ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Автоматизований бот для торгівлі криптовалютою на основі машинного навчання»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
(код,найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Дмитро АНІСІМОВ
(підпис) (ім'я, ПРІЗВИЩЕ здобувача)

Виконав: здобувач вищої освіти група ІСД-42
Дмитро АНІСІМОВ
(ім'я, ПРІЗВИЩЕ)
Керівник Каміла СТОРЧАК
д.т.н.,професор (ім'я, ПРІЗВИЩЕ)
Рецензент: Сергій ПРОКОПОВ
к.т.н.,доцент (ім'я, ПРІЗВИЩЕ)

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедрою ІСТ

_____ Каміла СТОРЧАК

“____” _____2026 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Анісімову Дмитру Олеговичу

(прізвище, ім’я, по батькові здобувача)

1.Тема кваліфікаційної роботи**:** Автоматизований бот для торгівлі криптовалютою на основі машинного навчання. Керівник кваліфікаційної роботи: Каміла СТОРЧАК, доктор тех. наук. Затверджена наказом Державного університету інформаційно-комунікаційних технологій від “20” лютого 2026 р. № 51.

2**.** Строк подання кваліфікаційної роботи**:** “1” червня 2026 р.

3. Вихідні дані кваліфікаційної роботи:

1.Науково-методична література з фінансового машинного навчання (AFML).

2.Історичні тикові дані за криптовалютною парою BTCUSDT, агреговані у бари за грошовим обсягом, та набір зафіксованих артефактів експериментального запуску.

3.Програмна реалізація модульного дослідницького прототипу автоматизованої торгової системи із використанням алгоритмів класифікації (Random Forest) та підсистем управління ризиками.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

1. Дослідження теоретико-методичних засад автоматизованої торгівлі, обґрунтування вибору методів машинного навчання, розмітки подій (схема потрійного бар'єра) та фінансової валідації на криптовалютному ринку.
2. Проєктування архітектури та конструктивна реалізація модульного програмного прототипу.
3. Проведення аналітико-дослідницького оцінювання експериментального сценарію на даних BTCUSDT та аналіз результатів.

5. Перелік ілюстраційного матеріалу: презентація

6. Дата видачі завдання: "____" _____ 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	Добір і аналіз теоретичних джерел за темою роботи	25.01.2026 – 15.02.2026	
2	Аналіз предметної області та формування методики дослідження	16.02.2026 – 10.03.2026	
3	Розроблення експериментального конвеєра машинного навчання та модулів системи	11.03.2026 – 15.04.2026	
4	Проведення експерименту та аналіз результатів	16.04.2026 – 30.04.2026	
5	Оформлення пояснювальної записки та демонстраційних матеріалів	01.05.2026 – 11.05.2026	

Здобувач вищої освіти _____ Дмитро АНІСІМОВ
підпис *Ім'я, ПРІЗВИЩЕ*

Керівник
кваліфікаційної роботи _____ Каміла СТОРЧАК
підпис *Ім'я, ПРІЗВИЩЕ*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 48 стор., 11 рис., 13 табл., 22 джерела.

Мета роботи – розроблення та дослідження автоматизованої системи машинного навчання для формування, фільтрації та історичного оцінювання торгових рішень на криптовалютному ринку.

Об'єкт дослідження – процес автоматизованого прийняття та оцінювання торгових рішень на криптовалютному ринку на основі аналізу історичних даних.

Предмет дослідження – методи подієвого подання ринкових даних, схеми розмітки торгових подій, алгоритми машинного навчання для фільтрації сигналів та процедури історичного оцінювання у дослідницькому прототипі автоматизованої торгової системи.

Короткий зміст роботи: У кваліфікаційній роботі досліджено теоретико-методичні основи автоматизованої торгівлі із застосуванням сучасних підходів фінансового машинного навчання. Спроектовано та реалізовано модульний дослідницький прототип, який включає підсистеми підготовки даних, навчання мета-моделі на основі алгоритму Random Forest та управління ризиками. Проведено експериментальний сценарій на ліквідному інструменті BTCUSDT, у межах якого сформовано 597 розмічених подій та проведено 184 торгові операції в режимі хронологічного оцінювання. Отримані результати продемонстрували позитивну динаміку капіталу в межах зафіксованих артефактів запуску (підсумковий realized PnL склав понад 309 тис. USDT). Здійснено аналіз обмежень системи, оцінено чутливість до порогових значень фільтрації та визначено пріоритетні напрями подальшого розвитку конвеєра даних.

КЛЮЧОВІ СЛОВА**:** КРИПТОВАЛЮТНИЙ РИНОК, ФІНАНСОВЕ МАШИННЕ НАВЧАННЯ (AFML), БАРИ ЗА ГРОШОВИМ ОБСЯГОМ, РОЗМІТКА ЗА СХЕМОЮ ПОТРІЙНОГО БАР'ЄРА, МЕТА-РОЗМІТКА, ІСТОРИЧНЕ ОЦІНЮВАННЯ (BACKTESTING), УПРАВЛІННЯ РИЗИКОМ, ДОСЛІДНИЦЬКИЙ ПРОТОТИП.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AFML	Advances in Financial Machine Learning – Методологічний підхід до застосування алгоритмів машинного навчання у фінансовій сфері, який враховує специфіку ринкової мікроструктури та використаний як теоретична основа дослідження.
API	Application Programming Interface – Прикладний програмний інтерфейс, що забезпечує взаємодію між ядром системи та веб-переглядачем результатів для читання експериментальних артефактів.
BTCUSDT	Торгова пара (Біткоїн до стейблкоїна Tether), що виступає основним високоліквідним криптовалютним активом у експериментальному сценарії.
CLI	Command Line Interface – Інтерфейс командного рядка, який використовується для конфігурації параметрів та ініціації експериментального запуску.
FFD	Fractional Differentiation (фракційне диференціювання) – метод перетворення часових рядів, що дозволяє досягти стаціонарності даних при максимальному збереженні довгострокової пам’яті (сигналу) ряду.
ML	Machine Learning (машинне навчання) – Клас методів штучного інтелекту, що базується на навчанні алгоритмів (зокрема Random Forest) на історичних даних для вирішення завдань фільтрації та класифікації торгових сигналів.
PCA	Principal Component Analysis (метод головних компонент) – метод зниження розмірності даних та усунення мультиколінеарності шляхом трансформації корельованих ознак у набір ортогональних (незалежних) компонентів.

PnL	Profit and Loss – фінансовий показник, що відображає реалізований прибуток або збиток за окремою угодою чи за фіксований період оцінювання.
Purged K-Fold	Спеціалізована схема перехресної валідації для фінансових даних, яка передбачає вилучення (очищення) спостережень, що перетинаються у часі, для запобігання витоку інформації.
R-multiple	Нормалізована метрика оцінки результативності, що відображає відношення отриманого результату (прибутку/збитку) до початково закладеного ризику в угоді.
Бари за грошовим обсягом	Метод подієвого вибіркування ринкових даних, за якого нове спостереження формується після накопичення заданого грошового обсягу транзакцій; дозволяє коректно відображати інтенсивність ринкової активності.
Витік даних	Data Leakage – помилка в побудові моделі, за якої інформація з майбутнього або з тестової вибірки потрапляє в навчальний набір, що призводить до штучно завищених результатів.
Історичне оцінювання	Хронологічне оцінювання – процес імітаційного тестування відібраних торгових сигналів на історичних даних із суворим дотриманням хронологічної послідовності, правил управління ризиками та обмежень капіталу.
Мета-розмітка	Процес навчання вторинної моделі машинного навчання, призначеної для оцінки ймовірності успіху та фільтрації первинних торгових подій (замість прямого прогнозування напрямку ціни).
Нестаціонарність	Non-stationarity – властивість фінансових рядів змінювати свої статистичні характеристики (середнє, дисперсію) з часом, що є основним викликом для стабільності моделей машинного навчання.

Переглядач	Локальний аналітичний інструмент із веб-інтерфейсом, розроблений для інтерактивного візуального аналізу, діагностики та демонстрації завершених експериментальних запусків.
Поріг	Числове значення оцінки моделі, після перевищення якого система ухвалює рішення про прийняття торгової події та ініціацію угоди.
Походження запуску	Набір збережених метаданих (параметри конфігурації, хеші даних, середовище виконання), що забезпечують відтворюваність експерименту та зв'язок зі сформованими артефактами.
Схема потрійного бар’єра	Метод розмітки фінансових подій, що визначає результат потенційної угоди на основі досягнення одного з трьох критеріїв: рівня фіксації прибутку (Take Profit), рівня обмеження збитку (Stop Loss) або вичерпання вертикального часового горизонту.

ВСТУП

Актуальність теми. Останніми роками децентралізовані фінанси та криптовалюти стали невід'ємною частиною глобальної економіки. Особливістю криптовалютного ринку є те, що він працює цілодобово і має дуже високу волатильність. Кожної секунди тут відбуваються тисячі операцій, які генерують великі масиви даних.

Через таку динаміку ручний аналіз графіків та прийняття рішень людиною стають недостатньо ефективними. Трейдерам складно миттєво реагувати на зміни курсу, а емоційний фактор часто заважає об'єктивно оцінювати ризики та системно перевіряти свої торгові ідеї. Тому виникає потреба у створенні автоматизованих систем. Такі програми мають поєднувати сучасні способи обробки даних, алгоритми машинного навчання та надійний ризик-менеджмент.

Сьогодні мало просто знайти один тимчасовий сигнал для входу в угоду. Набагато важливіше побудувати стабільну систему для тестування торгових стратегій. Такий підхід дозволить чітко фіксувати правила, перевіряти їх на історичних даних і відрізняти реальні закономірності від випадкового везіння. Це робить розробку торгової системи чудовим прикладом прикладного дослідження для кваліфікаційної роботи.

Багато сучасних рішень для алгоритмічної торгівлі намагаються просто вгадати майбутню ціну за допомогою стандартних часових графіків. Це не найкращий підхід, оскільки він не враховує реальну активність торгів на ринку і часто дає хибні результати при тестуванні. Наша робота спрямована на вирішення цієї проблеми. Ми пропонуємо розробити прототип, який покаже весь шлях: від збору сирих даних з біржі до детального аналізу результатів торгівлі без спотворення показників.

Мета роботи – дослідження та розробка автоматизованої системи машинного навчання для прийняття, фільтрації та оцінювання торгових рішень на криптовалютному ринку.

Для досягнення мети, у роботі успішно виконано наступні завдання:

- Дослідження теоретичних основ автоматизованої торгівлі та методів обробки ринкових даних;
- Проєктування архітектури прототипу, що включає підготовку даних та алгоритми машинного навчання;
- Реалізація системи тестування на основі історичних даних для торгового інструменту BTCUSDT;
- Аналіз результатів роботи системи, перевірка прибутковості (PnL) та оцінка ризиків;
- Визначення практичної цінності та обмежень розробленої системи для її подальшого покращення.

Об’єкт дослідження – процес автоматизованого прийняття та оцінювання торгових рішень на криптовалютному ринку.

Предмет дослідження – алгоритми машинного навчання, методи подієвої обробки ринкових даних та процедури тестування в межах автоматизованої торгової системи.

Методи дослідження. Під час написання кваліфікаційної роботи були використані методи машинного навчання, методи подієвої обробки даних (агрегація барів за грошовим обсягом), алгоритми мета-розмітки (схема потрійного бар’єра) та методи імітаційного моделювання для перевірки результатів на історичних даних.

Наукова новизна одержаних результатів. У ході дослідження адаптовано підходи фінансового машинного навчання для роботи з криптовалютами. Вперше у межах роботи продемонстровано комплексне використання подієвої обробки даних та мета-розмітки, що дозволяє суттєво підвищити прозорість та обґрунтованість торгових рішень [1, 2].

Практична значущість одержаних результатів. Розроблений дослідницький прототип є готовим інструментом для тестування торгових гіпотез та аналізу історичних даних. У майбутньому його можна розширити та підключити до API біржі для повної автоматизації створення та супроводу угод.

Апробація результатів дослідження. Окремі положення, пов’язані із застосуванням методів штучного інтелекту та машинного навчання, були представлені на:

VII Науково-технічній конференції «Сучасний стан та перспективи розвитку IoT» з темою доповіді: «Штучний інтелект як інструмент підвищення безпеки в IoT-мережах»;

III Всеукраїнській науково-технічній конференції «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу» з темою доповіді: «Штучний інтелект та машинне навчання у побуті і промисловості».

Структура роботи. Кваліфікаційна робота складається з титульного аркуша, завдання, реферату, змісту, переліку умовних позначень і термінів, вступу, трьох розділів, висновків, переліку посилань і додатків.

- У першому розділі досліджено теоретико-методичні основи застосування машинного навчання в автоматизованій торгівлі.
- У другому розділі описано конструктивну реалізацію системи, її модульну архітектуру та інструментарій для відтворення досліджень.
- У третьому розділі наведено результати експериментального дослідження на даних BTCUSDT, їх аналітичне оцінювання та аналіз роботи системи.

1 ТЕОРЕТИКО-МЕТОДИЧНІ ОСНОВИ АВТОМАТИЗОВАНОЇ ТОРГІВЛІ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

У першому розділі досліджуються теоретичні засади функціонування сучасного криптовалютного ринку та методичні підходи до розробки автоматизованих торгових систем. Проводиться аналіз специфіки ринкових даних, методів їх агрегування та принципів мікроструктурного аналізу. Особлива увага приділяється методології фінансового машинного навчання (AFML), зокрема питанням формування стаціонарних ознак та розмітки подій. Такий теоретичний огляд дозволяє закласти фундамент для подальшого проєктування інтелектуальних фільтрів торгових сигналів у межах розроблюваного прототипу.

1.1 Специфіка криптовалютного ринку та аналіз ринкової мікроструктури

Криптовалютний ринок у межах цього дослідження розглядається як специфічне нестаціонарне середовище для розробки та апробації автоматизованих торгових рішень на основі аналізу історичних транзакційних даних [6]. Фундаментальною відмінністю криптовалютних інструментів від традиційних біржових активів є режим безперервної торгівлі (24/7), що виключає поняття торгових сесій та фіксованих пауз у надходженні котирувань [5]. Така безперервність у поєднанні з фрагментацією ліквідності між численними торговими майданчиками створює умови, за яких використання календарного часу як універсальної одиниці аналізу стає малоефективним [3, 4].

У криптовалютному середовищі періоди низької ринкової активності можуть раптово чергуватися з різкими сплесками волатильності, обсягів торгів та кількості угод. Оскільки ціна в історичному ряду є лише підсумковим результатом безлічі короткострокових процесів, для завдань машинного навчання критичним стає розуміння ринкової мікроструктури – механізмів взаємодії учасників торгів, де потік заявок та ліквідність безпосередньо впливають на ціноутворення [4]. Надто

грубе агрегування таких даних призводить до втрати цінної інформації про дисбаланс попиту і пропозиції та інші мікроструктурні характеристики [3].

Теоретичний підхід до вирішення проблеми нестаціонарності базується на концепції «volume clock» (годинник обсягу), яка виходить із постулату про нерівномірне надходження інформації на ринок [1]. Календарна хвилина у період ринкового штилю та хвилина під час високої активності не є еквівалентними за обсягом економічної інформації. Таким чином, інтенсивність транзакцій та обсяг торгів виступають більш релевантними індикаторами ринкових подій, ніж астрономічний час [1, 7].

Класичним інструментом кількісної оцінки такої мікроструктурної динаміки є показник неліквідності Аміхуда, який оцінює ціновий вплив торгового обсягу. Цей показник розраховується за формулою (1.1):

$$\lambda_t = \frac{|R_t|}{V_t \cdot P_t} \tag{1.1}$$

де

λ_t – коефіцієнт неліквідності за період t;

$|R_t|$ – абсолютна доходність активу за період t;

V_t – обсяг торгів у базових одиницях активу;

P_t – ціна активу.

Отже, криптовалютний ринок визначається як складне середовище, де застосування алгоритмів машинного навчання вимагає не лише спеціальної підготовки даних і суворого контролю витоку інформації, а й використання мікроструктурних характеристик як теоретично мотивованого фундаменту для побудови ознак моделі [2].

1.2 Методологія підготовки даних: вибіркування та стаціонарність ознак

Традиційне представлення ринкових даних у вигляді фіксованих часових інтервалів часто створює статистично неоднорідну картину: у спокійні періоди ринок надмірно деталізується, генеруючи надлишкові спостереження, тоді як в активні фази критично важлива динаміка стискається в один часовий бар [1]. Вирішенням цієї проблеми є подієве вибіркування, зокрема формування барів за грошовим обсягом (рис. 1.1). За цього підходу новий бар формується лише після того, як накопичений обсяг угод досягне визначеного порогу [1]. Це повністю узгоджується з концепцією інформаційного часу: у періоди високої ліквідності генерується більше даних для аналізу, а в періоди стагнації – менше. Таким чином, бари за грошовим обсягом доцільно розглядати як одну з найбільш придатних одиниць спостереження для побудови дослідницької системи [7].

Рисунок 1.1 – Порівняння рівномірності надходження інформації при часовому та подієвому вибіркуванні [20]

Для вибору оптимального методу агрегування даних у межах розробки торгового алгоритму було проведено порівняльний аналіз основних типів барів (табл. 1.1).

Таблиця 1.1

Порівняльна характеристика методів формування ринкових барів

Тип бару	Критерій формування	Статистичні властивості	Основні недоліки
Часовий (Time)	Фіксований інтервал часу	Гетероскедастичність, значний рівень шуму	Ігнорує інтенсивність торгів
Тиковий (Tick)	Кількість транзакцій	Наближення до нормального розподілу доходностей	Не враховує обсяг (лотаж) угод
Обсягу (Volume/ Dollar)	Накопичений обсяг активу/валюти	Найвища стаціонарність та стабільність характеристик	Складність обробки сирих даних
Дисбалансу (Imbalance)	Відхилення співвідношення купівель/продажів від очікуваного рівня	Дозволяють виявляти активність інформованих трейдерів та зміну ринкових настроїв	Найвища обчислювальна складність; потребують постійного перерахунку очікуваного дисбалансу

Після формування подієвого ряду ключовим етапом є побудова ознак (features), придатних для алгоритмів класифікації. Безпосереднє використання сирих цінових рівнів з побудованих барів є методологічно помилковим через їхню нестаціонарну природу: статистичні характеристики цін (середнє значення, дисперсія) постійно змінюються в часі [8]. Це створює високий ризик того, що модель замість фундаментальних закономірностей вивчить випадковий ринковий шум, який не повторюватиметься за межами навчальної вибірки.

Теоретична мета проєктування ознак полягає у перетворенні ринкових даних у такий статистичний простір, де мінімізується нестаціонарність, але зберігається максимальна кількість корисної інформації. Складним завданням на цьому етапі є пошук математичного балансу між стаціонарністю ряду та збереженням його довгострокової «пам'яті» [1]. Оскільки класичне цілочисельне диференціювання

(наприклад, перехід до відносних доходностей) хоч і досягає стаціонарності, але безповоротно видаляє значну частину структурних залежностей, одним із методично обґрунтованих рішень у фінансовому машинному навчанні є метод фракційного (дробового) диференціювання фіксованої ширини [1].

Математично оператор фракційного диференціювання $(1 - \mathrm{L})^{\mathrm{D}}$ визначається через розклад у біноміальний ряд згідно з виразом (1.2):

$$(1 - \mathrm{L})^{\mathrm{d}} = \sum_{\mathrm{k}=0}^{\infty} \omega_{\mathrm{k}} \mathrm{L}^{\mathrm{k}} = \sum_{\mathrm{k}=0}^{\infty} \frac{\prod_{\mathrm{i}=0}^{\mathrm{k}-1}(\mathrm{d} - \mathrm{i})}{\mathrm{k}!} (-\mathrm{L})^{\mathrm{k}} \qquad (1.2)$$

де

L – оператор лагу (зворотного зсуву);

d – дійсне число, що визначає порядок диференціювання;

ω_{k} – вагові коефіцієнти ряду;

K – індекс підсумовування (глибина пам’яті ряду).

При цілочисельному значенні d = 1 оператор перетворюється на класичну першу різницю, де ваги швидко обнуляються, що призводить до повної втрати пам'яті ряду. При дробовому $\mathrm{d} \in (0,1)$ ваги ω_{k} спадають поступово (рис. 1.2), дозволяючи досягти стаціонарності при збереженні довгострокових структурних залежностей цін.

Для об’єктивної оцінки результатів трансформації у межах дослідження використовується розширений тест Дікі-Фуллера (ADF). Процес підбору параметра d вважається завершеним, коли нульова гіпотеза про наявність одиничного кореня відхиляється на рівні значущості $\alpha = 0{,}05$ (значення $p < 0{,}05$). Це забезпечує статистичну коректність та надійність багатовимірного масиву вхідних даних для подальшого навчання мета-моделі, гарантуючи, що отриманий ряд ознак є стаціонарним у широкому сенсі, що є критично важливою умовою для запобігання перенавчанню моделі на ринковому шумі [9].

1.3 Методологія формування та маркування торгових подій

У задачах фінансового машинного навчання критично важливим етапом є не лише конструювання стаціонарних ознак, а й математично коректне визначення цільової змінної. Традиційний підхід із фіксованим часовим горизонтом («ціна зросте або впаде через n хвилин») часто виявляється нерелевантним для реальних торгових стратегій [1]. Це зумовлено тим, що він ігнорує траєкторію руху ціни всередині інтервалу: позиція може бути примусово закрита за алгоритмом обмеження збитків задовго до закінчення терміну, або ж ціна може досягти цільового рівня прибутку та розвернутися назад.

Оскільки торговельний алгоритм не повинен здійснювати аналіз на кожному доступному барі (що призвело б до надмірного перенавчання на ринковому шумі), першочерговим завданням стає виявлення інформаційно значущих подій. Для цього у роботі застосовано симетричний фільтр кумулятивної суми.

Цей метод базується на відстеженні накопичувального відхилення поточних доходностей від їхнього очікуваного значення. Фільтр ініціює подію лише тоді, коли абсолютна сума відхилень перевищує заданий поріг h, адаптований до поточної волатильності:

$$S_t = \max(0, S_{t-1} + y_t - E_{t-1}[y_t]) \tag{1.3}$$

де

S_t – значення кумулятивної суми відхилень у момент часу t;

y_t – доходність активу (або зміна ціни) за поточний період;

$E_{t-1}[y_t]$ – очікувана доходність (у фінансових рядах зазвичай приймається рівною 0).

Використання кумулятивної суми дозволяє системі ігнорувати фази ринкової консолідації та фокусуватися виключно на моментах потенційної зміни тренду або сплесків волатильності [1]. Саме ці ідентифіковані моменти стають точками входу для подальшої розмітки за схемою потрійного бар'єра.

Ефективним вирішенням цієї проблеми є метод розмітки за схемою потрійного бар’єра (triple-barrier labeling), який моделює результат події з урахуванням трьох динамічних умов [1]:

1. Верхній бар’єр (Profit-Taking): рівень фіксації прибутку, що розраховується як динамічне відхилення від ціни входу на основі поточної волатильності. Для адаптації до мінливого стану ринку ширина бар'єрів визначається на основі експоненційно зваженого ковзного стандартного відхилення доходностей (щоденної волатильності), що дозволяє автоматично розширювати цільові рівні під час високої турбулентності та звужувати їх у періоди консолідації.
2. Нижній бар’єр (Stop-Loss): рівень обмеження збитків, який забезпечує контроль ризику кожної окремої угоди.
3. Вертикальний бар’єр (Time-Trigger): часове обмеження, яке припиняє спостереження за подією, якщо жоден із цінових рівнів не був досягнутий.

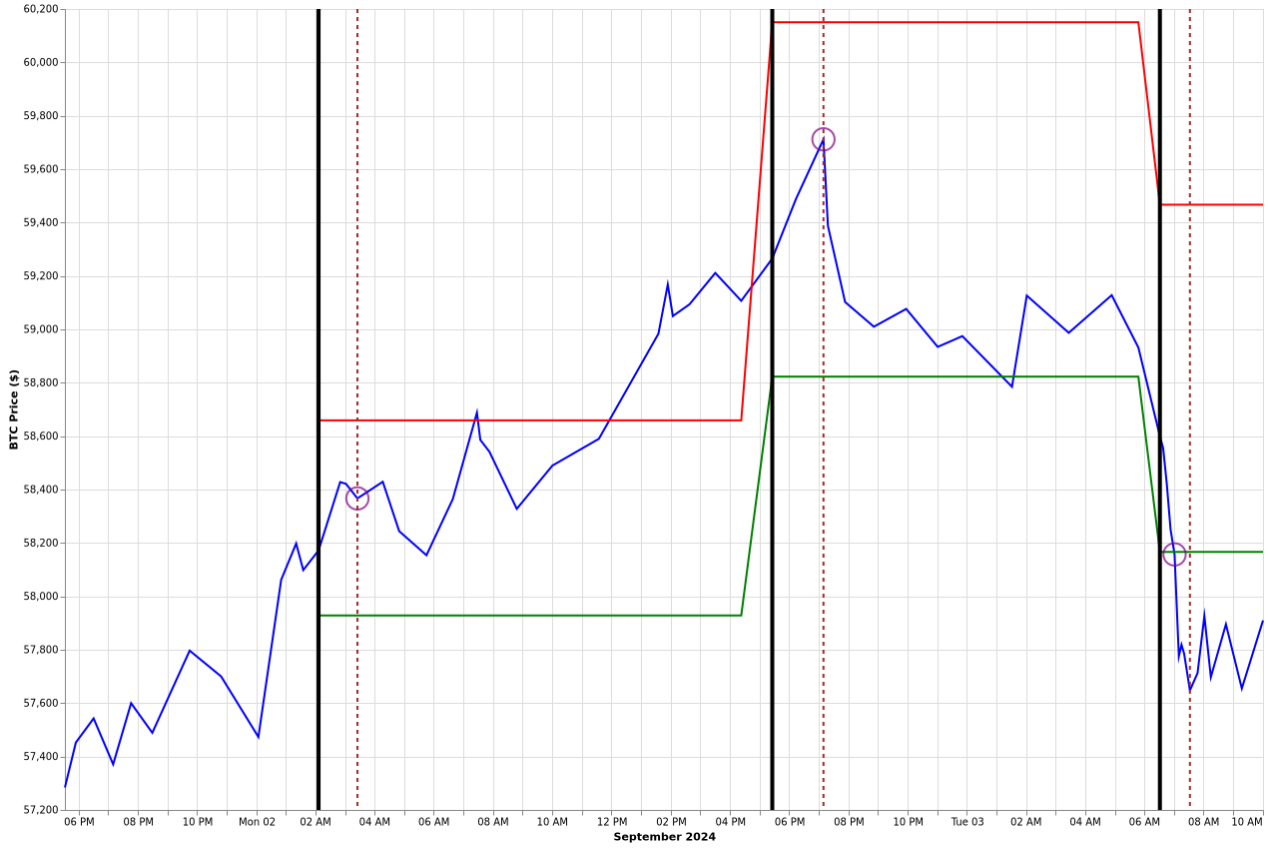


Рисунок 1.2 – Схема методу потрійного бар’єра: спрацювання верхнього (PT), нижнього (SL) або вертикального (T) обмеження [21]

На наведеній схемі (рис. 1.2) візуалізовано процес розмітки даних на прикладі пари BTC/USDT. Чорні вертикальні лінії фіксують момент входу, після чого модель відстежує рух ціни відносно динамічних рівнів (зелена лінія – Profit-Taking, червона лінія – Stop-Loss) та часового ліміту (вертикальна пунктирна лінія). Фіолетове коло вказує на подію, яка першою ініціювала закриття бар'єра, що дозволяє об’єктивно визначити клас для кожного навчального прикладу.

Для обґрунтування вибору даної методики було проведено порівняльний аналіз із традиційними підходами до розмітки цільової змінної, результати якого наведено у таблиці 1.2.

Таблиця 1.2

Порівняльний аналіз методів розмітки цільової змінної

Параметр порівняння	Фіксований час	Фіксований %	Потрійний бар’єр
Умова закриття	Після закінчення T	Досягнення цілі ±$X\%$	PT, SL або Time (що раніше)
Адаптивність	Відсутня	Відсутня	Динамічна (через σ)
Врахування траєкторії	Відсутнє	Наявне	Повне (Path-dependent)
Контроль ризику	Статичний	Статичний	Адаптивний
Статистичний шум	Високий	Середній	Мінімальний
Клас виходу	Binary (Buy/Sell)	Binary (TP/SL)	Triple (1, 0, -1)
Тип стратегії	Класична регресія	Технічний аналіз	Фінансове ML

Такий підхід дозволяє формувати навчальні приклади, що максимально наближені до структури реальної торговельної операції. Завдяки цій схемі кожній події присвоюється об'єктивна мітка успішності, яка стає основою для наступного етапу – мета-розмітки [1, 7].

Мета-розмітка (meta-labeling) – це інноваційна концепція, що дозволяє розділити процес прийняття торгового рішення на два архітектурні рівні: виявлення потенційної можливості та оцінювання ймовірності її успіху [1]. На

першому рівні первинний алгоритм формує «сирий» торговий сигнал (напрям угоди: Long або Short). На другому рівні мета-модель аналізує стан ринку в момент виникнення цього сигналу та прогнозує, чи призведе він до спрацювання позитивного бар’єра (фіксації прибутку).

Цей підхід є доцільним для систем, які генерують велику кількість сигналів із високим рівнем хибних спрацьовувань («шуму») [1]. Мета-модель не намагається лінійно передбачити майбутню ціну, її роль полягає у фільтрації: якщо модель класифікує подію як успішну (мітка 1), сигнал приймається до виконання; якщо ж прогнозується негативний або сумнівний результат (мітка 0), система ухвалює рішення «не торгувати», уникаючи просідань капіталу.

У межах розробленого прототипу для пари BTC/USDT мета-розмітка формалізує задачу як бінарну класифікацію доцільності входу в ринок після врахування транзакційних витрат [7]. Вихідні ймовірності такої моделі надалі використовуються не лише для жорсткої фільтрації сигналів, а й для динамічного калібрування розміру позиції [1, 7].

Формально задача мета-розмітки полягає у навчанні вторинної моделі прогнозувати бінарний результат:

Y = 1: первинний сигнал є вірним (ціна досягла верхнього бар'єра);

Y = 0: первинний сигнал є хибним (спрацював нижній або вертикальний бар’єр).

Таким чином, мета-модель мінімізує кількість хибнопозитивних спрацювань (false positives), що є критичним для стратегій із високою частотою угод.

Слід зауважити, що застосування мета-розмітки часто призводить до виникнення дисбалансу класів у навчальній вибірці. Для вирішення цієї проблеми в межах дослідження передбачено використання методів балансування (наприклад, зважування прикладів), що забезпечує адекватне навчання класифікатора на рідкісних, але важливих позитивних сигналах.

1.4 Валідація фінансових моделей та управління ризиками

На відміну від класичних задач машинного навчання, фінансові спостереження не є незалежними та однаково розподіленими. Ринкові спостереження мають сильні серійні залежності, а торгові події часто перетинаються у часі: одна позиція може залишатися відкритою, коли система вже реєструє наступний вхід [1]. Використання стандартної випадкової крос-валідації в таких умовах призводить до критичної помилки – «витоку даних» (data leakage), коли інформація з майбутнього опосередковано потрапляє в навчальну вибірку через перекриття міток.

Для мінімізації цих ризиків у системі реалізовано адаптовані методи фінансової валідації. Зокрема, використовується метод Purged K-fold cross-validation, за якого з навчальної вибірки вилучаються всі спостереження, часові інтервали яких перетинаються з тестовим періодом [1]. Додатково вводиться буферна зона (Embargo) після тестового інтервалу, щоб розірвати статистичний зв’язок між сусідніми періодами, зумовлений серійною кореляцією ознак [1, 16]. Крім того, для врахування нерівномірної унікальності спостережень застосовується послідовний бутстреп (sequential bootstrap). Оскільки перекривні події несуть частково ідентичну ринкову інформацію, цей метод дозволяє зважувати приклади так, щоб зменшити домінування надмірно схожих паттернів і підвищити здатність моделі до узагальнення [1, 7].

Візуалізація процесу партиціювання даних, що демонструє практичне застосування процедур очищення та ембарго для забезпечення об'єктивності валідації, представлена на рисунку 1.3.

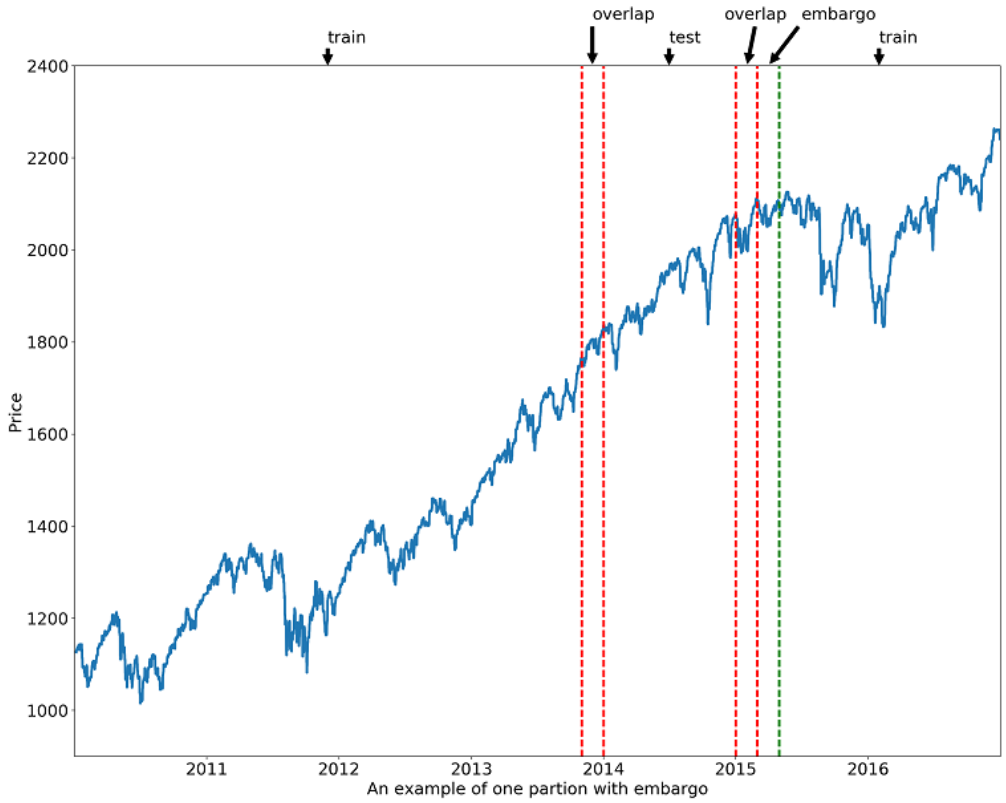


Рисунок 1.3 – Схема партиціювання даних для крос-валідації із застосуванням процедур очищення (Purging, червоні лінії) та ембарго (Embargo, зелена лінія) [17]

Завершальним етапом теоретичного обґрунтування є перехід від метрик класифікації до фінансового оцінювання стратегії. У торговельному моделюванні частка правильних прогнозів (accuracy) не є репрезентативним показником, оскільки один великий збиток може нівелювати серію дрібних прибутків [10]. Результат роботи моделі розглядається не як ізольована статистична метрика, а як невід'ємна частина цілісного процесу, що включає управління капіталом, облік транзакційних комісій та ринкового проковзування (slippage) [11, 12]. Натомість, основний акцент при валідації зміщується на показники, що враховують ризик та розподіл прибутків: коефіцієнт Шарпа (оцінка доходності на одиницю ризику), коефіцієнт Сортіно (оцінка доходності відносно волатильності від'ємних доходів) та максимальна просадка, що визначає стійкість системи до пікових навантажень [10].

Система управління ризиками у розробленому прототипі функціонує на двох ієрархічних рівнях. На рівні окремої угоди розмір позиції визначається динамічно,

виходячи з відстані до алгоритму обмеження збитків та наперед заданого відсотка ризику на угоду. Математично обсяг позиції (S_t) розраховується таким чином, щоб потенційний збиток не перевищував встановлений ліміт ризику, незалежно від волатильності інструменту:

$$S_t = (\text{Equity} * R)/(P_{entry} - P_{sl}) \quad (1.4)$$

де

S_t – обсяг позиції у момент часу t;

Equity – поточний баланс капіталу;

R – фіксований відсоток ризику на одну операцію;

P_{entry} – ціна входу в позицію;

P_{sl} – ціна спрацювання нижнього бар'єра (Stop-Loss).

На портфельному рівні здійснюється контроль сукупної експозиції та обмеження кількості одночасних позицій для запобігання небезпечній концентрації ризику [7].

Окрему увагу приділено проблемі «упередження відбору» (selection bias), коли багаторазове тестування на одних і тих самих історичних даних призводить до підгонки під результат [1]. Саме тому хронологічне оцінювання (backtesting) у даній роботі трактується як дослідницька процедура для перевірки внутрішньої логічної узгодженості системи та динаміки капіталу в умовах реального ринкового хаосу, а не як абсолютна гарантія майбутньої прибутковості алгоритму [1, 7].

Таким чином, сформований у першому розділі теоретичний фундамент – від подієвого вибіркування та фракційного диференціювання до методів захищеної валідації – створює необхідні умови для розробки стійкої торговельної системи. Описана методологія дозволяє перейти до практичної реалізації архітектури алгоритмічного бота та проведення експериментальних досліджень на реальних ринкових даних у наступному розділі роботи.

2 ПРОЄКТУВАННЯ ТА КОНСТРУКТИВНА РЕАЛІЗАЦІЯ СИСТЕМИ МАШИННОГО НАВЧАННЯ

Для забезпечення автономної роботи, високої продуктивності та відтворюваності розрахунків, архітектура системи спирається на сучасний стек технологій аналізу даних. Базовою мовою програмування виступає Python (версії 3.13–3.14), для якого налаштовано ізольоване віртуальне середовище (venv) із фіксацією програмних залежностей.

Обробка багатовимірних масивів інформації в системі концептуально розділена між двома основними бібліотеками: Polars та Pandas. Бібліотека Polars використовується у найбільш навантажених частинах конвеєра: для нормалізації сирих тикових даних, потокового формування подієвих барів та генерації ознак. Використання механізму лінивого сканування (lazy scan) дозволяє ефективно обробляти гігабайти історичних угод без переповнення оперативної пам'яті. Натомість Pandas застосовується на етапах машинного навчання, валідації та бектестингу, де критично важливою є робота зі складними часовими індексами та безшовна інтеграція з екосистемою Scikit-Learn.

Зберігання проміжних результатів та обмін даними між підсистемами реалізовано на базі формату Parquet з використанням бібліотеки PyArrow. Такий підхід забезпечує колонкове зберігання інформації з компресією zstd, що значно пришвидшує операції читання та запису (I/O) під час ітеративних експериментів.

Математичне ядро та модулі машинного навчання побудовані на базі бібліотек Scikit-Learn, NumPy, SciPy та Statsmodels. Зокрема, SciPy використовується для операцій із розрідженими матрицями (csr_matrix) під час послідовного бутстрепу (Sequential Bootstrap), що оптимізує споживання пам'яті при обробці подій, які перекриваються в часі. Перевірка часових рядів на стаціонарність під час фракційного диференціювання здійснюється за допомогою розширеного тесту Дікі-Фуллера (adfuller) з пакета Statsmodels.

У даному розділі описується програмна архітектура, структура компонентів та технічні особливості реалізації розробленого дослідницького прототипу. Окрема

увага приділяється принципам модульності та автоматизації експериментального циклу, що забезпечують відтворюваність отриманих фінансових результатів.

2.1 Загальна архітектура дослідницького прототипу та робочий простір

Архітектура розробленої системи побудована за модульним принципом, що дозволяє розглядати її як гнучке середовище для проведення складних експериментів у сфері алгоритмічної торгівлі. Основною концептуальною ідеєю є жорстке розділення функціональних задач на послідовні етапи: від нормалізації сирих ринкових даних до генерації діагностичних артефактів та візуального аналізу [7].

Центральним елементом системи є Координатор дослідження, який керує потоком даних між підсистемами відповідно до конфігураційного файлу експерименту. Такий підхід забезпечує цілісність конвеєра: параметри барів, обрані моделі ML та правила управління ризиком фіксуються в єдиному контексті запуску [1, 7]. Приклад структури конфігураційного файлу експерименту представлено у Додатку А.

Саме така модульна організація дозволяє розглядати кожен етап дослідження як незалежний процес. Це забезпечує не лише високу відтворюваність результатів, а й можливість швидкої адаптації системи до нових типів ринкових активів без зміни фундаментальних принципів роботи конвеєра. Узагальнену логіку взаємодії підсистем, ієрархію управління та послідовність проходження інформаційних потоків у межах розробленого прототипу наведено на рисунку 2.1.

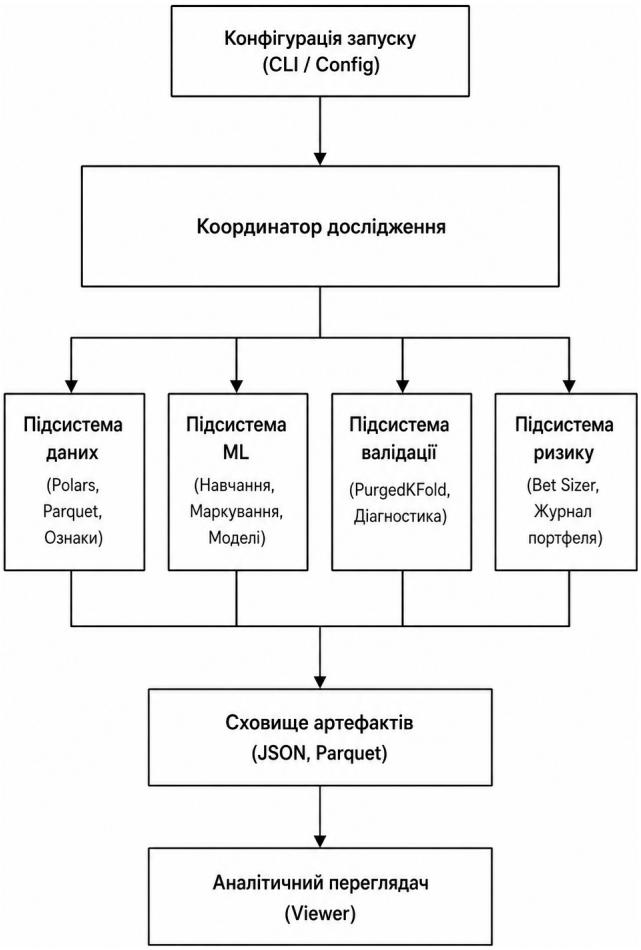


Рисунок 2.1 – Узагальнена архітектура дослідницького прототипу

Розподіл функціональної відповідальності між основними програмними модулями наведено у таблиці 2.1.

Таблиця 2.1

Функціональна відповідальність модулів системи

Компонент	Роль у системі	Вхідні дані	Ключові артефакти
Інтерфейс запуску	Формування умов експерименту	Параметри CLI / Config	config.json.
Координатор	Управління життєвим циклом	Об’єкти підсистем	Підсумкові метрики.
Підсистема даних	Агрегація та побудова ознак	Тикові дані (Parquet)	Таблиці ознак, бари.
Підсистема ML	Маркування та навчання	Навчальна таблиця	Навчена модель, мітки.

Продовження таблиці 2.1

Компонент	Роль у системі	Вхідні дані	Ключові артефакти
Підсистема валідації	Оцінка та діагностика	Результати прогнозів	feature_importance, метрики фолдів.
Підсистема ризику	Розрахунок позицій та PnL	Прогнози моделі	equity_curve, журнал угод.
Переглядач	Аналітична візуалізація	Артефакти запусків	Веб-інтерфейс аналізу.

З інженерної точки зору вихідний код згруповано у доменну структуру. Ядро системи (src/afml_crypto) містить доменні типи (TradeTick, DollarBar, LabeledEvent, MetaPrediction), які імплементовані як об'єкти класів даних (dataclasses) і відповідають за жорстку валідацію цін, обсягів та часових меж.

Функціонально система складається з чотирьох ключових підсистем:

1. Підсистема даних: відповідає за приймання даних (ingestion_worker.py), нормалізацію (tick_loader.py), формування барів та обчислення мікроструктурних ознак (features_factory.py).
2. Підсистема машинного навчання: реалізує логіку маркування (triple_barrier.py) та навчання випадкового лісу (meta_model.py).
3. Підсистема валідації: забезпечує процедури очищення (PurgedKFold) та аналіз важливості ознак.
4. Підсистема управління ризиком: виконує розрахунок розміру позиції (bet_sizing.py) та веде імітаційний торговий журнал (PortfolioLedger).

Управління цими підсистемами здійснюється через ResearchCoordinator. Для режимів із кількома активами (multi-asset) реалізовано дворівневу архітектуру: спочатку для кожного інструменту локально будуються бари, ознаки та мітки, після чого формується зведена панель (meta_panel.parquet), на якій запускається крос-валідація та оцінювання.

Організація робочого простору проєкту спроектована таким чином, щоб забезпечити повний аудит кожного експериментального кроку. Структура

передбачає чітке розділення між статичними джерелами даних та динамічними результатами запусків.

Сховище результатів функціонує як цифровий журнал експериментів. Для кожного запуску автоматично створюється директорія, що містить:

- повну копію конфігурації;
- навчальні таблиці для кожного валідаційного розбиття;
- артефакти моделей та аналізу важливості ознак;
- часові ряди капіталу та деталізовані звіти про угоди.

Така стандартизація робочого простору та автоматичне збереження всіх проміжних артефактів сприяють відтворюваності експериментів і дозволяють ефективно масштабувати систему для подальших досліджень на нових торгових активах.

2.2 Система підготовки ринкових даних та генерації ознак

Підсистема даних є базовим рівнем процесу підготовки даних, що відповідає за перетворення сирих транзакційних записів у структуроване подієве представлення та підготовку математичних ознак. Для забезпечення високої продуктивності завантаження даних реалізовано через модуль TickLoader на базі бібліотеки Polars. Використання механізму lazy scan для форматів Parquet та CSV дозволяє ефективно обробляти мільярди тикових записів без надмірного навантаження на оперативну пам’ять. Система виконує обов'язкову нормалізацію полів (timestamp, price, quantity, side), що робить експериментальний конвеєр незалежним від специфіки форматів різних постачальників даних.

Першим етапом обробки після нормалізації є трансформація сирих тикових записів у подієве представлення. Перехід від астрономічного часу до «інформаційного» реалізується через побудову барів за грошовим обсягом (dollar volume bars) [1]. Такий підхід дозволяє стабілізувати статистичні властивості ряду, оскільки кожне нове спостереження формується лише після накопичення критичної маси ринкової активності. У межах цього дослідження основним

інструментом виступає криптовалютна пара BTCUSDT, що обумовлено її високою ліквідністю та значним обсягом доступних історичних даних.

Для сценарію BTCUSDT було проведено процедуру калібрування порога на основі 60-денного історичного вікна [7]. Цільова частота була встановлена на рівні 48 барів на добу (target_bars_per_day=48), що дозволяє детально фіксувати ринкову динаміку з інтервалом приблизно у 30 хвилин інформаційного часу. За результатами калібрування методом медіанного грошового обсягу було визначено поріг на рівні приблизно 32,63 млн USDT за бар. У підсумковому наборі даних для сценарію зафіксовано 114 624 такі бари. Оптимізацію збереження реалізовано за допомогою бібліотеки PyArrow [14]: дані зберігаються у форматі Parquet із застосуванням zstd-стиснення, що забезпечує високу швидкість доступу [17, 18]. Важливою особливістю алгоритмічної реалізації є ігнорування незавершених останніх барів (include_partial=False), що гарантує цілісність кожного спостереження у навчальній вибірці [19].

Після формування подієвого ряду система ініціює конвеєр генерації ознак (features), які виступають вхідними даними для мета-моделі. Координатор ознак вилучає з кожного бару базові поля: ціни відкриття, закриття, мінімуму та максимуму (OHLC), а також обсяг, середньозважену ціну (VWAP) та кількість тиків. На їх основі обчислюються похідні метрики, зокрема логарифмічна доходність (log_return), спред VWAP та показник номіналу на один тик (notional_per_tick) [13, 14].

Окремий інтерес становить модуль generate_microstructure_features, який обчислює проксі-метрики ліквідності та інформаційної асиметрії:

- Amihud Lambda: показник неліквідності, що відображає вплив обсягу на зміну ціни [14];
- Roll Measure: оцінка прихованого спреду на основі автокореляції змін цін [22];
- VPIN Proxy: наближений показник дисбалансу потоку замовлень на основі синхронізованої за обсягом ймовірності інформованої торгівлі [18];

- Corwin-Schultz Volatility: оцінка волатильності на основі спреду High-Low [15].

Інтеграція ознак ринкової мікроструктури дозволяє мета-моделі враховувати динаміку ліквідності, яка на криптовалютних ринках часто виступає випереджальним індикатором волатильності. Зокрема, використання метрики VPIN дає змогу ідентифікувати періоди «токсичного» потоку замовлень, що передують зміні локального тренду. Це дозволяє враховувати додатковий ринковий контекст, який не відображається безпосередньо в самих цінових рівнях. Фрагмент програмного коду генерації мікроструктурних ознак (модуль features_factory.py) наведено у Додатку Б.

Для забезпечення коректної роботи алгоритмів машинного навчання особлива увага приділяється стаціонарності рядів. У системі реалізовано метод фракційного диференціювання фіксованої ширини (FFD) [1]. За допомогою вбудованого модуля StationarityChecker для кожної ознаки ітераційно підбирається мінімальний порядок диференціювання d, що дозволяє пройти розширений тест Дікі-Фуллера (ADF) при максимальному збереженні довгострокової пам’яті ряду [1, 9]. У процесі пошуку оптимального коефіцієнта алгоритм використовує ітераційний крок 0,05, перевіряючи досягнення критичного значення статистики на рівні значущості 5%.

Завершується підготовка даних етапом стандартизації та PCA-трансформації (за необхідності) в межах кожного навчального вікна, що формує фінальну очищену основу для роботи мета-моделі. Підсумкова структура ознак, використана у запуску, систематизована у таблиці 2.2.

Таблиця 2.2

Групи ознак у навчальній таблиці

Група ознак	Приклади колонок	Джерело	Роль у системі
Базові поля барів	close, vwap, volume	Первинна таблиця	Фундаментальні показники ринкової активності.
FFD-ознаки	ffd_close, ffd_log_close	fractional_diff.py	Стаціонарні часові ряди зі збереженою пам'яттю.
Мікроструктурні проксі	amihud_lambda, vpin_proxy	features_factory.py	Оцінка ліквідності та інформаційної асиметрії.
Характеристики активу	asset_trailing_ volatility	asset_characteristics.py	Контекстуальна волатильність інструменту.
Стан напряму (Side state)	side_state_zscore	Навчальна таблиця	Динамічні характеристики поточного торгового сигналу.

Варто підкреслити, що кожен етап обробки, наведений у таблиці 2.2, реалізований як окремий незалежний компонент модульної архітектури. Це забезпечує гнучкість системи: за необхідності перелік ознак може бути розширений без переписування логіки всього конвеєра. Процес перетворення "сирих" транзакційних даних у фінальний набір стаціонарних ознак є багатокомпонентним ітераційним циклом, де результат попереднього етапу (наприклад, формування бару) є фундаментом для наступного (обчислення мікроструктурних метрик та подальше диференціювання).

Така послідовність операцій дозволяє нівелювати вплив ринкового "шуму" та підготувати дані у форматі, найбільш придатному для навчання градієнтного бустингу або нейронних мереж. Для наочного відображення архітектурної цілісності та взаємозв'язків між окремими модулями трансформації, логічну схему проходження даних від початкового завантаження до формування стаціонарного вектору ознак представлено на рисунку 2.2.

Рисунок 2.2 – Логічна схема трансформації ринкових даних у стаціонарні ознаки

Таким чином, описаний конвеєр генерації ознак забезпечує глибоке перетворення необроблених ринкових даних у структурований набір стаціонарних показників. Використання мікроструктурних проксі-змінних у поєднанні з фракційним диференціюванням дозволяє виділити приховані закономірності ціноутворення. Отриманий масив багатовимірних даних формує надійний аналітичний фундамент і є повністю підготовленим для використання як вхідний вектор при навчанні мета-моделі.

2.3 Підсистема генерації сигналів та машинного навчання

Підсистема машинного навчання реалізована як сукупність програмних модулів, що відповідають за перехід від первинних ознак та результатів розмітки до побудови моделі, яка виконує роль фільтра торгових подій. Архітектурно підсистема машинного навчання не є самостійним сервісом, що дозволяє уникнути зайвих затримок при пакетній обробці великих масивів історичних даних [7]. Система не розв'язує задачу прямого прогнозування майбутнього напряму руху ринку, а оцінює ймовірність того, чи має поточний контекст достатні ознаки для успішної реалізації торгової ідеї.

Ключові етапи роботи підсистеми виконуються у чітко визначеній послідовності:

1. Генерація первинних сигналів та фільтрація

Визначення первинного напряму події (сторона угоди) формується до етапу навчання мета-моделі [1]. Програмну реалізацію цієї функції забезпечує модуль HysteresisPrimarySide, який отримує на вхід впорядковану за часом послідовність цін закриття барів за грошовим обсягом. Для кожного активу розраховуються швидка та повільна експоненційні ковзні середні (EWMA), їхній спред (spread) та EWMA-оцінка волатильності цього спреду [8]. На основі отриманих значень формується z-оцінка (z-score), що керує автоматом станів із гістерезисом. У сценарії для інструменту BTCUSDT зафіксовано такі параметри: fast_span = 20, slow_span = 80, vol_span = 50, entry_z = 1,0 та exit_z = 0,25 [7].

Застосована логіка станів є більш консервативною порівняно зі звичайним порівнянням середніх: стан flat переходить у long, якщо z-оцінка перевищує поріг входу, і в short, якщо вона стає нижчою за відповідний від’ємний поріг. Вихід із позиції відбувається за нижчим порогом, а при виникненні сильного протилежного сигналу допускається прямий реверс позиції [11]. Такий підхід дозволяє суттєво зменшити частоту хибних перемикань. Додатково застосовується фільтр кумулятивних сум (CUSUM)

для формування індексу вибірки на основі цін закриття барів, який згодом поєднується з послідовністю напрямів для виділення подій [1]. У разі відсутності перехідних подій після CUSUM-фільтрації передбачено використання резервної логіки на основі часових штампів. У наборі даних зафіксовано збалансований розподіл напрямів: 301 подія типу long та 296 подій типу short.

2. Впровадження схеми потрійного бар’єра

Після визначення напряму компонент TripleBarrierLabeller застосовує процедуру розмітки до подій, сформованих на подієвих барах. Для кожного випадку система перевіряє досягнення одного з трьох цінових або часових рівнів: фіксації прибутку (profit-taking), обмеження збитку (stop-loss) або вичерпання вертикального часового горизонту [1]. Отриманий результат події виступає основою для побудови цільової змінної мета-моделі [1, 7].

У сценарії BTCUSDT процедура розмітки сформувала 597 подій. Модуль розраховує цільові рівні на основі ціни входу, напряму позиції та поточної волатильності (σ) [7]. Для довгих позицій рівень фіксації прибутку встановлюється вище ціни входу, а для коротких – нижче, з дзеркальною логікою для рівнів обмеження збитку. Процес маркування (label_events_streaming_from_parquet) реалізовано через пакетне зчитування сирих тикових даних, що дозволяє коректно ідентифікувати момент першого досягнення будь-якого з бар’єрів у межах встановленого часового ліміту [7, 18]. На виході формується детальна таблиця подій. У навчальній вибірці розподіл результатів склав: 108 спрацювань за прибутком (pt), 167 – за збитком (sl) та 322 – за часом (time). Фрагмент програмного коду реалізації алгоритму розмітки потрійного бар'єра наведено у Додатку Г.

3. Формування та структура навчальної таблиці

Для коректного навчання моделі попередньо застосовується алгоритм SequentialBootstrap. Він будує матрицю одночасності подій та оцінює їхню середню унікальність (average uniqueness), що надалі використовується для

зважування спостережень під час навчання [1]. Це критично важливо для мінімізації впливу перекривних у часі угод на якість моделі.

Після цього система об'єднує визначені первинні напрями, результати розмітки та мікроструктурні ознаки у єдину навчальну таблицю. Побудова таблиці базується на процедурі «as-of merging» (злиття станом на момент часу), що дозволяє для кожної події використовувати останній доступний стан ознак безпосередньо перед її стартом [13]. Така схема жорстко дотримується хронологічного порядку і запобігає критичній помилці витоку даних (look-ahead bias) [1]. Для кожної події розраховується чиста дохідність з урахуванням транзакційних комісій. Фінальна цільова змінна (meta_label) дорівнює 1 лише у випадку отримання додатної чистої дохідності, в усіх інших ситуаціях вона класифікується як 0 [1, 7].

Підсумкова навчальна таблиця для інструменту BTCUSDT містить 597 рядків та 42 колонки, де зафіксовано 267 позитивних (схвалених) та 330 нульових (відхилених) мета-міток. Структура зведеного набору даних представлена у таблиці 2.3.

Таблиця 2.3

Структура навчальної таблиці

Група колонок	Склад полів	Роль у конвеєрі	Межа інтерпретації
Параметри набору	597 рядків; 42 колонки	Загальний обсяг даних.	Характеристика конкретного запуску.
Ідентифікація та час	event_id, symbol, start_time, end_time	Зв’язок подій з активами та часовими межами.	Необхідні для коректної валідації.
Цінова геометрія	entry_price, exit_price, pt_level, sl_level	Опис рівнів потрійного бар’єра.	Орієнтовні рівні для моделювання.

Продовження таблиці 2.3

Фінансові результати	trigger, realized_return_net, meta_label	Формування цільової змінної (meta_label = 1 при return > 0).	Цільова мітка базується на чистій дохідності.
Контекст напряму	side, side_state_*	Опис первинного сигналу та стану гістерезису.	Визначає вхідні умови моделі.
Ознаки (Features)	ffd_*, amihud_lambda, roll_measure, vpin_proxy	Вхідні дані для навчання мета-моделі.	Статистичні характеристики ринку.

4. Навчання мета-моделі

Фінальним етапом роботи підсистеми є навчання інтелектуального фільтра. Оскільки торговий напрям уже визначено, цільова змінна відображає оцінку доцільності прийняття конкретного сигналу, відокремлюючи потенційно якісні торгові події від хибних кандидатів [1, 7].

Для реалізації цього завдання обрано алгоритм RandomForestClassifier [14]. Даний вибір є методологічно обґрунтованим для фінансових ознак: модель стійка до викидів, не потребує надмірних обсягів даних та дозволяє проводити аналіз важливості ознак (feature importance) [10, 11]. Архітектура підтримує незалежне навчання моделей для довгого (long) та короткого (short) напрямів [7]. У разі недостатньої кількості спостережень застосовується спільна модель (fallback). У BTCUSDT-запуску було використано окремі спеціалізовані моделі. Вихід моделі трактується як числовий бал (score) для порогової фільтрації, що є критичною основою для наступних етапів валідації та управління ризиками [1, 7]. Програмну реалізацію логіки навчання мета-моделі та маркування подій наведено у Додатку В.

2.4 Модулі валідації, управління ризиком та історичного оцінювання

Розділення оцінки класифікаційної якості моделі та її фінансової поведінки є критичною вимогою для об'єктивного аналізу алгоритмічних стратегій. У

розробленій архітектурі за перевірку математичної точності відповідає підсистема фінансової валідації, а за виконання угод – підсистема управління ризиком та історичного оцінювання.

Головним завданням модуля валідації є організація фінансово коректної перевірки моделей та створення умов, що запобігають витоку інформації (data leakage). Основним інструментом виступає процедура Purged K-Fold [1], інженерна реалізація якої включає:

- Очищення (Purging): вилучення з навчальної вибірки подій, що мають часові перетини з тестовим блоком.
- Ембарго (Embargo): встановлення додаткового часового буфера після тестового блоку для запобігання використанню серійно корельованих ознак у наступних навчальних циклах.

Для аналізу структури ознак підсистема використовує метод Mean Decrease Accuracy (MDA) [10]. Оцінювання внеску змінних проводиться як на індивідуальному рівні, так і на рівні логічних груп ознак. Це дозволяє перевірити, чи не є прибутковість моделі наслідком перенавчання на шумових характеристиках. Важливо зазначити, що результати валідації інтерпретуються як заходи з мінімізації ризиків витоку, а не як їх повне гарантоване усунення.

Після етапу порогової фільтрації система перетворює схвалені мета-моделлю сигнали у конкретні параметри торгових позицій. Підсистема ризику формалізує перехід від ймовірнісних оцінок моделі до реального виконання в імітаційному середовищі за допомогою модуля BetSizer [1]. У прототипі реалізовано два методологічні підходи до визначення розміру позиції (position sizing):

Оцінювальний підхід: розмір позиції розраховується як нелінійна функція від оцінки моделі (probability score) з використанням параметрів стиснення (shrink) та дискретизації [1].

Підхід із фіксованим ризиком: розмір позиції залежить від заданої частки капіталу, яку дослідник готовий втратити у конкретній угоді [12].

У сценарії для BTCUSDT активним є спеціалізований режим fixed_risk. Розрахунок обсягу позиції виконується за алгоритмом, що безпосередньо враховує геометрію розмітки потрійного бар’єра [7]:

- Спочатку обчислюється відносна відстань до рівня обмеження збитків: stop_frac = abs(entry_price - sl_level) / entry_price.
- Далі визначається сума ризику в грошовому еквіваленті: risk_dollars = current_equity * risk_per_trade_pct.
- Номінальний розмір позиції розраховується як risk_dollars / stop_frac з подальшим застосуванням обмеження максимального маржинального плеча.

Для експерименту частку ризику на угоду встановлено на рівні 0,01 (1%), а мультиплікатор номінального розміру – 2,0. Якщо вхідні параметри ціни або рівні бар'єрів визначені некоректно, модуль автоматично блокує відкриття позиції, присвоюючи їй нульовий розмір.

Облік результатів здійснюється через модуль хронологічного оцінювання (backtesting) та реєстр PortfolioLedger, який моделює роботу торгового рахунку в реальному часі [7]. Під час імітаційного проходження подій (event-driven simulation) система нормалізує часові мітки та динамічно перевіряє три ключові портфельні обмеження:

- Максимальна кількість одночасних позицій (max_concurrent).
- Частка заблокованого ризику від поточного капіталу (max_locked_risk_pct).
- Валова експозиція (gross exposure) відносно капіталу.

Якщо будь-яке з обмежень порушується, система відхиляє сигнал, фіксуючи причину в діагностичному журналі. Важливою технічною особливістю є пріоритетність обробки подій виходу у випадку їхнього збігу в часі з подіями входу: це запобігає штучному завищенню обсягу задіяного капіталу в симуляції [7]. Журнал портфеля також фіксує транзакційні витрати (fee_bps) та можливе проковзування ціни (slippage_bps), що забезпечує високу точність моделювання. Зведені характеристики механізмів виконання та їх обмеження представлені у таблиці 2.4.

Таблиця 2.4

Визначення розміру позиції, історичне оцінювання та межі інтерпретації

Механізм	Функція контролю	Підтвердження у сценарії	Межі моделювання
Фіксований ризик	Розрахунок номіналу через капітал та відстань до стоп-лоссу.	sizing_mode=fixed_risk; ризик 0,01.	Не гарантує ідентичний ризик у реальних торгах.
Діагностика розміру	Контроль нульових обсягів та некоректних цінових рівнів.	Нульові відхилення (zero=0, invalid_stop=0).	Описує конкретний запуск, а не систему в цілому.
Журнал портфеля	Облік активів, комісій та проковзування ціни.	Реалізація у portfolio.py; артефакти equity_curve.	Не моделює стакани заявок (order book).
Хронологічне оцінювання	Контроль експозиції та послідовності угод.	184 угоди опрацьовано без системних відхилень.	Моделювання після відбору сигналів.
Обмеження системи	Максимальна кількість позицій та валова експозиція.	max_concurrent=2; max_gross_mult=0,84.	Не враховує затримки та часткові виконання.

Слід підкреслити, що дана процедура є перевіркою внутрішньої узгодженості моделі й не враховує складні аспекти біржового виконання, зокрема мікроструктурна черга заявок або динамічна зміна ліквідності [1, 4]. Проте цей етап забезпечує прозорість експерименту та дозволяє аналізувати систему як цілісний дослідницький прототип.

Керування дослідницьким процесом реалізовано через поєднання двох способів запуску: графічного веб-інтерфейсу та CLI-інтерфейсу. Дослідник має змогу ініціювати обчислення та гнучко конфігурувати експеримент як через графічний веб-інтерфейс, так і за допомогою CLI-інтерфейсу (Command Line Interface) на базі модуля argparse. Це дозволяє поєднувати зручність візуального налаштування із можливостями глибокої автоматизації через скрипти командного рядка. Користувач може задавати цільові активи (--assets), сітку порогових значень

(--threshold-grid), режими розрахунку ризику (--sizing-mode), параметри крос-валідації (--embargo-pct) та інші налаштування у зручному форматі

Усі результати виконання конвеєра автоматично ізолюються. Система генерує унікальну директорію для кожного запуску (outputs/<UTC timestamp>), де зберігаються серіалізовані моделі (.pkl), матриці прогнозів, криві капіталу (equity_curve.json) та зведений маніфест артефактів (summary.json). Вхідні тикові дані очікуються у вигляді нормалізованих Parquet-файлів (наприклад, BTCUSDT.parquet) у локальному сховищі, що виключає залежність системи від стабільності мережевого підключення до біржових API під час тестування.

Для пост-експериментального аналізу та інтерпретації результатів розроблено локальний веб-переглядач на базі мікрофреймворку Flask та бібліотеки Plotly (tools/viewer/app.py). Ця підсистема зчитує артефакти з директорії outputs і формує інтерактивну аналітичну панель (Dashboard). Інтерфейс переглядача дозволяє візуалізувати динаміку капіталу, деталізований реєстр закритих позицій, діагностику стійкості порогів та графіки важливості мікроструктурних ознак. Завдяки окремому модулю експорту (export_viewer_data.py), система також здатна накладати маркери реалізованих угод на інтерактивні свічкові графіки ринкового активу, забезпечуючи детальний візуальний контроль за своєчасністю та логікою алгоритмічних сигналів.

3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

У цьому розділі наведено результати експериментальної перевірки розробленої автоматизованої системи алгоритмічної торгівлі. Основна увага приділяється аналізу ефективності застосованих алгоритмів машинного навчання на історичних даних криптовалютної пари BTC/USDT, оцінці стабільності мета-моделі та перевірці відтворюваності проведеного експерименту.

3.1 Конфігурація експерименту та забезпечення відтворюваності

Оркестрація експерименту в розробленій системі побудована навколо декларативного підходу: всі умови дослідження фіксуються у централізованій конфігурації до початку виконання. Це дозволяє уникнути ручного втручання в процес обробки та забезпечує незмінність параметрів на всіх етапах конвеєра [7]. Після ініціалізації координатор експерименту створює ізольовану директорію результатів, де зберігаються всі проміжні стани системи. Такий підхід перетворює інтерфейс запуску на потужний засіб інженерного керування, що дозволяє досліднику фокусуватися на аналізі гіпотез, а не на технічній підтримці скриптів.

Експериментальна частина роботи спирається на зафіксований сценарій для торгової пари BTC/USDT. Вибір цього інструменту є методично обґрунтованим: BTC/USDT є високоліквідним і ринково значущим активом, а наявність репрезентативних історичних даних дозволяє коректно перевірити працездатність усього реалізованого конвеєра машинного навчання [1, 7]. Хоча архітектура системи підтримує режим мультиактивного аналізу (multi-asset), фактичні результати цього експерименту слід інтерпретувати як спеціалізоване тематичне дослідження поведінки моделі на одному високоліквідному активі.

Ключові гіперпараметри та налаштування середовища, застосовані в експерименті, наведено у таблиці 3.1.

Таблиця 3.1

Конфігурація експерименту для пари BTC/USDT

Параметр	Значення	Джерело	Коментар
Актив	BTCUSDT	Підсумковий паспорт	Сценарій охоплює лише один актив.
Джерело даних	data/processed	Конфігурація запуску	Вхідний каталог із підготовленими даними.
Цільова частота барів	48	Метадані калібрування	Цільове значення: 48 барів на день.
Поріг обсягу	32 631 443,44	threshold_metadata.json	Відкалібрований поріг грошового обсягу для BTCUSDT.
Сітка порогів	0,52, 0,54, 0,56, 0,58	Аргументи конфігурації	Сітка значень для налаштування порога фільтрації.
Розбиття	3 / 2	Паспорт запуску	Задано 3 розбиття, фактично оцінено 2 фолди.
Структура мета-моделі	side_specific	Паспорт запуску	Застосовано окремі моделі для Long та Short напрямів.
Політика порогів	asset_side_aware	Паспорт запуску	Поріг налаштовується індивідуально для активу та напряму.
Визначення розміру позиції	fixed_risk	Паспорт запуску	Режим фіксованого ризику на кожну угоду.
Ризик на угоду	0,01 (1%)	Паспорт запуску	Параметр risk-per-trade-pct.
Макс. маржинальне плече	2,0	Паспорт запуску	Верхнє обмеження номінального розміру позиції (max-position-notional-mult).
Обмеження хронологічного оцінювання	max_concurrent=3, locked=0,03, gross=2,0	Аргументи конфігурації	Ризикові обмеження для бектестингу відфільтрованих сигналів.

Зазначені в таблиці 3.1 параметри формують чіткий регламент проведення експерименту. Застосування режиму фіксованого ризику, обмеження

маржинального плеча та використання роздільних мета-моделей для різних торгових напрямів дозволяють надійно контролювати ризики та мінімізувати вплив випадкових ринкових коливань.

Критично важливим аспектом наукового дослідження є його відтворюваність. У розробленій системі вона реалізована через механізм автоматичного збереження узгодженого набору артефактів, що описують кожен крок трансформації даних [1, 7]. Центральним документом кожного запуску є підсумковий паспорт експерименту, який виконує функцію реєстру і пов’язує вхідні параметри, цифрові підписи даних та посилання на створені файли. Перелік та стан ключових артефактів для сценарію наведено у таблиці 3.2.

Таблиця 3.2

Артефакти, походження даних та межі відтворюваності

Артефакт / Показник	Призначення та зміст	Стан у сценарії	Межі інтерпретації
Паспорт запуску	Зведення метрик, конфігурації та походження.	Наявний; містить агреговані звіти.	Є агрегатом результатів.
Навчальна таблиця	База для навчання мета-моделі.	597 рядків; 42 ознаки.	Не є журналом угод.
Важливість ознак	Аналітичні звіти MDA.	Сформовано для всіх груп.	Оціночна характеристика.
Журнал бектесту	Хронологічна послідовність подій.	368 рядків (входи/виходи).	Моделювання стратегії.
Data Signature	Хеш параметрів джерела даних.	Наявний; враховує шлях та розмір.	Не є повним хешем вмісту.

Крім підсумкових показників, розроблена система генерує комплекс діагностичних матеріалів, спрямованих на всебічне виявлення слабких місць у послідовності обробки даних (таблиця 3.3). Основними компонентами є аналіз чутливості до порога та діагностика за фіксованими календарними періодами.

Таблиця 3.3

Діагностичні артефакти та засоби контролю якості

Діагностичний засіб	Джерело / Артефакт	Об'єкт перевірки	Межі інтерпретації
Стійкість порогів	Звіт чутливості порогів	Корисність моделі за сіткою значень.	Демонструє стабільність, не використовується для оптимізації.
Періодичний аналіз	frozen_period_diagnostics	Метрики (PnL, R-multiple) за роками.	Виявляє календарну неоднорідність.
Важливість ознак	feature_importance	Відносний внесок змінних у модель.	Не вказує на прямий причинно-наслідковий зв'язок.
Хронологічна крива	Журнал оцінювання	Послідовність 368 подій у часі.	Моделювання стратегії, а не ринкового виконання.
Аналітичний переглядач	Веб-інтерфейс	Візуалізація артефактів та діагностик.	Інструмент для пост-експериментального аналізу.
Модульні тести	Набір unittest	Коректність обчислень (BetSizer, PnL).	Інженерний контроль, не гарантує відсутність логічних помилок.

Діагностика чутливості дозволяє порівнювати навчальні та тестові криві корисності за сіткою порогів для кожного валідаційного розбиття. Це критично важливо для виявлення випадків, коли робочий поріг обрано на межі заданої сітки або коли спостерігається значна деградація ефективності моделі на позавибірковій основі (out-of-sample) [1].

Окремим модулем реалізовано діагностику за фіксованими часовими інтервалами (frozen-period diagnostics), яка агрегує результати за календарними роками [7]. Для кожного періоду розраховуються такі метрики, як коефіцієнт прийняття угод (acceptance ratio), статистики кратності ризику (R-multiple), частка успішних угод (Win Rate) та показники максимального просідання капіталу

(Maximum Drawdown). Це дозволяє оцінити стабільність алгоритму в різних ринкових фазах.

Одним із ключових елементів інтерпретації моделі є оцінка важливості ознак (рисунок 3.1), що дозволяє визначити, які саме параметри ринкової мікроструктури мали найбільший вплив на формування фільтраційних рішень мета-моделі.

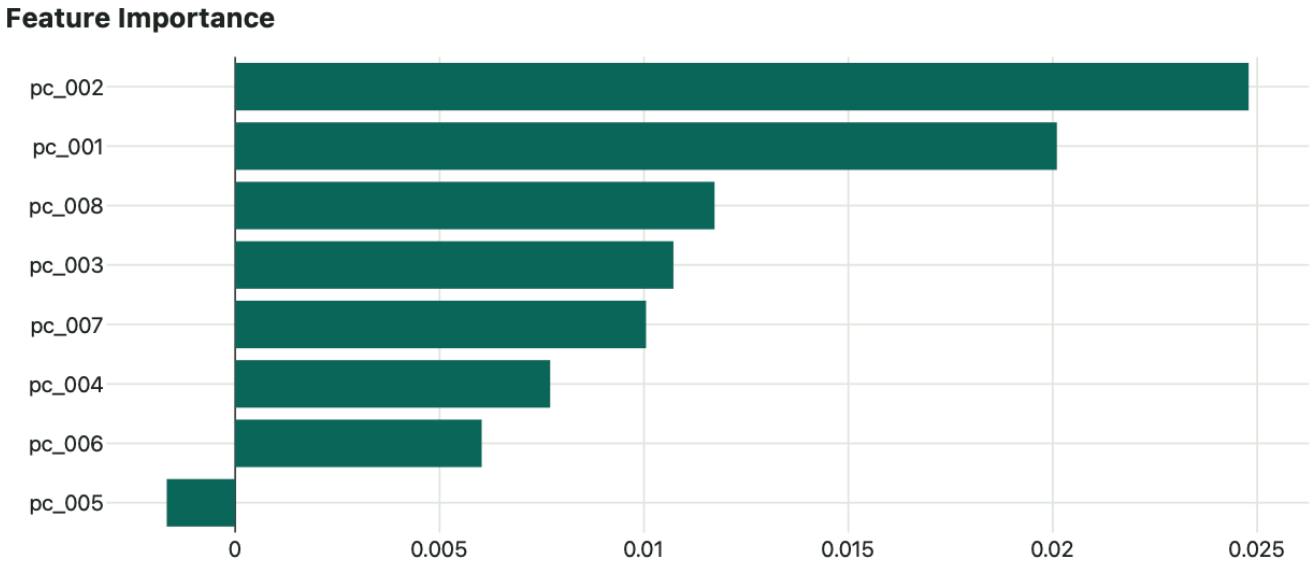


Рисунок 3.1 – Діагностика важливості мікроструктурних ознак

Оскільки на фінальному етапі підготовки даних застосовується PCA-трансформація (метод головних компонент) для усунення мультиколінеарності, ознаки на графіку позначаються як pc_001, pc_002 тощо (від англ. Principal Component – головна компонента). Найбільшу роздільну здатність для мета-моделі демонструють перші компоненти (pc_002 та pc_001), які агрегують у собі ключову дисперсію початкового набору мікроструктурних ознак (таких як волатильність та метрики ліквідності), трансформуючи їх в ортогональні, математично незалежні вектори.

Варто відзначити специфічну поведінку моделі: висока класифікаційна важливість компоненти pc_008 (яка за своєю математичною природою несе малу частку загальної дисперсії) свідчить про те, що алгоритм Random Forest знаходить високу предиктивну силу не в домінуючих ринкових коливаннях, а у специфічних, очищених від макро-шуму мікроструктурних патернах. Це підтверджує

доцільність використання нелінійних моделей, здатних виявляти приховані залежності у низькодисперсійних даних.

Аналіз ієрархії ознак за їхньою роздільною здатністю в алгоритмі RandomForest допомагає переконатися, що модель спирається на економічно обґрунтовані фактори (такі як волатильність, обсяг або мікроструктурні проксі), а не на випадкові шумові характеристики. Наявність такого детального аудиту дозволяє трактувати запуск як прозорий експериментальний об'єкт, готовий до перевірки.

3.2 Аналіз результатів сценарію на парі BTC/USDT

Оцінка ефективності розробленого прототипу базується на результатах імітаційного моделювання, проведеного за суворими правилами фінансової валідації. Для забезпечення достовірності результатів та виключення ефекту «підглядання в майбутнє» (look-ahead bias) у системі реалізовано процедуру Purged K-Fold валідації з механізмом ембарго [1, 7]. Через високу щільність подій та необхідність очищення часових перетинів, у поточному сценарії було фактично оцінено два тестові блоки: другий і третій фолди.

Першим етапом аналізу результатів є перевірка стабільності налаштування порогів фільтрації. Відповідно до обраної політики asset_side_aware, система ітераційно перебирала сітку значень (0,52–0,58) для максимізації очікуваної корисності сигналу на навчальних даних. У підсумку для інструменту BTC/USDT в обох торгових напрямах було обрано поріг 0,52. Зведені параметри валідаційного процесу представлені у таблиці 3.4.

Таблиця 3.4

Параметри валідації та налаштування порогів

Етап	Реалізація та логіка	Підтвердження у сценарії	Обмеження методу
Схема PurgedKFold	Вилучення подій, що перекриваються з тестом; застосування ембарго.	Валідація з очищенням перетинів.	Зменшує ризик витоку даних, але не усуває його повністю.
Хронологічний контроль	Навчання лише на даних, що передують тестовому блоку.	Координація послідовності експерименту.	Залежність від обсягу попередньої історії.
Результати розбиттів	Формування метрик для кожного вікна.	Оцінено 2 розбиття (fold 2 та fold 3).	Кількість оцінених фолдів може бути меншою за задану.
Оптимізація порога	Перебір сітки значень за критерієм корисності.	Обрано поріг 0,52 для всіх груп BTCUSDT.	Тестові дані не використовуються для вибору порога.
Політика налаштування	Гнучкий підбір порогів за активом та напрямом.	Використано режим asset_side_aware.	Потребує достатньої кількості подій у групі.

У ході експерименту на парі BTC/USDT було згенеровано 114 624 бари за грошовим обсягом, що дозволило виділити 597 потенційних торгових подій. Після мета-фільтрації та перевірки на відповідність обмеженням портфеля, в імітаційному середовищі було реалізовано 184 угоди.

За результатами хронологічного оцінювання було зафіксовано підсумковий реалізований прибуток у розмірі 309 985,52 USDT. З урахуванням початкового капіталу в розмірі 1 000 000 USDT, кінцевий стан рахунку склав 1 309 985,52 USDT, що відповідає загальній прибутковості 31% за період дослідження. Динаміку зростання капіталу відображено на рисунку 3.2.

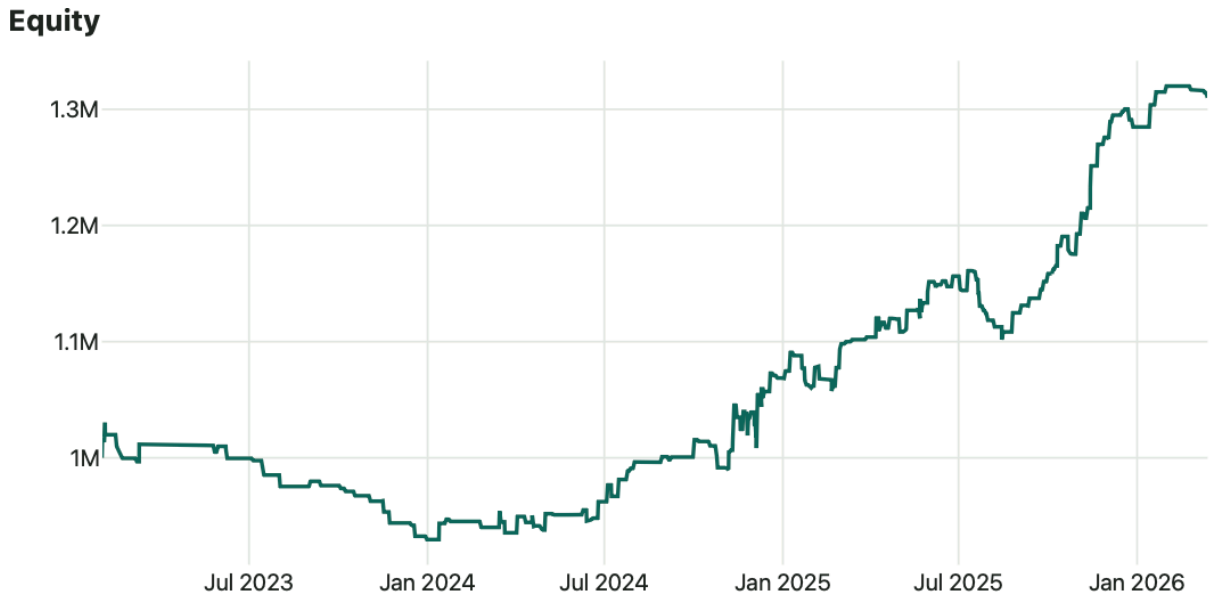


Рисунок 3.2 – Динаміка капіталу в межах сценарію BTC/USDT

Аналіз ризикових метрик підтверджує стабільність роботи системи. Максимальне просідання капіталу склало 100 531,00 USDT, або 10,05% у відносному вираженні (рисунок 3.3). Консервативний характер стратегії підтверджується також показником валової експозиції, яка не перевищувала 0,84, та обмеженою кількістю одночасно відкритих позицій (не більше двох).

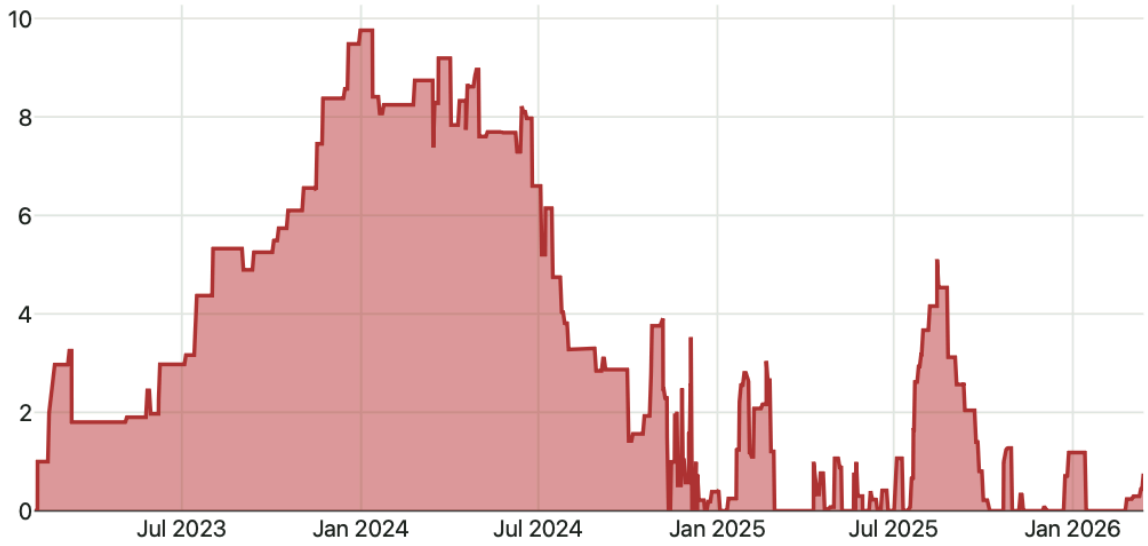


Рисунок 3.3 – Просідання капіталу під час хронологічного оцінювання

Для глибшого розуміння джерел прибутку було проведено аналіз на рівні окремих валідаційних фолдів. Результати тестування на позавибірковій основі (out-of-sample) продемонстрували значну статистичну неоднорідність:

- Fold 2: згенеровано 62 угоди, сумарний результат – 15 594,26 USDT.
- Fold 3: згенеровано 122 угоди, реалізований прибуток – 294 391,27 USDT.

Така диспропорція, де основна частина прибутку сформувалась у третьому фолді, є характерною ознакою нестаціонарності криптовалютного ринку. Це вказує на те, що предиктивна сила мета-моделі суттєво залежить від поточної ринкової фази та рівня волатильності (рисунок 3.4).

Рисунок 3.4 – Розподіл реалізованого PnL за валідаційними періодами

Незважаючи на часову нерівномірність доходів, автоматично встановлений поріг 0,52 виявився стабільним для обох фолдів. Проте діагностичний аналіз стійкості порогів показав, що у третьому фолді для коротких позицій результат міг бути вищим при встановленні порога 0,54. Це підтверджує тезу про те, що оптимальні параметри фільтрації не є константами і потребують регулярної адаптації до динамічного ринкового контексту, що реалізовано в архітектурі розробленого прототипу.

3.3 Оцінка результатів відносно ринкового контексту та візуалізація

Для об'єктивної оцінки ефективності розробленого прототипу необхідно зіставити результати імітаційного моделювання з пасивним ринковим контекстом.

У межах цієї роботи такий аналіз має допоміжний характер і дозволяє візуалізувати поведінку системи на тлі загальних трендів активу BTC/USDT.

Згідно з підготовленими артефактами, ринковий огляд охоплює період з січня 2023 по березень 2026 року. Зафіксована динаміка тижневих свічок за цей період демонструє потужний висхідний тренд: від першої ціни відкриття 22 719,28 USDT до останньої ціни закриття 70 722,61 USDT, що відповідає зростанню ціни активу приблизно на +211,29%.

Водночас, розроблена торгова система зафіксувала прибуток у розмірі +31,0% (309 985,52 USDT). Слід наголосити, що таке зіставлення не слід трактувати як відставання від ринку. Стратегія пасивного утримання (Buy & Hold) передбачає 100% безперервну експозицію на капітал та прийняття повного ринкового ризику. Натомість реалізована система оперує з жорстким обмеженням ризику (fixed_risk = 1%), перебуває на ринку лише під час підтверджених сигналів та утримує максимальну експозицію на рівні 0,84. Це робить отриманий результат математично більш стійким до можливих глибоких корекцій ринку. Деталізація ринкового контексту наведена у таблиці 3.5.

Таблиця 3.5

Ринковий контекст і межі порівняння

Джерело / Рівень	Зафіксовані параметри	Роль у системі	Межі інтерпретації
Макро-рівень (Тижні)	163 тижневі свічки (2023–2026); зростання +211%.	Формує загальний контекст тренду.	Не є формальним еталоном Buy & Hold.
Мікро-рівень (Години)	27 220 погодинних свічок із trade_ticks.	Деталізація контексту в точках входу.	Наявний технологічний пропуск даних.
Підсумок оцінювання	Прибуток +31,0% при 184 відфільтрованих угодах.	Демонстрація масштабу результатів.	Порівняння з ринком потребує поправки на ризик.

Важливим компонентом забезпечення прозорості експериментів є розроблена візуально-аналітична підсистема (переглядач результатів). Це локальне веб-

середовище на базі Flask та Plotly, що дозволяє проводити інтерактивний аналіз сформованих артефактів без перегляду сирих логів. Інтерфейс головної аналітичної панелі розробленого додатка наведено на рисунку 3.5.

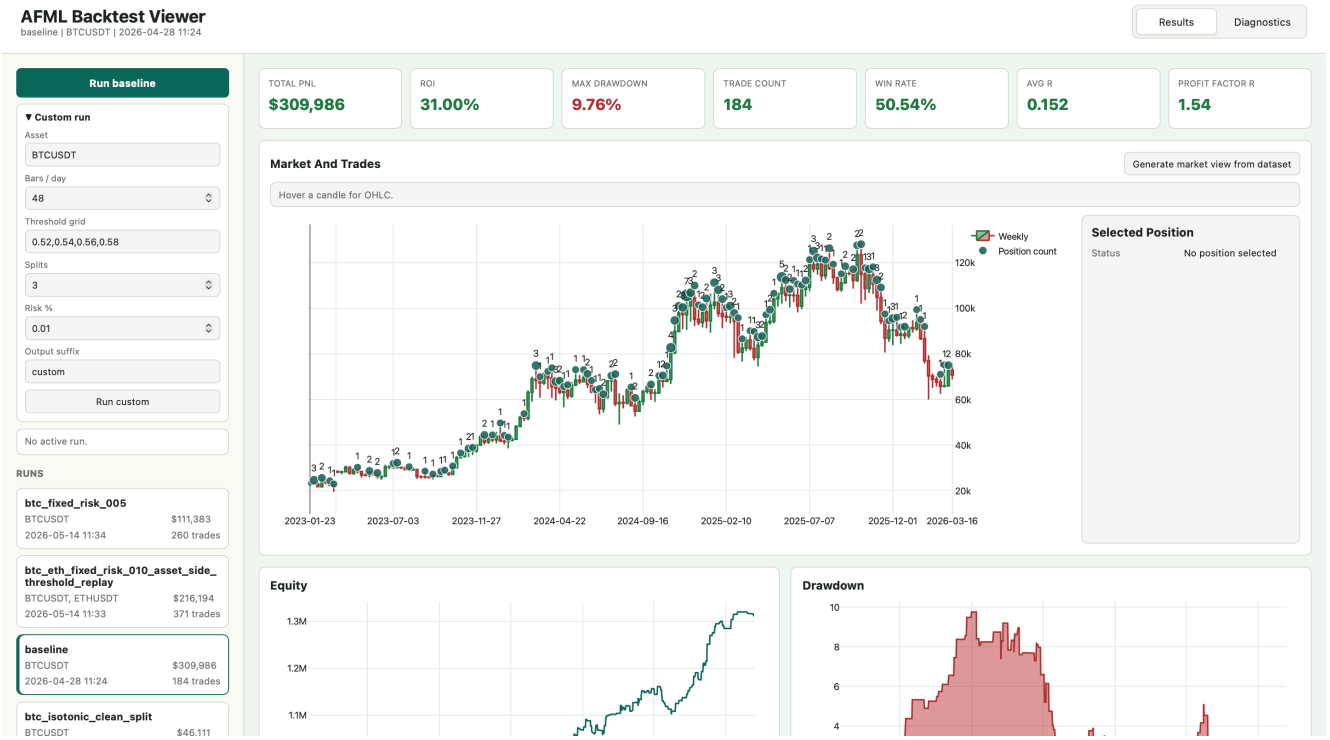


Рисунок 3.5 – Головна аналітична панель веб-застосунку

Функціональність переглядача включає:

- Dashboard-огляд: швидка оцінка фінального капіталу, PnL та графіка просадок (Equity/Drawdown).
- Інтерактивний ринковий графік: візуалізація свічок OHLC з накладеними маркерами точок входу та виходу сигналів моделі.
- Деталізація угод: інтерактивна таблиця позицій, що автоматично реконструюється з журналів бектесту.
- Діагностична вкладка: виведення звітів щодо стійкості порогів та графіків важливості ознак. Детальний вигляд діагностичної панелі зображено на рисунку 3.6.

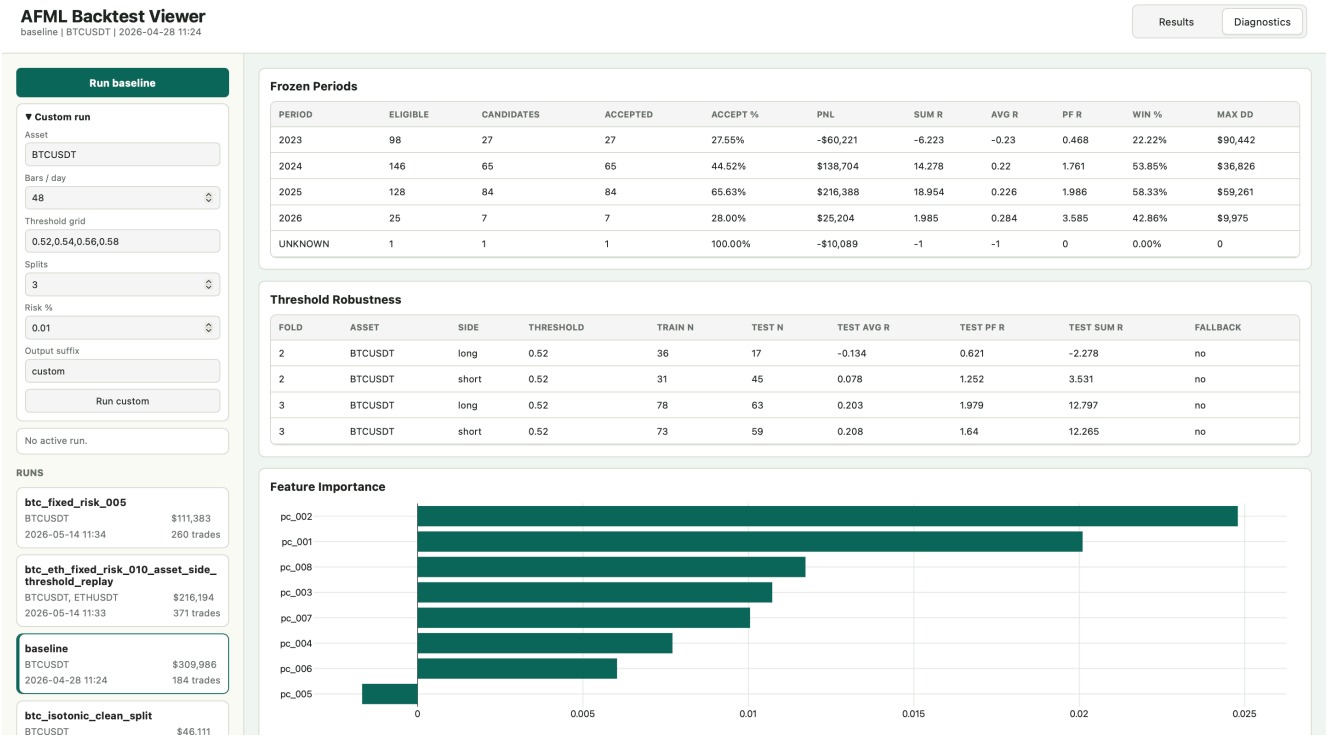


Рисунок 3.6 – Діагностична панель: аналіз мікроструктурних ознак

Зведені характеристики підсистеми візуалізації наведено у таблиці 3.6.

Таблиця 3.6

Реалізована функціональність переглядача результатів

Компонент	Підтверджена поведінка	Межі застосування
Список запусків	Парсинг метаданих та виведення деталей експерименту.	Інструмент читання готових артефактів (Read-only).
Динаміка капіталу	Побудова графіків Equity, PnL та Drawdown.	Візуалізація математичного моделювання, а не реальних торгів.
Журнал угод	Формування реєстру виконаних позицій.	Дані відновлюються з журналу бектесту.
Ринковий графік	Свічковий аналіз із маркерами відкриття/закриття позицій.	Візуальний контроль своєчасності сигналів моделі.
Вкладка діагностик	Зведення тестів на стабільність та оцінки важливості ознак.	Важливість ознак не доводить причинно-наслідковий зв'язок.

Реалізація такого інтерфейсу перетворює розроблену систему зі звичайного набору скриптів на повноцінне аналітичне середовище. Це дозволяє досліднику

виконувати аналіз – від загальної кривої капіталу до конкретної хвилинної свічки, на якій мета-модель прийняла рішення про вхід у позицію.

3.4 Обмеження реалізованої системи та перспективи розвитку

Проведений експеримент є важливим етапом перевірки дослідницького прототипу, проте він має низку методологічних та технологічних обмежень. Для коректної інтерпретації отриманих результатів та визначення векторів подальшого розвитку системи необхідно чітко окреслити межі поточної реалізації.

Головним обмеженням є зовнішня валідність результатів. Хоча архітектура системи підтримує мультиактивний режим (multi-asset), емпірична перевірка була виконана переважно на інструменті BTC/USDT. Отже, зафіксовані показники прибутковості не можна автоматично екстраполювати на широкий спектр криптовалютних активів без додаткових тестів.

Крім того, результати свідчать про режимну нестаціонарність: ефективність мета-моделі суттєво залежить від ринкової фази (волатильності та ліквідності), що підтверджується нерівномірним внеском різних валідаційних фолдів у підсумковий PnL.

Зведену інформацію про виявлені обмеження та стратегічні напрями вдосконалення системи наведено у таблиці 3.7.

Таблиця 3.7

Комплексний аналіз обмежень та напрями розвитку системи

Обмеження	Точніше формулювання та наслідки	Напрям подальшого розвитку
Обмежена зовнішня валідність	Емпірична перевірка виконана на парі BTC/USDT; результати потребують підтвердження на активах із меншою ліквідністю.	Проведення серії експериментів на широкому кошику інструментів (ETH, SOL, BNB) для перевірки універсальності ознак.

Продовження таблиці 3.7

Спрощена модель виконання	Ураховано комісії та проковзування (fee/slippage) у параметризованій формі, але не моделюються order book depth, затримки (latency) та market impact.	Заміна константної моделі на повноцінний симулятор виконання з урахуванням мікроструктури стакана та черги заявок.
Якість тикових даних	Результати критично залежать від повноти, відсутності дублікатів та точності часових міток (timestamps) у вхідних Parquet-джерелах.	Впровадження модулів автоматичного очищення аномалій та детекції пропусків у потокових даних.
Ризик перенавчання (Selection Bias)	Вибір ознак та налаштування порогів проводилися на історичних даних; залишається ризик специфічної адаптації під конкретний період.	Впровадження вкладеної часової крос-валідації (nested walk-forward) та валідація на незалежному «замороженому» періоді.
Обмежений простір ознак	Ознаки побудовані переважно на базі ринкової мікроструктури; ігноруються зовнішні фактори (funding rates, sentiment, open interest).	Інтеграція альтернативних джерел даних та макроекономічних індикаторів ринкового режиму.
Спрощене управління капіталом	Реалізовано режим fixed-risk, проте відсутня повноцінна портфельна оптимізація та адаптивний контроль просадки.	Розробка системи динамічного управління ризиком на основі волатильності портфеля та оцінок впевненості моделі.
Відсутність Live-валідації	Система демонструє історичну дослідницьку придатність, але не проходила тестування в реальному часі через API.	Перехід до paper-trading інфраструктури (через CCXT) для перевірки роботи системи в режимі реального часу.

Таким чином, розроблений прототип є цілісним інженерним середовищем для кількісних досліджень. Визначені обмеження не знецінюють отримані результати, а слугують дорожньою картою для трансформації дослідницької системи у повноцінний програмний комплекс промислового рівня.

ВИСНОВКИ

У дипломній роботі спроєктовано, реалізовано та досліджено автоматизовану систему машинного навчання для формування, фільтрації й оцінювання торгових рішень на ринку криптовалют. Поставлену мету роботи досягнуто шляхом створення цілісного дослідницького прототипу, що забезпечує прозорий та відтворюваний експериментальний цикл – від обробки сирих тикових даних до візуального аналізу результатів.

Ключові результати та інженерні досягнення роботи:

- Розроблено архітектуру конвеєра, яка базується на агрегації барів за грошовим обсягом. Це дозволило стабілізувати статистичні характеристики вхідних даних та перейти від хронологічного часу до об’єктивної ринкової активності.
- Реалізовано систему мета-розмітки за схемою потрійного бар’єра, що дозволило пов’язати навчання моделі з реальними параметрами ризику (Profit-Taking та Stop-Loss) замість наївного прогнозування напряму ціни.
- Впроваджено методику фінансової валідації Purged K-Fold, яка з урахуванням часового ембарго та очищення перетинів дозволила мінімізувати ризики витоку інформації з майбутнього та забезпечити об’єктивність позавибіркового тестування.
- Створено підсистему управління капіталом на основі фіксованого ризику, що забезпечило суворе дотримання маржинальних лімітів та стабільність портфеля під час історичного оцінювання.

Перевірка та аналіз результатів:

Апробація системи на сценарії BTCUSDT підтвердила працездатність розробленого інструментарію. У межах зафіксованого історичного проміжку система опрацювала 114 624 бари та згенерувала 597 подій, з яких після мета-фільтрації було реалізовано 184 угоди.

Фінансовий результат: підсумковий реалізований прибуток склав 309 985,52 USDT на умовний початковий капітал у 1 000 000 USDT (загальна дохідність 31,0%).

Ризиковий профіль: максимальне відносне просідання капіталу не перевищило 10,05%, що свідчить про високу ефективність алгоритмів фільтрації та ризик-менеджменту.

Зіставлення з ринковим контекстом BTCUSDT (зростання ціни на +211,29%) показало, що система здатна генерувати позитивний результат в умовах обмеженого ризику та низької експозиції (max gross exposure = 0,84), проте пряме порівняння з пасивною стратегією Buy & Hold потребує додаткового дослідження з поправкою на ризик (через коефіцієнти Шарпа або Сортіно).

Напрями подальшого розвитку системи включають:

1. масштабування експериментів на мультиактивні портфелі та різні часові періоди;
2. впровадження алгоритмів калібрації виходів класифікатора для точнішого визначення розміру позицій;
3. інтеграцію складніших моделей симуляції ринкового виконання з урахуванням ліквідності та затримок exchange API.

Загалом, розроблений дослідницький прототип є завершеним інженерним об’єктом, що створює надійну основу для подальших контрольованих експериментів у сфері інтелектуальних торгових систем.

ПЕРЕЛІК ПОСИЛАНЬ

1. López de Prado M. Advances in Financial Machine Learning. Hoboken, New Jersey: John Wiley & Sons, Inc., 2018. 400 p.
2. López de Prado M. Machine Learning for Asset Managers. Cambridge: Cambridge University Press, 2020. 152 p.
3. Lehalle C.-A., Laruelle S. Market Microstructure in Practice. 2nd Edition. World Scientific Publishing Company, 2018. 368 p.
4. Bouchaud J.-P., Bonart J., Donier J., Gould I. Trades, Quotes and Prices: Financial Markets Under the Microscope. Cambridge University Press, 2018. 444 p.
5. Kyriazis N. A. A Survey on Volatility Fluctuations in the Decentralized Cryptocurrency Financial Assets. Journal of Risk and Financial Management. 2021. Vol. 14, No. 7. Article 293. DOI: 10.3390/jrfm14070293.
6. Sezer O. B., Gudelek M. U., Ozbayoglu A. M. Financial time series forecasting with deep learning: A systematic literature review (2005–2019). Applied Soft Computing. 2020. Vol. 90. P. 106181.
7. Jansen S. Machine Learning for Algorithmic Trading: Predictive models to extract signals from market and alternative data for systematic trading strategies with Python. 2nd Edition. Packt Publishing, 2020. 820 p.
8. Hyndman R. J., Athanasopoulos G. Forecasting: Principles and Practice. 3rd Edition. OTexts, 2021.
9. Dixon M. F., Halperin I., Bilokon P. Machine Learning in Finance: From Theory to Practice. Springer, 2020. 566 p.
10. Molnar C. Interpretable Machine Learning: A Guide for Making Black Box Models Explainable. 2nd Edition. 2022. 320 p.
11. James G., Witten D., Hastie T., Tibshirani R., Taylor J. An Introduction to Statistical Learning: with Applications in Python. Springer, 2023. 607 p.
12. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 3rd Edition. O'Reilly Media, 2022. 864 p.
13. Chen D. Y. Pandas for Everyone: Python Data Analysis. 2nd Edition. Addison-Wesley Professional, 2022. 512 p.

14. Pedregosa F. et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011. Vol. 12. P. 2825–2830. DOI: 10.5555/1953048.2078195.
15. Plotly Python Open Source Graphing Library. Plotly Technologies Inc. 2024. URL: https://plotly.com/python/ (дата звернення: 12.05.2026).
16. Amihud Y. Illiquidity and stock returns: cross-section and time-series effects. Journal of Financial Markets. 2002. Vol. 5, No. 1. P. 31–56.
17. Roll R. A Simple Implicit Measure of the Effective Bid-Ask Spread in an Efficient Market. The Journal of Finance. 1984. Vol. 39, No. 4. P. 1127–1139.
18. Easley D., López de Prado M. M., O'Hara M. Flow Toxicity and Liquidity in a High-Frequency World. The Review of Financial Studies. 2012. Vol. 25, No. 5. P. 1457–1493.
19. Corwin S. A., Schultz P. A Simple Way to Estimate Bid-Ask Spreads from Daily High and Low Prices. The Journal of Finance. 2012. Vol. 67, No. 2. P. 719–760.
20. Sefidian S. M. Introduction to Advanced Candlesticks in Finance: Tick Bars, Dollar Bars, Volume Bars, and Imbalance Bars. Sefidian.com. 2021. URL: https://www.sefidian.com/2021/06/12/introduction-to-advanced-candlesticks-in-finance-tick-bars-dollar-bars-volume-bars-and-imbalance-bars/ (дата звернення: 12.05.2026).
21. Yairoz. The Triple Barrier Method: Labeling financial time series for ML in Elixir. Medium. 2019. URL: https://medium.com/@yairoz/the-triple-barrier-method-labeling-financial-time-series-for-ml-in-elixir-e539301b90d6 (дата звернення: 12.05.2026).
22. Velázquez Bustamante A. K-fold Cross-validation with Purging and Embargo: The Ultimate Cross-validation Technique for Time Series. Medium. 2024. URL: https://antonio-velazquez-bustamante.medium.com/kfold-cross-validation-with-purging-and-embargo-the-ultimate-cross-validation-technique-for-time-2d656ea6f476 (дата звернення: 13.05.2026).

ДОДАТОК А

```
def run_experiment(self, raw_data_path: str | Path) -> ResearchExperimentResult:
    self._validate_ticks_source(raw_data_path)
    self._purge_cached_event_artifacts(raw_data_path)
    self._trace_allocations("RunExperiment:Start")

    data_phase = self._run_data_phase(
        raw_data_path,
        symbol=self._extract_symbol_from_source(raw_data_path),
    )
    n_dollar_bars = int(data_phase["n_dollar_bars"])
    gc.collect()
    self._trace_allocations("RunExperiment:AfterDataPhase")

    label_phase = self._run_labeling_phase(
        ticks_source=raw_data_path,
        bars_path=data_phase["bars_path"],
        symbol=self._extract_symbol_from_source(raw_data_path),
    )
    gc.collect()
    self._trace_allocations("RunExperiment:AfterLabelingPhase")

    meta_dataset = self._build_meta_dataset(
        labels_source=label_phase["labels_path"],
        features_source=data_phase["ffd_features_path"],
    )
    gc.collect()
    self._trace_allocations("RunExperiment:AfterMetaDataset")

    cv_phase = self._run_training_validation_backtest_phase(
        meta_dataset=meta_dataset,
        ffd_feature_columns=data_phase["ffd_feature_columns"],
        asset_feature_columns=data_phase["asset_feature_columns"],
        dollar_bar_timestamps=data_phase["dollar_bar_timestamps"],
    )
    gc.collect()
    self._trace_allocations("RunExperiment:AfterTrainingValidation")

    analysis_phase = self._run_analysis_phase(
        meta_dataset=meta_dataset,
        ffd_feature_columns=data_phase["ffd_feature_columns"],
        asset_feature_columns=data_phase["asset_feature_columns"],
        dollar_bar_timestamps=data_phase["dollar_bar_timestamps"],
        tuned_thresholds=cv_phase["tuned_thresholds"],
        oof_feature_matrix=cv_phase["oof_feature_matrix"],
    )
    gc.collect()
    self._trace_allocations("RunExperiment:AfterAnalysis")
```

ДОДАТОК Б

```
out = (
    df.with_columns(
        [
            pl.col("symbol").cast(pl.Utf8,
strict=False).str.strip_chars().str.to_uppercase(),
            pl.col("timestamp").cast(pl.Datetime("ns", "UTC"), strict=False),
            pl.col("close").cast(pl.Float64, strict=False),
            pl.col("high").cast(pl.Float64, strict=False),
            pl.col("low").cast(pl.Float64, strict=False),
            pl.col("volume").cast(pl.Float64, strict=False),
        ]
    )
    .sort(["symbol", "timestamp"])
    .with_columns(
        [
            pl.when(pl.col("close") > 0.0)
            .then(pl.col("close").log().diff().over("symbol"))
            .otherwise(None)
            .alias("_log_return"),
            pl.col("close").diff().over("symbol").alias("_dp"),
            pl.when((pl.col("high") > 0.0) & (pl.col("low") > 0.0))
            .then((pl.col("high") / pl.col("low")).log())
            .otherwise(None)
            .alias("_log_hl"),
        ]
    )
    .with_columns(
        [
            pl.col("_dp").shift(1).over("symbol").alias("_dp_lag1"),
            pl.when(pl.col("_dp") > 0.0)
            .then(pl.lit(1.0))
            .when(pl.col("_dp") < 0.0)
            .then(pl.lit(-1.0))
            .otherwise(pl.lit(0.0))
            .alias("_sign_dp"),
        ]
    )
    .with_columns(
        [
            (pl.col("_dp") * pl.col("_dp_lag1")).alias("_dp_x_dp_lag1"),
            (pl.col("volume") * pl.col("_sign_dp")).alias("_signed_volume"),
        ]
    )
    .with_columns(
        [
            pl.when(pl.col("volume") > 0.0)
            .then(pl.col("_log_return").abs() / pl.col("volume"))
            .otherwise(None)
            .alias("amihud_lambda"),
```

```
pl.col("_dp_x_dp_lag1").rolling_mean(window_size=20).over("symbol").alias("_e_xy"),

pl.col("_signed_volume").abs().rolling_sum(window_size=10).over("symbol").alias("_abs_sign
ed_vol_10"),

pl.col("volume").rolling_sum(window_size=10).over("symbol").alias("_volume_10"),

pl.col("_log_hl").rolling_std(window_size=20).over("symbol").alias("corwin_schultz_vol"),
        ]
    )
    .with_columns(
        [
            pl.when(pl.col("_volume_10") > 0.0)
            .then(pl.col("_abs_signed_vol_10") / pl.col("_volume_10"))
            .otherwise(None)
            .alias("vpin_proxy"),
        ]
    )
)
```

ДОДАТОК В

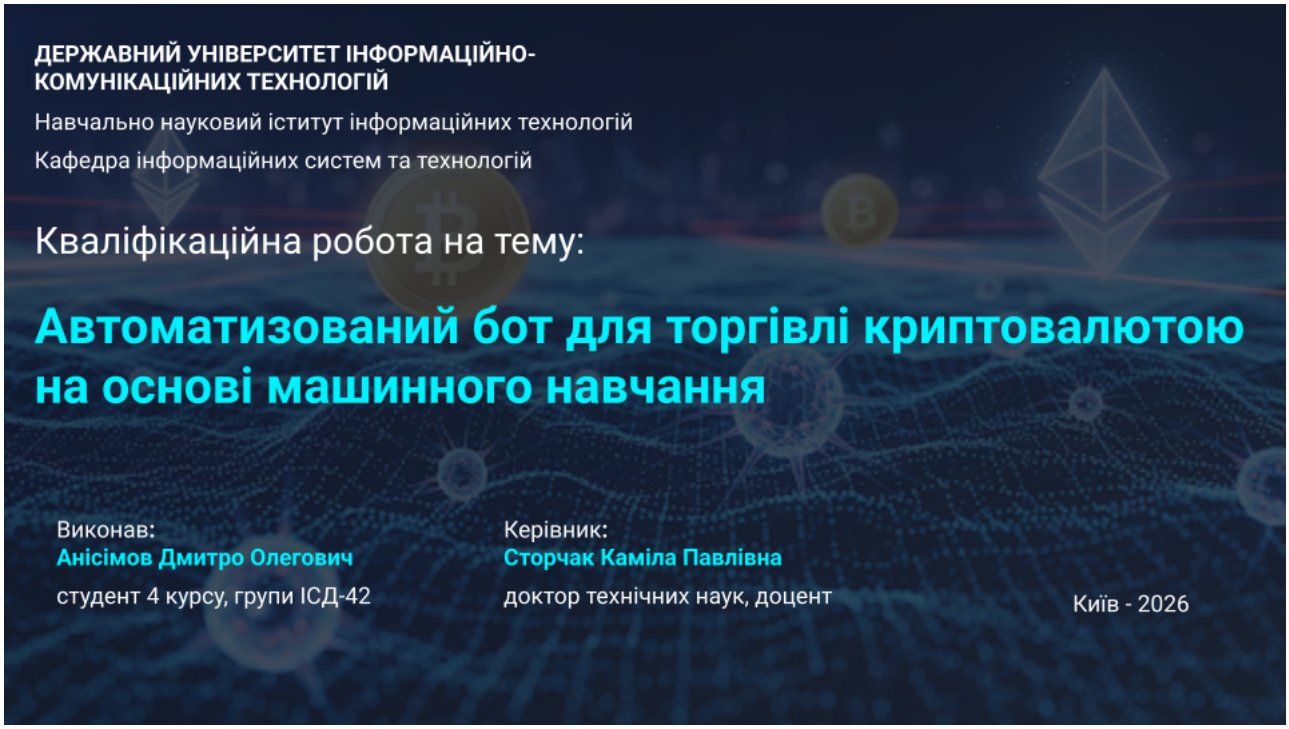
```
def fit(
    self,
    X: pd.DataFrame,
    y: pd.Series | np.ndarray | Sequence[int],
    *,
    sample_weights: pd.Series | np.ndarray | Sequence[float] | None = None,
    feature_columns: Sequence[str] | None = None,
) -> MetaModelFitSummary:
    x_clean, columns = self._prepare_features_for_fit(X, feature_columns=feature_columns)
    y_clean = self._prepare_target(y, expected_len=x_clean.shape[0])
    w_clean = self._prepare_sample_weights(sample_weights, expected_len=x_clean.shape[0])
    valid_rows = np.isfinite(y_clean) & np.isfinite(x_clean).all(axis=1)
    if w_clean is not None:
        valid_rows &= np.isfinite(w_clean) & (w_clean >= 0.0)
    x_fit = x_clean[valid_rows]
    y_fit = y_clean[valid_rows].astype(np.int8, copy=False)
    if w_clean is not None:
        w_fit = w_clean[valid_rows].astype(np.float64, copy=False)
    else:
        w_fit = None
    if x_fit.shape[0] == 0:
        raise ValueError("No valid rows available for fitting MetaModel.")
    unique = np.unique(y_fit)
    if unique.size < 2:
        raise ValueError("MetaModel requires both classes in target for training.")
    model = self._new_model()
    model.fit(x_fit, y_fit, sample_weight=w_fit)
```

```
    self.model_ = model
    self.feature_columns_ = columns
    self.is_fitted_ = True
    positive_rate = float(y_fit.mean())
    weighted_positive_rate: float | None = None
    if w_fit is not None and float(w_fit.sum()) > 0.0:
        weighted_positive_rate = float(np.average(y_fit, weights=w_fit))
    return MetaModelFitSummary(
        n_samples=int(x_fit.shape[0]),
        n_features=int(x_fit.shape[1]),
        positive_rate=positive_rate,
        weighted_positive_rate=weighted_positive_rate,
    )
def _new_model(self) -> Any:
    from sklearn.ensemble import RandomForestClassifier
                    return RandomForestClassifier(**self.model_params)
```

ДОДАТОК Г

```
def _barrier_levels(self, *, entry_price: float, sigma: float, side: int) -> tuple[float,
float]:
    if side == 1:
        pt = entry_price * (1.0 + self.pt_mult * sigma)
        sl = entry_price * (1.0 - self.sl_mult * sigma)
    else:
        pt = entry_price * (1.0 - self.pt_mult * sigma)
        sl = entry_price * (1.0 + self.sl_mult * sigma)
    return float(pt), float(sl)

def _first_touch(
    self,
    *,
    side: int,
    pt_level: float,
    sl_level: float,
    path_times: np.ndarray,
    path_prices: np.ndarray,
    vertical_time: pd.Timestamp,
    fallback_entry_price: float,
) -> tuple[BarrierTrigger, pd.Timestamp, float]:
    if path_prices.size == 0:
        return BarrierTrigger.TIME, vertical_time, float(fallback_entry_price)

    if side == 1:
        pt_hits = np.flatnonzero(path_prices >= pt_level)
        sl_hits = np.flatnonzero(path_prices <= sl_level)
    else:
        pt_hits = np.flatnonzero(path_prices <= pt_level)
        sl_hits = np.flatnonzero(path_prices >= sl_level)

    first_pt = int(pt_hits[0]) if pt_hits.size else None
    first_sl = int(sl_hits[0]) if sl_hits.size else None

    if first_pt is None and first_sl is None:
        return (
            BarrierTrigger.TIME,
            pd.Timestamp(path_times[-1], tz="UTC"),
            float(path_prices[-1]),
        )

    if first_sl is None or (first_pt is not None and first_pt <= first_sl):
        idx = first_pt
        assert idx is not None
        return      BarrierTrigger.PT,      pd.Timestamp(path_times[idx],      tz="UTC"),
float(path_prices[idx])

    idx = first_sl
    assert idx is not None
    return       BarrierTrigger.SL,       pd.Timestamp(path_times[idx],       tz="UTC"),
float(path_prices[idx])
```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



3/14

Підготовка даних та стаціонарність

Бари за грошовим обсягом

Перехід до «інформаційного» часу шляхом формування бару після накопичення грошового обсягу.

Проблема нестаціонарності

Сирі цінові рівні нестаціонарні, а класичне диференціювання видаляє структурні залежності.

Лямбда Аміхуда:

Кількісна оцінка неліквідності для вимірювання цінового впливу торгового обсягу на мікроструктуру.

Фракційне диференціювання (FFD)

Застосовується для досягнення стаціонарності рядів із максимальним збереженням довгострокової пам'яті (сигналу).

4/14

Генерація торгових подій

Формування та маркування подій

Ідентифікація значущих подій за допомогою симетричного фільтра кумулятивної суми (CUSUM).

Потрійний бар'єр

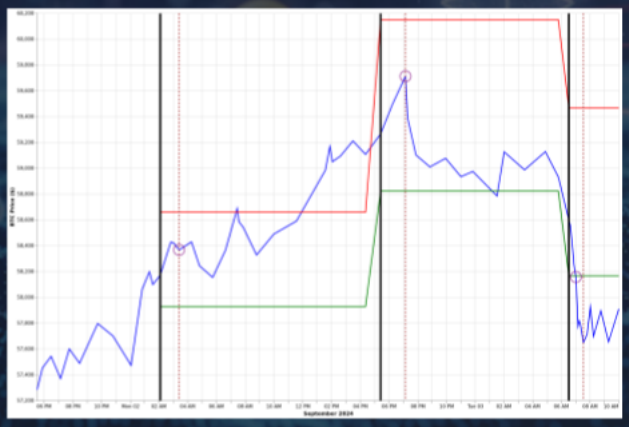
Цільова змінна моделюється за схемою Triple-barrier labeling для комплексного маркування.

Вертикальний бар'єр

Time-Trigger: ліміт часу для примусового закриття спостереження.

Верхній та Нижній бар'єри

Profit-Taking: динамічний рівень фіксації прибутку.
Stop-Loss: рівень обмеження збитків.



Інтелектуальна фільтрація

5/14

Концепція мета-розмітки

Архітектура розділена на два рівні: первинний сигнал (напрям) та мета-модель (фільтрація хибних сигналів).

Задача класифікації

Мета-модель вирішує задачу Y=1 (сигнал прибутковий) або Y=0 (спрацював стоп-лос або часовий ліміт).

Алгоритм Random Forest

Стійкий до викидів та здатний працювати з нелінійними фінансовими залежностями ринку.

Головна мета

Мінімізація хибнопозитивних спрацювань (false positives) для алгоритмів із високою частотою угод.

Фінансова валідація

6/14

Запобігання витоку даних

Стандартна крос-валідація неприпустима через серійну кореляцію та перекриття подій у часі.

Purged K-fold

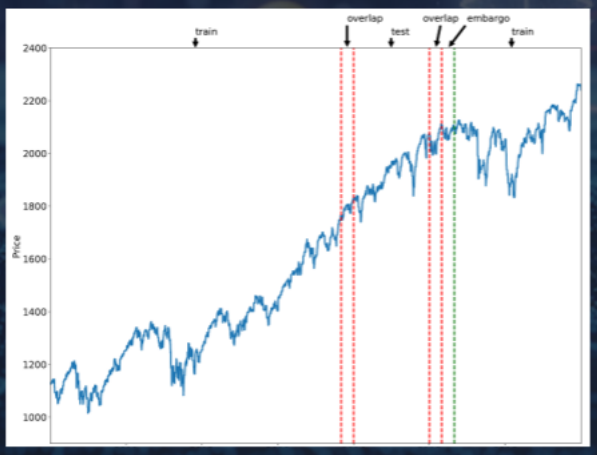
Очищення: вилучення спостережень, що перетинаються у часі з тестовим набором.

Ембарго

Встановлення часового буфера після тестового вікна для розриву статистичних зв'язків.

Послідовний Бутстреп

Зважування подій, що перетинаються у часі, для підвищення здатності моделі до узагальнення.



Програмна реалізація

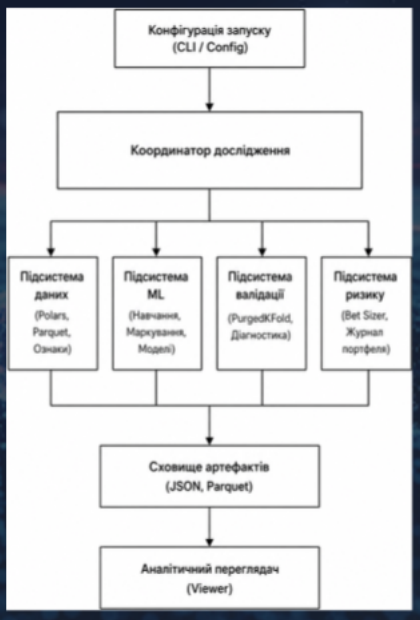
7/14

Стек та Обробка
Python (3.14), Polars (lazy scan) для барів, Pandas для індексів.

Зберігання
Parquet за допомогою PyArrow, zstd компресія для оптимізації I/O.

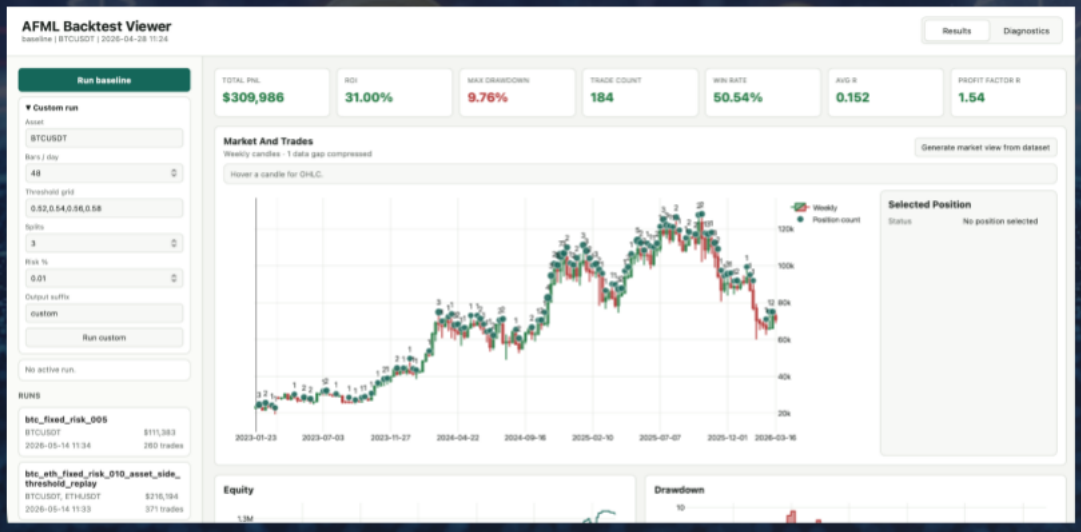
ML ядро та Аналіз
Scikit-Learn, SciPy (csr_matrix), Statsmodels (ADF тест).

Центральний Координатор
Управління: Data, ML, Validation, Risk.



Інтерфейс аналітичного переглядача

8/14



Управління капіталом

9/14

Підсистема ризик-менеджменту

Управління ризиком реалізовано за підходом фіксованого ризику на рівні кожної угоди.

Обсяг позиції розраховується математично залежно від відстані до рівня обмеження збитків та маржинальних лімітів.

Ризик на угоду
1.0%

Портфельні обмеження

Не більше двох одночасних позицій у портфелі.

Валова експозиція не перевищує ліміт ≤ 0,84.

Експериментальний запуск

10/14

Конфігурація сценарію BTC/USDT

Актив: високоліквідна пара BTCUSDT. Калібрування порога сформовано на цільову частоту 48 барів на добу.

Поріг бару
~$32.6млн

Навчальних подій
597

Згенеровано 114 624 бари за грошовим обсягом для аналізу.

Поріг фільтрації мета-моделі (L/S): **0.52**

Підсумкова навчальна таблиця

42 стаціонарні ознаки для моделювання.

Збалансований розподіл цільових міток для стабільного навчання.

Оптимізована структура даних для високої швидкості I/O.

11/14

Результати оцінювання

Фінансові результати моделювання

При початковому балансі 1 000 000 USDT підтверджено стабільну прибутковість алгоритму з контрольованим рівнем ризику.

Реалізований прибуток
+31.0%

Кількість угод
184

- Прибуток (PnL): **309 985,52 USDT**
- Max Drawdown (просідання): **10,05%**
- Max валова експозиція: **≤ 0,84**

12/14

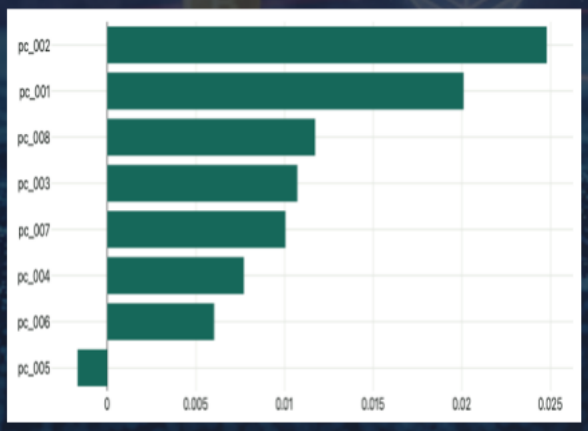
Аналітика ознак

Діагностика мікроструктурних ознак

Аналіз MDA підтвердив, що предиктивна сила базується на мікроструктурі. Застосовано PCA для усунення мультиколінеарності.

Найбільша важливість: **спред Corwin-Schultz, Amihud, VPIN.**

Модель знаходить закономірності в очищених від макро-шуму патернах.



13/14

Висновки: Підсумки та перспективи

Ключові досягнення проєкту

Розроблено цілісний програмно-аналітичний комплекс на базі методології фінансового машинного навчання. Відмова від класичних часових свічок на користь барів за грошовим обсягом стабілізувала статистичні властивості даних.

Мета-розмітка **Random Forest** спільно зі схемою потрійного бар'єра довела свою життєздатність у генерації позитивного результату:

+31% PnL

Перспективи розвитку

Мультиактивні портфелі:
Масштабування стратегії на ETH, SOL та інші високоліквідні активи.

Інфраструктура:
Перехід до paper-trading на базі CCXT для перевірки в реальному часі.

Глибина ринку:
Симуляція реальної глибини стакана заявок (order book depth).

14/14

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Анісімов Д.О. VII Науково-технічній конференції «Сучасний стан та перспективи розвитку IoT» з темою доповіді: «Штучний інтелект як інструмент підвищення безпеки в IoT-мережах»;

Анісімов Д. О. III Всеукраїнській науково-технічній конференції «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу» з темою доповіді: «Штучний інтелект та машинне навчання у побуті і промисловості».