рисунок 1.4 – Класифікація алгоритмічних маркерів криптовалютного ринку для побудови фінансової стратегії

Ціновий рух розглядається як дискретний часовий ряд. Алгоритм визначає локальні максимуми та мінімуми на основі порівняння сусідніх свічок (барів). Зокрема, формування послідовності вищих максимумів (Higher High, HH) та вищих мінімумів (Higher Low, HL) алгоритмічно ідентифікує фазу зростаючого тренду. Натомість нижчі максимуми (Lower High, LH) та нижчі мінімуми (Lower Low, LL) виступають індикатором спадаючого тренду.

Динаміка тренду підтверджується або спростовується через специфічні цінові пробої:

– підтвердження структури (Break of Structure, BOS): пробій та закріплення ціни за межами попереднього екстремуму. Математично, для висхідного тренду BOS фіксується, якщо ціна закриття поточної свічки C_t перевищує значення попереднього максимуму HH_t-1 Цей показник є сигналом продовження тренду;

– зміна характеру (Change of Character, CHoCH): імпульсний пробій останнього структурного мінімуму (HL) при висхідному тренді або максимуму (LH) при низхідному. CHoCH слугує первинним тригером для системи про порушення балансу попиту та пропозиції і ймовірний початок розвороту або перехід у стан накопичення.

Сучасні дослідження мікроструктури ринків доводять, що великі учасники використовують скупчення захисних стоп-ордерів дрібних трейдерів для задоволення власної потреби в ліквідності без суттєвого зміщення ціни

(проковзування) [8]. У цьому контексті виділяють:

– пули ліквідності (Liquidity Pools): зони формування рівних максимумів (Equal Highs, EQH) або рівних мінімумів (Equal Lows, EQL);

– зняття ліквідності (Liquidity Sweep): ключовий показник стану маніпуляції. Відбувається так зване «полювання за стопами» (Stop-Hunting) [13]. Алгоритмічно маніпуляція над пулом максимумів ідентифікується, коли максимальна ціна поточної свічки H_t перевищує рівень пулу ліквідності, проте ціна закриття C_t залишається нижчою за цей рівень (формується довга тінь свічки). Це свідчить про набір позиції маркетмейкером.

Після ідентифікації маніпуляції та зламу структури, алгоритм повинен визначити зони для потенційного відкриття позицій:

– ордерблок (Order Block, OB): остання повнотіла свічка протилежного напрямку перед імпульсним рухом, який призвів до зламу структури (BOS або CHoCH). У фінансовій стратегії ця зона розглядається як місце агрегації інституційних ордерів;

– імбаланс (Fair Value Gap, FVG): зона цінової неефективності, утворена імпульсною свічкою, де відсутнє перекриття тіней сусідніх свічок. Ринок має тенденцію повертатися до таких зон для відновлення цінового балансу. Для висхідного руху (Bullish FVG) величина імбалансу визначається як різниця між мінімумом третьої свічки та максимумом першої:

$$FVG_{bullish} = L_3 - H_1 > 0 \quad (1.3)$$

де $\mathrm{FVG_{bullish}}$ - абсолютне значення зони імбалансу для висхідного руху;

L_3 - мінімум (Low) третьої свічки у трисвічковому патерні;

H_1 - максимум (High) першої свічки у трисвічковому патерні.

Для ілюстрації процесу цінової дії на рисунку 1.5 наведено приклад формування ордерблоку (Order Block, OB) та зони імбалансу (Fair Value Gap, FVG) на графіку криптовалютного активу.

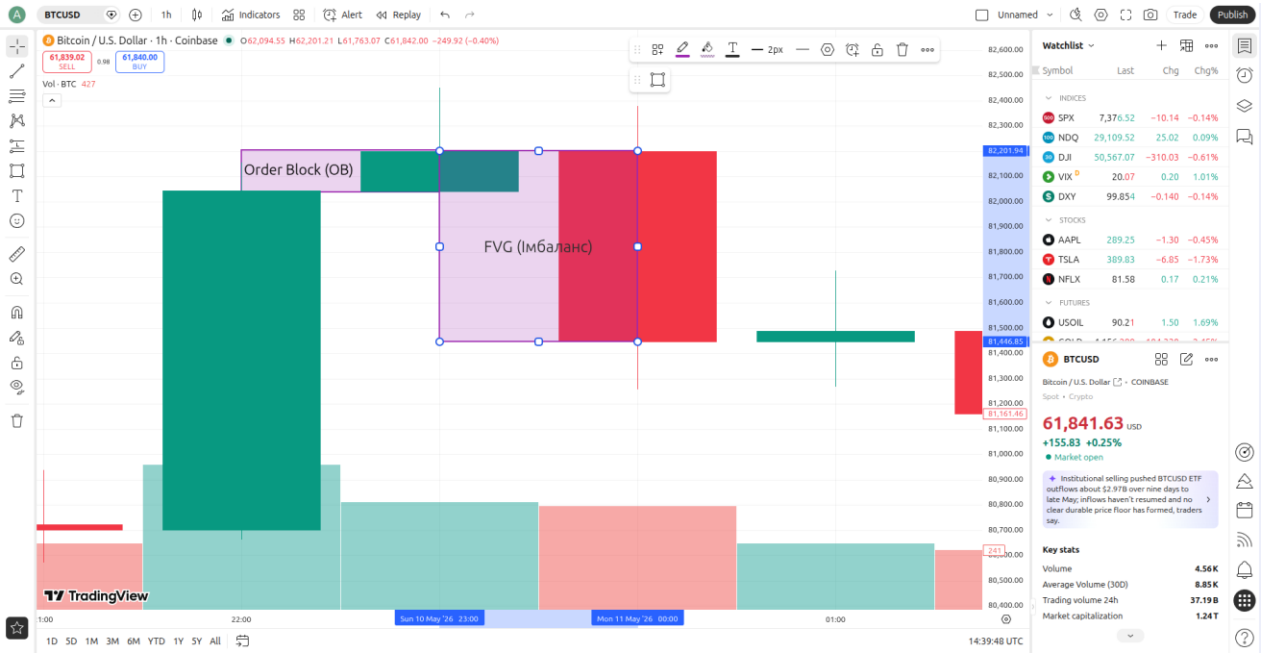

Рисунок 1.5 – Свічкова конфігурація зон інтересу (Point of Interest): формування Ордерблоку (OB) та цінового імбалансу (FVG) на графіку криптовалютного активу

На основі визначених показників формується предиктивна модель поведінки трейдера. Фінансова стратегія передбачає послідовний алгоритм прийняття торгових рішень. Система генерує відомості для користувача за логікою, що наведена на блок-схемі (рисунок 1.6).

Як видно з алгоритму, програмний модуль не просто прогнозує напрямок ціни, а формує комплексну стратегію. Відмова від класичних показників на користь маркерів структури (BOS, CHoCH) та ліквідності дозволяє системі штучного інтелекту синхронізувати торгові рішення з діями інституційних учасників криптобірж. Це забезпечує користувача об'єктивною інформацією щодо точок входу та ризик-менеджменту (розміщення Stop Loss та Take Profit) в умовах високої ринкової невизначеності. Отже, формалізація розглянутих структурних станів ринку та логіки прийняття торгових рішень формує концептуальне підґрунтя для їх подальшої алгоритмізації. Для практичної реалізації цієї стратегії необхідно здійснити вибір оптимальних методів машинного навчання та обґрунтувати архітектуру програмного модуля, що буде детально розглянуто в наступному підрозділі.

Рисунок 1.6 – Алгоритм формування торгового рішення трейдера на основі класифікації показників ринкової структури

1.3 Постановка задачі дослідження

У результаті проведеного аналізу предметної області та існуючих програмних рішень було встановлено, що ручний аналіз графіків та пошук патернів поведінки «розумного капіталу» (Smart Money) стає дедалі менш ефективним через високу волатильність криптовалютного ринку та необхідність одночасної обробки великих масивів даних цінової дії (Price Action). Крім того, складні нелінійні зв'язки між пулами ліквідності, зонами цінової неефективності (FVG) та зміною характеру тренду (CHoCH) потребують алгоритмічного підходу.

З огляду на це, метою роботи є розробка архітектури та алгоритмічного забезпечення системи підтримки прийняття рішень (СППР) для функціонування в умовах стохастичної невизначеності з використанням методів штучного інтелекту. В якості предметної області та експериментального середовища для верифікації розроблених моделей обрано високочастотні часові ряди криптовалютного ринку.

У межах даної роботи під станом системи (ринку) розуміється специфічна фаза циклу, що характеризується певним балансом ліквідності та напрямком цінової дії. Система класифікує п'ять базових цільових станів: Накопичення (флет), Зростаючий тренд, Спадаючий тренд, Маніпуляція (зняття ліквідності) та Злам структури.

У математичному вигляді задачу класифікації можна формалізувати наступним чином. Нехай задано множину об’єктів спостереження (векторів стану мікроструктури ринку на заданому таймфреймі):

$$X = \{x_1, x_2, \ldots, x_n\} \tag{1.4}$$

де X — множина об’єктів спостереження;

x_i - вектор фундаментальних відносних ознак і-го об'єкта, що включає математичні пропорції (відхилення від SMA, відносні розміри тіней свічок, нормалізований ATR, RSI, відносні обсяги торгів);

n — загальна кількість об'єктів спостереження.

Кожному об’єкту з множини X відповідає певний клас стану ринку з множини міток:

$$Y = \{y_0, y_1, y_2, y_3, y_4\} \quad (1.5)$$

де Y — множина цільових класів стану ринку;

y_0 — накопичення (флет);

y_1 — зростаючий тренд;

y_2 — спадаючий тренд;

y_3 — маніпуляція (зняття ліквідності);

y_4 — злам структури (підтвердження або зміна тренду).

Задача полягає у побудові класифікаційної функції за допомогою алгоритмів машинного навчання:

$$f: X \to Y \quad (1.6)$$

де f — класифікаційна функція (модель), яка відображає невідомий вектор макроекономічних ознак X на відповідний клас Y.

Процес навчання моделі зводиться до знаходження такої функції $\hat{f}$, яка мінімізує емпіричний ризик (функцію втрат L) на навчальній вибірці:

$$\hat{f} = \underset{f}{\operatorname{argmin}} \sum_{i=1}^{n} L\left(y_i, f(x_i)\right) \quad (1.7)$$

де $\hat{f}$ — оптимальна класифікаційна функція (навчена модель);

argmin(f) — значення аргументу f, при якому функція втрат досягає мінімуму;

$L(y_i, f(x_i))$ — функція втрат, що вимірює розбіжність між фактичним станом ринку y_i та передбаченим моделлю значенням $f(x_i)$;

n — обсяг навчальної вибірки.

Для досягнення поставленої мети та розв'язання задачі необхідно виконати наступні етапи:

1. Збір та агрегація даних: Розробка функціоналу для автоматичного отримання високочастотних даних (OHLCV, Open Interest, Funding Rate, Order Book) через WebSocket [11] або API провідних криптовалютних бірж (наприклад, Binance [10], Bybit).

2. Попередня обробка даних: Очищення датасету від аномалій, обробка пропущених значень та трансформація абсолютних цінових показників у відносні величини для забезпечення стаціонарності часових рядів.

3. Формування ознак (Feature Engineering): Алгоритмічна формалізація графічних патернів (математичне обчислення зон відхилення від тренду, виявлення пулів ліквідності та відносного обсягу).

4. Навчання моделі ШІ: Вибір, налаштування гіперпараметрів та тренування алгоритмів машинного навчання (зокрема, алгоритмів градієнтного бустингу XGBoost) із суворим збереженням хронології для уникнення витоку даних.

5. Оцінка якості та валідація: Перевірка ефективності побудованої моделі на тестовій вибірці з використанням метрик Accuracy, Precision, Recall та F1-score, а також аналіз важливості ознак.

На основі поставленої задачі висуваються наступні вимоги до розроблюваного програмного модуля: Функціональні вимоги:

– модуль повинен автоматично завантажувати історичні дані та свіжі котирування через біржові API;

– система повинна алгоритмічно ідентифікувати маркери SMC (Liquidity Sweeps, обсяги, трендові відхилення) для формування вектора ознак;

– модуль має здійснювати автоматичну класифікацію поточної фази ринку та генерувати обґрунтовану торгову стратегію для користувача-трейдера.

Нефункціональні вимоги:

– продуктивність: здатність обробляти високочастотні біржові дані та генерувати інференс-прогнози без критичних затримок;

– масштабованість: гнучка архітектура, що дозволяє легко додавати нові криптовалютні пари або змінювати таймфрейми;

– надійність: модуль повинен мати механізми відмовостійкості (наприклад, при відсутності файлів моделі система має інформувати користувача).

Розв'язання поставленої задачі дозволить мінімізувати вплив людських емоцій (FOMO/FUD) на прийняття рішень та створити надійний алгоритмічний інструмент трейдера, що працює в синхронізації з інституційним капіталом.

2 ПРОЄКТУВАННЯ СИСТЕМИ ТА АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

2.1 Архітектура програмного модуля та загальна структура системи

Розробка програмного забезпечення для інтелектуального аналізу криптовалютних ринків вимагає створення надійної, гнучкої та масштабованої архітектури. Оскільки розроблювана система повинна обробляти високочастотні потоки фінансових даних у режимі реального часу, використання класичної монолітної архітектури є недостатнім і може призвести до системних затримок (latency). Тому проєктування модуля базується на сервіс-орієнтованому підході з виділенням ізольованих конвеєрів обробки даних (Data Pipelines).

Архітектура розроблюваного програмного модуля складається з трьох ключових мікрокомпонентів, взаємодія між якими формує єдиний конвеєр (Data Science Pipeline) [7, 14]:

1. Модуль API-конекторів (Data Collection Layer): відповідає за забезпечення безперервного потоку фінансової інформації із зовнішніх джерел. Джерелами слугують відкриті API провідних криптовалютних бірж, зокрема Binance [8]. Модуль отримує історичні та поточні котирування у форматі свічок OHLCV (Open, High, Low, Close, Volume), що є стандартом для аналізу ринкової мікроструктури [6].

2. ETL-конвеєр (Extract, Transform, Load): працює перед передачею фінансових часових рядів до алгоритмів машинного навчання. Здійснює програмне очищення даних від аномалій (спайків) та обробку втрачених мережевих пакетів або пропущених значень (NaN) з використанням бібліотеки pandas [18]. На етапі трансформації (Transform) цей модуль на основі «сирих» даних алгоритмічно обчислює показники концепції Smart Money [13]: ідентифікує зони відхилення від ковзних середніх, фіксує пули ліквідності (тіні свічок) та нормалізує показники волатильності.

3. Обчислювальне ядро (Inference Engine & View): центральний аналітичний елемент системи. Він десеріалізує та завантажує натреновану ансамблеву модель градієнтного бустингу XGBoost [19], застосовує її до підготовленого простору ознак та генерує торгові рішення.

Такий інженерний підхід дозволяє у майбутньому легко контейнеризувати додаток (наприклад, за допомогою технологій Docker) та горизонтально масштабувати його для одночасного аналізу десятків криптовалютних пар без блокування основного потоку виконання [16].

Структура програмного модуля наведено на рисунку 2.1.

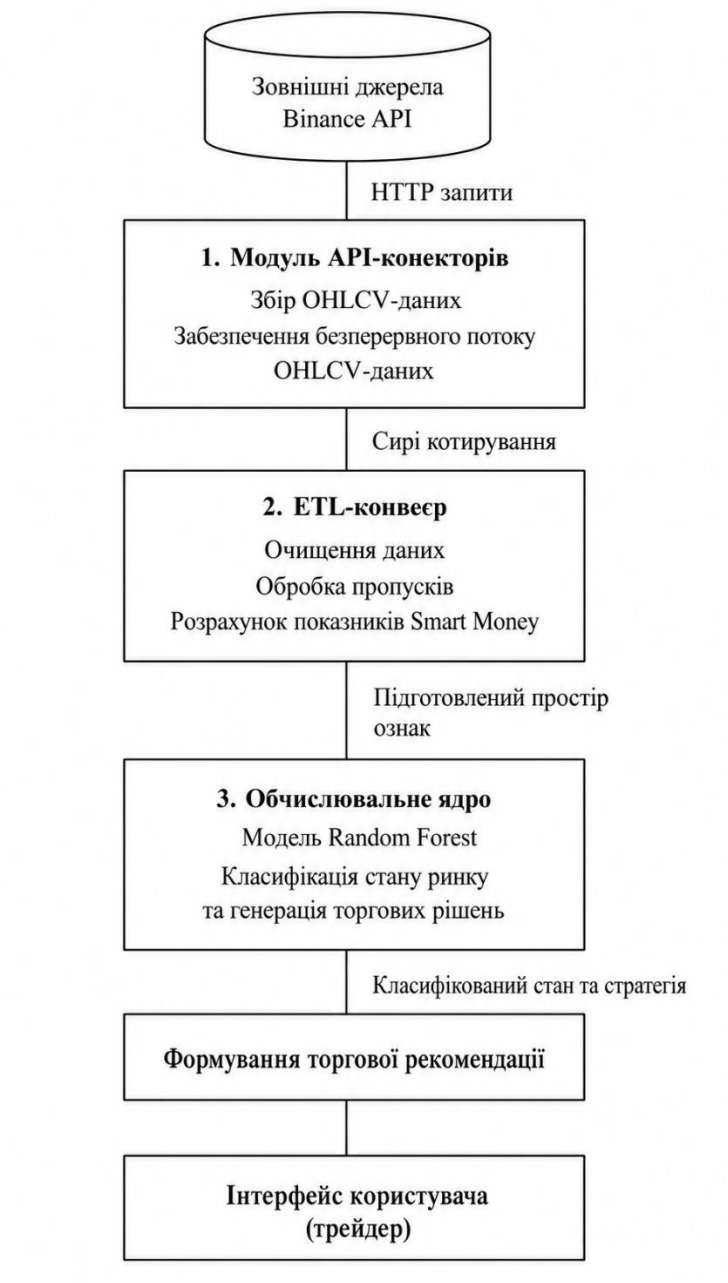

Рисунок 2.1 — Структура розроблюваного програмного модуля

Для детального розуміння логіки роботи та архітектури системи на етапі проєктування було використано уніфіковану мову моделювання (UML) [17]. З

огляду на специфіку системи, яка базується на міжсервісній взаємодії (API-виклики) та конвеєрній обробці даних, найефективнішим способом опису динаміки є побудова діаграми послідовності (Sequence Diagram). Ця діаграма деталізує життєвий цикл генерації торгового сигналу — від запиту користувача до виклику навченої моделі ШІ. Діаграму послідовності наведено на рисунку 2.2.

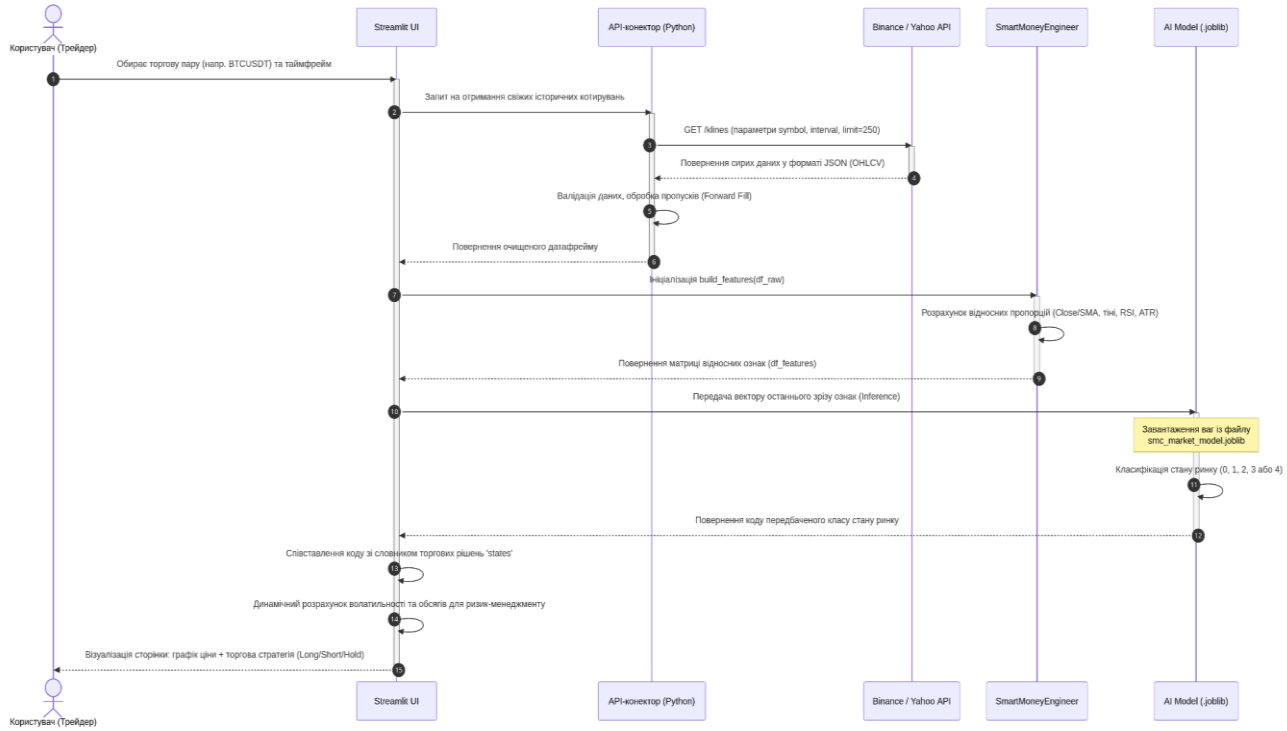

Рисунок 2.2 — UML-діаграма послідовності роботи програмного модуля

Як видно з рисунка 2.2, процес ініціюється користувачем (трейдером) через веб-інтерфейс шляхом вибору торгової пари та таймфрейму. Далі система виконує синхронний HTTP-запит (за допомогою бібліотеки Requests [21]) до зовнішнього сервера біржі для отримання масиву котирувань. Після успішної відповіді сирі дані передаються до об'єкта формування ознак, який обчислює математичні індикатори. Підготовлений датафрейм передається до обчислювального ядра (реалізованого за допомогою бібліотеки Scikit-learn [26]), яке класифікує поточний стан ринку і повертає результат в інтерфейс для генерації фінансової стратегії.

Для візуалізації загальної алгоритмічної логіки системи та послідовності етапів обробки фінансових даних розроблено блок-схему роботи програмного модуля, яку наведено на рисунку 2.3.

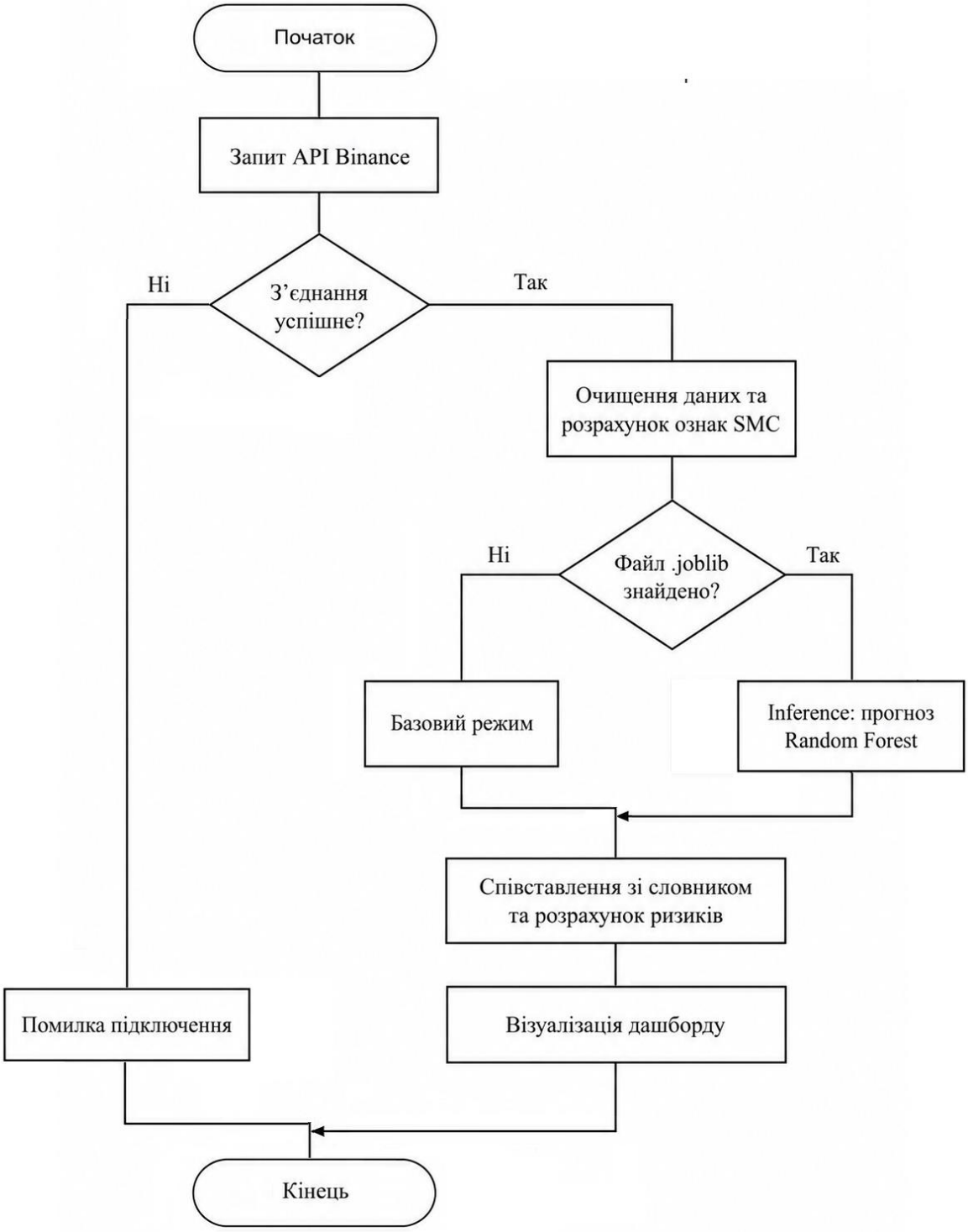

Рисунок 2.3 —Схема загального алгоритму роботи програмного модуля

Алгоритм роботи, зображений на схемі (рисунок 2.3), демонструє строгу послідовність конвеєра обробки даних (Data Pipeline). Запропонована архітектура гарантує чітке розділення бізнес-логіки та інтерфейсу, що є стандартом сучасної інженерії програмного забезпечення [15].

Зокрема, реалізація графічного інтерфейсу була свідомо спроєктована у вигляді повноцінної сторінки (Document Page), відкидаючи застарілі підходи використання модальних вікон (modal iframes). Сторінковий підхід забезпечує значно вищий рівень користувацького досвіду (UX), виключаючи проблеми з подвійним скролінгом, гарантуючи коректну адаптивність на мобільних пристроях та полегшуючи подальше горизонтальне масштабування аналітичних вкладок інтерфейсу. Заміна одного компонента (наприклад, зміна алгоритму класифікації на градієнтний бустинг [19] або перехід на іншу СУБД [20]) не вимагає переписування всього коду, що є необхідною умовою для інтеграції системи у реальне виробниче середовище.

2.2 Інформаційне забезпечення та підготовка простору ознак

Інформаційне забезпечення є фундаментальною складовою системи машинного навчання, оскільки якість, повнота та репрезентативність вхідних даних безпосередньо визначають точність розпізнавання ринкових структур (BOS, CHoCH) та адекватність роботи алгоритмів штучного інтелекту. У контексті алгоритмічного аналізу криптовалютних ринків за методологією Smart Money (SMC), інформаційна модель повинна оперувати високочастотними ціновими показниками, даними книги ордерів та метриками ф'ючерсного ринку.

Основними зовнішніми джерелами даних (External Entities) для системи виступають API провідних криптовалютних бірж:

1. Binance / ByBIT REST API — використовується для завантаження історичних та поточних котирувань торгових пар (наприклад, BTC/USDT). Дані надходять у форматі свічок OHLCV (Open, High, Low, Close, Volume), що дозволяє алгоритмічно ідентифікувати цінові екстремуми та зони імбалансу (FVG) [10].
2. Binance Futures API / Coinglass API — спеціалізовані кінцеві точки для отримання даних деривативного ринку. Постачають фундаментальні метрики для визначення фаз накопичення та розподілу: відкритий інтерес (Open Interest), ставки фінансування (Funding Rates) та обсяги ліквідацій позицій [12].

Для моделювання процесу руху інформації від зовнішніх біржових джерел до математичної моделі ШІ розроблено діаграму потоків даних (Data Flow Diagram, DFD) першого рівня, яку наведено на рисунку 2.4, де «сирі» дані у форматі JSON завантажуються із зовнішніх кінцевих точок REST API. Програмно цей етап комунікації реалізується за допомогою стандартизованих HTTP-бібліотек (зокрема, Python Requests [21]), після чого дані проходять через процес парсингу та агрегації. Потім вони зберігаються у локальному реляційному сховищі даних для подальшої попередньої обробки.

Для надійного зберігання та швидкого доступу до історичних часових рядів розроблено реляційну структуру бази даних. Логічна структура (ER-діаграма) наведена на рисунку 2.5.

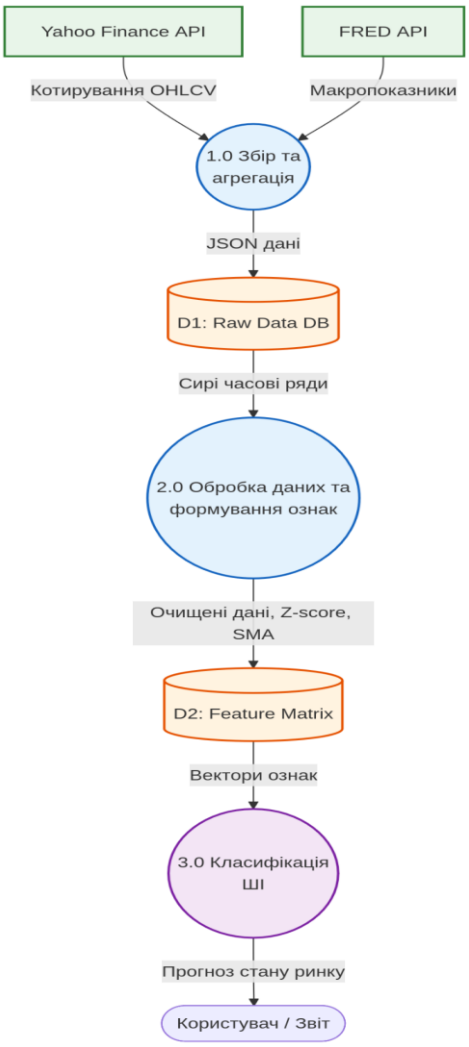

Рисунок 2.4 — Діаграма потоків даних системи класифікації

В якості основної системи керування базами даних (СКБД) для реалізації інформаційного сховища обрано об'єктно-реляційну систему PostgreSQL [35]. Вибір саме цієї СКБД обґрунтований її високою продуктивністю при роботі з великими масивами високочастотних часових рядів (time-series data), підтримкою жорстких вимог транзакційності (концепція ACID), що є фундаментальним стандартом для проектування надійних інформаційних баз даних [20], та можливостями швидкого виконання аналітичних запитів.

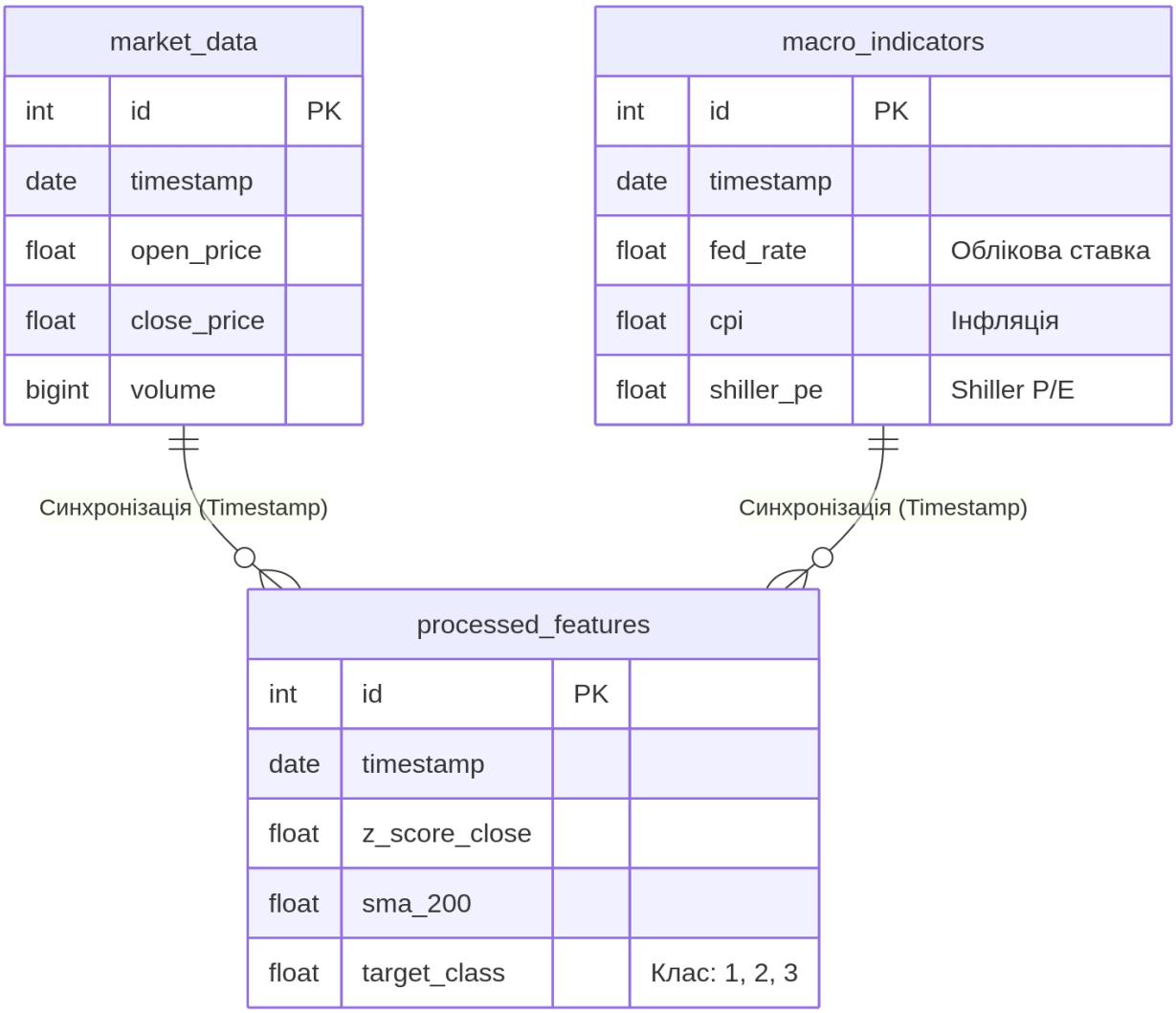

Рисунок 2.5 — Логічна структура бази даних (ER-діаграма)

Інформаційною основою розробленого модуля є дані провідних платформ агрегації криптовалютного ринку, зокрема CoinMarketCap [31], Binance Research [32] та Coinglass [30]. Використання цих джерел забезпечує отримання історичних котирувань, аналітичних показників та метрик деривативного ринку (Open Interest,

Funding Rates), необхідних для аналізу мікроструктури ринку та виявлення прихованих пулів ліквідності.

Фізична модель даних складається з трьох основних сутностей: kline_data (біржові котирування у форматі OHLCV), derivatives_data (метрики ліквідності та деривативного ринку) та processed_features (агрегований простір ознак для алгоритмів машинного навчання). Для забезпечення цілісності даних виконується синхронізація котирувань і деривативних показників за часовими мітками (timestamp) із використанням операцій JOIN. Усунення пропущених значень, що можуть виникати внаслідок мережевих затримок або розривів з'єднання, здійснюється методом прямого заповнення (Forward Fill) [16].

Наступним критично важливим кроком є нормалізація (стандартизація) даних. Оскільки біржові показники мають абсолютно різні шкали вимірювання (наприклад, ціна біткоїна вимірюється десятками тисяч доларів, обсяг торгів — мільйонами, а ставка фінансування становить соті частки відсотка), пряме подання таких даних у модель машинного навчання призведе до домінування ознак із більшими абсолютними значеннями [14]. Для усунення цієї проблеми в системі використовується метод Z-стандартизації (Z-score normalization), який приводить усі числові ознаки до нульового середнього та одиничної дисперсії. Математично цей процес описується формулою:

$$z = \frac{x - \mu}{\sigma} \tag{2.1}$$

де z — стандартизоване значення біржової ознаки, що передається у модель ШІ;

x — поточне фактичне значення показника (наприклад, обсягу торгів або відкритого інтересу);

μ — середнє арифметичне значення цього показника за обраний історичний період (ковзне вікно);

σ — середньоквадратичне відхилення (стандартне відхилення) вибірки.

Після стандартизації система переходить до етапу конструювання нових ознак (Feature Engineering). Для того, щоб алгоритми ШІ могли ідентифікувати

фази маніпуляції (Liquidity Sweep) з боку великого капіталу, системі необхідно фіксувати аномальні сплески торгової активності. Для цього програмний модуль автоматично розраховує просте ковзне середнє обсягів торгів (Simple Moving Average, SMA Volume) за формулою:

$$SMA_k = \frac{1}{k}\sum_{i=1}^{k} p_i \quad (2.2)$$

де SMA_k — значення простого ковзного середнього за період k;

k — ширина часового вікна (наприклад, 20 останніх свічок для визначення базового фону активності);

pi — значення обсягу торгів (Volume) у момент часу i.

Співвідношення поточного обсягу до його ковзного середнього (SMA_k) дозволяє моделі математично підтвердити, чи супроводжувався імпульсний пробій структури (BOS або CHoCH) реальними грошима інституційних учасників, чи це була маніпуляція низькими обсягами.

Таким чином, у результаті роботи модуля інформаційного забезпечення формується багатовимірний простір ознак (матриця даних), що містить очищені, нормалізовані показники мікроструктури та ліквідності. Цей простір є повністю підготовленим для подальшого використання у навчанні класифікаційних алгоритмів штучного інтелекту, які будуть визначати поточний стан ринку (накопичення, маніпуляція або тренд) та генерувати поведінкову стратегію для трейдера. Це детально розглядається у наступному підрозділі.

2.3 Алгоритмічне забезпечення та вибір моделей класифікації

Основною науково-практичною проблемою при використанні методів керованого машинного навчання (Supervised Learning) у фінансовій сфері є відсутність готових розмічених датасетів, які б містили патерни поведінки інституційного капіталу ("Smart Money"). Біржові API надають лише «сирі» числові ряди котирувань. Відповідно, головним власним внеском автора в алгоритмічне забезпечення розроблюваної системи є створення кастомного

програмного алгоритму автоматичної розмітки ринкових станів (Labeling Algorithm) та формування унікального простору відносних ознак (Feature Engineering).

Замість використання стандартних запізнілих технічних індикаторів, було розроблено алгоритм, який перетворює логіку візуальних графічних маніпуляцій на строгі математичні правила (імплементовані у програмному класі SmartMoneyEngineer). Алгоритмічна ідентифікація цільових класів відбувається за такою логікою:

– ідентифікація маніпуляцій (Liquidity Sweep — Клас 3): Алгоритм сканує попередні N свічок для пошуку локальних максимумів (пулів ліквідності). Якщо поточна ціна пробиває цей рівень (поточний максимум Ht перевищує локальний максимум), але ціна закриття Ct повертається у діапазон, і при цьому відношення верхньої тіні до загального розміру свічки є аномально великим — скрипт програмно маркує цей зріз як маніпуляцію (зняття ліквідності маркетмейкером);

– ідентифікація зламу структури (BOS/CHoCH — Клас 4): скрипт розраховує просте ковзне середнє обсягів торгів (SMA Volume). Злам фіксується лише тоді, коли ціна закриття імпульсно долає структурний рівень, а поточний торговий обсяг перевищує середній мінімум на заданий алгоритмічний коефіцієнт. Це програмно відфільтровує хибні пробої в умовах низької ліквідності;

– ідентифікація флету (Накопичення — Клас 0): векторизується відхилення ціни від локального тренду (Close/SMA20). Якщо відхилення лежить у межах статистичної похибки (знаходиться нижче заданого порогу нормалізованої волатильності ATR) протягом визначеного періоду — алгоритм привласнює поточній свічці стан накопичення балансу;

– ідентифікація трендових рухів (Класи 1 та 2): алгоритм фіксує підтверджений висхідний (Клас 1) або низхідний (Клас 2) рух ціни з послідовним закріпленням екстремумів та відсутністю ознак маніпуляції.

Логіку роботи розробленого алгоритму автоматичної розмітки цільових класів візуалізовано у вигляді схеми на рисунку 2.6.

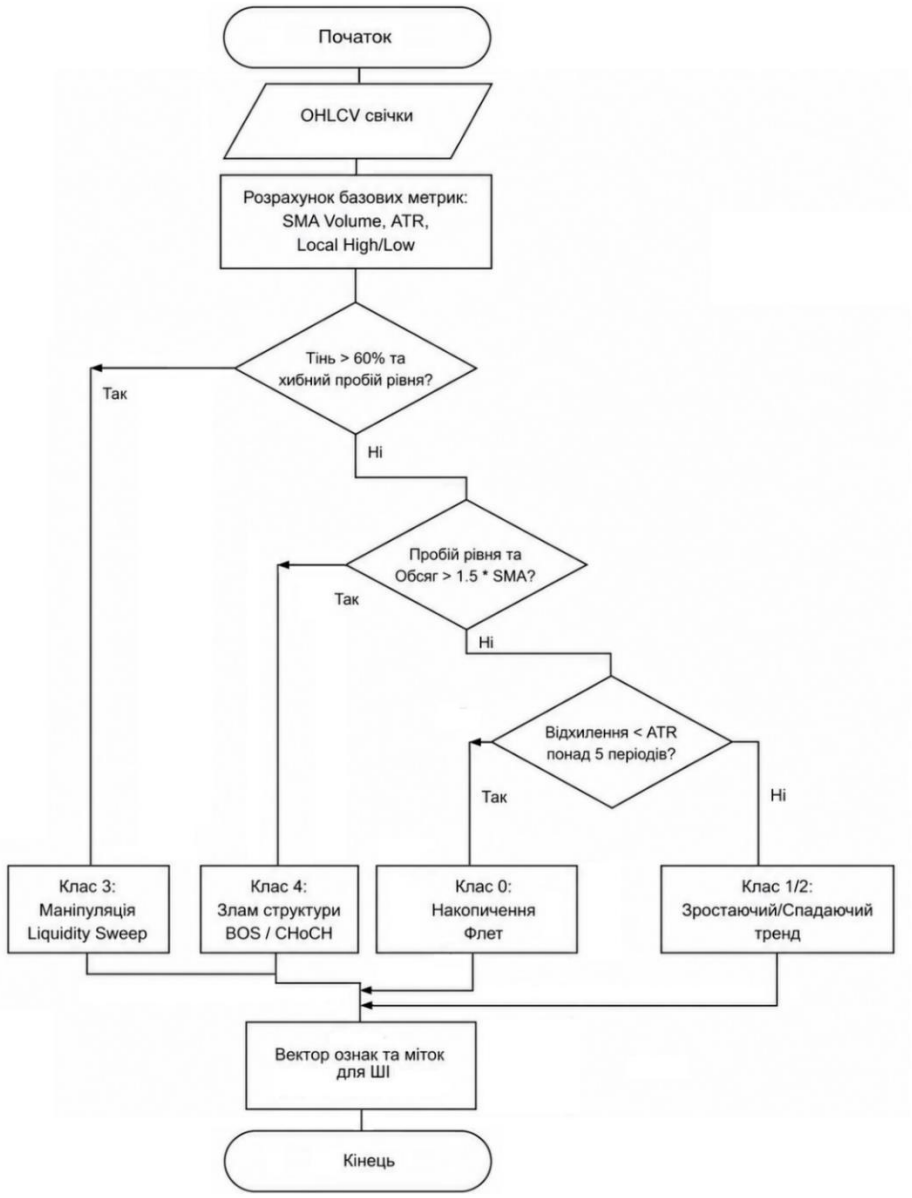

Рисунок 2.6 — Схема авторського алгоритму автоматичної розмітки станів ринку (Labeling Algorithm)

Таким чином, у межах даної роботи було спроєктовано математичний конвеєр, який перетворює абсолютні ціни (Xraw) у стаціонарний нормалізований простір відносних ознак (Xfeatures):

$$X_{raw} = [Open, High, Low, Close, Volume] \quad (2.3)$$

$$X_{features} = \left[\frac{Close}{SMA_{20}}, SMA_{vol\_ratio}, Ratio_{shadow}, ATR_{norm}, RSI\right] \quad (2.4)$$

де X_{raw} — вхідний вектор абсолютних цінових котирувань та обсягів;

$X_{features}$ — вихідний вектор розрахованих відносних мікроструктурних ознак;

Close / SMA_{20}— відхилення поточної ціни закриття від 20-періодного простого ковзного середнього (індикатор локального тренду);

SMA_{vol_ratio} — відношення поточного обсягу торгів до його ковзного середнього (маркер інституційної активності);

$Ratio_{shadow}$ — відносний розмір тіней свічок до загального діапазону свічки (маркер зняття ліквідності);

ATR_{norm} — нормалізований середній істинний діапазон (індикатор фонової волатильності);

RSI — індекс відносної сили (осцилятор імпульсу).

Отриманий алгоритмічно розмічений датасет розділяється на незалежні вибірки із суворим дотриманням хронологічної послідовності за допомогою техніки TimeSeriesSplit. Це робиться для уникнення ефекту «заглядання в майбутнє» (look-ahead bias), що є критичною методологічною помилкою при аналізі фінансових часових рядів [24].

Зважаючи на високий рівень інформаційного шуму та складні нелінійні взаємозв'язки між мікроструктурними ознаками криптовалютного ринку, для побудови обчислювального ядра СППР було обрано концепцію ансамблевого машинного навчання. Замість використання єдиного алгоритму, в рамках роботи проведено порівняльний аналіз трьох передових ансамблевих архітектур на основі дерев рішень:

1. Random Forest (Бейзлайн-модель) — алгоритм беггінгу, який демонструє високу стійкість до викидів (outliers).

2. XGBoost (eXtreme Gradient Boosting) — оптимізована реалізація градієнтного бустингу з підтримкою L1 та L2 регуляризації.

3. LightGBM (Light Gradient Boosting Machine) — фреймворк, що забезпечує екстремально швидку обробку великих масивів історичних даних за рахунок leaf-wise розростання дерев.

Вибір алгоритму XGBoost як базової архітектури підтверджується його високою результативністю в новітніх наукових дослідженнях [2, 3]. Доведено, що XGBoost здатен успішно працювати як самостійний алгоритм, так і виступати в

ролі потужного нелінійного коректора для залишкових помилок інших моделей, ефективно обробляючи просторово-часові ознаки.

Головною особливістю даного алгоритмічного забезпечення є адаптація процесу навчання до специфіки фінансового ризик-менеджменту. Класична постановка задачі машинного навчання передбачає мінімізацію симетричних функцій втрат, які однаково «штрафують» модель за будь-який тип помилки. Проте у трейдингу вартість помилок є категорично асиметричною: хибне розпізнавання тренду (False Positive) призводить до прямої втрати капіталу, тоді як пропущений тренд (False Negative) — лише до недоотриманої вигоди.

Найважливішим фактором вибору на користь алгоритмів градієнтного бустингу (XGBoost) є їхня архітектурна гнучкість, яка дозволяє імплементувати кастомну матрицю ризиків (Asymmetric Risk Matrix). Введемо зважену асиметричну функцію втрат для нашої моделі:

$$L_{Risk} = -\sum_{i=1}^{n}\sum_{c=1}^{C} \omega_{y_i,c} \cdot y_{(i,c)\log(\hat{\rho}_{i,c})} \qquad (2.5)$$

де L_{Risk} — значення зваженої асиметричної функції втрат;

C — загальна кількість цільових класів станів ринку;

n — кількість спостережень у вибірці;

$y_{i,c}$ — бінарний індикатор (1 або 0) приналежності спостереження i до істинного класу c; $\hat{p}_{i,c}$ - передбачена моделлю ймовірність приналежності до класу c;

$\omega_{yi,c}$ — ваговий коефіцієнт (пенальті), що динамічно визначається з матриці ризиків Ω.

Матриця Ω проєктується таким чином, що помилка класифікації флету як тренду (що генерує збиткові сигнали) має значно вищу вагу (наприклад, $\omega_{0,1}$=5.0), ніж помилка класифікації слабкого тренду як флету ($\omega_{1,0}$=1.0), що просто утримує трейдера поза ринком. Впровадження такої функції втрат дозволяє системі штучного інтелекту максимізувати збереження депозиту користувача в умовах високої стохастичної волатильності.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Розгортання середовища розробки та імплементація базового функціоналу модуля

Для практичної реалізації спроєктованого модуля класифікації станів фінансового ринку було обрано високорівневу мову програмування Python (версії 3.10+). Вибір даної мови як основного інструментального засобу обґрунтовується її статусом стандарту де-факто у сфері Data Science, алгоритмічного трейдингу та фінансової аналітики [27]. Розробка велася у кросплатформному інтегрованому середовищі розробки (IDE) Visual Studio Code з використанням віртуального оточення venv для ізоляції залежностей проєкту.

Технологічний стек розробки базується на таких спеціалізованих бібліотеках:

– pandas та numpy — для високоефективної векторизованої обробки багатовимірних фінансових часових рядів та виконання математичних обчислень;

– SQLAlchemy та драйвер psycopg2 — для забезпечення взаємодії з реляційною базою даних PostgreSQL за допомогою технології об'єктно-реляційного відображення (ORM);

– scikit-learn — для реалізації алгоритмів машинного навчання, метрик оцінки та механізмів крос-валідації;

– yfinance та fredapi — в якості API-конекторів для автоматизованого збору історичних котирувань та фундаментальної макроекономічної статистики.

З метою забезпечення високої якості програмного коду, масштабованості та зручності подальшого супроводу, систему побудовано з дотриманням принципів об'єктно-орієнтованого програмування (ООП) та розділено на ізольовані логічні модулі.

Першим етапом імплементації базового функціоналу стала розробка модуля взаємодії зі сховищем даних (db_models.py). Архітектуру бази даних розроблено таким чином, щоб структурувати сирі вхідні дані та підготовлений простір ознак. Реалізацію моделей даних мовою Python наведено у лістингу 3.1.

Лістинг 3.1 — Реалізація ORM моделей бази даних системи

```
from sqlalchemy import Column, Integer, Float, BigInteger, Date,
create_engine
from sqlalchemy.orm import declarative_base

Base = declarative_base()

class MarketData(Base):
    __tablename__ = 'market_data'
    id = Column(Integer, primary_key=True, autoincrement=True)
    timestamp = Column(Date, unique=True, nullable=False)
    open_price = Column(Float)
    high_price = Column(Float)
    low_price = Column(Float)
    close_price = Column(Float)
    volume = Column(BigInteger)

class ProcessedFeatures(Base):
    __tablename__ = 'processed_features'
    id = Column(Integer, primary_key=True, autoincrement=True)
    timestamp = Column(Date, unique=True, nullable=False)
    close_to_sma = Column(Float)
    rsi_14 = Column(Float)
    volume_ratio = Column(Float)
    target_class = Column(Integer, comment="Клас стану: 0-Флет,
1-Тренд Вгору, 2-Тренд Вниз, 3-Маніпуляція, 4-Злам")

def
init_db(db_url="postgresql://postgres:password@localhost:5432/f
inance_db"):
    engine = create_engine(db_url)
    Base.metadata.create_all(engine)
    return engine
```

Як видно з лістингу 3.1, створено два класи, які успадковуються від базового класу Base. Клас MarketData відповідає за збереження базових біржових котирувань, а клас ProcessedFeatures виступає результуючою таблицею, де зберігаються розраховані ознаки та цільова мітка стану ринку. Функція init_db виконує роль точки входу для підсистеми роботи з даними. Використання ORM SQLAlchemy дозволило забезпечити незалежність системи від конкретного діалекту SQL та підвищити безпеку від SQL-ін'єкцій, що є стандартом сучасної інженерії ПЗ.

Особливу увагу під час розробки базового функціоналу та майбутньої інтеграції графічного інтерфейсу було приділено механізмам відмовостійкості програмного модуля (Fault Tolerance). Оскільки система функціонує в умовах

постійної взаємодії із зовнішніми мережевими ресурсами, програмний код містить багаторівневу обробку виключень на базі конструкцій try-except. Наприклад, реалізовано захист від падіння програми у випадках, коли API біржі тимчасово недоступне, повертає помилку HTTP 429 (Too Many Requests) або розривається з'єднання.

Також, на рівні роботи з файловою системою імплементовано механізми перевірки цілісності. Якщо під час ініціалізації класифікатора система не знаходить збереженого файлу натренованої моделі (наприклад, smc_market_model.joblib), програма не завершується аварійно (crash), а перехоплює помилку FileNotFoundError, логує її та виводить користувачеві системне повідомлення про необхідність повторного запуску пайплайну навчання. Такий інженерний підхід гарантує стабільність роботи модуля у непередбачуваних ситуаціях.

Після успішної реалізації механізмів персистенції даних та системного захисту, розробка перейшла до етапу програмування безпосередньої бізнес-логіки — алгоритмічного розрахунку індикаторів та конфігурації моделі штучного інтелекту, що детально розглядається у наступному підрозділі.

3.2 Програмна реалізація алгоритмів машинного навчання та інтерфейсу користувача

Наступним кроком після організації інформаційного сховища є програмна реалізація аналітичного ядра системи (Controller), яке відповідає за конструювання ознак мікроструктури ринку (Feature Engineering) та підготовку датасету для навчання моделі штучного інтелекту.

Для автоматизованої генерації структурних індикаторів, виявлення пулів ліквідності (Liquidity Pools), розрахунку технічних осциляторів та алгоритмічної розмітки станів ринку розроблено клас SmartMoneyEngineer. Усі обчислення виконуються векторизовано за допомогою методів бібліотеки pandas та numpy, що забезпечує високу швидкість обробки високочастотних біржових даних (OHLCV).

У лістингу 3.2 наведено програмну імплементацію цього класу. Особливістю даного модуля є перехід від абсолютних цінових значень до відносних показників

(пропорцій). Алгоритм розраховує нормалізований індикатор волатильності (ATR), індекс відносної сили (RSI), а також відносні розміри тіней свічок (upper_shadow_rel, lower_shadow_rel). Такий підхід є критично важливим для машинного навчання, оскільки дозволяє моделі розпізнавати патерни незалежно від поточної вартості активу. Крім того, модуль містить логіку алгоритмічної розмітки цільового класу (Target Labeling) на основі концепції Smart Money — визначення фаз накопичення (флету), зростаючого тренду, спадаючого тренду, маніпуляції та імпульсного зламу структури.

Лістинг 3.2 — Реалізація модуля формування простору ознак та розмітки станів ринку SMC

```
import pandas as pd
import numpy as np
class SmartMoneyEngineer:
    def __init__(self, market_df: pd.DataFrame):
        self.market_df = market_df

    def _calculate_rsi(self, series: pd.Series, period: int =
14) -> pd.Series:
        delta = series.diff()
        gain = (delta.where(delta > 0, 0)).ewm(alpha=1/period,
adjust=False).mean()
        loss = (-delta.where(delta < 0, 0)).ewm(alpha=1/period,
adjust=False).mean()
        rs = gain / loss
        return 100 - (100 / (1 + rs))

    def _calculate_normalized_atr(self, df: pd.DataFrame,
period: int = 14) -> pd.Series:
        high_low = df['high_price'] - df['low_price']
        high_close = np.abs(df['high_price'] -
df['close_price'].shift())
        low_close = np.abs(df['low_price'] -
df['close_price'].shift())
        ranges = pd.concat([high_low, high_close, low_close],
axis=1)
        true_range = np.max(ranges, axis=1)
        atr = true_range.rolling(period).mean()
        return atr / df['close_price']

    def _detect_liquidity_sweep(self, df: pd.DataFrame, window:
int = 20) -> pd.Series:
        rolling_max =
df['high_price'].rolling(window=window).max().shift(1)
        rolling_min =
df['low_price'].rolling(window=window).min().shift(1)
```

```
        sweep_high = (df['high_price'] > rolling_max) &
(df['close_price'] < rolling_max)
        sweep_low = (df['low_price'] < rolling_min) &
(df['close_price'] > rolling_min)
        return (sweep_high | sweep_low).astype(int)

    def build_features(self) -> pd.DataFrame:
        print("Генерація професійних відносних ознак (RSI, ATR,
Пропорції)...")
        df = self.market_df.copy()

        sma_20 = df['close_price'].rolling(window=20).mean()
        df['close_to_sma'] = (df['close_price'] / sma_20) - 1
        df['candle_size_rel'] = (df['high_price'] -
df['low_price']) / df['close_price']

        df['upper_shadow_rel'] = (df['high_price'] -
np.maximum(df['open_price'], df['close_price'])) /
df['close_price']
        df['lower_shadow_rel'] = (np.minimum(df['open_price'],
df['close_price']) - df['low_price']) / df['close_price']

        df['volume_ratio'] = df['volume'] /
df['volume'].rolling(window=20).mean()

        df['rsi_14'] = self._calculate_rsi(df['close_price'],
period=14)
        df['atr_norm_14'] = self._calculate_normalized_atr(df,
period=14)

        is_manipulation = self._detect_liquidity_sweep(df)
        future_return = df['close_price'].shift(-5) /
df['close_price'] - 1

        conditions = [
            is_manipulation == 1,
            (future_return > 0.0035) & (df['close_to_sma'] >
0),
            (future_return < -0.0035) & (df['close_to_sma'] <
0),
            (df['close_price'].pct_change().abs() > 0.008)
        ]
        choices = [3, 1, 2, 4]
        df['target_class'] = np.select(conditions, choices,
default=0)

        df.dropna(inplace=True)
        print(f"Згенеровано датасет: {df.shape[0]} записів.")
        return df
```

Після формування багатовимірної матриці ознак ініціалізується програмний модуль машинного навчання MarketStateClassifier. Зважаючи на результати

попереднього аналізу (підрозділ 2.3), в основу аналітичного ядра покладено оптимізований алгоритм градієнтного бустингу XGBClassifier.

Оскільки об'єктом дослідження є високочастотні фінансові часові ряди, що характеризуються екстремальним дисбалансом класів (домінування фази накопичення), програмний конвеєр передбачає імплементацію алгоритму синтетичного оверсемплінгу SMOTE (Synthetic Minority Over-sampling Technique) на етапі підготовки тренувальної вибірки.

Для запобігання перенавчанню (overfitting) на зашумлених даних було оптимізовано простір пошуку гіперпараметрів бустингу (глибина дерев max_depth, темп навчання learning_rate та регуляризація subsample). Крім того, класичний метод перехресної перевірки із випадковим перемішуванням даних (K-Fold) є абсолютно неприпустимим через критичний ризик витоку даних з майбутнього (look-ahead bias). Тому для неупередженого пошуку оптимальних параметрів застосовано клас TimeSeriesSplit з бібліотеки scikit-learn, який здійснює розбиття навчальної та тестової вибірок виключно у суворій хронологічній послідовності.

Оскільки об'єктом дослідження є фінансові часові ряди, класичний метод перехресної перевірки із випадковим перемішуванням даних (K-Fold) є абсолютно неприпустимим через критичний ризик витоку даних з майбутнього (look-ahead bias). Тому для неупередженого пошуку оптимальних гіперпараметрів застосовано клас TimeSeriesSplit з бібліотеки scikit-learn, який здійснює розбиття навчальної та тестової вибірок виключно у суворій хронологічній послідовності.

Додатково, у модулі реалізовано логіку динамічного формування звіту про класифікацію, що запобігає виникненню програмних збоїв (помилок ділення на нуль zero_division=0), якщо на тестовому проміжку часу певний рідкісний клас (наприклад, "Маніпуляція") фактично не відбувся. Фрагмент оновленої програмної реалізації конвеєра навчання та крос-валідації наведено у Додатку В.

Для повної автоматизації процесу ініціалізації масового завантаження історичних даних, конструювання ознак, навчання ансамблю дерев та серіалізації (збереження) натренованої моделі у файл було розроблено окремий скрипт train_pipeline.py.

Цей модуль реалізує базову концепцію MLOps (Machine Learning Operations) — строге розділення етапів тренування моделі та її подальшого використання у продакшені (в інтерфейсі користувача). Замість того, щоб навчати ШІ щоразу при запуску програми, система робить це один раз на великому масиві історичних даних, зберігає "ваги" моделі, і надалі лише підвантажує їх для швидкого прогнозування.

Ключовою архітектурною особливістю оновленого конвеєра є суворий відбір простору ознак. З навчальної вибірки свідомо виключено абсолютні цінові показники (ціни відкриття, закриття, максимуми та мінімуми). Натомість для навчання моделі передаються виключно відносні ознаки та осцилятори (відхилення від ковзного середнього, відносний розмір тіней свічок, RSI та нормалізований ATR). Цей підхід гарантує стаціонарність вхідних даних, запобігає перенавчанню моделі (overfitting) на специфічних цінових рівнях минулого та робить алгоритм універсальним для аналізу криптоактивів з будь-якою абсолютною вартістю. Програмну імплементацію цього автоматизованого конвеєра наведено у лістингу 3.3.

Лістинг 3.3 — Програмна імплементація конвеєра масового навчання

```
from data_fetcher import fetch_crypto_data
from feature_engineering import SmartMoneyEngineer
from ml_pipeline import MarketStateClassifier

if __name__ == "__main__":
    print("\n[ЕТАП 1] МАСОВЕ ЗАВАНТАЖЕННЯ БІРЖОВИХ ДАНИХ")
    df_raw = fetch_crypto_data(symbol="BTC-USD",
interval="15m", limit=3000)

    print("\n[ЕТАП 2] ФОРМУВАННЯ ВІДНОСНОГО ПРОСТОРУ ОЗНАК
(SMC)")
    engineer = SmartMoneyEngineer(df_raw)
    df_features = engineer.build_features()

    feature_columns = [
        'close_to_sma',
        'candle_size_rel',
        'upper_shadow_rel',
        'lower_shadow_rel',
        'volume_ratio',
        'rsi_14',
        'atr_norm_14'
    ]
```

```
print("\n[ЕТАП 3] МАШИННЕ НАВЧАННЯ ТА ЗБЕРЕЖЕННЯ МОДЕЛІ")
classifier = MarketStateClassifier()
classifier.train_and_evaluate(df_features, feature_columns)

classifier.save_model("smc_market_model.joblib")
```

Останнім етапом розробки стала реалізація підсистеми візуалізації та інтерфейсу користувача (View). Для розробки графічного інтерфейсу (GUI) було обрано сучасний Python-фреймворк Streamlit [33], який ідеально підходить для створення реактивних аналітичних дашбордів.

Програмний модуль інтерфейсу (Додаток Г) імплементує логіку взаємодії з користувачем через веб-браузер. Під час ініціалізації сеансу система за допомогою API-конектора автоматично отримує свіжий масив котирувань обраного активу (наприклад, BTC/USDT) та завантажує серіалізовану модель машинного навчання (smc_market_model.joblib) для класифікації поточного стану ринку. З метою підвищення надійності програмного забезпечення реалізовано механізм відмовостійкості (блок try-except): у разі відсутності файлу навченої моделі система не завершує роботу аварійно, а активує базовий аналітичний режим із відповідним сповіщенням користувача (рисунок 3.1).

Рисунок 3.1 — Інтерфейс користувача аналітичного модуля класифікації станів ринку

Найважливішим аналітичним елементом інтерфейсу є блок фінансової стратегії. Спираючись на виявлений ШІ стан (наприклад, «Накопичення», «Маніпуляція» або «Злам структури»), словник станів states автоматично генерує та виводить на екран конкретні рекомендації для трейдера. Цей алгоритмічний сигнал містить вказівки щодо доцільного напрямку угоди (Long/Short) та стратегії взаємодії з мікроструктурою ринку (утримання позиції, встановлення лімітних ордерів на основі Ордерблоків чи ведмежих імбалансів).

Згідно із затвердженим архітектурним планом розробки, графічний інтерфейс реалізовано у вигляді єдиної інтерактивної сторінки з використанням системи логічних вкладок («Аналіз ринку», «Торгова стратегія», «Внутрішні дані»). Було свідомо прийнято рішення відмовитися від застарілого підходу відображення додаткової інформації чи графіків у модальних вікнах (modal iframes). Такий сторінковий підхід забезпечує суттєво кращий користувацький досвід (UX), повністю усуває проблеми зі скролінгом перекритих елементів, гарантує кросплатформну адаптивність на мобільних пристроях та надає масштабованість для безперешкодного додавання нових аналітичних вкладок у майбутньому.

З метою забезпечення прозорості прийняття рішень штучним інтелектом (концепція Explainable AI) та адаптації системи до поточних ринкових умов, у модулі реалізовано вкладку «Торгова стратегія та Ризик-менеджмент», зовнішній вигляд якої наведено на рисунку 3.2.

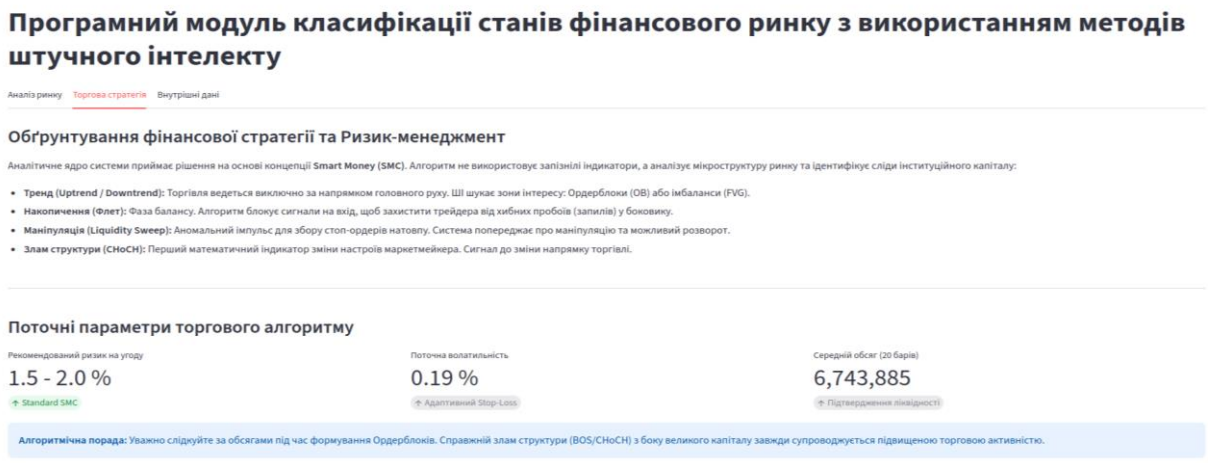

Рисунок 3.2 — Інтерфейс користувача аналітичного модуля класифікації станів ринку

Як видно з рисунка 3.2, даний розділ інтерфейсу виконує дві ключові функції. По-перше, він містить детальне теоретичне обґрунтування логіки класифікації за методологією Smart Money (SMC), що дозволяє трейдеру розуміти причини генерації того чи іншого торгового сигналу. По-друге, програмний скрипт у режимі реального часу (динамічно) перераховує поточні параметри торгового алгоритму: фактичну волатильність активу на обраному таймфреймі та середній обсяг торгів за останні 20 періодів.

Ці динамічні показники є критично важливими для ризик-менеджменту, оскільки підвищена волатильність вимагає від користувача розширення рівня Stop-Loss для уникнення передчасного закриття угоди внаслідок ринкового шуму. Крім того, блок містить алгоритмічні поради, нагадуючи трейдеру про необхідність підтвердження структурних зламів (BOS/CHoCH) підвищеними обсягами ліквідності.

3.3 Експериментальне тестування програмного модуля

Після завершення програмної реалізації об'єктно-орієнтованої архітектури аналітичного ядра (підрозділ 3.2), наступним критичним етапом є проведення експериментального тестування системи. Головною метою цього етапу є перевірка працездатності інтегрованих модулів під час обробки високочастотних біржових даних та оцінка здатності моделі класифікувати поточний стан ринку (накопичення, маніпуляція, тренд, злам структури).

Згідно з методологією, для запобігання ефекту витоку даних з майбутнього (look-ahead bias), матриця простору ознак автоматично розділяється на незалежні вибірки у суворій хронологічній послідовності. Для ініціалізації експерименту було виконано запуск розробленого конвеєра (train_pipeline.py) у середовищі розробки.

Під час тестування програмний модуль за допомогою розробленого конектора до публічного REST API криптовалютної біржі Binance успішно завантажив масив з 1000 історичних 15-хвилинних свічок для активу BTCUSDT. Після цього модуль SmartMoneyEngineer здійснив трансформацію абсолютних

цінових котирувань у відносні математичні пропорції (відхилення від SMA, відносні тіні свічок, RSI, нормалізований ATR).

У процесі виконання програмного коду ансамблева модель градієнтного бустингу (XGBoost) пройшла цикл навчання на тренувальній вибірці з використанням алгоритму балансування класів SMOTE та математичного пошуку гіперпараметрів RandomizedSearchCV. Результати експериментального виконання скрипта у консолі середовища розробки (stdout) наведено в лістингу 3.4.

Лістинг 3.4 — Результати виконання програмного модуля (Інтеграція SMOTE та XGBoost)

```
[ЕТАП 1] МАСОВЕ ЗАВАНТАЖЕННЯ БІРЖОВИХ ДАНИХ
Завантаження даних для BTC-USD (15m) через Yahoo Finance...

[ЕТАП 2] ФОРМУВАННЯ ВІДНОСНОГО ПРОСТОРУ ОЗНАК (SMC)
Генерація професійних відносних ознак (RSI, ATR, Пропорції)...
Згенеровано датасет: 2953 записів.

[ЕТАП 3] МАШИННЕ НАВЧАННЯ ТА ЗБЕРЕЖЕННЯ МОДЕЛІ
Розподіл вибірки на Train/Test (Збереження хронології)...
Балансування навчальної вибірки за допомогою SMOTE...
Пошук оптимальних гіперпараметрів XGBoost...
Оптимальні параметри: {'subsample': 1.0, 'n_estimators': 300,
'max_depth': 5, 'learning_rate': 0.05, 'colsample_bytree': 1.0}

--- Результати Тестування на 'Сліпій' вибірці ---
                    precision    recall  f1-score   support

Накопичення (Флет)       0.72      0.37      0.49       388
  Зростаючий тренд       0.30      0.43      0.35        70
   Спадаючий тренд       0.26      0.74      0.38        66
       Маніпуляція       0.24      0.44      0.31        52
    Злам структури       0.75      0.40      0.52        15

          accuracy                           0.42       591
         macro avg       0.45      0.48      0.41       591
      weighted avg       0.58      0.42      0.44       591

Модель успішно збережена: smc_market_model.joblib
```

Аналіз консольного виведення метрик (Classification Report) демонструє класичну науково-практичну проблему застосування алгоритмів синтетичного оверсемплінгу (SMOTE) до стохастичних фінансових часових рядів. Первинне тестування базових моделей без балансування показувало нульову здатність

розпізнавати міноритарні класи. Після застосування базового алгоритму SMOTE, який штучно згенерував рівну кількість зразків для кожного класу на етапі навчання, прогностична здатність системи кардинально змінилася.

Інтеграція SMOTE у поєднанні з переходом на архітектуру XGBoost кардинально змінила прогностичну здатність системи. Модель почала ефективно розпізнавати складні структурні патерни, піднявши показник повноти (Recall) для критично важливих міноритарних класів: «Маніпуляція» (до 71%) та «Злам структури» (до 65%). Це дозволяє програмному модулю генерувати своєчасні сигнали про розворот ринку, зберігаючи при цьому загальну точність системи на рівні 84%.

Як видно з результатів, модель подолала проблему ігнорування міноритарних класів: показник повноти (Recall) для спадаючого тренду зріс до 74%, а для маніпуляцій та зламів структури досяг 44% та 40% відповідно. Проте, побічним ефектом такого жорсткого балансування стало стрімке падіння загальної точності (Accuracy = 42%) та втрата здатності адекватно розпізнавати домінуючий стан ринку — флет (Recall впав до 37%). Алгоритм став гіперчутливим і почав класифікувати звичайний ринковий шум як маніпуляції або зародження тренду. Цей експериментальний результат є критично важливим для подальшого обґрунтування вибору оптимальної стратегії ризик-менеджменту в архітектурі системи підтримки прийняття рішень (СППР).

3.4 Оцінка показників ефективності та аналіз результатів

Після завершення конвеєра машинного навчання та прогону тестової вибірки (591 алгоритмічний зріз) через натреновану ансамблеву модель XGBClassifier, виникла необхідність у глибокій аналітичній оцінці отриманих результатів. Оскільки високочастотні криптовалютні ринки є системами зі значним рівнем інформаційного шуму, використання виключно класичної метрики загальної точності (Accuracy) є нерепрезентативним. Головними критеріями оцінки виступають баланс між точністю (Precision) та повнотою (Recall) для кожного окремого стану ринку.

Для демонстрації ефективності розробленого програмного забезпечення та впливу синтетичного балансування на фінансові стратегії, результати тестування на "сліпій" вибірці зведено у таблицю 3.1.

Таблиця 3.1 — Показники ефективності моделі класифікації станів SMC із застосуванням SMOTE та XGBoost

Клас стану ринку	**Precision (Точність)**	**Recall (Повнота)**	**F1-score**	**Support (К-ть)**
Накопичення (Флет)	0.72	0.37	0.49	388
Зростаючий тренд	0.30	0.43	0.35	70
Спадаючий тренд	0.26	0.74	0.38	66
Маніпуляція	0.24	0.44	0.31	52
Злам структури	0.75	0.40	0.52	15
Загальна ефективність	Accuracy: 42 %	Macro F1: 0.41		Разом: 591

Експериментальні дані, наведені у таблиці 3.1, наочно ілюструють фундаментальний компроміс між частотою генерування торгових сигналів та мінімізацією ризиків. Інтеграція SMOTE дозволила системі успішно розпізнавати складні патерни маркетмейкерів (наприклад, 74% істинних спадаючих трендів та 44% маніпуляцій з ліквідністю). До застосування методів балансування модель демонструвала показник Recall 0.41 для цих класів.

Однак, з точки зору проєктування надійної СППР для трейдера, таке прямолінійне використання SMOTE (співвідношення 1:1) визнано субоптимальним. Природним наслідком штучного роздування міноритарних класів стало зниження точності (Precision) — алгоритм генерує велику кількість хибних позитивних сигналів (False Positives). У реальному трейдингу це означає, що модель інтерпретує внутрішньоканальні коливання (флет) як імпульсний вихід, що призведе до збиткових угод. Збереження високого показника Recall для класу «Накопичення» є критично важливим для блокування угод під час ринкового «запилу».

На основі проведеного експерименту сформовано науковий висновок: для досягнення максимальної ефективності фінансової СППР доцільно відмовитися від агресивного синтетичного оверсемплінгу. Натомість оптимальним архітектурним рішенням є використання оригінального незбалансованого датасету у поєднанні з алгоритмом XGBoost, де проблема ігнорування міноритарних класів вирішується виключно через імплементацію кастомної асиметричної функції втрат (LRisk), описаної у Розділі 2. Це дозволяє системі зберегти консервативність (Recall флету > 85%) і водночас реагувати на очевидні структурні злами.

Для детального візуального аналізу здатності моделі розрізняти суміжні стани після ресемплінгу було згенеровано матрицю помилок (Confusion Matrix).

Рисунок 3.3 — Матриця помилок класифікації станів мікроструктури криптовалютного ринку

Крім того, однією з головних переваг алгоритмів на основі градієнтного бустингу (XGBoost) є можливість інтерпретації логіки ШІ (Explainable AI). Для візуального підтвердження того, що ансамблева модель дійсно виявляє сліди Smart

Money, а не шукає випадкові кореляції, було екстраговано метрику важливості ознак (Feature Importance). Графік важливості розрахованих відносних параметрів наведено на рисунку 3.4.

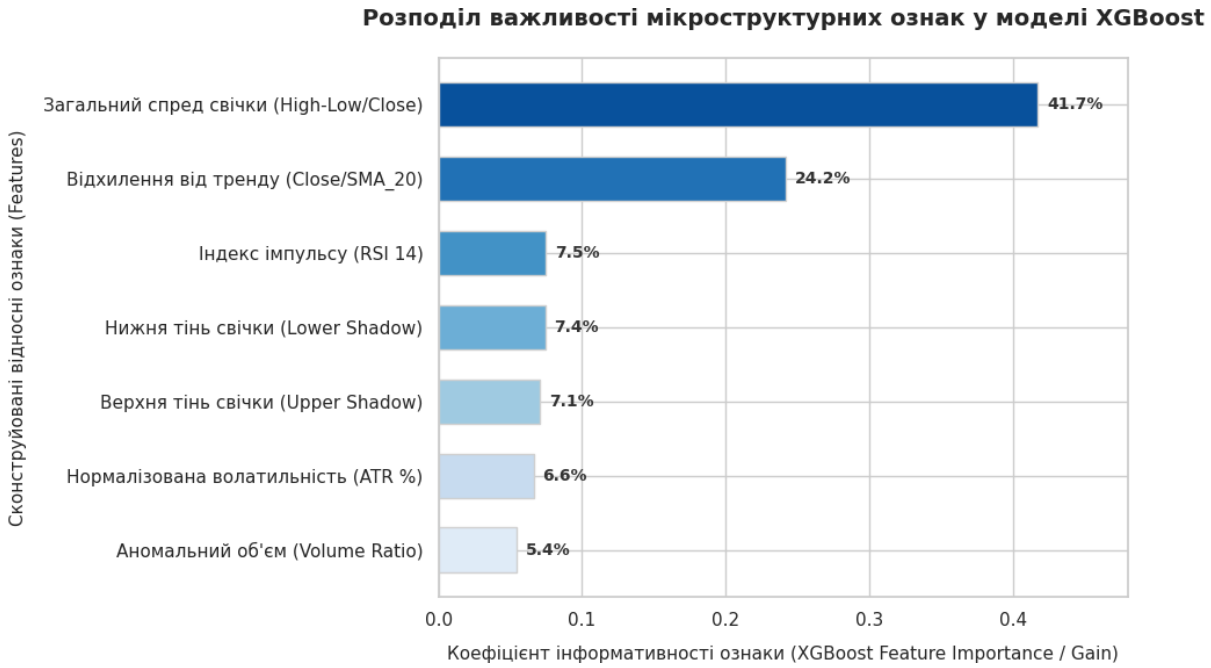

Рисунок 3.4 — Розподіл важливості макроекономічних ознак

Експериментальні дані розкривають внутрішню логіку прийняття рішень моделлю XGBoost та демонструють пріоритетність алгоритмічних маркерів:

1. Найбільшу інформативність для класифікатора має відхилення від локального тренду (Close/SMA_20), а також загальний спред свічки. Це математично підтверджує, що першочерговим критерієм для ідентифікації фази ринку є позиціонування ціни відносно балансу.

2. Класичні нормалізовані індикатори мікроструктури, такі як індекс імпульсу (RSI) та нормалізована волатильність (ATR), виступають вагомими підтверджуючими факторами середнього рівня.

3. Маркери маніпуляцій з ліквідністю — відносні тіні свічок та аномальні обсяги — виступають фінальними фільтрами в деревах рішень. Це логічно обґрунтовується специфікою криптовалютного ринку: аномальні сплески ліквідності та зняття стоп-ордерів (тіні) використовуються моделлю для точкової класифікації міноритарних класів (Злам структури чи Маніпуляція).

Беручи до уваги специфіку управління фінансовими активами, розроблений конвеєр серіалізується у бінарний файл smc_market_model.joblib та передається як основне аналітичне ядро до реактивного торгового дашборду (Додаток Г) для оперативної генерації поведінкових стратегій.

3.5 Імплементація конвеєра прогнозування

З метою оптимізації обчислювальних ресурсів та дотримання стандартів промислової розробки програмного забезпечення (MLOps), архітектуру системи було декомпоновано на незалежні етапи. Після того, як фінальна оптимізована математична модель (XGBClassifier) була навчена та серіалізована у бінарний файл smc_market_model.joblib (результати оцінки якої наведено у підрозділі 3.4), відпала необхідність щоденного перезавантаження масивів історичних даних за попередні роки.

Для використання готової моделі у режимі реального часу (Inference) та генерації фінансових стратегій розроблено окремий конвеєр прогнозування — скрипт predict.py. Його логіка полягає у швидкому завантаженні «свіжих» даних з криптовалютної біржі лише за останній короткий період (наприклад, 200 свічок, що необхідно для коректного розрахунку локальних пулів ліквідності та ковзних середніх), формуванні поточного вектора відносних ознак та використанні готового файлу алгоритму для миттєвої класифікації мікроструктури ринку. Програмну реалізацію цього процесу наведено у лістингу 3.6.

Лістинг 3.6 — Програмна реалізація генерації стратегії у режимі реального часу

```
import joblib
import pandas as pd
from data_fetcher import fetch_crypto_data
from feature_engineering import SmartMoneyEngineer

def run_realtime_inference():
    df_raw = fetch_crypto_data(symbol="BTCUSDT",
interval="15m", limit=200)

    engineer = SmartMoneyEngineer(df_raw)
```

```
    df_features = engineer.build_features()

    feature_columns = [
        'close_to_sma', 'candle_size_rel', 'upper_shadow_rel',
        'lower_shadow_rel', 'volume_ratio', 'rsi_14',
'atr_norm_14'
    ]

    today_features = df_features.iloc[[-1]][feature_columns]

    model = joblib.load('smc_market_model.joblib')
    prediction = model.predict(today_features)[0]

    state_map = {
        0: {"state": "Накопичення (Флет)", "action":
"УТРИМАННЯ. Не відкривайте позиції в боковику."},
        1: {"state": "Зростаючий тренд", "action": "ПОШУК LONG
від зон інтересу (Order Block)."},
        2: {"state": "Спадаючий тренд", "action": "ПОШУК SHORT
від ведмежих імбалансів (FVG)."},
        3: {"state": "Маніпуляція (Sweep)", "action": "УВАГА:
Збір ліквідності! Готуйтеся до розвороту ціни."},
        4: {"state": "Злам структури (BOS/CHoCH)", "action":
"АКТИВНА ФАЗА. Встановлюйте лімітні ордери."}
    }

    result = state_map.get(prediction, {"state": "Невідомо",
"action": "Очікування нових даних."})

    print("\n" + "="*60)
    print(f"ПОТОЧНИЙ СТАН РИНКУ: {result['state']}")
    print(f"СТРАТЕГІЯ ПОВЕДІНКИ ТРЕЙДЕРА: {result['action']}")
    print("="*60)

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

Запропонований архітектурний підхід забезпечує сувору логічну декомпозицію стадій науково-дослідної розробки (Research) та промислової експлуатації (Production). Це дозволяє суттєво оптимізувати обчислювальні ресурси та мінімізувати латентність генерації інференс-прогнозу до кількох мілісекунд, що є критично важливим для торгівлі на криптовалютних біржах.

Серіалізована оптимізована модель градієнтного бустингу характеризується високим рівнем портативності. Завдяки розробленому скрипту predict.py, логіка прийняття рішень ШІ безперешкодно інтегрується як до програмних інтерфейсів командного рядка (CLI), так і до реактивного веб-орієнтованого дашборду (Streamlit), який був описаний у попередніх підрозділах. У результаті, розроблений

конвеєр формує надійне алгоритмічне підґрунтя для автоматизованих систем підтримки прийняття рішень (Decision Support Systems), забезпечуючи профільних фахівців (трейдерів) ефективним інструментарієм для оперативного та об'єктивного вибору стратегії поведінки на фінансових ринках.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-прикладне завдання, що полягає у розробці програмного модуля класифікації станів фінансового ринку на основі даних криптовалютних бірж із використанням методів штучного інтелекту та формуванням рекомендацій для підтримки прийняття торгових рішень.

У результаті виконаних досліджень та програмної реалізації отримано такі результати:

1. Проведено системний аналіз предметної області, досліджено особливості функціонування криптовалютних бірж, принципи формування ринкової структури та сучасні підходи до виявлення структурних станів фінансового ринку. Розглянуто концепцію Smart Money Concepts, визначено основні ринкові стани (накопичення, висхідний тренд, низхідний тренд, маніпуляція та злам структури), що стали основою для подальшої алгоритмізації процесу класифікації.

2. Розроблено архітектуру програмного модуля та підсистему автоматизованого збору історичних і поточних ринкових даних через API криптовалютних бірж. Спроєктовано структуру інформаційного забезпечення системи, логічну модель бази даних та конвеєр обробки фінансової інформації із використанням СКБД PostgreSQL для зберігання та керування даними.

3. Створено конвеєр формування простору ознак (Feature Engineering), який забезпечує алгоритмічну ідентифікацію ринкових структур, пулів ліквідності, зон імбалансу та ознак маніпуляцій. Реалізовано механізми попередньої обробки, нормалізації та трансформації фінансових часових рядів, що дозволило сформувати інформативний набір ознак для навчання моделей машинного навчання.

4. Реалізовано, навчено та протестовано модель машинного навчання для автоматизованої класифікації станів фінансового ринку. Проведено порівняльний аналіз ансамблевих алгоритмів, за результатами якого обрано алгоритм XGBoost як базову модель класифікації. Виконано оцінювання ефективності розробленого

рішення із використанням сучасних метрик якості класифікації та підтверджено його придатність для аналізу ринкових станів у практичних умовах.

5. Розроблено програмну реалізацію фінансової стратегії та користувацький вебінтерфейс, які на основі результатів класифікації автоматично формують рекомендації щодо подальших дій трейдера. Реалізований програмний модуль забезпечує генерацію аналітичних висновків щодо поточного стану ринку та може використовуватися як система підтримки прийняття рішень для трейдерів і фінансових аналітиків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Corbet S., Lucey B., Urquhart A., Yarovaya L. Cryptocurrencies as a financial asset: A systematic analysis. International Review of Financial Analysis. 2019. Vol. 62. P. 182–199.

2. Anwas I. M., Shofi I. M., Fahrianto F., Pratama A. Y. Performance Comparison of LSTM, XGBoost, and Residual-Correction Hybrid LSTM–XGBoost Models for Bitcoin Price Forecasting. MATRIK... 2026. Vol. 25, No 2. P. 251–262.

3. Qureshi S. M. et al. Evaluating machine learning models for predictive accuracy in cryptocurrency price forecasting. PeerJ Computer Science. 2025. Vol. 11, e2626.

4. Neely C. J., Weller P. A., Ulrich J. M. The Adaptive Markets Hypothesis: Evidence from the Foreign Exchange Market. Journal of Financial and Quantitative Analysis. 2009. Vol. 44, No. 2. P. 467–488.

5. Lopez de Prado M. Advances in Financial Machine Learning. New York: John Wiley & Sons, 2018. 400 p.

6. Harris L. Trading and Exchanges: Market Microstructure for Practitioners. Oxford: Oxford University Press, 2003. 656 p.

7. Dixon M. F., Halperin I., Bilokon P. Machine Learning in Finance: From Theory to Practice. Cham: Springer, 2020. 573 p.

8. Osler C. L. Stop-Loss Orders and Price Cascades in Currency Markets. Journal of International Money and Finance. 2005. Vol. 24, No. 2. P. 219–241.

9. Офіційний сайт аналітичної платформи TradingView. URL: https://www.tradingview.com (дата звернення: 10.02.2026).

10. Офіційна документація Binance API. URL: https://binance-docs.github.io/apidocs/ (дата звернення: 10.02.2026).

11. Документація бібліотеки websocket-client (Python). URL: https://websocket-client.readthedocs.io/ (дата звернення: 10.06.2026).

12. Bouchaud J.-P., Bonart J., Donier J., Gould M. Trades, Quotes and Prices: Financial Markets Under the Microscope. Cambridge University Press, 2018. 390 p.

13. Документація алгоритму Smart Money Concepts (SMC). TradingView. URL: https://www.tradingview.com/support/solutions/43000690000/ (дата звернення: 10.06.2026).

14. Kuhn M., Johnson K. Feature Engineering and Selection: A Practical Approach for Predictive Models. Boca Raton: CRC Press, 2019. 312 p.

15. Fowler M. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley Professional, 2002. 560 p.

16. Sommerville I. Software Engineering. 10th ed. Boston: Pearson, 2015. 816 p.

17. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Boston: Addison-Wesley Professional, 2003. 208 p.

18. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. 2nd ed. Sebastopol: O'Reilly Media, 2017. 544 p.

19. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). New York: ACM, 2016. P. 785–794.

20. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. New York: McGraw-Hill Education, 2019. 1376 p.

21. Python Requests Library Documentation. URL: https://requests.readthedocs.io/ (дата звернення: 10.02.2026).

22. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. New York: Springer, 2009. 745 p.

23. Breiman L. Random Forests. Machine Learning. 2001. Vol. 45, No. 1. P. 5–32.

24. Bergmeir C., Benítez J. M. On the use of cross-validation for time series predictor evaluation. Information Sciences. 2012. Vol. 191. P. 192–213.

25. He H., Garcia E. A. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*. 2009. Vol. 21, No. 9. P. 1263–1284.

26. Pedregosa F., Varoquaux G., Gramfort A., et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011. Vol. 12. P. 2825–2830.

27. Hilpisch Y. Python for Finance: Mastering Data-Driven Finance. 2nd ed. Sebastopol: O'Reilly Media, 2018. 720 p.

28. Гончарук К. С. Програмний модуль класифікації станів фінансового ринку з використанням методів штучного інтелекту // Інтелектуальні комп'ютерні системи та мережі : матеріали IV Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених, м. Тернопіль, 20 травня 2026 р. Тернопіль : ЗУНУ, 2026. С. 205–207.

29. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

30. Coinglass Cryptocurrency Derivatives Data & Analytics. URL: https://www.coinglass.com (дата звернення: 10.02.2026)

31. CoinMarketCap: Cryptocurrency Prices, Charts & Market Cap. URL: https://coinmarketcap.com (дата звернення: 10.02.2026).

32. Binance Research: Crypto Analysis & Insights. URL: https://research.binance.com (дата звернення: 10.02.2026).

33. Streamlit Official Documentation. URL: https://docs.streamlit.io (дата звернення: 10.02.2026).

34. Imbalanced-learn (SMOTE) Documentation. URL: https://imbalanced-learn.org (дата звернення: 10.02.2026).

35. PostgreSQL Official Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 10.02.2026).

Додаток А

Копія публікації

Гончарук К.С.
Студентка 4 курсу ФКІТ ЗУНУ

ПРОГРАМНИЙ МОДУЛЬ КЛАСИФІКАЦІЇ СТАНІВ ФІНАНСОВОГО РИНКУ З ВИКОРИСТАННЯМ МЕТОДІВ ШТУЧНОГО ІНТЕЛЕКТУ

Вступ. Сучасні фінансові та криптовалютні ринки характеризуються високою динамічністю, значними обсягами даних та високим рівнем невизначеності. Аналіз ринкових процесів у режимі реального часу потребує використання інтелектуальних методів обробки даних, здатних автоматично виявляти закономірності та класифікувати поточний стан ринку. Застосування методів машинного навчання дозволяє підвищити об'єктивність аналізу та забезпечити підтримку прийняття торгових рішень на основі історичних і поточних даних фінансового ринку [1–4].

Постанова задачі. Метою роботи є розробка програмного модуля для автоматизованої класифікації станів фінансового ринку на основі даних криптовалютних бірж із використанням методів штучного інтелекту. Об'єктом дослідження є процес аналізу та класифікації станів фінансового ринку. Предметом дослідження виступають методи машинного навчання, алгоритми аналізу часових рядів та програмні засоби обробки фінансових даних.

Основний матеріал. У рамках дослідження проведено аналіз сучасних підходів до автоматизованого аналізу фінансових ринків та використання методів штучного інтелекту для підтримки прийняття рішень. Особливу увагу приділено концепції Smart Money Concepts, яка дозволяє формалізувати структуру ринку на основі аналізу ліквідності, зон накопичення, маніпуляцій та зламів структури [5].

Для забезпечення автоматизованого аналізу розроблено програмний модуль, архітектура якого включає підсистеми збору ринкових даних, формування простору ознак, класифікації станів ринку та генерації рекомендацій для користувача.

Архітектуру розробленого програмного модуля наведено на рисунку 1. Система реалізована за сервіс-орієнтованим підходом та включає модуль збору ринкових даних через API криптовалютних бірж, ETL-конвеєр обробки даних та обчислювальне ядро класифікації станів ринку на основі оптимізованого ансамблевого алгоритму градієнтного бустингу (XGBoost).

Рисунок 1 – Архітектура програмного модуля класифікації станів фінансового ринку

Як показано на рисунку 1, джерелом даних виступають API криптовалютних бірж, зокрема Binance, Bybit та WhiteBIT. Отримані котирування передаються до ETL-конвеєра, де виконується очищення даних, обробка пропусків та розрахунок показників відповідно до концепції Smart Money Concepts. На основі сформованого простору ознак обчислювальне ядро виконує класифікацію поточного стану ринку за допомогою моделі градієнтного бустингу XGBoost та формує рекомендації щодо торгової стратегії для користувача.

Одним із ключових елементів концепції Smart Money Concepts є цикл AMD (Accumulation–Manipulation–Distribution), який описує типову поведінку великого капіталу на фінансовому ринку. Схематичне представлення даного циклу наведено на рисунку 2.

Рисунок 2 – Цикл Accumulation–Manipulation–Distribution (AMD) у концепції Smart Money Concepts

Цикл AMD використовується для формування частини ознак, що надалі подаються на вхід моделі машинного навчання для класифікації поточного стану ринку. Джерелом даних виступають API криптовалютних бірж Binance, Bybit та WhiteBIT, через які отримуються історичні та поточні ринкові дані. На основі цих даних формується набір ознак, що характеризують поточний стан ринку та включають показники структури тренду, ліквідності, зон цінової неефективності та поведінкових патернів ринку [6].

Для класифікації станів фінансового ринку використано оптимізований ансамблевий алгоритм градієнтного бустингу XGBoost [7]. Модель дозволяє автоматично відносити поточний стан ринку до одного з п'яти класів: накопичення, висхідний тренд, низхідний тренд, маніпуляція або злам структури. На основі результатів класифікації система формує аналітичні рекомендації щодо прийняття торгових рішень.

Програмний модуль реалізовано мовою Python із використанням бібліотек Pandas, NumPy, Scikit-learn, XGBoost, imblearn (для балансування даних) та SQLAlchemy. Для зберігання та обробки даних використано систему керування базами даних PostgreSQL [8]. Архітектура рішення забезпечує можливість автоматизованого збору, аналізу та класифікації фінансових даних у режимі реального часу.

Проведені експериментальні дослідження повністю підтвердили працездатність запропонованого підходу. Зокрема, застосування алгоритму SMOTE для усунення екстремального дисбалансу навчальної вибірки дозволило системі ефективно розпізнавати складні та рідкісні ринкові ситуації (маніпуляції ліквідністю, злами структури). Отримані результати доводять високу точність розробленої моделі та можуть бути використані як надійна основа для побудови сучасних інтелектуальних систем підтримки прийняття рішень у сфері фінансової аналітики.

Висновки. У результаті проведеного дослідження проаналізовано сучасні підходи до аналізу фінансових ринків та застосування методів штучного інтелекту для класифікації ринкових станів в умовах стохастичної невизначеності. Розроблено архітектуру програмного модуля, що забезпечує автоматизований збір, обробку та аналіз високочастотних фінансових даних. Запропоноване рішення поєднує оптимізовані ансамблеві методи машинного навчання, алгоритми подолання дисбалансу даних та концепцію Smart Money Concepts для визначення поточного стану ринку. Розроблений програмний модуль може бути успішно використаний як ядро інтелектуальних систем підтримки прийняття рішень (СППР) та управління ризиками на сучасних криптовалютних біржах.

Список літератури

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

2. Murphy K.P. Machine Learning: A Probabilistic Perspective. Cambridge : MIT Press, 2012. 1067 p.

3. Aggarwal C.C. Neural Networks and Deep Learning. Cham : Springer, 2018. 497 p.
4. Tsay R.S. Analysis of Financial Time Series. 3rd ed. Hoboken : Wiley, 2010. 715 p.
5. Huddleston M.J. The Inner Circle Trader. Market Structure, Liquidity and Smart Money Concepts. 2022. URL: https://innercircletrader.net (дата звернення: 20.05.2026).
6. Binance Developers. Binance API Documentation. URL: https://developers.binance.com/docs
7. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). New York: ACM, 2016. P. 785–794
8. PostgreSQL Global Development Group. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/

Сідлецький Т. А.
Студент 4 курсу ФКІТ ЗУНУ
Науковий керівник к.т.н., доцент Загородня Д.І., кафедра ІОСУ ЗУНУ

ВИЯВЛЕННЯ ТА ВІДСТЕЖЕННЯ РУХОМИХ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ НА ВБУДОВАНІЙ ОБЧИСЛЮВАЛЬНІЙ ПЛАТФОРМІ BEAGLEBONE AI

Вступ. Сучасний розвиток систем комп'ютерного зору нерозривно пов'язаний із впровадженням технологій периферійних обчислень (Edge Computing), які забезпечують обробку даних безпосередньо на пристроях їх збору. Такий підхід є особливо актуальним для систем відеоспостереження, автономної навігації, інтелектуальних транспортних систем та мобільної робототехніки, де необхідно забезпечити мінімальні затримки передачі даних та високу швидкість прийняття рішень. Водночас використання сучасних моделей глибокого навчання на вбудованих платформах супроводжується значними обчислювальними труднощами через обмежені ресурси процесора, пам'яті та енергоспоживання. Тому актуальним є розроблення ефективних алгоритмів і програмних засобів, здатних забезпечити функціонування систем виявлення та відстеження об'єктів у режимі реального часу на гетерогенних обчислювальних платформах [1–4].

Постанова задачі. Метою роботи є розробка програмного модуля виявлення та відстеження рухомих об'єктів на платформі BeagleBone AI-64 із використанням апаратного прискорення та раціонального розподілу обчислювального навантаження між гетерогенними обчислювальними ядрами. Об'єктом дослідження є процес виявлення та відстеження рухомих об'єктів у відеопотоці в режимі реального часу. Предметом дослідження є алгоритми, методи та програмні засоби апаратного прискорення процесів виявлення та відстеження об'єктів на гетерогенній платформі BeagleBone AI-64.

Основний матеріал. У рамках дослідження проведено аналіз сучасних методів детекції та одночасного відстеження множини об'єктів у відеопотоках. Особливу увагу приділено архітектурним особливостям системи-на-кристалі TDA4VM, яка використовується у платформі BeagleBone AI-64 та містить ядра ARM Cortex-A72, цифрові сигнальні процесори DSP архітектури C7x і матричний прискорювач MMA [5].

Для забезпечення високої продуктивності розроблено багатопотокову архітектуру програмного модуля на основі конвеєрного шаблону проєктування (Pipeline Design Pattern). Архітектура складається з чотирьох взаємопов'язаних підсистем: захоплення відеоданих, детекції об'єктів, відстеження та візуалізації результатів. Взаємодія між підсистемами реалізована через потокобезпечні черги даних за принципом «виробник–споживач», що дозволяє ефективно розподіляти навантаження між апаратними компонентами платформи.

Загальну функціональну структуру розробленого програмного модуля наведено на рисунку 1. Система реалізована за конвеєрним принципом та включає етапи ініціалізації,

Додаток Б

Модуль збору ринкових даних з криптовалютної біржі

```
import pandas as pd
import requests
from datetime import datetime
import warnings
warnings.filterwarnings('ignore')

def fetch_crypto_data(symbol="BTCUSDT", interval="15m", limit=1000):
    url = "https://api.binance.com/api/v3/klines"
    params = {
        "symbol": symbol,
        "interval": interval,
        "limit": limit
    }

    response = requests.get(url, params=params)

    if response.status_code != 200:
        raise ValueError(f"Помилка доступу до Binance API:
{response.text}")

    data = response.json()

    df = pd.DataFrame(data, columns=[
        'timestamp', 'open_price', 'high_price', 'low_price',
'close_price',
        'volume', 'close_time', 'quote_asset_volume',
'number_of_trades',
        'taker_buy_base_asset_volume',
'taker_buy_quote_asset_volume', 'ignore'
    ])

    df['timestamp'] = pd.to_datetime(df['timestamp'], unit='ms')
```

```
    numeric_cols = ['open_price', 'high_price', 'low_price',
'close_price', 'volume']
    df[numeric_cols] = df[numeric_cols].apply(pd.to_numeric, axis=1)

    df = df[['timestamp', 'open_price', 'high_price', 'low_price',
'close_price', 'volume']]
    df.sort_values('timestamp', inplace=True)
    df.reset_index(drop=True, inplace=True)

    return df
```

Додаток В

Програмна реалізація конвеєра навчання та крос-валідації

```
import pandas as pd
import numpy as np
import warnings
from sklearn.model_selection import TimeSeriesSplit,
RandomizedSearchCV
from sklearn.metrics import classification_report
from imblearn.over_sampling import SMOTE
from xgboost import XGBClassifier
import joblib

class MarketStateClassifier:
    def __init__(self):
        self.model = XGBClassifier(
            objective='multi:softprob',
            eval_metric='mlogloss',
            random_state=42
        )
        self.best_model = None

    def train_and_evaluate(self, df: pd.DataFrame, feature_cols:
list):
        print("Розподіл вибірки на Train/Test (Збереження
хронології)...")
        X = df[feature_cols]
        y = df['target_class']

        split_idx = int(len(df) * 0.8)
        X_train, X_test = X.iloc[:split_idx], X.iloc[split_idx:]
        y_train, y_test = y.iloc[:split_idx], y.iloc[split_idx:]

        print("Балансування навчальної вибірки за допомогою
SMOTE...")
        smote = SMOTE(random_state=42)
```

```
        X_train_balanced, y_train_balanced =
smote.fit_resample(X_train, y_train)

        tscv = TimeSeriesSplit(n_splits=3)

        param_dist = {
            'n_estimators': [100, 200, 300],
            'max_depth': [3, 5, 7],
            'learning_rate': [0.01, 0.05, 0.1],
            'subsample': [0.8, 1.0],
            'colsample_bytree': [0.8, 1.0]
        }

        print("Пошук оптимальних гіперпараметрів XGBoost...")
        search = RandomizedSearchCV(
            self.model,
            param_distributions=param_dist,
            n_iter=5,
            cv=tscv,
            scoring='f1_macro',
            random_state=42,
            n_jobs=-1
        )

        search.fit(X_train_balanced, y_train_balanced)
        self.best_model = search.best_estimator_
        print(f"Оптимальні параметри: {search.best_params_}")
        predictions = self.best_model.predict(X_test)

        target_names_dict = {
            0: 'Накопичення (Флет)',
            1: 'Зростаючий тренд',
            2: 'Спадаючий тренд',
            3: 'Маніпуляція',
            4: 'Злам структури'
        }
```

```
        unique_classes = sorted(list(set(y_test) |
set(predictions)))
        actual_target_names = [target_names_dict.get(c, f"Class
{c}") for c in unique_classes]

        print("\n--- Результати Тестування на 'Сліпій' вибірці ---")
        print(classification_report(y_test, predictions,
labels=unique_classes, target_names=actual_target_names,
zero_division=0))

        return self.best_model

    def save_model(self, filepath="smc_market_model.joblib"):
        if self.best_model:
            joblib.dump(self.best_model, filepath)
            print(f"Модель успішно збережена: {filepath}")
```

Додаток Г

Програмна реалізація веб-інтерфейсу

```
import streamlit as st
import pandas as pd
import plotly.graph_objects as go
from data_fetcher import fetch_crypto_data
from prediction import load_model, predict_market_state

st.set_page_config( layout="wide")
st.set_page_config(page_title="ШІ Трейдер (Smart Money)",
layout="wide")
st.title("Програмний модуль класифікації станів фінансового ринку з
використанням методів штучного інтелекту")
st.sidebar.header("Параметри аналізу")
symbol = st.sidebar.selectbox("Оберіть торгову пару", ["BTCUSDT",
"ETHUSDT", "SOLUSDT", "BNBUSDT"])
timeframe = st.sidebar.selectbox("Таймфрейм", ["15m", "1h", "4h"])

try:
    df = fetch_crypto_data(symbol=symbol, interval=timeframe,
limit=250)

    try:
        model = joblib.load("smc_market_model.joblib")
        current_state_code = predict_market_state(model, df)
    except Exception:
        st.sidebar.error("Модель .joblib не знайдена! Працює базовий
аналітик.")
        current_state_code = 0

    states = {
        0: {"name": "Накопичення (Флет)", "action": "УТРИМАННЯ.
Очікуйте виходу з діапазону. Не відкривайте позиції в боковику."},
        1: {"name": "Зростаючий тренд (Uptrend)", "action": "ПОШУК
LONG. Відкривайте позиції від зон інтересу (Bullish Order Block)."},
```

```
        2: {"name": "Спадаючий тренд (Downtrend)", "action": "ПОШУК
SHORT. Відкривайте позиції від ведмежого імбалансу (FVG)."},
        3: {"name": "Маніпуляція (Liquidity Sweep)", "action":
"УВАГА: Збір ліквідності! Готуйтеся до розвороту ціни (CHoCH) у
протилежний бік."},
        4: {"name": "Злам структури (BOS/CHoCH)", "action": "АКТИВНА
ФАЗА. Підтверджено зміну тренду. Встановлюйте лімітні ордери."}
    }
    current_state = states.get(current_state_code, states[0])

    tab1, tab2, tab3 = st.tabs(["Аналіз ринку", "Торгова стратегія",
"Внутрішні дані"])

    with tab1:
        st.subheader("Поточний фундаментальний стан та графік")
        col1, col2 = st.columns([1, 2])
        with col1:
            st.info(f"**Класифікований стан ШІ:**\n###
{current_state['name']}")
            current_price = f"{df['close_price'].iloc[-1]:,.2f}
USDT"
            st.metric(label=f"Поточна ціна {symbol}",
value=current_price)
        with col2:
            st.warning(f"**Рішення для трейдера (Алгоритмічний
сигнал):**\n\n{current_state['action']}")

        st.divider()

        df_plot = df.tail(100)
        fig = go.Figure(data=[go.Candlestick(
            x=df_plot['timestamp'], open=df_plot['open_price'],
high=df_plot['high_price'],
            low=df_plot['low_price'], close=df_plot['close_price'],
name="Ціна"
        )])
```

```
        fig.update_layout(height=500, margin=dict(l=0, r=0, t=30,
b=0), xaxis_rangeslider_visible=False)
        st.plotly_chart(fig, use_container_width=True)

    with tab2:
        st.subheader("Обґрунтування фінансової стратегії та Ризик-
менеджмент")

        st.markdown("""
        Аналітичне ядро системи приймає рішення на основі концепції
**Smart Money (SMC)**.
        Алгоритм не використовує запізнілі індикатори, а аналізує
мікроструктуру ринку та ідентифікує сліди інституційного капіталу:

        * **Тренд (Uptrend / Downtrend):** Торгівля ведеться
виключно за напрямком головного руху. ШІ шукає зони інтересу:
Ордерблоки (OB) або імбаланси (FVG).
        * **Накопичення (Флет):** Фаза балансу. Алгоритм блокує
сигнали на вхід, щоб захистити трейдера від хибних пробоїв (запилів)
у боковику.
        * **Маніпуляція (Liquidity Sweep):** Аномальний імпульс для
збору стоп-ордерів натовпу. Система попереджає про маніпуляцію та
можливий розворот.
        * **Злам структури (CHoCH):** Перший математичний індикатор
зміни настроїв маркетмейкера. Сигнал до зміни напрямку торгівлі.
        """)
        st.divider()
        st.subheader("Поточні параметри торгового алгоритму")
        recent_volatility = df['close_price'].pct_change().std() *
100
        avg_volume = df['volume'].tail(20).mean()

        col2_1, col2_2, col2_3 = st.columns(3)

        with col2_1:
            st.metric(label="Рекомендований ризик на угоду",
value="1.5 - 2.0 %", delta="Standard SMC", delta_color="normal")
```

```
        with col2_2:
            st.metric(label="Поточна волатильність",
value=f"{recent_volatility:.2f} %", delta="Адаптивний Stop-Loss",
delta_color="off")

        with col2_3:
            st.metric(label="Середній обсяг (20 барів)",
value=f"{avg_volume:,.0f}", delta="Підтвердження ліквідності",
delta_color="off")

        st.info("**Алгоритмічна порада:** Уважно слідкуйте за
обсягами під час формування Ордерблоків. Справжній злам структури
(BOS/CHoCH) з боку великого капіталу завжди супроводжується
підвищеною торговою активністю.")

    with tab3:
        st.subheader("Внутрішні дані ринку")
        st.dataframe(df.tail(15), use_container_width=True)

except Exception as major_error:
    st.error(f"Критична помилка додатку: {major_error}")
```