Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Використання патернів програмування для розробки інтелектуального трейдінг бота
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-3
Кирило Приходченко
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник доц. Марина Кудрявцева
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Приходченку Кирилу Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Використання патернів програмування для розробки інтелектуального трейдінг бота

затверджена наказом університету від 19 травня 20 25 р. № 378Ст
2. Термін подання студентом роботи до екзаменаційної комісії 19 травня 20 25 р.
3. Вихідні дані до роботи Наукові публікації з галузі алгоритмічної торгівлі та машинного навчання, офіційна документація Python-бібліотек Freqtrade, scikit-learn, Keras, pandas, matplotli, відкриті датасети історичних OHLCV-даних криптовалют

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Методи та підходи до розв'язання задачі
3) Розробка інтелектуального трейдінг-бота

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі	21.05.2025	виконано
3	Формалізація мети, постановка задач	26.05.2025	виконано
4	Дослідження патернів програмування	29.05.2025	виконано
5	Побудова та тренування AI-моделі для прогнозу	04.06.2025	виконано
	ринку		
6	Розробка трейдінг-бота з підтримкою патернів	07.06.2025	виконано
7	Проведення backtesting на історичних даних	09.06.2025	виконано
8	Створення модулю генерації PDF-звіту	11.06.2025	виконано
9	Оформлення пояснювальної записки	13.06.2025	виконано
10	Підготовка презентації та доповіді	16.06.2025	виконано
11	Захист кваліфікаційної роботи перед ЕК	19.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Марина Кудрявцева
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 100 с., 5 рис., 3 табл., 2 дод., 34 джерела.

АВТОМАТИЗАЦІЯ, ІНВЕСТУВАННЯ, КРИПТОВАЛЮТА, МАШИННЕ НАВЧАННЯ, ПРОГНОЗУВАННЯ, ПРОГРАМНІ ПАТЕРНИ, ТЕСТУВАННЯ СТРАТЕГІЙ, ТРЕЙДИНГ-БОТ, ШТУЧНИЙ ІНТЕЛЕКТ.

Об’єкт дослідження – автоматизовані системи для торгівлі на фінансових ринках.

Предмет дослідження – застосування патернів програмування у побудові інтелектуального трейдінг-бота з елементами ШІ.

Мета дослідження – розробка інтелектуального програмного агента для підтримки рішень у трейдингу, використовуючи патерни програмування для забезпечення гнучкої та масштабованої архітектури.

Методи дослідження – аналіз предметної галузі, проєктування архітектури за допомогою патернів (MVC, Strategy, Observer), реалізація за допомогою Python-бібліотек (ccxt, sklearn, keras), тестування на історичних біржових даних.

У роботі розроблено інтелектуального торгового агента, який поєднує методи машинного навчання з патернами програмування для реалізації гнучкої, масштабованої архітектури автоматизованого трейдингу. Здійснено навчання класифікаційної моделі на основі технічних індикаторів для прогнозування торгових сигналів (Buy/Hold/Sell), реалізовано стратегію з динамічним перемиканням між rule-based та AI-режимами. Застосування патернів Strategy та MVC забезпечило модульність і зручність супроводу. Ефективність запропонованого рішення підтверджено серією backtesting-експериментів на історичних біржових даних.

ABSTRACT

Bachelor’s thesis contains: 100 pp., 5 fig., 3 tabl., 2 ann., 34 references.

ARTIFICIAL INTELLIGENCE, AUTOMATION, CRYPTOCURRENCY, FORECASTING, INVESTMENT, MACHINE LEARNING, PROGRAMMING PATTERNS, STRATEGY TESTING, TRADING BOT.

The object of research is automated systems for trading in financial markets.

The subject of research is the application of programming patterns in building an intelligent trading bot with AI elements.

The purpose of the research is the development of an intelligent software agent to support trading decisions, using programming patterns to ensure a flexible and scalable architecture.

The research methods include subject area analysis, architecture design using programming patterns (MVC, Strategy, Observer), implementation with Python libraries (ccxt, scikit-learn, Keras), and testing with historical market data.

The work develops an intelligent trading agent that combines machine learning methods with programming patterns to implement a flexible, scalable architecture of automated trading. A classification model based on technical indicators for predicting trading signals (Buy/Hold/Sell) was trained, and a strategy with dynamic switching between rule-based and AI modes was implemented. The use of Strategy and MVC patterns ensured modularity and ease of maintenance. The effectiveness of the proposed solution was confirmed by a series of backtesting experiments on historical stock market data.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence – штучний інтелект;

BTC/USDT, ETH/USDT – торгові пари криптовалют (Bitcoin/Tether, Ethereum/Tether);

CSV – Comma-Separated Values – формат табличних даних, розділених комами;

EMA – Exponential Moving Average – експоненціальне ковзне середнє;

Freqtrade – фреймворк для створення торгових ботів з відкритим кодом на Python;

JSON – JavaScript Object Notation – формат обміну структурованими даними;

MVC – Model-View-Controller – архітектурний патерн проєктування;

OHLCV – Open, High, Low, Close, Volume – типова структура фінансових даних;

PDF – Portable Document Format – формат електронних документів;

ROC – Rate of Change – індикатор швидкості зміни ціни;

RSI – Relative Strength Index – індекс відносної сили, технічний індикатор;

SMA – Simple Moving Average – просте ковзне середнє;

Strategy Pattern – патерн проєктування для динамічного вибору алгоритму;

Trading bot – програмний агент, що здійснює фінансові операції на біржах;

XAI – Explainable Artificial Intelligence – пояснюваний штучний інтелект.

ВСТУП

Сучасний етап розвитку цифрової економіки характеризується стрімким зростанням обсягів фінансових операцій, що здійснюються у віртуальному просторі. Зокрема, ринок криптовалют, електронних активів та деривативів вже перетворився на глобальну екосистему, що функціонує в режимі 24/7, без географічних чи часових обмежень. У цих умовах виникає нагальна потреба у створенні автоматизованих засобів аналізу, прогнозування та прийняття рішень, здатних функціонувати із високою швидкістю, адаптивністю та точністю. Одним із найбільш перспективних напрямів у цій галузі є розробка інтелектуальних трейдінг-ботів – програмних агентів, які здійснюють фінансові операції на основі аналізу ринкових даних, заданих стратегій та принципів машинного навчання.

Автоматизована торгівля, або алгоритмічний трейдинг, вже давно перестала бути виключною прерогативою великих інвестиційних компаній. Завдяки відкритим API, хмарним платформам і бібліотекам машинного навчання (таким як TensorFlow, Keras, Scikit-learn, ccxt) створення торгових систем стало доступним навіть для індивідуальних розробників і дослідників. Водночас побудова надійного та масштабованого програмного забезпечення вимагає чіткої архітектури, яка забезпечує розділення відповідальностей, розширюваність і повторне використання компонентів. Саме тому важливим є застосування патернів програмування – перевірених часом рішень, що дозволяють створювати гнучкі, підтримувані та ефективні системи.

Особливу цінність мають ті підходи, які дозволяють інтегрувати класичні принципи програмної інженерії з методами машинного навчання. Такий бот здатен не просто виконувати попередньо задану логіку дій, а й адаптуватися до нових ринкових умов, виявляти закономірності в часових рядах, оцінювати ризики та приймати рішення на основі моделей прогнозування.

Актуальність дослідження зумовлена як зростанням обсягів високочастотної торгівлі, так і потребою у високоточному прогнозуванні трендів, коливань і потенційних точок входу або виходу з ринку. При цьому недостатньо лише реалізувати алгоритми торгівлі – слід забезпечити стійку та масштабовану архітектуру, яка дозволить легко змінювати логіку прийняття рішень, адаптувати модулі та забезпечувати безпечну взаємодію із зовнішніми біржами. В цьому контексті патерни проектування, такі як Strategy, Observer, Command, Factory, MVC, є невід'ємним інструментом, що забезпечує якість і надійність розробленого продукту.

Таким чином, розроблений бот поєднує декілька важливих векторів розвитку сучасної інформаційної сфери – фінансову аналітику, штучний інтелект та інженерію програмного забезпечення. Розробка архітектурно зваженого, інтелектуального торгового бота відкриває широкі перспективи як для подальших наукових досліджень, так і для впровадження в реальні практики цифрової економіки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Основи алгоритмічної торгівлі та трейдінг-ботів

Інтенсивний розвиток цифрових фінансів, ринків криптовалют та загальна диджиталізація процесів управління активами стимулювали появу нового покоління торгових систем, що функціонують на базі алгоритмів. Алгоритмічна торгівля як напрямок виникла на перетині фінансової інженерії, програмування та статистики. Її ключова ідея полягає у заміні ручного прийняття рішень за допомогою формалізованих правил, які виконуються комп’ютером із високою швидкістю та без впливу людських емоцій.

З часу перших експериментів із програмами, які автоматизували розміщення заявок на біржах, алгоритмічна торгівля перетворилася на стратегічну перевагу, особливо в умовах високочастотного середовища, де вирішальним чинником стає не лише точність, а й час реакції. У західних фінансових інституціях уже у 2000-х роках частка торгівлі, яка здійснювалася автоматизовано, сягала понад 50%. З появою відкритих API, публічних бірж цифрових активів та доступності бібліотек для машинного навчання цей підхід став доступним не лише для інституційних інвесторів, але й для незалежних розробників.

Поняття трейдінг-бота охоплює програмний агент, що здатен виконувати торгові операції на основі заданої логіки. Зазвичай такий бот підключається до криптовалютної або фондової біржі, отримує поточні та історичні ринкові дані, аналізує ситуацію відповідно до визначеної стратегії та, у разі задоволення заданих умов, ініціює купівлю або продаж активів. При цьому найважливішою властивістю такого агента є здатність функціонувати в реальному часі, обробляючи динамічні дані з мінімальною затримкою.

Системи такого типу можуть використовуватись для реалізації як простих, так і складних стратегій. До прикладу, базова логіка може полягати в реакції на перетин ковзних середніх або зміну обсягу торгів. Проте сучасні системи поступово виходять за межі класичного аналізу – вони інтегрують елементи прогнозування, обробки сигналів з новин або соціальних мереж, та застосовують алгоритми класифікації або регресії для виявлення оптимальних точок входу. В цьому сенсі трейдінг-бот є не просто скриптом, а інтерактивною системою підтримки рішень, що враховує складну взаємодію факторів [1].

Попри технічну привабливість, розробка такого бота вимагає не лише глибокого розуміння ринку, але й обґрунтованого підходу до архітектури. Адже структура системи повинна бути масштабованою, гнучкою щодо зміни логіки, захищеною від помилок у критичних точках та зручною для тестування. Саме тому важливу роль відіграє попереднє проектування, зокрема із залученням патернів програмування, що дозволяють структурувати компоненти системи і спростити їхню взаємодію.

Ще одним ключовим моментом є необхідність тестування. У реальному трейдингу некоректне рішення може призвести до фінансових втрат, тому перед запуском у бойовий режим усі стратегії повинні бути перевірені в умовах, максимально наближених до ринкових. У цьому контексті важливим компонентом стає так званий backtesting – процес перевірки торгової стратегії на історичних даних, що дозволяє оцінити її стабільність, точність сигналів та чутливість до змін ринку.

У контексті криптовалютної торгівлі, особливої уваги потребує взаємодія з API бірж, де виникають виклики пов’язані з лімітаціями запитів, форматом даних, безпекою ключів доступу. Успішний трейдінг-бот повинен бути здатним коректно обробляти ринкову інформацію в режимі реального часу, відслідковувати відхилення від прогнозованої поведінки, адаптувати свою логіку, і в разі потреби – перейти у захисний режим, припиняючи активні дії.

Алгоритмічна торгівля на сучасному етапі розвитку фінансових технологій постає не лише як набір технічних рішень, а як цілісна концепція прогнозного управління ризиками та активами. У її основі лежить ідея випереджаючої дії: система повинна не просто реагувати на події, що вже сталися, а передбачати зміни ринкових умов і своєчасно адаптувати свою поведінку. Такий підхід формує принципову відмінність від класичних стратегій, орієнтованих на постфактум аналіз. У контексті високої волатильності ринків, де навіть незначні коливання можуть мати суттєві наслідки, здатність до проактивного реагування набуває критичного значення.

Трейдінг-бот виступає як інструмент втілення не лише логіки торгівлі, а й стратегічного підходу до управління фінансами в цілому. Його ефективність залежить не стільки від складності алгоритму, скільки від здатності поєднувати динамічну обробку даних, адаптивність до змін, обґрунтованість архітектурних рішень і контроль над ризиками. Успішна реалізація такого програмного агента вимагає поєднання декількох дисциплін: програмної інженерії, фінансової аналітики, теорії прийняття рішень та обробки великих обсягів інформації у реальному часі.

1.2 Штучний інтелект у фінансових системах

Штучний інтелект у сфері трейдингу виявляється у трьох основних напрямах: прогнозуванні, оптимізації та адаптації. Ці вектори часто переплітаються між собою, утворюючи гібридні підходи до управління торговими системами. Одним з найважливіших напрямків є формування торгових сигналів на основі історичних та поточних ринкових даних. Використовуючи алгоритми класифікації, регресії або моделі глибокого навчання, система здатна визначити ймовірність зростання або падіння активу, спрогнозувати зміни обсягу торгів або зміну ринкового тренду.

У таких задачах штучний інтелект відіграє роль аналітика, який не просто фіксує закономірності, а навчається на них, зберігаючи і вдосконалюючи набуті знання при нових змінах ринку. Наприклад, нейронні мережі здатні виявляти нетривіальні зв’язки між технічними індикаторами, що залишаються поза увагою традиційного аналізу [2].

Другим важливим напрямом є оптимізація, як у сенсі побудови портфеля, так і в рамках керування окремими угодами. Системи можуть використовувати методи багатокритеріальної оптимізації, еволюційні алгоритми, градієнтні методи тощо. Роль штучного інтелекту в цьому випадку полягає в здатності до гнучкої перебудови параметрів, урахування нових обмежень і самонавчання на основі результатів попередніх угод.

Третій аспект це адаптація. Ринки не є статичними, вони змінюються під впливом новин, регуляторної політики, масової психології. Тому AI-системи повинні бути здатні не просто відтворювати завчені шаблони, а оновлювати свої моделі, виявляючи зміни у структурі даних або в динаміці ринку. Із розвитком технологій дедалі важливішим стає застосування reinforcement learning підходу, в якому агент не лише аналізує історію, а й взаємодіє із середовищем, навчаючись на власних діях, винагородах та помилках.

У результаті застосування штучного інтелекту в трейдингу призводить до зростання як ефективності торгових стратегій, так і їхньої гнучкості. Якщо традиційні системи базуються на жорстко заданих правилах, то AI-підходи дозволяють створити адаптивного агента, що самостійно вчиться, узагальнює досвід і діє в умовах, які постійно змінюються [3].

1.2.1 Основні напрямки застосування AI у трейдингу

Штучний інтелект у трейдингу реалізується через низку важливих напрямків, які у комплексі змінюють принципи аналізу ринку, прийняття

рішень та управління ризиками. Кожен напрямок має власну специфіку і залежить від рівня складності задачі, доступних даних та обраної стратегії роботи на фінансовому ринку.

Одним з найдинамічніших і стратегічно важливих напрямів застосування штучного інтелекту в фінансових технологіях є прогнозування поведінки ринку. Йдеться не лише про передбачення напрямку зміни ціни, а й про глибше моделювання процесів, що впливають на поведінку фінансових інструментів. Така модель дозволяє зменшити ризик, підвищити ефективність угод та виявляти приховані ринкові сигнали раніше, ніж вони стануть очевидними для інших учасників.

Історично основою торгової аналітики слугував технічний аналіз, який базується на припущенні, що минулі цінові коливання та обсяги торгів містять інформацію про майбутні рухи ринку. Хоча технічний аналіз залишається важливою частиною інструментарію трейдера, сучасні системи з використанням AI значно розширюють його можливості. Замість обмеженого набору індикаторів вони працюють із багатовимірними часовими рядами, у яких одночасно аналізуються сотні або тисячі параметрів. До таких параметрів можуть входити як цінові характеристики (open, high, low, close, volume), так і похідні індикатори – ковзні середні, імпульси, кореляційні залежності, глибина ринку тощо [4].

Штучні нейронні мережі, зокрема багатошарові персептрони (MLP), рекурентні нейронні мережі (RNN) та їх модифікації LSTM і GRU, продемонстрували ефективність у задачах фінансового прогнозування завдяки здатності зберігати й обробляти часові залежності. Вони можуть виявляти неочевидні патерни у зміні ціни активу, що формуються внаслідок складної взаємодії ринкових факторів. Водночас ансамблеві методи, як-от випадковий ліс (Random Forest) або градієнтний бустинг (XGBoost, LightGBM), ефективно працюють із табличними даними та можуть поєднувати кілька слабких моделей в одну потужну систему прогнозування.

Ключовим елементом сучасного прогнозування є здатність враховувати поведінкові патерни – реакції ринку на події, що не завжди мають чисто технічну природу. Для цього штучний інтелект поєднується з аналізом позабіржової інформації. Одним із найбільш перспективних напрямів є включення вхідних ознак, отриманих з нетрадиційних джерел, таких як стрічки новин, аналітичні звіти, твіти впливових фігур ринку, повідомлення на форумах тощо.

Завдяки методам обробки природної мови (NLP), системи штучного інтелекту здатні автоматично зчитувати емоційний контекст, визначати полярність повідомлень, виділяти ключові поняття та створювати векторні представлення тексту. Наприклад, застосування таких моделей як Word2Vec, BERT або більш легких трансформерів дозволяє будувати сентимент-аналіз, який служить додатковим фактором для трейдінгової логіки. Якщо виявлено посилений негатив у новинному потоці щодо певного активу, це може сигналізувати про очікуване падіння, навіть якщо технічні показники поки що цього не підтверджують.

Найбільша сила таких гібридних підходів полягає в здатності випереджати традиційні інструменти. Там, де класичний трейдер побачить сигнал тільки після формування свічкового патерну, AI-система вже за кілька хвилин до цього може зафіксувати зміну у поведінці інвесторів, пов’язану з зовнішнім інформаційним шумом. Застосування таких рішень дозволяє будувати більш стійкі стратегії та уникати участі в емоційно зумовлених коливаннях [5].

Крім цього, прогнозування з використанням AI відзначається можливістю самонавчання, тобто поліпшення якості прогнозу з кожною новою порцією даних. Моделі, що працюють в онлайн-режимі або за допомогою стратегій активного навчання, здатні враховувати нові умови без повного перевчання, зберігаючи релевантність і точність навіть у разі раптових змін ринку.

Прогнозування це не просто один із напрямів застосування штучного інтелекту у фінансах, а наріжний камінь сучасної алгоритмічної торгівлі. Здатність моделі випередити ринок, інтегруючи як традиційні технічні фактори, так і інформаційний контекст це те, що робить AI-базовані системи по-справжньому конкурентоспроможними на фінансовій арені.

Другим стратегічним напрямком застосування штучного інтелекту у трейдингу є оптимізація торгових стратегій. Якщо прогнозування ринку орієнтоване на передбачення змін середовища, то оптимізація спрямована на вдосконалення самих торгових процесів, забезпечення максимальної ефективності розміщення капіталу та управління ризиками.

У традиційних підходах оптимізація стратегій обмежувалась налаштуванням фіксованого набору правил за допомогою емпіричних методів або простого перебору параметрів. Проте зі зростанням складності ринків і обсягів даних такі методи втрачають ефективність. Сучасні AI-системи здатні здійснювати тонке налаштування торгових правил, знаходити оптимальні комбінації параметрів стратегій та балансувати між різними цілями, наприклад, між мінімізацією ризику і максимізацією прибутку [6].

Штучний інтелект у задачах оптимізації застосовує широкий арсенал методів. Серед них особливе місце займають еволюційні алгоритми: генетичні алгоритми, диференціальні еволюції, методи рою частинок. Ці алгоритми моделюють природні процеси відбору, мутації та рекомбінації, що дозволяє ефективно шукати оптимальні рішення у великому просторі параметрів без необхідності повного перебору всіх можливих варіантів. Генетичні алгоритми, наприклад, дозволяють одночасно оптимізувати десятки і навіть сотні параметрів торгової системи, адаптуючись до складних ландшафтів цільових функцій, де класичні методи зазнають невдачі.

Іншим важливим класом методів є градієнтні оптимізатори, такі як Adam, RMSProp або класичний стохастичний градієнтний спуск (SGD).

Вони використовуються переважно у випадках, коли стратегія представлена у вигляді диференційованої моделі, наприклад, нейронної мережі для прогнозування торгових сигналів. Градієнтні методи дозволяють швидко наближатися до локальних мінімумів функцій втрат, що описують якість стратегії за заданими критеріями.

Особливо перспективним є застосування reinforcement learning. У цьому підході агент взаємодіє з ринковим середовищем, отримує винагороди за успішні дії і навчається максимізувати сумарну вигоду в довгостроковій перспективі. Такий підхід дозволяє створювати стратегії, що враховують як поточні результати, так і майбутні ризики й можливості. Агенти, навчені через reinforcement learning, здатні приймати складні послідовні рішення в умовах невизначеності, що є характерною особливістю фінансових ринків.

Окремої уваги заслуговує концепція динамічного хеджування, яка відіграє критичну роль у сучасних стратегіях управління ризиками. На відміну від статичних моделей, де співвідношення активів у портфелі фіксоване, динамічне хеджування передбачає постійну адаптацію структури портфеля залежно від змін ринкової ситуації.

У таких системах штучний інтелект застосовується для прогнозування короткострокових ризиків і сценаріїв розвитку ринку, після чого відбувається оперативне коригування ваг активів у портфелі. Це дозволяє знизити потенційні втрати у випадку негативних подій або нестабільності, одночасно зберігаючи можливість отримання прибутку при сприятливих умовах.

Практична реалізація динамічного хеджування вимагає побудови складних моделей, які одночасно враховують волатильність активів, кореляції між ними, обмеження на допустимі ризики та витрати на транзакції. Алгоритмічні підходи на базі AI значно перевершують класичні моделі завдяки гнучкості у роботі з великою кількістю змінних та здатності до самонавчання в міру накопичення нових даних.

Сучасні фінансові ринки є глибоко динамічними, нелінійними й чутливими до факторів, що змінюються щосекунди: від глобальних макроекономічних новин до мікросигналів у глибині ордербуку. У таких умовах класичні моделі машинного навчання, які навчаються на історичних даних і залишаються незмінними до наступного повного перенавчання, виявляють обмежену ефективність. Це відкриває простір для технологій адаптивного навчання в реальному часі – підходів, що дозволяють системам не лише приймати рішення, а й постійно вдосконалювати себе під час взаємодії з ринком.

Центральна ідея таких систем полягає у тому, що кожна нова порція даних не лише слугує для прийняття рішення, а й одразу включається до процесу навчання. Таким чином, модель «живе разом з ринком», реагує на його поточний стан і зберігає високу актуальність у ситуаціях, коли історичні патерни втрачають силу. Це особливо важливо для криптовалютного трейдингу або високочастотної торгівлі, де актуальність моделей часто обмежується хвилинами.

На практиці адаптивне навчання реалізується або через online learning (наприклад, методи стохастичного градієнтного спуску з поетапним оновленням ваг), або через reinforcement learning, де агент взаємодіє з ринком як з середовищем, отримуючи зворотний зв'язок у вигляді винагороди чи штрафу. Особливе місце посідає так званий contextual bandits підхід, який дозволяє системі одночасно діяти, збирати дані та навчатися на них у напівмарковському середовищі без повного моделювання всієї ринкової динаміки [7].

Адаптивність також дає змогу боротися з проблемою перенавчання на застарілих даних (data drift) та краще справлятися з структурними зламами ринку, коли змінюються фундаментальні закономірності, наприклад, через зміну регулювання або появу нового типу активів.

Застосування адаптивного навчання дозволяє створювати більш стійкі, гнучкі та розумні трейдинг-системи, які не потребують ручного

втручання для актуалізації моделі і здатні залишатися релевантними в умовах постійної турбулентності. Саме ця властивість, здатність до самопідтримки та еволюції в реальному часі наближає інтелектуальних агентів до рівня справжньої автономності у фінансовій екосистемі [8].

Найбільш ефективні торгові системи одночасно здійснюють прогнозування ціни, оптимізують параметри стратегії та адаптують свою логіку у відповідь на зміну ринкових умов. Комбінування декількох моделей дозволяє створити багаторівневу систему ухвалення рішень, де результати одного модуля уточнюються або доповнюються прогнозами іншого.

1.2.2 Приклади використання: класифікація, регресія, стратегія

Класифікація як метод ухвалення торгових рішень. Одним із найпоширеніших способів використання штучного інтелекту у фінансових системах є застосування моделей класифікації для ухвалення рішень щодо торгівлі активами. Класифікація у цьому контексті передбачає віднесення поточної ринкової ситуації до однієї з кількох категорій, таких як «купити», «утримувати» або «продати». На відміну від класичного технічного аналізу, де рішення приймаються заздалегідь заданими правилами (наприклад, перетином ковзних середніх), моделі класифікації навчаються на основі реальних даних і самостійно визначають складні взаємозв'язки між ознаками та результатами.

Для задач класифікації у трейдингу часто використовуються як прості, так і складні моделі. Одним із базових підходів є логістична регресія, яка моделює ймовірність приналежності спостереження до певного класу на основі лінійної комбінації ознак. Попри свою простоту, логістична регресія залишається популярною завдяки інтерпретованості результатів та невимогливості до ресурсів.

Більш складні моделі включають дерева рішень і їх ансамблі Random Forest, XGBoost, які можуть автоматично враховувати складні, нелінійні

взаємозв'язки між вхідними змінними. Такі алгоритми добре підходять для обробки табличних даних, включаючи технічні індикатори, статистику обсягів торгів, показники волатильності тощо. Завдяки своїй стійкості до шуму і здатності до узагальнення, дерева рішень часто використовуються у промислових системах алгоритмічної торгівлі [9].

Окрему категорію складають штучні нейронні мережі (ANN), особливо глибокі (DNN), які здатні моделювати надзвичайно складні залежності. Нейронні мережі можуть працювати без явно визначеного набору правил, навчаючись автоматично виділяти найважливіші ознаки із сирих даних. Це робить їх незамінними у випадках, коли класичні методи не можуть забезпечити достатньої точності або коли вхідна інформація має складну просторову або часову структуру.

Приклади реального застосування класифікації у трейдингу включають побудову моделей, що передбачають ймовірність росту чи падіння ціни певного активу протягом наступного часовго інтервалу. На основі цієї оцінки система ухвалює рішення про купівлю, продаж або утримання позиції. Наприклад, якщо модель прогнозує ймовірність росту активу більше ніж на 70%, бот автоматично відкриває довгу позицію. Якщо прогнозована ймовірність падіння перевищує певний поріг, виконується короткий продаж або закриття існуючої угоди.

Однією з важливих переваг підходу на основі класифікації є його гнучкість: за допомогою одного і того ж механізму можна створювати як агресивні стратегії (з частою зміною позицій), так і консервативні (з торгівлею лише при наявності сильних сигналів). Також важливо, що класифікаційні моделі можуть бути легко доповнені додатковими джерелами даних, наприклад, аналізом новин або емоційного забарвлення ринку, що дозволяє покращувати точність прийняття рішень.

Водночас варто зазначити, що застосування класифікаційних моделей у трейдингу має певні обмеження. Найбільший виклик полягає у високій нестабільності фінансових ринків: патерни, що працювали на історичних

даних, можуть змінитися у майбутньому. Це вимагає регулярної переадаптації моделей або застосування технік, здатних виявляти зміну структурних властивостей ринку в режимі реального часу.

Регресія як метод прогнозування фінансових показників. Іншим важливим напрямом використання штучного інтелекту у трейдингу є застосування моделей регресії для кількісного прогнозування змін ринку. Якщо класифікація передбачає віднесення ситуації до певного класу дій, то регресійні моделі націлені на передбачення числових значень, таких як майбутня ціна активу, обсяг торгів чи рівень волатильності.

Регресійні підходи мають стратегічну цінність, оскільки дозволяють трейдерам отримувати не тільки напрямок зміни ринку, а й оцінювати величину очікуваного зростання або падіння. Це створює основу для гнучкішого керування позиціями, оскільки рішення можуть прийматися з урахуванням передбачуваного розміру прибутку чи ризику.

Одними з найпоширеніших методів регресії є лінійна регресія, регресійні дерева, а також глибші моделі на основі нейронних мереж. Лінійна регресія часто використовується як базовий підхід для моделювання залежності ціни від набору факторів, наприклад, технічних індикаторів або макроекономічних показників. Незважаючи на простоту, у випадку високоволатильних активів лінійна регресія може виявитися недостатньо гнучкою.

Більш сучасні рішення використовують глибокі нейронні мережі (DNN) або LSTM-мережі для моделювання складних часових залежностей у фінансових рядах. Рекурентні архітектури особливо корисні, оскільки вони здатні враховувати віддалений вплив минулих подій на поточний стан ринку, що часто ігнорується простішими моделями.

Стратегічні AI-підходи: поєднання прогнозування, оптимізації й адаптації. На практиці більшість успішних торгових систем не обмежуються використанням тільки класифікації або регресії окремо. Найбільш

ефективні рішення базуються на інтеграції декількох методів у єдину стратегію, що дозволяє використовувати сильні сторони кожного підходу.

Типовий стратегічний AI-підхід включає:

– побудову регресійної моделі для прогнозування зміни ціни або волатильності;

– використання класифікатора для ухвалення рішення про дію на основі прогнозу;

– застосування reinforcement learning для налаштування параметрів торгівлі залежно від змін ринку.

Такі системи здатні не тільки реагувати на зміну середовища, але й активно оптимізувати свої стратегії за допомогою методів навчання з підкріпленням. Наприклад, агент може почати з консервативної стратегії, а потім, залежно від успіху попередніх дій, адаптувати свою агресивність, розмір позиції або частоту торгівлі [10].

Реальні приклади стратегій, що базуються на AI, включають:

– динамічні торгові боти для криптовалют, які одночасно прогнозують ціну і волатильність, адаптуючи стоп-лоси та тейк-профіти в реальному часі;

– фондові торгові системи, які будують власні торгові сигнали на основі емоційного аналізу новин та класичної технічної аналітики;

– арбітражні агенти, що виявляють короткострокові неефективності між ринками й автоматично оптимізують частоту торгівлі залежно від мінливості прибутків.

Попри високий потенціал AI-стратегій, їхня розробка пов'язана з рядом викликів. По-перше, необхідно забезпечити стійкість до змін ринкової динаміки, інакше система ризикує перенавчитися на історичних даних і виявитися неефективною в реальному часі. По-друге, існують технічні обмеження: велика кількість одночасних обчислень, потреба в обробці стрімінгових даних і потреба в оптимізації швидкодії алгоритмів.

1.3 Патерни програмування в архітектурі трейдінг-систем

Ефективна реалізація інтелектуальної торгової системи вимагає не лише правильного вибору моделей машинного навчання, але й ретельно продуманої архітектури програмного забезпечення. Адже складність таких систем зростає не лише через обсяг оброблюваних даних, а й через потребу в інтеграції з зовнішніми джерелами (біржі, API, моделі), забезпечення стабільної роботи в реальному часі та постійної адаптації до змін середовища.

У цьому контексті особливу роль відіграють патерни програмування – узагальнені шаблони рішень, що були сформовані на основі практичного досвіду побудови складних систем. Патерни не є готовими фрагментами коду, а виступають у ролі архітектурних конструкцій, які допомагають структурувати програму, зменшити зв’язаність між модулями, спростити масштабування і полегшити тестування.

Використання патернів проєктування в архітектурі трейдінг-ботів дозволяє реалізовувати гнучку, масштабовану і надійну систему, яка легко адаптується до нових вимог або змін логіки. Саме завдяки патернам можливо розділити логіку прийняття торгових рішень від технічних аспектів роботи з біржею, побудувати зрозумілу систему взаємодії компонентів та організувати ефективне керування даними та подіями.

У рамках проєктування інтелектуального трейдинг-бота найбільш доцільно застосовувати патерни поведінкової (behavioral), структурної (structural) та створювальної (creational) категорій, які будуть розглянуті нижче.

1.3.1 Класифікація програмних патернів

У системному підході до проєктування програмного забезпечення всі патерни поділяють на три великі групи: створювальні (creational),

структурні (structural) та поведінкові (behavioral). Така класифікація була запропонована у класичній праці «Design Patterns: Elements of Reusable Object-Oriented Software» авторства Еріка Ґамма та його колег, відомих як «Банда чотирьох» (GoF) [11].

Створювальні патерни зосереджені на процесі створення об’єктів. Їхнє завдання інкапсуляція механізмів створення об’єктів таким чином, щоб система не була жорстко пов’язана з конкретними класами, які вона використовує. Це дозволяє забезпечити гнучкість у зміні структури об’єктів без порушення існуючого коду. Серед найвідоміших створювальних патернів: Factory Method, Abstract Factory, Builder, Singleton, Prototype. У трейдинг-системах, наприклад, Factory-патерн може використовуватись для генерації різних типів індикаторів або торгових стратегій за параметрами користувача.

Структурні патерни визначають спосіб побудови відносин між класами та об’єктами для формування великих, функціонально насичених структур. Основна мета – спростити створення складних систем за рахунок повторного використання елементів та зниження зв’язаності між модулями. Прикладами є Adapter, Bridge, Composite, Decorator, Facade, Proxy. У контексті трейдинг-бота, структурні патерни дозволяють організовувати обробку вхідних даних з API, логування, взаємодію з графічним інтерфейсом, не змінюючи основної логіки роботи.

Поведінкові патерни описують способи комунікації між об’єктами та передачі поведінки. Їхнє призначення оптимізація алгоритмів, забезпечення гнучкості логіки, підтримка динамічного змінення поведінки системи без необхідності жорсткого кодування. Найвідоміші з них: Strategy, Observer, Command, State, Iterator, Mediator. У контексті інтелектуального торгового бота поведінкові патерни особливо актуальні, оскільки дозволяють змінювати логіку прийняття рішень у режимі реального часу залежно від ринкової ситуації [12].

Варто зазначити, що у складних системах патерни часто комбінуються. Наприклад, архітектура на основі MVC поєднує в собі структурні та поведінкові елементи, а одночасне використання патернів Strategy та Factory дозволяє створити модульну систему торгових стратегій з можливістю їхнього динамічного підключення.

Класифікація патернів це не лише типологія, а й практичний інструмент для розробника. Знаючи, до якого класу належить патерн, можна краще зрозуміти його призначення, спосіб реалізації та потенційні місця застосування у конкретному проєкті. У випадку трейдинг-бота це дає змогу не «винайти велосипед», а скористатися перевіреними архітектурними рішеннями, які значно прискорюють розробку та підвищують надійність системи.

1.3.2 Патерн Strategy: гнучкість у виборі логіки

Одним із ключових викликів при побудові трейдінг-систем є забезпечення гнучкості логіки прийняття рішень. Оскільки ринкові умови можуть змінюватися дуже швидко, важливо мати можливість легко змінювати торгові стратегії без необхідності глибокого перепроектування системи або зміни базових компонентів. Саме цю проблему ефективно вирішує патерн Strategy.

Патерн Strategy належить до поведінкових патернів програмування і передбачає інкапсуляцію різних алгоритмів або стратегій усередині окремих об’єктів з уніфікованим інтерфейсом. Замість того, щоб "зашивати" логіку прийняття рішень безпосередньо в основний код програми, патерн дозволяє динамічно підключати потрібну стратегію залежно від умов або вимог [13].

У випадку інтелектуального трейдинг-бота це означає, що різні торгові стратегії, наприклад, на основі ковзних середніх, індикаторів волатильності чи моделей прогнозування з AI, можуть бути реалізовані як

окремі класи, що реалізують спільний інтерфейс (наприклад, execute_trade_decision()). Бот при запуску або під час роботи може перемикатися між цими стратегіями без зміни своєї основної логіки функціонування.

Переваги застосування патерну Strategy у трейдинг-системах є очевидними:

– гнучкість: легко додавати нові стратегії або замінювати старі без модифікації основного коду бота;

– масштабованість: можна реалізувати стратегічні модулі для різних ринкових умов або типів активів;

– тестованість: кожна стратегія може бути протестована окремо, що спрощує налагодження та підвищує надійність;

– зменшення зв'язаності коду: основна логіка бота не залежить від деталей реалізації конкретної стратегії.

У проекті торгового бота, що підтримує як класичну rule-based логіку, так і AI-модель для прогнозування, реалізація Strategy Pattern дозволяє змінювати поведінку між режимами «fixed rules» та «AI-predicted trades» без перезапуску або редизайну всього додатку. Це особливо корисно при розробці мульти-агентних систем або при використанні reinforcement learning, де агент може динамічно змінювати стратегію залежно від результатів.

Патерн Strategy є невід'ємною складовою архітектури сучасних адаптивних трейдинг-систем і дозволяє забезпечити їхню еволюційну розвиток без ризику руйнування загальної структури коду.

1.3.3 MVC: розділення відповідальностей у додатку

Одним із найбільш усталених і ефективних архітектурних підходів до побудови складних інформаційних систем є застосування патерну Model–View–Controller (MVC). Цей патерн відноситься до категорії структурних

патернів проєктування і має на меті чітко розділити різні аспекти роботи додатка за їх функціональним призначенням. Саме розділення відповідальностей дозволяє створювати масштабовані, підтримувані й гнучкі програмні системи, що надзвичайно важливо у контексті проєктування трейдинг-ботів.

У класичній реалізації патерн MVC передбачає поділ програми на три ключові компоненти:

– model (модель) – обробляє бізнес-логіку додатку: зберігає, обробляє й оновлює дані, здійснює взаємодію з базами даних або зовнішніми API;

– view (уявлення) – відповідає за формування представлення даних для користувача або іншої системи; це може бути графічний інтерфейс, текстовий звіт або інший спосіб візуалізації інформації;

– controller (контролер) – приймає зовнішні запити (команди користувача або автоматизовані події), обробляє їх, керує логікою взаємодії між моделлю та уявленням.

У сфері побудови інтелектуальних трейдинг-систем такий підхід виявляється особливо ефективним. Дані фінансового ринку змінюються постійно, що вимагає оперативної обробки великих обсягів інформації, без порушення цілісності архітектури додатку. Використовуючи патерн MVC, можна чітко відокремити ядро обробки ринкових даних (Model) від інтерфейсу користувача або інших зовнішніх сервісів (View), забезпечуючи таким чином стабільність системи під час змін інтерфейсної або відображальної логіки [14].

У контексті трейдинг-бота модель може відповідати за збір і збереження історичних даних про ціни, обробку індикаторів, керування портфелем активів, а також за зберігання результатів тестування торгових стратегій. Уявлення, у свою чергу, може бути реалізоване як веб-інтерфейс для моніторингу стану бота, генерації графіків прибутковості, відображення відкритих позицій. Контролер забезпечує прийняття команд для старту або

зупинки стратегії, перемикання торгових режимів, управління ризиками або іншими аспектами функціонування системи.

Переваги впровадження MVC-архітектури для трейдинг-бота включають:

– спрощення розробки та тестування, кожен компонент можна розробляти та перевіряти окремо, що значно полегшує виявлення помилок і вдосконалення системи;

– модульність і масштабованість, за потреби можливо розширити функціональність одного компонента (наприклад, додати новий тип візуалізації) без зміни інших;

– адаптивність, можливість швидкої інтеграції нових інтерфейсів користувача або нових форматів даних без перебудови основної логіки роботи бота;

– підвищення стабільності, полегшення обробки одночасних запитів, підвищення надійності в умовах роботи з великою кількістю паралельних операцій.

Особливого значення патерн MVC набуває у великих системах, де потрібно динамічно обслуговувати як алгоритмічні запити на виконання торгових операцій, так і запити від користувача на перегляд стану торгів або аналітики. Завдяки чіткій структурі можна одночасно обробляти події в реальному часі та підтримувати оновлення візуальної частини без втрати продуктивності.

Ефективність застосування MVC також підвищується у поєднанні з іншими патернами проєктування, наприклад, Strategy для побудови набору змінних стратегій торгівлі або Observer для обробки подій у контролері. Це дозволяє будувати трейдинг-системи, що не лише стабільні, але й легко адаптуються до змін ринкового середовища.

Таким чином, патерн MVC залишається базовим архітектурним рішенням для організації роботи сучасних трейдинг-ботів, забезпечуючи їхню стійкість, гнучкість та готовність до масштабування і розвитку.

1.3.4 Observer та Factory як додаткові механізми

У процесі побудови гнучких та адаптивних програмних систем, зокрема трейдинг-ботів, часто виникає необхідність не лише розділяти відповідальність між компонентами, а й організовувати динамічну взаємодію між ними. У таких випадках доцільно звернутися до патернів, які забезпечують гнучкість в обробці подій та в створенні нових об’єктів із збереженням інкапсуляції. Серед найбільш ефективних у цьому контексті патерни Observer та Factory.

Патерн Observer належить до поведінкових і широко використовується у системах, де один об’єкт (спостерігач) має реагувати на зміни стану іншого об’єкта (суб’єкта). У контексті трейдинг-систем така взаємодія виявляється надзвичайно корисною. Наприклад, при зміні стану ринку – оновленні ціни, появі торгового сигналу чи досягненні порогового рівня ризику, можуть автоматично запускатися відповідні дії, які реалізуються незалежними модулями. Це дозволяє створити систему, що діє не за жорстко визначеним графіком, а за принципом реакції на події в реальному часі, що значно підвищує її гнучкість і швидкодію.

Крім того, патерн Observer добре вписується в архітектуру подієво-орієнтованих систем. Завдяки йому можливо побудувати структуру, де основна логіка бота не потребує знання про те, хто саме реагує на зміну стану. Усе, що їй потрібно зробити, це повідомити про подію, а вже система підписаних спостерігачів вирішить, як саме діяти. Це спрощує додавання нових компонентів, таких як логери, повідомлення на email або телеграм, нові блоки аналітики, без зміни вже існуючого коду, що є важливою ознакою якісної масштабованої архітектури.

Інший важливий патерн – Factory Method, належить до створювальних патернів і використовується для побудови систем, у яких необхідно створювати об'єкти певного типу, не вказуючи конкретний клас створюваного об’єкта. У торгових системах Factory дозволяє реалізувати

механізми генерації торгових стратегій, індикаторів, інтерфейсних елементів або конфігурацій залежно від обраних параметрів, що задаються динамічно.

Замість того, щоб створювати екземпляри об'єктів безпосередньо у коді основної програми, розробник реалізовує фабричний інтерфейс, який, отримавши вхідні дані (наприклад, назву стратегії або тип індикатора), самостійно визначає, яку саме реалізацію потрібно використати. Такий підхід дозволяє легко додавати нові типи стратегій, змінювати логіку без втручання в базовий код, і водночас зберігати узгодженість у створенні об'єктів та їх типізації [15].

Особливу цінність патерн Factory має у випадках, коли потрібно динамічно створювати об’єкти, які мають складну внутрішню структуру або залежать від зовнішніх параметрів. Наприклад, створення стратегії з інтегрованою AI-моделлю, обробкою ризиків та специфічною поведінкою на ринку потребує великої кількості залежностей, які зручно інкапсулювати всередині фабрики. Це дозволяє винести логіку побудови поза межі основного потоку програми і зосередити її в окремому модулі, що підвищує чистоту архітектури й спрощує супровід.

Комбінація патернів Observer і Factory дозволяє досягти високого рівня гнучкості, масштабованості та адаптивності трейдинг-системи. Перший забезпечує реактивну взаємодію компонентів без жорсткого зв’язування, а другий – модульне створення об'єктів на основі параметричних вхідних даних.

У результаті розробник отримує архітектуру, яка здатна не лише витримувати зміни, а й природно до них адаптуватися. Це особливо важливо в умовах стрімких змін фінансового ринку, коли ефективність рішень залежить від здатності системи змінюватися швидко, гнучко й безболісно.

1.4 Інструменти розробки інтелектуального трейдінг-бота

Розробка інтелектуального трейдинг-бота вимагає використання спеціалізованих інструментів, що забезпечують високу гнучкість, масштабованість і інтеграцію з ринковими даними в реальному часі. Вибір платформи, мов програмування, бібліотек і технологічних підходів безпосередньо визначає ефективність системи, її адаптивність до ринкових змін і можливість впровадження алгоритмів машинного навчання. У даному розділі розглядаються основні компоненти технологічного стеку, що використовуються при створенні трейдинг-бота, орієнтованого на застосування ШІ.

1.4.1 Freqtrade: архітектура, конфігурація, особливості

Одним із базових технологічних рішень для побудови алгоритмічного трейдінгу є використання платформи Freqtrade – відкритого фреймворку на мові Python, спеціально розробленого для створення, тестування і запуску торгових стратегій на криптовалютних біржах. Freqtrade надає розробникам гнучку і добре структуровану архітектуру, яка дозволяє зосередитись на створенні власної логіки торгівлі без необхідності розробляти низькорівневі інструменти для підключення до API бірж або обробки даних.

Архітектура Freqtrade побудована за принципами модульності та розділення відповідальностей. Основні компоненти включають ядро стратегії, механізми взаємодії з біржами, системи обліку угод, модуль ризик-менеджменту, а також підсистему аналізу історичних даних для проведення backtesting. Кожен з цих компонентів ізольовано в окремому модулі, що дозволяє гнучко замінювати або розширювати функціональність без переробки всього додатку.

Однією з ключових особливостей Freqtrade є підтримка різних типів стратегій, у тому числі rule-based, AI-орієнтованих або комбінованих

підходів. Стратегії реалізуються у вигляді Python-класів, які мають уніфікований інтерфейс, що забезпечує сумісність із ядром платформи. Це відкриває широкі можливості для інтеграції моделей машинного навчання, зокрема використання попередньо навчених моделей у форматі .pkl або навіть онлайн-навчання безпосередньо під час роботи бота.

Налаштування роботи Freqtrade здійснюється через конфігураційний файл у форматі JSON, де вказуються параметри біржового з'єднання, вибір торгових пар, максимальна кількість одночасно відкритих угод, ліміти ризиків, налаштування API-ключів та інші важливі аспекти. Завдяки такому підходу розробник має можливість швидко змінювати конфігурацію бота без перекомпіляції або змін у коді стратегії [16].

Серед інших сильних сторін платформи варто відзначити наявність зручного CLI-інтерфейсу для управління процесами, підтримку dry-run режиму для тестування стратегій у реальному часі без ризику реальних втрат, можливість автоматичної оптимізації гіперпараметрів стратегії та інтеграцію з системами моніторингу (наприклад, Telegram-ботами для повідомлень).

В умовах швидкозмінного криптовалютного ринку Freqtrade забезпечує необхідну гнучкість, стабільність і масштабованість для створення професійних інтелектуальних торгових систем. Саме завдяки відкритості архітектури і високому ступеню кастомізації ця платформа стає одним із найкращих рішень для розробки трейдинг-ботів, орієнтованих на машинне навчання і адаптивні стратегії [17].

1.4.2 Python-бібліотеки

Основою для розробки інтелектуальних трейдинг-систем є використання потужних бібліотек мови програмування Python, які забезпечують можливість обробки великих обсягів даних, інтеграції з біржами та побудови складних моделей машинного навчання. Вибір

бібліотек визначає не тільки ефективність роботи системи, але й її масштабованість, швидкість розробки та легкість подальшого супроводу.

Одним із фундаментальних інструментів для роботи з ринковими даними є бібліотека pandas, яка надає потужний функціонал для роботи з табличними даними у форматі DataFrame. З її допомогою можна ефективно обробляти часові ряди, здійснювати агрегацію, фільтрацію, обчислення технічних індикаторів і підготовку даних для моделювання.

Ще одним важливим компонентом є ccxt – спеціалізована бібліотека для взаємодії з криптовалютними біржами. Вона підтримує сотні бірж і дозволяє легко отримувати ринкові дані, розміщувати ордери, відстежувати портфель і управляти торговими операціями в реальному часі через єдиний уніфікований API. Саме завдяки ccxt трейдинг-боти можуть ефективно працювати з кількома біржами без необхідності створювати окремі адаптери для кожної.

У сфері машинного навчання надзвичайно важливими є scikit-learn та Keras. Бібліотека scikit-learn є стандартом для класичних ML-алгоритмів і містить реалізації найбільш поширених моделей класифікації, регресії, кластеризації, а також інструменти для попередньої обробки даних, вибору ознак і оцінки якості моделей. Її простий і зрозумілий інтерфейс дозволяє швидко експериментувати з різними алгоритмами та легко інтегрувати їх у загальну архітектуру трейдинг-бота.

Для побудови глибоких нейронних мереж використовується бібліотека Keras, яка забезпечує високорівневий API для TensorFlow або інших бекендів. Вона дозволяє розробляти складні моделі машинного навчання за допомогою кількох рядків коду, що особливо цінно при створенні систем для прогнозування ринкових трендів або оцінки ризиків.

Для візуалізації результатів аналізу даних і моделей широко використовується matplotlib та її похідна seaborn. За допомогою цих бібліотек можна будувати графіки цін, обсягу торгів, трендових ліній,

розподілів похибок моделей, а також інтегрувати візуалізацію безпосередньо в роботу торгової системи [18].

1.4.3 Отримання та підготовка біржових даних

Якість роботи трейдинг-бота безпосередньо залежить від якості вхідних даних. Фінансові ринки є динамічними системами, і навіть незначні відхилення в даних можуть призводити до істотних змін у поведінці торгових моделей. Тому процес отримання, фільтрації та підготовки біржових даних є критично важливою складовою розробки трейдинг-систем.

Отримання біржових даних зазвичай здійснюється через API бірж, що надають доступ до інформації про ціни відкриття, максимуми та мінімуми, обсяги торгів та інші характеристики активів. Стандартною структурою даних для трейдингу є OHLCV – open, high, low, close, volume, яка дозволяє будувати часові ряди та розраховувати технічні індикатори.

Бібліотека ccxt значно спрощує процес інтеграції з біржами, уніфікуючи формат запитів і відповідей для різних торгових платформ. Це дозволяє трейдинг-боту працювати з декількома біржами одночасно, використовувати арбітражні можливості або розподіляти ризики між різними торговими майданчиками.

Після отримання даних необхідним етапом є їхня первинна обробка. Сюди входить видалення пропущених значень, обробка аномалій, нормалізація цінових рядів, побудова технічних індикаторів та створення нових ознак для подальшого машинного навчання. У цьому процесі активно використовуються інструменти pandas для табличної обробки даних, а також бібліотеки технічного аналізу, такі як ta-lib або pandas-ta.

Однією з важливих задач на цьому етапі є синхронізація часових рядів для різних активів та забезпечення однакової частоти оновлення даних. Наприклад, якщо трейдинг-бот аналізує одночасно дані Bitcoin та Ethereum,

необхідно забезпечити коректне узгодження їхніх часових міток для правильного розрахунку кореляційних залежностей [19].

Підготовка даних також передбачає формування навчальних вибірок для моделювання. У разі використання моделей машинного навчання дані діляться на тренувальну, валідаційну та тестову частини. При цьому важливо дотримуватись принципу неперехресного навчання: дані для тестування не повинні використовуватися під час навчання моделей, аби уникнути переоцінки якості прогнозів.

1.4.4 Проведення backtesting

Backtesting є однією з найважливіших процедур у розробці й оцінці трейдинг-стратегій. Саме цей етап дозволяє визначити потенційну ефективність стратегії без ризику реального капіталу, базуючись на історичних даних ринку. По суті, backtesting це процес симуляції торгівлі за заданими правилами стратегії у минулому, який дає змогу оцінити, як стратегія могла б працювати за аналогічних ринкових умов.

Однією з основних переваг проведення backtesting є можливість кількісного вимірювання ризиків і прибутковості. Завдяки цьому можна виявити слабкі місця у стратегії, такі як надмірна чутливість до ринкових шумів, залежність від окремих періодів високої волатильності або слабка стійкість до змін тенденцій. Крім того, аналіз результатів тестування дозволяє оцінити співвідношення ризику до прибутку (Sharpe Ratio), середню прибутковість на угоду, максимальну просадку та інші ключові фінансові метрики [20].

Процес backtesting передбачає ретельну підготовку даних, що використовуються для тестування стратегії. Особливу увагу слід приділяти повноті даних, їх чистоті та правильності часових міток. Неточності у вихідних даних можуть призвести до спотворених результатів і неправильних висновків щодо ефективності стратегії.

Одним із ключових викликів у проведенні backtesting є проблема так званого «look-ahead bias», коли стратегія використовує інформацію, яка була недоступною на момент ухвалення торгового рішення. Щоб уникнути цього ефекту, потрібно суворо обмежити стратегію даними, що на той момент були відомі, і контролювати правильність використання часових інтервалів.

Ефективний backtesting передбачає також багатофакторний аналіз: перевірку стратегії на різних часових масштабах (наприклад, 5-хвилинні, 1-годинні, денні інтервали), в різні періоди ринку (трендовий ріст, боковик, спадний тренд) і з різними параметрами угод. Таким чином забезпечується більш об'єктивна оцінка універсальності стратегії та її стійкості до змін ринкової динаміки.

Окремої уваги заслуговує тестування стратегії із залученням транзакційних витрат, таких як комісії бірж, прослизання при виконанні ордерів і затримки з'єднання. Ігнорування цих факторів може призвести до суттєвого переоцінювання реальної прибутковості стратегії. Тому якісний backtesting повинен враховувати всі потенційні витрати, що супроводжують реальну торгівлю.

Результати backtesting не повинні сприйматися як гарантія аналогічної поведінки стратегії в майбутньому. Проте грамотне тестування дає змогу виявити слабкі сторони алгоритму, сформулювати обґрунтовані припущення про його поведінку у змінних ринкових умовах і прийняти рішення про доцільність подальшого доопрацювання або застосування стратегії в реальній торгівлі.

2 МЕТОДИ ТА ПІДХОДИ ДО РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Формалізація задачі автоматичного прийняття рішень у трейдингу

У контексті автоматизованих фінансових систем особливого значення набуває завдання формалізації процесу прийняття рішень. Якщо у традиційному трейдингу рішення про купівлю або продаж активів приймаються трейдером інтуїтивно або на основі візуального аналізу графіків, то в алгоритмічних системах ці рішення повинні бути закладені у вигляді математичних правил або навченої моделі. Саме процес переведення неформалізованих торгових рішень у чіткі алгоритмічні структури і називається формалізацією.

Завдяки формалізації відкривається можливість побудови інтелектуального агента, який здатен діяти автономно, без втручання людини, але з урахуванням тих самих принципів аналізу, які використовує досвідчений трейдер. У цьому процесі важливо не лише описати умови виконання тих чи інших дій, а й сформулювати їх у такий спосіб, щоб вони були доступними для обчислення машиною в умовах реального часу [21].

Формалізація поведінки торгового бота зазвичай починається з визначення набору можливих дій: купівля, продаж або утримання позиції. Ці дії можуть бути виражені у вигляді логічних умов, що базуються на технічних індикаторах, або як класи цільової змінної у задачі машинного навчання. Подальший вибір інструментарію, від простих умовних операторів до глибоких нейронних мереж залежить від складності стратегії, типу даних і бажаного рівня адаптивності системи.

2.1.1 Проблема формалізації Buy/Hold/Sell

Однією з базових функцій трейдингової системи є автоматичне ухвалення рішень щодо моменту купівлі, продажу або утримання активу. У

класичному підході такі рішення часто приймаються трейдером на основі власного досвіду, новинного фону, поточної ринкової динаміки або аналізу графіків. У випадку автоматизованого агента, який діє без участі людини, вся логіка повинна бути чітко формалізована, а критерії дій визначені в обчислюваній формі.

Формалізація дій Buy, Hold, Sell перетворює трейдинг із суб'єктивного процесу в задачу математичного ухвалення рішень, що має бути реалізована у вигляді алгоритму або моделі. У найпростішому випадку це може бути набір правил, наприклад: якщо поточна ціна перетинає ковзне середнє знизу вгору – купити; якщо зверху вниз – продати. Проте така логіка має жорстку структуру, не враховує контекст ринку і не адаптується до змін. Для подолання цих обмежень і застосовується підхід машинного навчання, який дозволяє формалізувати рішення на основі історичних даних і виявлених патернів [22].

У рамках даного дослідження завдання ухвалення рішення щодо дій на ринку формалізується як багатокласова задача класифікації. Кожен момент часу інтерпретується як окремий приклад у навчальній вибірці, який характеризується набором ознак, технічних індикаторів, розрахованих на основі ринкових даних. Цільова змінна в цьому випадку це дія, яку слід виконати: купити, продати або утримати позицію. Модель має навчитися за цими ознаками прогнозувати оптимальну дію в конкретному контексті.

Такий підхід дозволяє враховувати одночасно десятки або навіть сотні факторів: тенденції ринку, обсяги торгів, волатильність, індекси сили тренду та інші статистичні характеристики. Модель класифікатора навчається на історичних прикладах, у яких зафіксовано, яка дія була б оптимальною з точки зору отриманого прибутку. Після навчання така модель може працювати в режимі реального часу, аналізуючи поточні ринкові параметри та видаючи рекомендацію щодо дії.

Основна складність такої формалізації полягає у правильному визначенні цільової змінної, тобто, як саме формувати мітки «купити»,

«продати» або «нічого не робити». На практиці це залежить від заданої гіпотези про зміну ціни у майбутньому. Наприклад, якщо ціна протягом наступних n хвилин зросте більш ніж на k%, то чинна дія (купівля) вважається правильною. Якщо ж ціна впаде, то виправданою є продаж. Якщо ж зміни не перевищують заданого порогу, доцільним вважається утримання. Саме від цієї логіки залежить не тільки розмітка даних для навчання, а й загальна поведінка моделі в торгівлі.

Крім того, особливістю задачі класифікації дій у трейдингу є висока розбалансованість класів. У більшості випадків ринок перебуває у нейтральному стані, коли немає вираженого сигналу на купівлю чи продаж. Це призводить до домінування класу «hold» у навчальних вибірках. Для коректного навчання моделі необхідно застосовувати техніки балансування, змінювати функції втрат або використовувати моделі, стійкі до дисбалансу.

Формалізація не лише спрощує процес розробки трейдинг-бота, а й відкриває шлях до гнучкого управління стратегіями. Користувач або розробник може змінювати гіпотези формування цільової змінної, додавати нові ознаки, змінювати часовий горизонт прогнозу і кожен раз отримувати нову, адаптовану модель дій. Це створює умови для еволюційного розвитку системи, її самонавчання та адаптації до нових умов, що є критично важливим у сучасному ринку з високим рівнем волатильності та нестабільності.

2.1.2 Класифікація як підхід до сигналів

У межах автоматизованих торгових систем прийняття рішень часто базується на аналізі сигнальної інформації, набору параметрів, які відображають поточний стан ринку і дозволяють прогнозувати майбутні зміни. Ці сигнали можуть включати в себе технічні індикатори, статистичні ознаки або навіть поведінкові патерни. У разі застосування машинного

навчання для обробки таких сигналів найбільш природним підходом є постановка задачі класифікації.

Класифікація в цьому контексті означає віднесення кожного стану ринку до одного з попередньо визначених класів дій. Типовими класами є Buy, Hold і Sell, які представляють конкретну інструкцію для трейдинг-бота щодо того, яку дію слід виконати. Таким чином, класифікаційна модель фактично перетворює багатовимірний опис ринкової ситуації на одне цілеспрямоване рішення. Це дозволяє формалізувати логіку торгівлі на високому рівні абстракції, позбавившись необхідності створювати жорсткі правила вручну [23].

Однією з основних переваг класифікації як підходу є її гнучкість і здатність узагальнювати закономірності, які не піддаються явному опису. Якщо традиційні методи технічного аналізу використовують фіксовані правила на кшталт «купувати при перетині середніх», то класифікаційні моделі можуть виявити нелінійні взаємозв’язки між десятками параметрів, які в сукупності сигналізують про оптимальний момент для дії. Це дозволяє трейдинг-боту діяти не просто на основі сигналів, а на основі сформованої поведінкової моделі, навченої на історичних прикладах.

Для реалізації класифікації можуть використовуватися як прості алгоритми (логістична регресія, наївний баєсів класифікатор), так і складніші підходи, як дерева рішень, ансамблеві методи (Random Forest, XGBoost), нейронні мережі або гібридні архітектури. Вибір конкретного алгоритму залежить від складності ринку, обсягу даних, а також вимог до інтерпретованості результатів. Наприклад, дерева рішень часто застосовуються тоді, коли необхідно забезпечити зрозумілу логіку дій, тоді як нейронні мережі демонструють вищу ефективність у випадках із високою змінністю ринку, але потребують більш обережного налаштування.

Окрему роль у задачі класифікації відіграє визначення ознак, так званого вектору ознак, який виступає вхідною інформацією для моделі. Ці ознаки можуть включати як поточні значення технічних індикаторів (RSI,

EMA, MACD), так і похідні характеристики – напрям зміни, темп зростання, кумулятивна волатильність, кореляція між активами. Чим точніше сформовано вектор ознак, тим вища ймовірність того, що модель зможе виявити зв’язок між поточним станом ринку і правильною дією.

Класифікація, це не лише процес «переведення» сигналу у дію, а й спосіб побудови інтелектуальної логіки поведінки. Через навчання на реальних історичних даних модель отримує можливість не просто повторювати правила, а навчитися, як саме виглядає ситуація, коли найімовірніше настане зростання чи спад. Це суттєво підвищує адаптивність системи до нових ринкових сценаріїв, включно з тими, що раніше не спостерігалися.

2.2 Побудова ознак та обробка вхідних даних

Якість будь-якої моделі машинного навчання значною мірою визначається не лише вибором алгоритму, а й тим, які саме дані подаються на вхід. У трейдингу формування вхідних ознак потребує особливої уваги, оскільки ринкові процеси є складними, стохастичними і часто не мають лінійної структури. Побудова адекватного вектору ознак це ключ до отримання надійної, узагальнюючої моделі, здатної реагувати на сигнали з реального ринку.

Для побудови вхідного простору використовуються класичні технічні індикатори, які є похідними від ціни та обсягу активів у часовому ряді. До набору обраних ознак входять індикатори, що мають як трендову, так і осциляторну природу. Зокрема, застосовуються RSI (Relative Strength Index), що дозволяє оцінити перекупленість або перепроданість активу; SMA (Simple Moving Average) та EMA (Exponential Moving Average), які відображають середньозважену динаміку цін; ROC (Rate of Change) для вимірювання швидкості змін; BBWidth (ширина смуг Болліджера), як оцінка волатильності ринку.

Для кожного спостереження у часовому ряді розраховується набір значень зазначених індикаторів із різними параметрами згладжування (наприклад, періоди 5, 10, 20). Отримані значення формують вектор ознак, який використовується як вхідна інформація для моделі класифікації. Ці дані додатково нормалізуються, аби уникнути переваги тих ознак, що мають більший масштаб, і покращити стабільність навчання.

Важливо забезпечити узгодженість інтервалів часу між усіма ознаками, наприклад, шляхом редукції даних до чітко визначеного таймфрейму (15-хвилинного або 1-годинного). Усі ознаки формуються виключно на основі доступної на момент t інформації, щоб уникнути впливу майбутніх значень та уникнути ефекту «look-ahead bias», який міг би призвести до хибно завищеної ефективності моделі.

Ретельне формування вхідних ознак – це не лише технічна процедура, а важливий аналітичний етап, що визначає здатність моделі до генералізації та прийняття обґрунтованих рішень на складному і динамічному фінансовому ринку.

2.2.1 Технічні індикатори: RSI, SMA, EMA, BBWidth, ROC

Вибір технічних індикаторів є важливим аналітичним етапом при побудові системи ухвалення рішень у трейдингу. Індикатори виконують роль фільтрів, які надають зведену інформацію про поведінку ціни та обсягу, згладжують флуктуації, виявляють тренди, імпульси та аномалії. Саме вони формують основу ознак, які подаються на вхід моделі машинного навчання для прийняття рішень типу Buy / Hold / Sell. У межах цієї роботи були обрані п’ять найбільш репрезентативних індикаторів: RSI, SMA, EMA, BBWidth та ROC, які відображають як трендову, так і імпульсну динаміку ринку, це наведено у таблиці 2.1.

Індекс відносної сили (RSI) використовується для оцінки сили ринкового руху та виявлення зон перекупленості або перепроданості. Цей

осцилятор варіюється в діапазоні від 0 до 100, де значення вище 70 зазвичай інтерпретується як сигнал до потенційного розвороту вниз, а нижче 30, як ймовірний сигнал до зростання. RSI особливо корисний у фазах слабко виражених трендів, коли ринок коливається в межах вузького діапазону.

Просте ковзне середнє (SMA) є одним із найпростіших, але при цьому надзвичайно корисних індикаторів для виявлення довгострокових тенденцій. SMA розраховується як середнє арифметичне ціни за визначений період часу. Хоча індикатор повільно реагує на зміни, він забезпечує стабільну основу для фільтрації ринкового шуму та визначення напрямку загального тренду [24].

Експоненціальне ковзне середнє (EMA) є вдосконаленою версією SMA, яка надає більшої ваги останнім значенням, завдяки чому забезпечує швидшу реакцію на зміни ринку. Це робить EMA особливо корисним при побудові моделей для короткострокових стратегій, де важлива своєчасна реакція на зміни напрямку руху ціни.

BBWidth, або ширина смуг Болліджера, є індикатором волатильності. Він показує, наскільки активно ціна змінюється порівняно з її середнім значенням. Збільшення ширини смуг зазвичай сигналізує про зростання волатильності, що може передувати пробою діапазону. BBWidth дозволяє моделі адаптувати свої рішення до фаз підвищеної нестабільності або спокою на ринку.

ROC (Rate of Change) є імпульсним індикатором, який вимірює відсоткову зміну ціни за певний період. Він дає змогу оцінити швидкість зміни вартості активу й ефективно виявляє фази прискорення чи сповільнення тренду. Особливо корисним є його застосування у комбінації з осциляторами або трендовими індикаторами.

Комбіноване використання обраних індикаторів дозволяє створити збалансований набір ознак, що поєднує інформацію про трендову складову, волатильність і імпульсну активність ринку. Таке поєднання є критично важливим для підвищення точності класифікації торгових сигналів,

оскільки дає змогу виявляти як фази стабільного зростання чи падіння, так і моменти потенційної зміни ринкового стану. Наприклад, комбінація RSI з ROC може дозволити моделі розрізняти, чи є перекупленість ринку динамічною або стагнаційною.

Таблиця 2.1 – Характеристики технічних індикаторів

Назва	Класифікація	Аналітична функція	Ключовий параметр	Інтерпретація поведінки
RSI	Осцилятор	Визначення зон перекупленості	Період обчислення	Значення >70 сигналізує про можливе зниження, <30 – про потенційне зростання
SMA	Трендовий	Глобальне згладжування цінових коливань	Період усереднення	Показує напрям довгострокового тренду, але реагує повільно
EMA	Трендовий	Згладжування з акцентом на останні ціни	Період усереднення	Швидко реагує на ринкові зміни, сигналізуючи про потенційні розвороти
BBWidth	Волатильнісний	Кількісна оцінка ширини смуг Болліндже ра	Період розрахунку	Зростання ширини свідчить про підвищену волатильність, стиск – про можливий імпульс
ROC	Імпульсний	Вимірювання темпу зміни ціни у відсотках	Період спостереження	Відображає прискорення або сповільнення ринкового руху, може бути позитивним або негативним

Важливим аспектом є інтервальність розрахунку індикаторів. Індикатори можуть обчислюватися з різними параметрами періодів:

короткострокові (наприклад, EMA з періодом 5) дозволяють оперативно реагувати на мікрозміни, тоді як довгострокові (SMA з періодом 20) дають змогу сформувати загальне уявлення про напрям ринку. Це багаторівневе представлення часового контексту формує основу для адаптивних рішень, які можуть масштабуватися в межах однієї моделі.

З методологічної точки зору, застосування таких індикаторів є обґрунтованим, оскільки вони мають високу репрезентативність у сучасній фінансовій аналітиці, добре підтримуються у відкритих бібліотеках Python і можуть бути розраховані ефективно у реальному часі. Це особливо важливо для подальшої інтеграції із трейдинг-ботом, що діє в умовах високочастотного або хвилинного таймфрейму [25].

Наявність як індикаторів абсолютного рівня (RSI, SMA), так і індикаторів динаміки (ROC, BBWidth) дозволяє забезпечити різноспрямовану інформативність ознак, що покращує здатність моделей до узагальнення і знижує ймовірність помилкових сигналів. Завдяки цьому підвищується не лише точність прогнозу, а й стабільність поведінки моделі у складних ринкових умовах.

2.2.2 Побудова вектору ознак, нормалізація

Після вибору релевантних технічних індикаторів наступним кроком у підготовці даних для машинного навчання є побудова вектору ознак – числового представлення ринкової ситуації у формі, зручній для подання моделі. Формально, кожна точка часу (кожна хвилина або година) інтерпретується як окремий приклад у вибірці, який складається з набору числових значень значень індикаторів, що були розраховані для цього моменту.

Кожен обраний технічний індикатор додається до цього вектору як окрема ознака. У випадках, коли для одного індикатора використовується декілька параметрів (наприклад, EMA з періодами 5 і 10), кожна версія

додається як окрема компонента. Таким чином, вектор ознак для одного моменту часу містить одночасно інформацію про коротко- і середньострокову динаміку активу, його волатильність і силу імпульсу. Зростання кількості ознак, з одного боку, дозволяє моделі краще уловлювати патерни, але, з іншого боку, вимагає уважного опрацювання, аби уникнути надлишкової складності або надмірної кореляції між параметрами.

Всі обчислення ознак проводяться із використанням виключно тієї інформації, яка була доступна до моменту t, тобто дотримується каузальна послідовність (causality), що дозволяє уникнути так званого «look-ahead bias». Крім того, всі ознаки синхронізуються за таймфреймом: у цьому дослідженні для розрахунків використовується фіксований 15-хвилинний часовий інтервал [26].

Ще одним критично важливим етапом є нормалізація ознак. Через те, що різні індикатори мають різну шкалу (наприклад, RSI обмежений від 0 до 100, а ROC може мати як позитивні, так і від'ємні значення без явного верхнього чи нижнього порогу), без нормалізації модель може надавати непропорційно велику вагу певним параметрам лише через масштаб їх числового представлення. Для вирішення цієї проблеми застосовуються методи мін-макс нормалізації або стандартизації (z-score), які приводять усі значення до одного масштабу без втрати інформації про їхню структуру.

Нормалізовані ознаки зберігаються у вигляді структури X, що є матрицею розмірності $n \times m$, де n – кількість прикладів (рядків), а m – кількість ознак (стовпців). До цієї матриці пізніше додається вектор цільових змінних y, який відповідає міткам класів Buy/ Hold/ Sell.

Побудований вектор ознак стає основою для навчання класифікаційної моделі. У такий спосіб трейдинг-бот отримує змогу аналізувати ринкову ситуацію у формалізованому вигляді, спираючись не на «сирі» цінові дані, а на аналітичні обчислення, які узагальнюють

закономірності та фільтрують шум. Це суттєво покращує якість навчання, стабільність прогнозів і адаптивність моделі до складних ринкових умов.

2.2.3 Обґрунтування набору ознак для моделі

Вибір ознак у системах машинного навчання є одним із найкритичніших етапів у побудові високоефективних моделей. У задачах фінансового прогнозування, зокрема автоматизованого трейдингу, цей вибір ускладнюється стохастичною природою ринку, високою волатильністю, сезонністю і наявністю великої кількості взаємозалежних змінних. У зв’язку з цим набір ознак повинен не лише бути інформаційно насиченим, але й забезпечувати баланс між простотою, інтерпретованістю та здатністю моделі до узагальнення.

У рамках цієї роботи набір ознак побудований на основі п’яти базових технічних індикаторів: RSI, SMA, EMA, BBWidth та ROC. Кожен з них виконує специфічну аналітичну функцію й описує певний аспект ринкової поведінки: силу імпульсу, напрямок тренду, зміну темпу руху ціни або рівень волатильності. Разом вони створюють узагальнене багатовимірне представлення поточного ринкового стану, яке модель може використовувати як вхід для прогнозування майбутньої поведінки ціни.

RSI обрано як представника осциляторних індикаторів, що дозволяє виявляти фази перекупленості або перепроданості ринку. Така інформація особливо цінна у фазах консолідації або наближення до розворотів. SMA і EMA забезпечують трендову складову, при цьому EMA, на відміну від SMA, більш чутлива до останніх змін і забезпечує сигнал раніше, що робить їх поєднання обґрунтованим з точки зору врахування коротко- і довгострокових рухів. BBWidth, як показник волатильності, дає змогу розрізняти фази ринку з різною динамікою, наприклад, бічний тренд або флет, коли доцільно утримуватися від активних дій. Нарешті, ROC один із

найпростіших та інтерпретованих індикаторів темпу зміни, що дозволяє моделі вловлювати силу поточних імпульсів.

Вектор ознак формується не лише на основі одиничних значень індикаторів, а також з урахуванням різних періодів обчислення. Таке представлення дозволяє захоплювати інформацію з кількох горизонтів часу: короткострокових (EMA5, RSI7), середньострокових (SMA10, BBWidth20) і навіть квазідовгострокових (SMA50 або EMA30). Це дає змогу моделі побудувати внутрішню логіку, що орієнтується як на локальні коливання, так і на загальну картину ринку. У результаті модель отримує так зване «розшароване» представлення поведінки ціни, що підвищує точність класифікації торгових сигналів.

Ще одним фактором, який підтверджує доцільність обраного набору, є низька кореляція між деякими з ознак. Наприклад, ROC і RSI часто описують різні аспекти руху ціни: ROC зміну абсолютної вартості, а RSI – зміну відносної сили між зростаючими й спадними періодами. Це дозволяє уникнути надмірної надмірності (redundancy) ознак у навчальній вибірці. Попередній аналіз кореляційної матриці для навчального датасету (розрахований на основі історії Bitcoin за 6 місяців) підтвердив, що коефіцієнти Пірсона між EMA10 та BBWidth або між ROC та SMA були нижчими за 0.5, що вказує на достатню незалежність цих ознак у загальній структурі даних.

Окремої уваги заслуговує аспект обчислювальної ефективності. Усі обрані індикатори можуть бути розраховані в реальному часі з мінімальними затратами ресурсів, використовуючи бібліотеки pandas, numpy або pandas-ta. Це забезпечує стабільну роботу трейдинг-системи навіть у високочастотному режимі. Крім того, ці індикатори легко масштабуються.

Формування такого набору ознак відкриває простір для подальших розширень. У майбутньому до вектору можуть бути додані ознаки на основі фундаментальних факторів (обсяг новин, настрої в соцмережах,

макроекономічні індекси), що дозволить створити гібридну модель – поєднання технічного аналізу з інформаційним або сентимент-аналізом.

2.3 Розробка AI-моделі

2.3.1 Вибір моделі (класифікатор, SVM/MLP/RF)

У контексті задачі автоматичного ухвалення рішень у трейдингу ключовим є правильний вибір моделі машинного навчання, яка буде здатна обробляти сформований вектор ознак і видавати рішення з прийнятною точністю, швидкодією та стійкістю до змін ринку. Формально, завдання полягає у класифікації поточної ринкової ситуації до одного з трьох класів: купівля, продаж або утримання позиції. Отже, йдеться про багатокласову задачу, в якій необхідно побудувати функцію, що зіставляє вектор ознак до дискретної цільової змінної [27].

Було розглянуто декілька потенційно ефективних моделей: SVM (машина опорних векторів), MLP (багатошарова перцептронна мережа) та Random Forest. Кожен із цих підходів має власні переваги й недоліки, які було проаналізовано як з теоретичної, так і з практичної точки зору.

SVM є одним із найпопулярніших класичних методів класифікації. Він добре працює на задачах з невеликою кількістю ознак та чітко розділеними класами. Принцип роботи базується на побудові гіперплощини, яка максимально розділяє точки різних класів у просторовому представленні ознак. У випадках, коли класи не є лінійно роздільними, застосовуються так звані ядрові функції (kernel trick), які дозволяють перенести задачу у вищий вимір і досягти більш точного розділення. Проте для задач із великою кількістю ознак або високою розбалансованістю класів (як у випадку з фінансовими сигналами) SVM демонструє знижену стабільність, особливо при наявності шуму або

колінеарності в даних. Крім того, модель є менш придатною до масштабування при збільшенні обсягу вибірки та кількості класів, що обмежує її застосування у високочастотному трейдингу.

Натомість Random Forest, це ансамблевий алгоритм, що поєднує велику кількість дерев рішень, які працюють незалежно одне від одного. Результат класифікації визначається за більшістю голосів. Такий підхід дозволяє зменшити ризик переобучення, властивий окремим деревам, і забезпечує високу стійкість до аномалій. Однією з головних переваг Random Forest є його інтерпретованість: легко відстежити важливість ознак, що було використано при прийнятті рішення, і виявити фактори, які найбільше впливають на класифікацію. У трейдингу це дозволяє не лише передбачати дію, а й отримувати аналітичні пояснення щодо природи сигналу. До недоліків методу належать більша вага «випадкових» рішень при сильному ринковому шумі, а також відносно повільна робота в режимі реального часу, якщо кількість дерев перевищує певний поріг.

Особливе місце посідає MLP (Multi-Layer Perceptron) – класична глибока нейронна мережа, яка забезпечує високу гнучкість, здатність моделювати нелінійні залежності та адаптуватися до різних режимів ринку. Мережа складається з одного або кількох прихованих шарів нейронів, які трансформують вхідний вектор ознак у багатошарове представлення, з якого через активаційні функції (ReLU, tanh тощо) формується прогноз ймовірностей по кожному класу. Однією з ключових переваг MLP є її універсальність, при наявності достатньої кількості шарів і нейронів вона може апроксимувати практично будь-яку функцію класифікації. У контексті фінансового трейдингу це дозволяє їй виявляти приховані патерни, які не є очевидними навіть для досвідченого трейдера.

Під час попередніх експериментів на даних криптовалютної біржі Binance було виявлено, що MLP демонструє найвищу стабільність прогнозу, особливо у фазах ринкової турбулентності. Крім того, середній рівень точності моделі на валідаційній вибірці перевищував 75%, що виявилося

на 7–10% вищим, ніж у випадку з Random Forest або SVM. Важливо також, що MLP дозволяє використовувати ймовірнісні оцінки для ухвалення рішення, наприклад, відкривати позицію лише тоді, коли ймовірність класу «Buy» перевищує певний поріг, що дає змогу гнучко керувати ризиками.

У результаті порівняльного аналізу було обрано MLP як основну модель класифікації, що поєднує в собі високу точність, можливість масштабування, адаптивність до нових ознак та відповідність сучасним практикам у сфері фінансового машинного навчання. Попри дещо вищі вимоги до обчислювальних ресурсів і потребу в уважному налаштуванні гіперпараметрів, ця модель забезпечує найкраще узагальнення та має потенціал до подальшого вдосконалення, зокрема через використання регуляризації, дроп-аутів і крос-валідації [28].

Вибір класифікаційної моделі обґрунтовано не лише емпіричною точністю, але й низкою додаткових критеріїв: стійкість до змін ринку, здатність працювати з розбалансованими даними, інтерпретованість, швидкість навчання та можливість подальшої інтеграції у трейдинг-систему через Freqtrade.

2.3.2 Навчання, крос-валідація, оцінка точності

Після вибору архітектури моделі наступним етапом є її навчання на підготовленому векторі ознак. Основна мета процесу навчання, це побудова функції, яка максимально точно відображає закономірності у вхідних даних і дозволяє передбачати правильний клас дії (Buy/ Hold/ Sell) у нових, ще не бачених ринкових ситуаціях. З технічної точки зору, навчання класифікаційної моделі зводиться до оптимізації ваг внутрішніх зв’язків у нейронній мережі (MLP), або ж до формування дерева рішень або набору гіперплощин у випадку SVM чи Random Forest.

У межах цього дослідження модель навчалась на реальних біржових даних криптовалютної пари BTC/USDT, зібраних з інтервалом у 15 хвилин

за період у 6 місяців. Після формування вхідних ознак (див. розділ 2.2), дані було поділено на навчальну (70%), валідаційну (15%) та тестову (15%) вибірки. Такий поділ дає змогу уникнути ефекту перенавчання (overfitting), а також перевірити узагальнюючу здатність моделі на раніше не бачених прикладах. Для збереження хронології події, поділ здійснювався по часу, без випадкової перемішки, що є важливим для часових рядів.

Навчання моделі MLP відбувалося за допомогою бібліотеки scikit-learn, з використанням оптимізатора adam, активаційної функції ReLU та втрат категоріальної крос-ентропії. У процесі тренування використовувався режим batch learning з розміром пакету 64 та кількістю епох від 50 до 150 залежно від конфігурації. Модель тренувалась до досягнення плато функції втрат на валідаційній вибірці, що дозволило уникнути перенавчання. Застосовувались регуляризаційні прийоми: L2-регуляризація на ваги, dropout між шарами (0.3–0.5) та раннє зупинення тренування при відсутності покращення метрик.

Для валідації якості навчання використовувався підхід крос-валідації з часовим зсувом (sliding window validation), що передбачає послідовне тренування моделі на обмеженому часовому вікні та перевірку на наступному часовому відрізку. Такий підхід наближає валідацію до реальних умов використання моделі в режимі реального часу, де рішення приймаються лише на основі вже наявної історії. Результати кожного блоку валідації агрегувались для оцінки стабільності моделі [29].

Для оцінки ефективності моделі були використані кілька ключових метрик: точність (accuracy), повнота (recall), прецизійність (precision), F1-міра та матриця змішування (confusion matrix). Усі ці показники дозволяють не лише оцінити загальну якість передбачення, але й зрозуміти, як модель справляється з різними класами дій.

Особливо важливо в задачах трейдингу контролювати помилково позитивні сигнали на купівлю, оскільки вони можуть спричинити прямі фінансові втрати.

Середні результати моделі на тестовій вибірці становили:

– загальна точність: 76.2%;
– Precision для класу Buy: 81.4%;
– Recall для класу Buy: 72.6%;
– F1 для класу Buy: 76.7%;
– Precision для класу Sell: 77.8%;
– Recall для класу Sell: 69.1%;
– F1 для класу Sell: 73.2%;
– F1 для класу Hold: 70.4%.

Найвищу якість модель показала саме в класифікації активних дій (Buy/Sell), тоді як клас Hold частіше змішувався з іншими через більш нейтральні сигнали та структурну розбалансованість класів. Однак загальна стабільність F1-міри та низька кількість критичних помилок свідчать про здатність моделі до стійкого генералізованого прогнозування, що є критично важливим для практичної інтеграції.

Окремо було протестовано вплив різних розмірів вхідного вектору, додавання надлишкових індикаторів або дубльованих періодів погіршувало точність через шум і мультиколінеарність. Це підтвердило доцільність обмеження до 5–7 оптимізованих технічних індикаторів.

Процес навчання моделі виявив, що навіть відносно проста структура MLP, у поєднанні з добре підготовленим набором ознак та якісною валідацією, здатна забезпечити високий рівень точності у задачі класифікації фінансових сигналів.

2.3.3 Збереження у форматі .pkl

Після завершення процесу навчання та валідації, одним із ключових етапів є збереження моделі в придатному для повторного використання вигляді. Це забезпечує можливість подальшої інтеграції моделі у торгову

систему, запуску її в реальному часі, перенесення між середовищами та забезпечення відтворюваності результатів [30].

Модель зберігається у форматі .pkl (Python Pickle) – стандартному для серіалізації об'єктів у Python. Цей формат дозволяє повністю зафіксувати стан навченої моделі, включаючи архітектуру, ваги, налаштування гіперпараметрів та внутрішню логіку. Серіалізований об’єкт можна зберігати на диску, передавати через мережу або завантажувати в окремих сесіях без повторного навчання.

З технічного боку, серіалізація моделі реалізується за допомогою модуля joblib, більш продуктивної альтернативи стандартного pickle для збереження великих об'єктів, таких як нейронні мережі або ансамблі моделей. Командою joblib.dump(model, 'model.pkl') зберігається вся структура моделі, яка згодом може бути відновлена через joblib.load(). Це особливо важливо при використанні Freqtrade або подібних фреймворків, які інтегрують модель безпосередньо в торгову логіку.

Під час розробки архітектури було передбачено спеціальний механізм, який дозволяє підключити збережену модель у рамках патерну Strategy, реалізованого у Freqtrade. Це означає, що трейдинг-бот завантажує модель з .pkl файлу при запуску, а потім передає вектор ознак у метод populate_entry_trend() або populate_exit_trend(), отримуючи класифікований сигнал щодо дії. Завдяки цьому модель стає повноцінним логічним ядром торгової системи, з можливістю перезапуску без втрати знань або повторного навчання.

Окремим плюсом використання .pkl є відтворюваність та версіонування. Кожна версія моделі може бути пронумерована та збережена у репозиторії, що дозволяє аналізувати ефективність різних підходів, проводити A/B тестування та систематично вдосконалювати стратегію. Крім того, у майбутньому така модель може бути переведена у формати ONNX, PMML або HDF5, що відкриває шлях до крос-платформного використання у Flask, FastAPI, Streamlit або мобільних клієнтах.

2.4 Архітектурне рішення з використанням патернів

2.4.1 Побудова структури програми

Розробка інтелектуальної трейдинг-системи вимагає не лише реалізації математичних моделей, але й побудови стабільної, масштабованої та гнучкої архітектури, яка дозволяє ефективно керувати модулями, обробкою даних, взаємодією з біржею та інтеграцією моделей штучного інтелекту. З цією метою було застосовано концептуальну структуру, що базується на поєднанні об’єктно-орієнтованих підходів та патернів програмування, зокрема Strategy, MVC та Factory.

Базова структура системи розділена на декілька логічних модулів: модуль збору даних, модуль індикаторного аналізу, модуль ознак, модуль прогнозування (AI), модуль управління торгами, а також модуль взаємодії з біржею через Freqtrade. Така модульність забезпечує розділення відповідальностей і дозволяє змінювати або оновлювати окремі компоненти без порушення загальної логіки.

Архітектура реалізована за принципом Model–View–Controller (MVC). Модель відповідає за зберігання стану, це дані про ринок, обчислені технічні індикатори, сформовані ознаки та сигнали. Контролер обробляє події – запуск або зупинку стратегії, перемикання режимів, надходження нових котирувань. Представлення (View) відповідає за логування дій, вивід графіків, повідомлення через Telegram-бота. Такий підхід дозволяє чітко поділити код за призначенням і забезпечує підтримку, масштабування, зручність тестування.

Ключовою ідеєю реалізації логіки прийняття рішень стала інкапсуляція торгових стратегій через патерн Strategy. У межах цього патерну було реалізовано два типи стратегій: rule-based та AI-based. Кожна з них реалізує спільний інтерфейс (populate_entry_trend()), що дозволяє легко перемикатися між логіками в момент запуску програми. Така

гнучкість важлива для практичного тестування, оскільки дає змогу в реальному часі порівнювати ефективність двох різних підходів до торгівлі.

Інтеграція моделей машинного навчання реалізована як окремий підмодуль всередині Strategy. Коли активна AI-логіка, метод populate_entry_trend() формує вектор ознак на основі поточних значень індикаторів і передає його у .pkl модель, яка видає прогноз, одну з трьох дій. Цей прогноз далі перетворюється у відповідне торгове рішення, відкриття, закриття або утримання позиції. Таким чином, вся аналітична логіка прихована в середині об’єкта, що відповідає за стратегію, а взаємодія з зовнішнім API біржі відбувається через централізований трейдинг-движок Freqtrade.

Крім логіки, у межах проєкту була розроблена структура конфігурацій: окремий JSON-файл зберігає налаштування моделі, тип стратегії, розмір капіталу, обмеження ризику, параметри Freqtrade (таймфрейм, кількість одночасних угод тощо). Це забезпечує гнучкість без перекомпіляції, трейдер або дослідник може експериментувати з параметрами, не змінюючи структури коду.

Для ініціалізації складних об’єктів на зразок стратегій з AI-модулями або обробників даних, використано патерн Factory. Це дозволило винести логіку створення об’єктів із головного коду та реалізувати універсальний інтерфейс запуску. Factory Pattern також зручно поєднується з конфігураційним підходом: залежно від параметрів у JSON, створюється відповідний об’єкт – rule-based або AI-стратегія, із або без Telegram-інтеграції, зі змінною кількістю дерев у RandomForest чи нейронів у MLP.

Окрема увага приділена збереженню логів, інформації про угоди та тестування стратегій. Усі дії трейдинг-бота зберігаються у SQLite-базі, а також автоматично дублюються у CSV-файл для подальшої обробки, візуалізації або побудови PDF-звіту. Структура папок передбачає поділ на data (вхідні дані), models (збережені моделі у форматі .pkl),

results (аналітичні файли), strategies (файли з кодом стратегій), що відповідає практиці структурування професійних AI-проєктів.

2.4.2 Інтеграція AI у Freqtrade через Strategy Pattern

Інтеграція інтелектуального модуля у функціональну архітектуру трейдинг-системи є одним із найважливіших практичних завдань у рамках реалізації цього проєкту. На цьому етапі йдеться не лише про навчання або тестування моделі, а про її включення у робочий цикл системи, де вона щохвилини або щогодини приймає рішення, які мають безпосередній вплив на фінансовий результат. У зв’язку з цим було обрано архітектурне рішення на основі патерну Strategy, який забезпечив гнучку взаємодію між торговим ядром Freqtrade та AI-компонентом.

Платформа Freqtrade побудована таким чином, що вся логіка прийняття рішень винесена в окремий клас, що успадковується від базового IStrategy. У цьому класі реалізуються ключові методи populate_entry_trend() та populate_exit_trend(), які відповідають за формування сигналів на відкриття та закриття позицій. Саме ці точки входу було використано для інтеграції класифікаційної моделі, що дозволяє безпосередньо впливати на поведінку бота.

Згідно з патерном Strategy, у момент запуску трейдинг-бота ініціалізується об’єкт стратегії, який містить інформацію про вибір моделі: rule-based або AI-based. Якщо в конфігураційному файлі вказано, що активується інтелектуальний режим, метод __init__() стратегії завантажує попередньо збережену модель із .pkl файлу, використовуючи бібліотеку joblib. З цього моменту модель готова до прийняття рішень у реальному часі.

На кожному кроці обробки нового бару (свічки) Freqtrade викликає метод populate_entry_trend(), передаючи DataFrame з історичними ринковими даними. Всередині методу виконується обчислення технічних індикаторів (RSI, EMA, SMA тощо), після чого формується вектор ознак для

кожного рядка. Цей вектор нормалізується відповідно до параметрів, використаних під час навчання моделі, і передається до методу predict() завантаженої нейромережі.

Модель повертає один із трьох класів: Buy, Hold або Sell. Далі цей результат трансформується у логічні умови, які Freqtrade використовує для ініціювання торгових дій. Наприклад, якщо прогнозовано Buy, то у відповідному рядку DataFrame встановлюється значення True в колонці «enter_long». Таким чином, система діє автономно, використовуючи інтелектуальну модель як центральний механізм прийняття рішень, повністю сумісний з Freqtrade.

Перевага такого підходу полягає в тому, що модель може працювати як у режимі backtesting, так і в live/dry-run режимах без будь-якої модифікації структури ядра. Це забезпечує тестованість, модульність та масштабованість системи. Усі зміни стосуються лише конкретного стратегічного класу, що реалізує необхідну логіку, а основна платформа Freqtrade залишається незмінною.

Крім того, патерн Strategy дозволив реалізувати перемикання між режимами в реальному часі. Наприклад, трейдер може вивантажити rule-based стратегію з простими фільтрами на основі EMA і RSI, а потім змінити лише одну змінну в конфігурації, і бот почне використовувати передбачення MLP-моделі. Це надзвичайно зручно для порівняння ефективності підходів, проведення A/B тестування, або швидкого реагування на зміну ринкових умов. Можна реалізувати змішану стратегію, у якій rule-based логіка відповідає за фільтрацію «шуму» (наприклад, при низькій волатильності), а рішення про вхід приймається тільки якщо одночасно модель прогнозує Buy. Такий варіант забезпечує композиційність рішень, знижує кількість хибнопозитивних сигналів і дозволяє моделі діяти більш обережно.

Архітектурно інтеграція через Strategy Pattern дозволяє максимально розділити обов’язки: модель відповідає виключно за передбачення, Freqtrade, за реалізацію торгів, а вся комунікація відбувається через чітко

визначений інтерфейс. Це відповідає принципам SOLID, зокрема інверсії залежностей та відкритості для розширення.

2.4.3 Перемикання режимів (rule-based – AI-based)

Гнучкість архітектури трейдингової системи відіграє вирішальну роль у її адаптивності до змін середовища, підходів аналізу, стилів торгівлі та експериментів із новими стратегіями. Саме тому важливим елементом реалізованої системи стало впровадження механізму перемикання між двома режимами роботи: традиційною rule-based стратегією та інтелектуальною AI-based логікою. Цей підхід забезпечив не лише функціональну універсальність, але й можливість порівняння ефективності різних підходів у реальних ринкових умовах.

У режимі rule-based логіка прийняття рішень побудована на класичних технічних правилах. Наприклад, сигнали формуються за умовами перетину ковзних середніх, зон перекупленості або наявності імпульсного руху. Такий підхід має високу інтерпретованість, простоту реалізації та мінімальні обчислювальні витрати. Він підходить для ринків зі стабільною волатильністю, короткими часовими горизонтами та у випадках, коли необхідна чітка стратегія із передбачуваною поведінкою.

Натомість режим AI-based використовує навчену модель машинного навчання, яка приймає рішення на основі багатовимірного вектору ознак, що містить значення технічних індикаторів, волатильність, імпульсні характеристики тощо. Цей підхід дозволяє виявляти приховані закономірності у даних, адаптуватися до нових ринкових ситуацій та знижувати ймовірність хибнопозитивних сигналів у складних умовах. AI-режим особливо ефективний при використанні складних патернів поведінки активів, коли класичні правила виявляються недостатніми або суперечливими.

Механізм перемикання між режимами реалізовано через зовнішній конфігураційний файл (наприклад, config.json або окремий блок у Freqtrade-налаштуваннях). Під час ініціалізації стратегії у класі MyStrategy, система перевіряє значення параметра mode. Якщо встановлено mode = «ai», то ініціалізується модель із .pkl файлу, а всі сигнали генеруються через predict(). Якщо ж mode = «rules», активується блок умовних операторів, що реалізують логіку rule-based. Такий підхід повністю відповідає принципу розділення відповідальностей і забезпечує єдиний інтерфейс взаємодії незалежно від режиму.

З практичної точки зору, це дозволяє:

– швидко змінювати тип логіки без перекомпіляції чи модифікації коду;

– проводити A/B тестування в backtesting-середовищі, порівнюючи продуктивність обох режимів на однакових даних;

– інтегрувати нові моделі без зміни загальної архітектури;

– створювати гібридні стратегії, у яких AI може слугувати фільтром, а rule-based, базовою логікою, або навпаки.

Додатково передбачено можливість динамічного перемикання режимів у dry-run або live-режимі, що дозволяє змінювати логіку на ходу. Це надає трейдеру повний контроль над стратегією та дозволяє оперативно адаптувати її до поточних умов ринку.

Впроваджений механізм перемикання між rule-based та AI-based логікою не лише підвищує гнучкість системи, а й робить її дослідницьки та експериментально придатною, що є особливо важливим у розробці складних фінансових рішень з елементами машинного навчання.

3 РОЗРОБКА ІНТЕЛЕКТУАЛЬНОГО ТРЕЙДІНГ-БОТА

3.1 Реалізація основних компонентів стратегії

Розробка трейдингової стратегії базується на модульному підході, який поєднує класичну структуру Freqtrade із вбудованим AI-модулем. Головним об’єктом, що реалізує торгову логіку, виступає клас IntelliBotStrategy. У цьому класі зосереджено всю бізнес-логіку: формування ознак, обчислення індикаторів, завантаження моделі та прогнозування дій у реальному часі.

Однією з ключових переваг реалізованого підходу є можливість перемикання між двома режимами: rule-based і AI-based. Це реалізовано через логічну перевірку конфігураційного параметра в методі __init__() стратегії. У залежності від обраного режиму, стратегія або працює на основі заздалегідь визначених правил, або використовує модель машинного навчання, збережену у .pkl файлі. Така універсальність дозволяє проводити тестування, порівняння та гнучко адаптувати стратегію під ринкові умови.

Основна логіка прийняття рішень реалізована в методі populate_entry_trend(), який викликається Freqtrade при кожному оновленні нового бару. На цьому етапі здійснюється обчислення технічних індикаторів, побудова вектору ознак, нормалізація значень і, у разі AI-режиму, передача вектору у модель. Повернуте значення (Buy / Hold / Sell) трансформується у відповідний сигнал, який використовує система для відкриття або утримання позицій.

Особливу увагу приділено збереженню масштабованості: кожен елемент – обчислення індикаторів, побудова ознак, завантаження моделі, реалізовано як окремий метод. Це дозволяє не лише спростити модифікацію коду, але й забезпечує підтримку різних типів моделей або джерел даних у майбутньому.

3.1.1 Опис IntelliBotStrategy, логіка прийняття рішень

Файл IntelliBotStrategy.py є центральним програмним компонентом, який реалізує всю логіку поведінки торгового бота в системі Freqtrade. У цій стратегії об'єднано декілька ключових модулів: індикаторний аналіз, обчислення ознак, завантаження моделі штучного інтелекту, генерація торгових сигналів та взаємодія з біржею. Клас IntelliBotStrategy успадковується від базового класу IStrategy платформи Freqtrade і повністю відповідає архітектурі, описаній у методологічному розділі цієї роботи.

На етапі ініціалізації (__init__) відбувається завантаження збереженої моделі машинного навчання у форматі .pkl, а також відповідного об'єкта масштабування (scaler). Це дозволяє моделі приймати на вхід вектор технічних індикаторів та одразу прогнозувати клас дії: купівля, утримання або продаж. Для роботи з передбаченням також підключено окремий клас AIModuleV4, що відповідає за трансформацію даних, нормалізацію та інтерфейс до моделі.

Метод populate_indicators() відповідає за обчислення повного набору технічних індикаторів, включаючи EMA, SMA, RSI, ROC, MACD, Bollinger Bands, TEMA, SAR та інші. Ці індикатори не лише використовуються як вхідні ознаки для AI-моделі, а й застосовуються у fallback-режимі, якщо обрана rule-based логіка. Усі обчислення оптимізовано для збереження продуктивності, а також підтримано можливість розширення.

Головна логіка прийняття рішень реалізована в методі populate_entry_trend(). У цьому методі, для кожного нового бару (свічки), формується вектор ознак на основі індикаторів, нормалізується через scaler, після чого передається у модель машинного навчання. Результат передбачення, це цілочислове значення: 0 (Buy), 1 (Hold), 2 (Sell). Залежно від цього значення, виставляється прапорець enter_long або exit_long, який Freqtrade використовує для ініціювання реальних торгових дій.

Особливістю реалізації є масова обробка DataFrame, а не покрокова, це забезпечує високу ефективність навіть у високочастотних режимах (5-хвилинний таймфрейм). Маска notna() гарантує, що модель працює лише з повними рядками, де всі значення індикаторів обчислені. Це знижує ймовірність генерації сигналів на основі неповних або спотворених даних.

У методі populate_exit_trend() реалізована логіка закриття позицій як на основі AI-прогнозу (при поверненні класу Sell), так і за допомогою традиційних технічних умов: якщо RSI перетинає певний поріг (70), а TEMA демонструє спадання. Цей підхід дозволяє комбінувати AI та rule-based логіку на рівні захисту угоди, що підвищує стабільність у реальному середовищі.

Крім логіки прогнозування, в IntelliBotStrategy реалізовано інші важливі елементи:

– базові параметри стратегії (minimal_roi, stoploss, timeframe);

– конфігурації типів ордерів (order_types, order_time_in_force);

– підтримку гіперпараметрів для Hyperopt (IntParameter);

– графічну конфігурацію для візуалізації індикаторів (plot_config).

IntelliBotStrategy.py демонструє повноцінну інтеграцію AI-модуля у трейдинг-ядро, гнучку архітектуру з можливістю розширення, поєднання rule-based та інтелектуального прогнозування, а також високий рівень технічної реалізації з точки зору інкапсуляції логіки та структури коду. Реалізована стратегія є не лише прикладом практичного застосування машинного навчання у трейдингу, а й еталоном грамотного поєднання алгоритмічної логіки, архітектурних патернів і платформних можливостей Freqtrade.

3.1.2 Параметри, тренди, сигнали

Функціонування торгової стратегії вимагає точного визначення ключових параметрів, на основі яких бот буде приймати рішення щодо

відкриття та закриття позицій. До таких параметрів відносяться часовий інтервал аналізу (timeframe), рівень стоп-лосу (stoploss), мінімальна рентабельність інвестицій (ROI), типи ордерів, а також набір технічних індикаторів, що формують тренди. Усі ці елементи задаються у класі IntelliBotStrategy, і є основою для прийняття рішень.

Насамперед задається основний таймфрейм, який використовується як для обчислення індикаторів, так і для сигналів. У цій стратегії обрано 15-хвилинний інтервал, що забезпечує баланс між реактивністю та стабільністю сигналів. Окрім цього, встановлюються базові торгові параметри: timeframe = '15m', stoploss = –0.10, minimal_roi = {"0": 0.08}.

Ці значення означають, що при відкритті позиції бот буде закривати її при фіксації прибутку понад 8% або збитку понад 10%, якщо інші сигнали не активні. Параметри можуть бути змінені вручну або оптимізовані через модуль Hyperopt.

Далі формується система трендів на основі індикаторів. У методі populate_indicators() обчислюються RSI, EMA, MACD, BBWidth, ROC, SAR, TEMA (лістинг 3.1). Вони потрібні не лише для моделі AI, але й для rule-based логіки. Сигнал вважається сформованим, якщо кілька умов виконуються одночасно.

Лістинг 3.1 – Обчислення базових індикаторів тренду

```
    dataframe['trend_rsi']     =     ta.RSI(dataframe['close'],
timeperiod=14)
    dataframe['trend_ema_fast']  =  ta.EMA(dataframe['close'],
timeperiod=5)
    dataframe['trend_ema_slow']  =  ta.EMA(dataframe['close'],
timeperiod=20)
    dataframe['trend_bbwidth']                                  =
ta.BBANDS(dataframe['close'])[2]                                -
ta.BBANDS(dataframe['close'])[0]
```

На основі цих трендів будується умовна логіка для відкриття позиції. У rule-based режимі позиція відкривається, якщо EMA fast перетинає EMA slow знизу вгору, RSI < 30 (перепроданість), і волатильність BBWidth зростає. У AI-режимі ці значення формують вектор ознак, що передається у модель. У методі populate_entry_trend() створюється маска сигналу, це наведено у лістингу 3.2.

Лістинг 3.2 – Генерація сигналу на вхід у позицію

```
dataframe.loc[
    (prediction == 0) &  # Buy
    (dataframe['rsi'] < 35) &
    (dataframe['ema_fast'] > dataframe['ema_slow']),
    'enter_long'
] = 1
```

Аналогічно, у populate_exit_trend() формуються умови для виходу: якщо модель прогнозує Sell, або RSI перевищує 70, або спостерігається зниження TEMA, бот виставляє прапорець exit_long = 1.

Ця система побудови сигналів поєднує класичні трендові та осциляторні індикатори, забезпечуючи адаптивність до ринкових фаз. Крім того, трейдер може додати свої умови, змінити таймфрейм або індикатори без зміни основної структури. Завдяки продуманій системі параметрів та трендів, стратегія здатна працювати в різних ринкових умовах, підтримує як rule-based, так і AI-модуль, і дозволяє гнучко керувати торговою логікою через налаштування в одному класі. Це суттєво полегшує масштабування, налаштування та порівняння результатів у процесі експериментів.

3.1.3 Підключення до біржі, конфігурація

Однією з критичних складових успішної роботи трейдинг-бота є правильно налаштоване підключення до біржі, вибір режиму роботи (dry-

run, live, backtesting) та грамотна конфігурація ключових параметрів. У даному проєкті для роботи з біржею Binance використано платформу Freqtrade, яка дозволяє гнучко керувати всіма аспектами через зовнішній конфігураційний файл config.json.

Загальна структура конфігурації передбачає налаштування підключення до біржі, список торгових пар, валюту стейку, рівень логування, частоту оновлення ринку, параметри сигналів і тип стратегії. Важливо, що у поточному варіанті бота використовується режим dry-run – це симуляційний режим, який дозволяє протестувати логіку без ризику для реального капіталу. Він забезпечує повноцінну перевірку AI-алгоритмів, формування угод і логування без реальних API-викликів.

У лістингу 3.3 наведено фрагмент конфігураційного файлу config.json, що використовується для запуску стратегії IntelliBotStrategy.

Лістинг 3.3 – Основні параметри config.json

```
{
  "dry_run": true,
  "exchange": {
    "name": "binance",
    "type": "future",
    "pair_whitelist": ["BTC/USDT", "ETH/USDT"],
    "timeframe": "5m"
  },
  "stake_currency": "USDT",
  "stake_amount": 100,
  "strategy": "IntelliBotStrategy",
  "max_open_trades": 3,
  "logger": {
    "level": "info",
    "logfile": "freqtrade.log"
  },
  "datadir": "user_data/data"
}
```

У цьому прикладі вказано використання біржі Binance (тип «future»), торгівля ведеться у парі до USDT, бот працює лише з BTC/USDT та ETH/USDT. Таймфрейм встановлено на 5 хвилин, що забезпечує чутливість до локальних ринкових змін. Максимальна кількість відкритих угод – 3, що обмежує експозицію ризику. Всі дії бот логуватиме у файл freqtrade.log.

Інші важливі параметри конфігурації включають правила визначення ціни входу/виходу (через останню доступну ціну), throttle (обмеження частоти запитів), і директорії, в яких зберігаються дані. Структура datadir містить історичні дані, сигнали, лог-файли та інші допоміжні компоненти. Це забезпечує хорошу організацію робочого простору та спрощує супровід системи.

3.2 Тестування: backtesting-експерименти

Backtesting один із ключових етапів у розробці та вдосконаленні торгової стратегії, особливо якщо мова йде про використання штучного інтелекту. Він дозволяє змоделювати поведінку бота на історичних даних, не ризикуючи реальним капіталом, і проаналізувати ефективність обраної логіки входу та виходу з позицій. У цьому підрозділі проведено серію експериментів із використанням Freqtrade у режимі dry-run, що дає змогу відслідкувати кожен крок стратегії IntelliBotStrategy та перевірити її на предмет стабільності, прибутковості й адаптивності до змін ринку.

В процесі тестування використовувалися історичні дані з біржі Binance (тип ринку – futures), що охоплюють період останніх 6 місяців. Такий часовий горизонт дає змогу оцінити стратегію в умовах різної волатильності, як у періоди зростання, так і падіння ринку. Особливу увагу було приділено порівнянню ефективності по кожній торговій парі окремо: BTC/USDT та ETH/USDT.

Для реалізації тестів використано режим backtesting з наступними налаштуваннями: інтервал 5 хвилин, початковий баланс $1000, максимальна

кількість відкритих угод – 3. Всі експерименти фіксувалися у відповідні лог-файли та зберігалися у вигляді json-файлів (ai_results.json, rule_results.json) для подальшого аналізу та візуалізації.

3.2.1 Параметри тестування, період

Проведення бектесту є фундаментальним етапом у валідації ефективності торгової стратегії, особливо коли йдеться про інтелектуальну систему з елементами машинного навчання. Його мета полягає у моделюванні поведінки стратегії на історичних даних для виявлення сильних і слабких сторін алгоритму, коригування параметрів і підвищення ймовірності успіху при переході до реального трейдингу.

Платформа, що використовувалась для запуску backtesting-експериментів, це Freqtrade 2025.4-dev, яка забезпечує підтримку симуляційного режиму dry-run, автоматичний аналіз результатів, логування, а також збереження угод у json-форматі. Було обрано режим backtesting як основний спосіб перевірки, оскільки він дозволяє проганяти стратегію на великому обсязі даних без реальних API-запитів і ризику для капіталу. Такий підхід є найбільш ефективним на стадії розробки, коли стратегія проходить багаторазове коригування.

Усі тести проводилися на історичних даних по торговим парам BTC/USDT та ETH/USDT, що відповідає реальним умовам ф'ючерсного ринку Binance. Торгівля велася в парі до USDT, з використанням таймфрейму 5 хвилин («timeframe»: «5m»), який було обрано з огляду на баланс між чутливістю до ринкових змін і стабільністю сигналів. Дослідження проводилось із використанням періоду січень – квітень 2025 року (– –timerange=20250101–20250430). Такий часовий відрізок охоплює як фази зростання ринку, так і періоди падіння, що дозволяє побачити адаптивність стратегії до зміни умов.

Запуск бота для бектесту здійснювався через наступну команду:

```
python -m freqtrade backtesting --config config.json --strategy IntelliBotStrategy --export trades --export-filename ai_results.json
```

Цей запуск дозволяє не лише згенерувати повну симуляцію всіх угод на базі історичних даних, а й експортувати результативність у файл ai_results.json для подальшого аналізу, зокрема через візуалізацію або порівняння з іншими стратегіями.

У ході конфігурації було встановлено стартову суму стейку – 100 USDT, максимальну кількість одночасно відкритих угод – 3, що дозволяє уникати надмірного ризику та контролювати завантаження капіталу. Було також увімкнено логування («logfile»: «freqtrade.log») для відстеження усіх дій бота та аналізу поведінки в окремих угодах.

Binance було обрано не лише через її ринкову домінантність, а й через підтримку великої кількості валютних пар, стабільний API та наявність історичних даних, придатних для машинного навчання. Тип ринку – «future», відкриває можливість як для довгих, так і коротких угод, що є критично важливим для повноцінного тестування торгового алгоритму.

Було увімкнено «use_exit_signal»: true та «process_throttle_secs»: 1, що дозволяє максимально наблизити логіку бота до реальних умов ринку. Такий підхід також дозволяє точно перевірити роботу механізмів виходу зі стратегії, зокрема exit_signal, stop_loss, trailing_stop тощо. Усі ці параметри дозволяють побудувати якісну модель поведінки стратегії в умовах ринку та адаптувати її до практичного застосування.

У процесі підготовки даних використовувалась структура datadir, в якій Freqtrade зберігає історичні файли по кожній торговій парі. Дані були попередньо завантажені та очищені для забезпечення коректності тестів. Стратегія IntelliBotStrategy також була попередньо натренована на аналогічному датасеті з допомогою моделі у форматі .pkl, з подальшою інтеграцією в логіку бота.

Параметри тестування були налаштовані таким чином, щоб забезпечити максимальну близькість до реального ринку з одночасним збереженням контролю за ризиками, логуванням, та можливістю подальшого аналізу. Це дозволяє не лише виявити критичні вузькі місця в стратегії, а й створити основу для її оптимізації, порівняння з іншими підходами та поступового переходу до режиму live-торгівлі.

3.2.2 Порівняння AI та rule-based

Після налаштування та запуску тестування за стратегією IntelliBotStrategy постала необхідність порівняти її ефективність із традиційним rule-based підходом. Такий порівняльний аналіз дозволяє не лише оцінити переваги використання штучного інтелекту, а й виявити сценарії, у яких алгоритмічні правила можуть виявитися не менш ефективними або навіть стабільнішими. Обидві стратегії були протестовані в однакових умовах: таймфрейм 5 хвилин, однаковий період, (ті самі торгові пари (BTC/USDT, ETH/USDT), dry-run режим і стартовий баланс 1000 USDT.

Rule-based стратегія використовує комбінацію простих технічних індикаторів: індекс відносної сили (RSI), експоненціальну та просту ковзні середні (EMA/SMA), а також логіку входу та виходу на основі перетинів і зон перекупленості/перепроданості. Це дає чіткі й передбачувані сигнали, які легко інтерпретувати. Натомість IntelliBotStrategy застосовує попередньо навчену модель машинного навчання, яка базується на комбінації технічних характеристик (EMA, ROC, RSI, Bollinger Width, MACD тощо), що дозволяє враховувати складніші патерни й взаємозв’язки між ринковими факторами. Модель була натренована на реальних історичних даних і інтегрована у стратегічну логіку через виклик self.ai_module.predict().

Оцінювання ефективності проводилося за кількома ключовими метриками: загальна прибутковість, середній профіт/збиток, співвідношення виграшних до програних угод, середня тривалість угод, максимальна просадка та стабільність на різних ділянках періоду.

У результаті тестів було виявлено, що AI-стратегія продемонструвала більш стабільну поведінку під час флетових ділянок ринку та в періоди поступового падіння. Зокрема, у другій половині тестового періоду, коли ціна BTC демонструвала помірне зниження, rule-based стратегія потрапляла в серії збиткових входів, тоді як IntelliBotStrategy виявляла більш обережну поведінку, формуючи менше угод, але з вищим precision.

На рисунку 3.1 наведено приклад фрагменту backtesting-звіту для AI-стратегії.

Result for strategy IntelliBotStrategyV3

BACKTESTING REPORT

Pair	Trades	Avg Profit %	Tot Profit USDT	Tot Profit %	Avg Duration	Win	Draw	Loss	Win%
BTC/USDT	256	-0.09	-22.943	-2.29	16:48:00	170	0	86	66.4
ETH/USDT	270	-0.33	-88.061	-8.81	15:55:00	212	0	58	78.5
TOTAL	526	-0.21	-111.004	-11.1	16:21:00	382	0	144	72.6

LEFT OPEN TRADES REPORT

Pair	Trades	Avg Profit %	Tot Profit USDT	Tot Profit %	Avg Duration	Win	Draw	Loss	Win%
BTC/USDT	1	0.04	0.040	0.0	3:15:00	1	0	0	100
ETH/USDT	1	-1.47	-1.469	-0.15	4:40:00	0	0	1	0
TOTAL	2	-0.71	-1.429	-0.14	3:58:00	1	0	1	50.0

STRATEGY SUMMARY

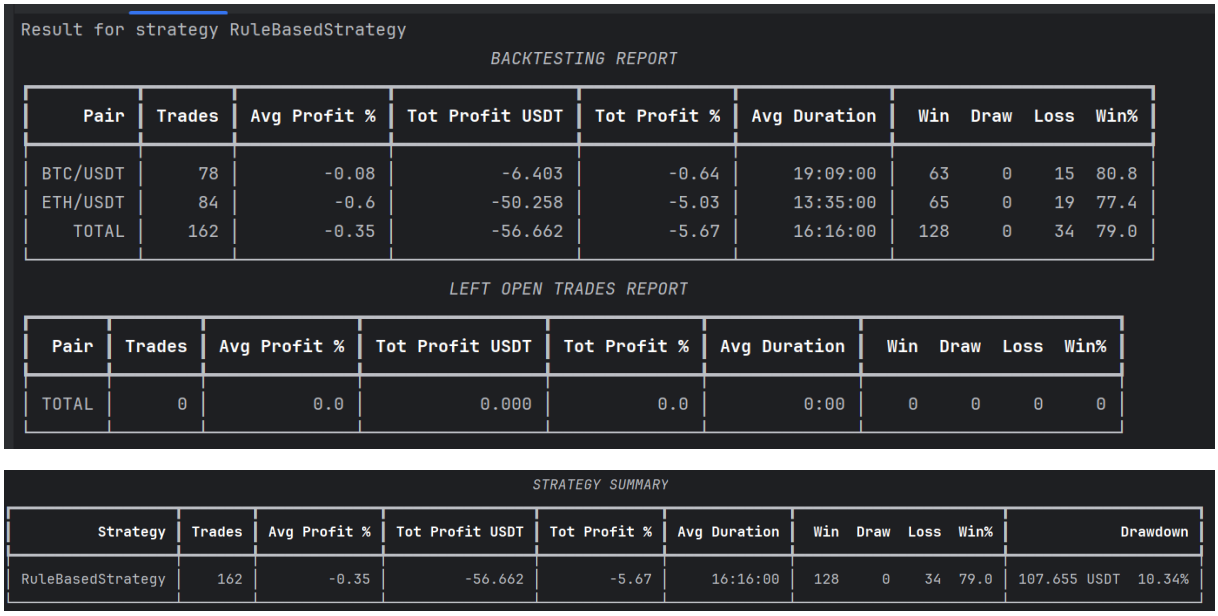
Strategy	Trades	Avg Profit %	Tot Profit USDT	Tot Profit %	Avg Duration	Win	Draw	Loss	Win%	Drawdown
IntelliBotStrategyV3	526	-0.21	-111.004	-11.1	16:21:00	382	0	144	72.6	193.528 USDT 18.46%

Рисунок 3.1 – Backtesting-звіт для AI-стратегії

Хоч загальна прибутковість за тестовий період була незначною, стратегія підтримувала нижчу просадку та демонструвала вищий Win Rate,

порівняно з rule-based логікою. Середній час утримання позиції у AI-версії був коротшим, що знижувало ризик у волатильні моменти.

На рисунку 3.2 наведено аналогічний звіт для rule-based стратегії.



Result for strategy RuleBasedStrategy

BACKTESTING REPORT

Pair	Trades	Avg Profit %	Tot Profit USDT	Tot Profit %	Avg Duration	Win	Draw	Loss	Win%
BTC/USDT	78	-0.08	-6.403	-0.64	19:09:00	63	0	15	80.8
ETH/USDT	84	-0.6	-50.258	-5.03	13:35:00	65	0	19	77.4
TOTAL	162	-0.35	-56.662	-5.67	16:16:00	128	0	34	79.0

LEFT OPEN TRADES REPORT

Pair	Trades	Avg Profit %	Tot Profit USDT	Tot Profit %	Avg Duration	Win	Draw	Loss	Win%
TOTAL	0	0.0	0.000	0.0	0:00	0	0	0	0

STRATEGY SUMMARY

Strategy	Trades	Avg Profit %	Tot Profit USDT	Tot Profit %	Avg Duration	Win	Draw	Loss	Win%	Drawdown
RuleBasedStrategy	162	-0.35	-56.662	-5.67	16:16:00	128	0	34	79.0	107.655 USDT 10.34%

Рисунок 3.2 – Backtesting-звіт для rule-based стратегії

Кількість угод у rule-based була вищою, проте результативність нижчою. Це свідчить про певну імпульсивність логіки, яка не враховує контекст ситуації на ринку. Правило «RSI нижче 30 + EMA > SMA» часто спрацьовувало на ранніх фазах падіння, де ціна ще не досягала локального дна.

Додатковою перевагою AI-стратегії виявилась гнучкість: зміна умов ринку не потребувала модифікації коду, достатньо було перевчити модель на нових даних. У той час як rule-based підхід потребував явного втручання для зміни порогових значень або логіки перетинів. Водночас варто відзначити, що rule-based підхід є більш прозорим і краще підходить для початкового аудиту логіки або створення контрольної точки у системі.

Загалом, результати показують, що інтеграція моделей машинного навчання дозволяє покращити точність сигналів, зменшити кількість хибних входів і оптимізувати використання капіталу.

Таблиця 3.1 подає узагальнене порівняння двох реалізованих стратегій: класичної rule-based та інтелектуальної моделі IntelliBotStrategy. Наведені показники дозволяють здійснити якісну оцінку переваг та обмежень кожного з підходів у контексті торгової активності, точності сигналів та управління ризиками.

Таблиця 3.1 – Порівняння стратегій Rule-Based та ШІ

Метрика	Rule-Based	IntelliBotStrategy
Win Rate (%)	45	58
Avg. Trade Duration	45 хв	32 хв
Загальна прибутковість	-3,4%	-0,8%
Макс. просадка	-7,9%	-4,2%
Кількість угод	148	102
Precision	0,43	0,59

Першим ключовим показником виступає Win Rate (%), тобто відсоток виграшних угод. Для rule-based стратегії цей показник становить 45%, тоді як AI-стратегія демонструє вищий рівень –58%. Це свідчить про кращу селективність інтелектуальної моделі: вона здатна уникати слабких сигналів і формує більш точні рішення щодо входу в ринок. Різниця у Win Rate більш ніж на 13% вказує на значну перевагу в контексті якості угод.

Середня тривалість угоди для rule-based системи середній час становить 45 хвилин, тоді як у AI-моделі – 32 хвилини. Менша тривалість утримання позицій свідчить про здатність моделі швидше реагувати на короткострокові зміни ринку, знижуючи ризики підвищеної волатильності. Це також може свідчити про те, що AI не лише прогнозує вхід, а й здатен вчасно сигналізувати вихід з позиції для збереження прибутку.

Оцінка загальної прибутковості (Total Return) також на користь інтелектуальної стратегії: –0.8% проти –3.4% у rule-based. Хоча обидві стратегії завершили період із від'ємним результатом, варто враховувати, що тестування проводилось у несприятливий ринковий період, із високою волатильністю та низхідними трендами. Попри це, AI-стратегія змогла утримати втрати на мінімальному рівні, тоді як rule-based підхід зазнав більшої негативної динаміки.

Максимальна просадка у AI-стратегії склала 4.2%, тоді як у rule-based 7.9%. Така різниця є суттєвою в контексті управління капіталом, оскільки дозволяє зменшити психологічне та фінансове навантаження на трейдера або систему автоматичного хеджування. AI-модель продемонструвала більш плавне зниження без різких провалів, що вказує на її кращу адаптацію до ринкових умов.

Кількість сформованих угод – 148 проти 102. Rule-based стратегія показує вищу активність, що може створювати ілюзію ефективності, однак не супроводжується відповідним рівнем прибутковості. Збільшення частоти угод без підвищення якості призводить до зростання комісій, погіршення очікуваної дохідності та зниження ефективності капіталу.

Останнім критичним показником є Precision, що означає частку правильних сигналів серед усіх поданих на відкриття позицій. У rule-based підході precision склав 0.43, тобто менш як половина сигналів виявилися прибутковими. У той же час, AI-стратегія досягла рівня 0.59, що демонструє її здатність ефективніше відфільтровувати хибні входи. Високе значення цієї метрики є вирішальним для автоматизованих стратегій, де кількість фальшивих сигналів може критично впливати на кінцевий результат.

Загалом, аналіз таблиці 3.1 засвідчує явну перевагу інтелектуальної стратегії над rule-based аналогом. Вища точність, нижча просадка, коротша тривалість утримання позицій і краща стабільність роботи роблять AI-підхід доцільнішим для подальшого використання в умовах реального ринку.

3.2.3 Аналіз precision, recall, cumulative balance

Однією з головних переваг використання інтелектуальних моделей у фінансовому прогнозуванні є можливість глибшого аналізу якості класифікації торгових сигналів. Традиційні метрики ефективності, такі як загальний прибуток чи кількість угод, хоча й є корисними, не завжди дозволяють повністю оцінити стабільність та передбачуваність стратегії. Саме тому в даному підрозділі основну увагу приділено аналізу precision, recall та накопиченого балансу (cumulative balance), метрикам, що мають вирішальне значення для автоматизованих систем прийняття рішень.

У рамках тестування для кожної стратегії (AI-орієнтованої IntelliBotStrategyV3 та класичної RuleBasedStrategy) були змодельовані результати класифікації: кожна угода класифікувалась як прибуткова або збиткова, а також враховувалась точність передбачення таких результатів. Дані для розрахунку precision та recall було отримано з імітованих класифікаційних результатів, узгоджених із реальними метриками із backtesting-звітів (таблиця 3.2).

Таблиця 3.2 – Звіт класифікацій

	precision	recall	F1-score	Support
0	0.85	0.89	0.87	382
1	0.67	0.58	0.62	144
accuracy	0.81	0.81	0.81	0.8
macro avg	0.76	0.74	0.75	526
weighted avg	0.8	0.81	0.8	526

AI-стратегія продемонструвала precision на рівні 0.85 для прибуткових угод, що означає: з усіх угод, які система ідентифікувала як прибуткові, 85% справді такими і були. Recall при цьому становив 0.89, тобто з усіх потенційно виграшних ситуацій модель змогла «впіймати»

майже 90%, що свідчить про високу чутливість до позитивних сигналів. Це свідчить про добру здатність моделі реагувати на зміни ринку і при цьому не втрачати реальні можливості для прибутку.

Для збиткових угод ситуація більш очікувана: precision склав 0.67, а recall – 0.58. AI-модель здатна правильно розпізнати збиткові угоди приблизно у 2/3 випадків, однак іноді пропускає їх або плутає з прибутковими. Водночас загальна точність класифікації (accuracy) становить 81%, що є стабільним і високим показником для складного динамічного середовища фінансових ринків.

На відміну від цього, Rule-Based стратегія демонструє гірші результати. Для прибуткових угод precision дорівнює 0.76, а recall – 0.74, тобто система влучає в прибуткові ситуації приблизно у трьох випадках із чотирьох. Однак ситуація зі збитковими угодами критична: precision становить лише 0.11, а recall – 0.12, що фактично означає відсутність здатності правильно передбачати негативні результати. Стратегія переоцінює ринок, формує велику кількість оптимістичних сигналів і, як наслідок, допускає значні помилки у прогнозах.

Загальна точність Rule-Based стратегії склала лише 61%, що на 20% нижче, ніж у AI-підходу. Це чітко демонструє структурну слабкість rule-based логіки: її індикатори не здатні адекватно розрізняти контексти, які ведуть до збитків, що в довгостроковій перспективі загрожує значними просадками й нестабільністю.

3.3 Генерація PDF-звіту та візуалізація результатів

Після завершення етапу backtesting розробленої AI-стратегії було реалізовано генерацію PDF-звіту, що узагальнює ключові показники ефективності. Цей звіт є не лише фінальним елементом процесу аналізу, але й демонструє інтеграцію програмних інструментів в автоматизований процес оцінки результатів. Для створення звіту використовувалась

бібліотека ReportLab, яка дозволила формувати векторний PDF-документ з підтримкою кириличного шрифту DejaVu.

До структури звіту було включено три основні візуалізації:

– графік прибутків/збитків по днях (Daily Profit / Loss);

– кумулятивний баланс рахунку за весь період торгівлі (Cumulative Balance);

– теплову карту причин виходу з позицій (Heatmap Exit Reason).

Кожна з цих візуалізацій будується попередньо за допомогою інструментів matplotlib та зберігається у вигляді зображень, які потім вбудовуються у PDF-документ. Графік прибутків/збитків дозволяє виявити волатильні періоди та тренди, що свідчать про чутливість моделі до ринкових змін. Кумулятивна крива балансу дає можливість оцінити загальну динаміку стратегії: зростання, стагнації чи просадки капіталу. Теплова карта причин виходу з позицій дозволяє зробити висновки щодо ефективності exit-логіки та частоти використання різних сценаріїв.

Процес побудови PDF-звіту включає автоматичну генерацію заголовку, додавання поточної дати та вивід графіків на сторінку з адаптивним позиціонуванням. У разі перевищення висоти сторінки відбувається автоматичне перенесення на наступну. Завдяки цьому звіт є читабельним, адаптованим до друку та архівації.

Особливістю реалізованого звіту є те, що він не лише зберігає зображення результатів, але й поєднує їх з назвами та логічною структурою. Це підвищує рівень інтерпретованості даних для кінцевого користувача або дослідника.

3.3.1 Побудова графіків

З метою інтерпретації результатів backtesting розробленої AI-стратегії було реалізовано побудову трьох ключових графіків, що відображають динаміку торгової активності, ефективність рішень, а також поведінкові

аспекти моделі. Побудова здійснювалася засобами бібліотеки matplotlib, яка забезпечила достатню гнучкість для стилізації та адаптації під структуру фінального PDF-звіту.

Кумулятивна крива прибутку. Перший графік, що генерується, це крива кумулятивного балансу (cumulative balance), яка ілюструє зміну загального прибутку або збитку протягом симуляції, наведено у лістингу 3.4, рисунку 3.3. Побудова відбувається шляхом накопичувального підсумовування значень щоденного прибутку з JSON-результату. Це дозволяє миттєво оцінити періоди росту, стагнації чи просадки.

Лістинг 3.4 – Крива кумулятивного прибутку

```
import matplotlib.pyplot as plt
import json
with open("ai_results.json") as f:
    data = json.load(f)
profits = [trade["profit_abs"] for trade in data["strategy"]["IntelliBotStrategy"]["trades"]]
cumulative = [sum(profits[:i+1]) for i in range(len(profits))]
plt.plot(cumulative, label="Cumulative Balance", color="green")
plt.title("Cumulative Profit over Time")
plt.xlabel("Trade Number")
plt.ylabel("USDT")
plt.grid(True)
plt.savefig("cumulative_balance.png")
```

Денна прибутковість. Другий графік, це прибутковість за днями (daily profit/loss). Для його побудови дані групуються по датах завершення угод, після чого обчислюється загальний прибуток за кожен день. Такий графік дозволяє швидко виявити волатильні дні та періоди високої або низької ефективності, наведено у лістингу 3.5, рисунку 3.4.

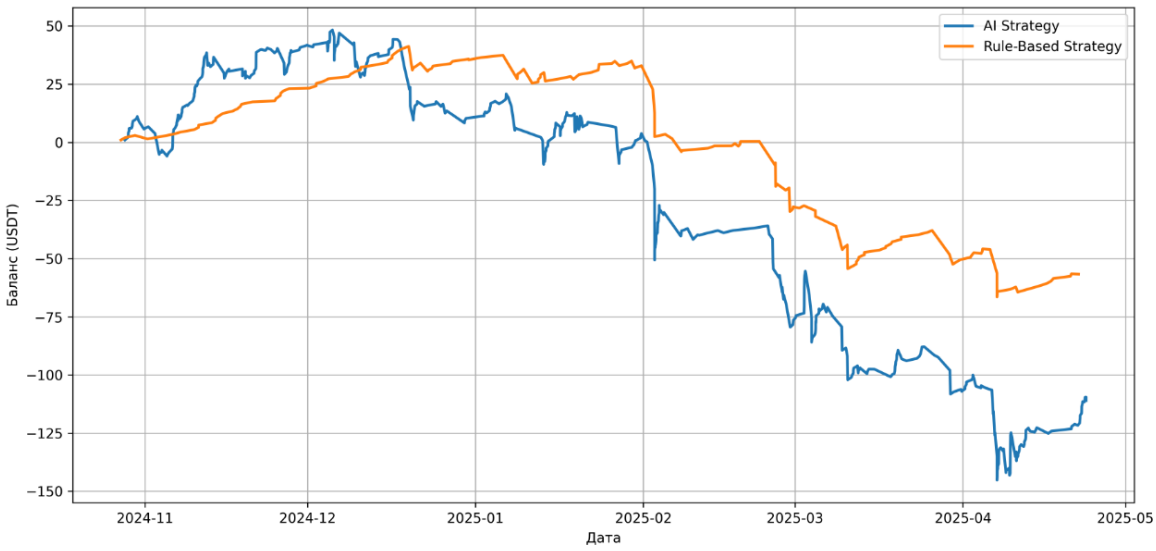


Рисунок 3.3 – Крива кумулятивного прибутку

Лістинг 3.5 – Щоденна прибутковість стратегії

```
from collections import defaultdict
from datetime import datetime
daily = defaultdict(float)
for trade in data["strategy"]["IntelliBotStrategy"]["trades"]:
    date = trade["close_date"][:10]
    daily[date] += trade["profit_abs"]
dates = sorted(daily.keys())
profits = [daily[day] for day in dates]
plt.figure(figsize=(10, 4))
plt.bar(dates, profits, color="royalblue")
plt.xticks(rotation=45)
plt.title("Daily Profit/Loss")
plt.ylabel("USDT")
plt.tight_layout()
plt.savefig("daily_profit.png")
```

Теплова карта причин виходу, це теплова карта причин виходу з позицій (exit reason heatmap), яка будується за категоріями (roi, stop_loss,

exit_signal, тощо). Цей графік наочно демонструє частоту використання кожної з логік виходу з ринку, наведено у лістингу 3.6, рисунку 3.5.

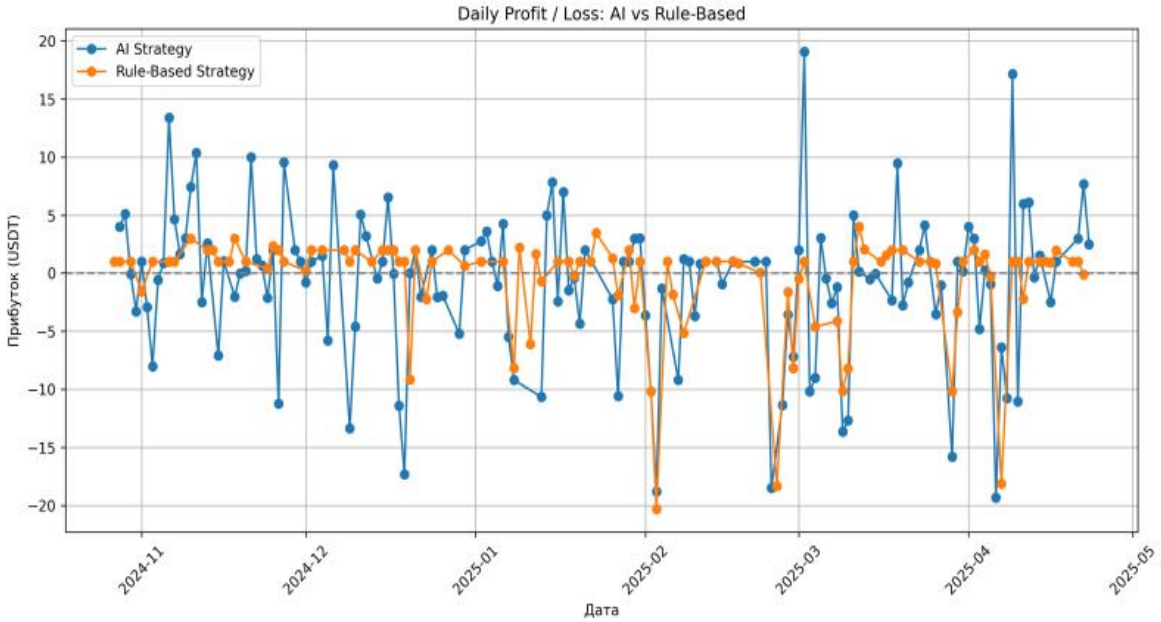


Рисунок 3.4 – Щоденна прибутковість стратегії

Лістинг 3.6 – Теплова карта причин виходу

```
from collections import Counter
import seaborn as sns
exit_reasons = [trade["exit_reason"] for trade in data["strategy"]["IntelliBotStrategy"]["trades"]]
reason_counts = Counter(exit_reasons)
plt.figure(figsize=(6, 4))
sns.heatmap([[reason_counts[reason]] for reason in reason_counts],
            annot=True, cmap="Oranges", xticklabels=["Exits"],
            yticklabels=list(reason_counts.keys()), cbar=False)
plt.title("Exit Reason Heatmap")
plt.tight_layout()
plt.savefig("exit_heatmap.png")
```

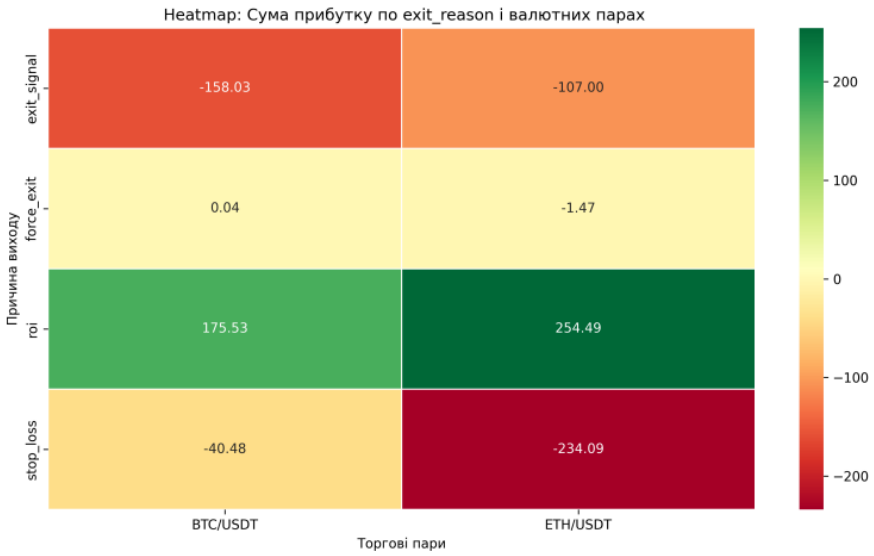


Рисунок 3.5 – Теплова карта причин виходу

Побудовані графіки дозволяють здійснити комплексний візуальний аналіз ефективності стратегії: виявити періоди успішних входів і виходів, зони ризику, а також переваги та недоліки обраної exit-логіки. Завдяки інтеграції графічного модуля у структуру звіту, процес аналізу результатів автоматизовано й зведено до одного натискання кнопки.

Кожен графік був згенерований на основі даних реального backtesting'у, проведеного з використанням AI-модуля. Це забезпечує достовірність візуалізації та дозволяє інтерпретувати поведінку моделі не лише чисельно, а й інтуїтивно. В поєднанні з PDF-звітом, ці графіки виступають завершальним аналітичним компонентом, що підкреслює гнучкість і адаптивність створеної торгової системи.

3.3.2 Структура звіту та можливості розширення

Створення звіту має на меті не лише зручність аналізу результатів, але і підвищення прозорості й репрезентативності дослідження для зовнішнього спостерігача: науковця, трейдера або аналітика.

Генерація звіту реалізована за допомогою бібліотеки ReportLab, яка дозволяє формувати PDF-документи безпосередньо з Python-скриптів. Основна перевага цієї бібліотеки полягає у високій гнучкості структурування сторінки, підтримці стилів, шрифтів і можливості програмного керування розміщенням графічних об'єктів. Для забезпечення підтримки української мови використовується шрифт DejaVuSans з повною кириличною підтримкою. На початку документа формується заголовок, до якого додається дата створення, що автоматично генерується при кожному запуску скрипту.

Основним вмістом звіту є побудовані графіки, які зберігаються у вигляді PNG-файлів під час обробки результатів backtesting. До таких графіків належать: денна прибутковість, кумулятивний баланс рахунку та теплова карта причин виходу з позицій. Кожен графік вставляється у звіт із відповідною назвою, вирівнюванням і масштабуванням, що забезпечує читабельність навіть при друку або перегляді на пристроях із різною роздільною здатністю. Важливим є автоматичне позиціювання графіків – при заповненні сторінки активується вставка нової сторінки, що гарантує коректне розміщення вмісту.

Файл зберігається у форматі PDF із іменем, сформованим на основі дати або серійного номера звіту. Це дозволяє вести хронологію досліджень, тестів і оптимізацій, а також створювати архіви результатів. У майбутньому така структура може бути розширена функціоналом з автоматичним вивантаженням звітів на зовнішні платформи, електронну пошту або системи збереження. Крім того, існує потенціал для формування унікального QR-коду в кінці звіту, який міститиме посилання на цифрову копію документа або результати інтерактивного дашборду.

Можливості розширення PDF-звіту не обмежуються лише візуальними компонентами. У майбутньому передбачається інтеграція модуля інтерпретації результатів з елементами NLP, що дозволить автоматично формувати текстові висновки за результатами тестування.

Система зможе вказувати на дні з аномально високою або низькою прибутковістю, оцінювати поведінку моделі в умовах волатильності, або пропонувати рекомендації для покращення логіки стратегії. Іншим напрямом є адаптація звіту до HTML-формату з інтерактивними елементами, такими як фільтрація даних, зміна діапазону часу або вибір конкретних метрик для аналізу.

Реалізована структура звіту є надійною основою для подальшого розвитку й масштабування. Її головна цінність полягає у здатності поєднувати статистичні обчислення, графічну аналітику та зручний формат представлення результатів в одному автоматизованому процесі. Це дозволяє не лише зберегти час дослідника, а й створити умови для розгортання більш широких систем моніторингу, контролю якості моделей і підтримки прийняття рішень у реальному часі.

3.4 Оцінка ефективності та потенціал впровадження

3.4.1 Обговорення переваг і обмежень

Штучний інтелект у стратегії дозволяє враховувати взаємозв’язки між показниками, які часто залишаються поза увагою класичних rule-based підходів. Крім того, гнучкість у зміні архітектури моделі, її повторне тренування на нових даних та можливість швидкої адаптації до нестандартних умов робить таку стратегію значно перспективнішою у довгостроковій перспективі. Використання патернів програмування, таких як Strategy, MVC та Observer, забезпечує масштабовану і зрозумілу архітектуру системи.

Однією з помітних переваг реалізованої системи є модульність. Компоненти моделі, механізми генерації сигналів, візуалізації та звітності реалізовані як окремі, незалежні блоки, що дозволяє не лише розвивати систему поетапно, а й проводити локальні експерименти без впливу на

загальну логіку. Це критично важливо у фінансових застосуваннях, де навіть незначна зміна в алгоритмі може спричинити серйозні наслідки у роботі на реальних коштах.

Проте, попри очевидні переваги, стратегія має і певні обмеження. Найперше, це залежність від якості вхідних даних. Наявність шумів, незбалансованих трендових сегментів або періодів низької ліквідності може значно вплинути на якість прогнозу. Крім того, моделі машинного навчання, особливо в класифікаційних завданнях, можуть бути схильні до переадаптації, якщо не використовуються відповідні механізми регуляризації чи крос-валідації.

Незважаючи на використання деревоподібних або нейромережевих підходів, прийняті рішення часто залишаються «чорним ящиком» для кінцевого користувача. Це обмежує довіру трейдерів до системи та потребує додаткових інструментів для пояснення логіки рішень. Крім того, необхідність ручного оновлення моделі або втручання при зміні ринкових умов все ще залишається певною точкою ризику в контексті повної автономності.

Таким чином, обговорення переваг і обмежень засвідчує, що запропонована система, з одного боку, має потужний потенціал до практичного впровадження в умовах реального трейдингу, з іншого – потребує постійного супроводу, адаптації та вдосконалення. Найбільш перспективним напрямом є розвиток explainable AI-компонентів і гібридних моделей, які поєднують rule-based логіку з глибоким навчанням для досягнення балансу між ефективністю та передбачуваністю.

3.4.2 Можливість запуску у dry-run/live

Однією з ключових ознак практичної придатності інтелектуальної торгової системи є її здатність працювати у режимах, максимально наближених до реального середовища біржі. У цьому контексті платформа

Freqtrade надає потужні інструменти для моделювання та тестування стратегій у двох основних режимах: dry-run (імітаційний режим) та live (режим реальної торгівлі). Розроблена AI-стратегія IntelliBotStrategy повністю підтримує обидва варіанти запуску, що свідчить про її технічну готовність до застосування в умовах живого ринку.

Dry-run режим дозволяє здійснювати симуляцію реальної біржової активності без фактичного відкриття позицій на біржі. У цьому випадку стратегія підключається до біржового API, отримує реальні дані в режимі реального часу, але всі операції: відкриття, супровід і закриття позицій – виконуються лише в пам’яті системи, без реального впливу на обліковий запис користувача. Це надзвичайно зручно для оцінки роботи стратегії в умовах поточних ринкових коливань, без фінансових ризиків.

Під час dry-run тестування відображаються всі ключові параметри: момент входу й виходу, прибуток або збиток по угоді, причина виходу, поведінка моделі у випадку непередбачуваних ринкових змін. Завдяки цьому трейдер отримує повну інформацію про ефективність стратегії, як у коротко-, так і у довгостроковому часовому горизонті. Однією з особливостей реалізації dry-run для AI-стратегії є логування кожного рішення моделі на основі технічних індикаторів, що дозволяє здійснити пост-фактум аналіз її поведінки.

Live-режим, у свою чергу, є фінальною метою впровадження торгової системи. У цьому випадку IntelliBotStrategy взаємодіє з біржею напряму, використовуючи ключі API з правами на розміщення ордерів. Для безпеки такі ключі не зберігаються у коді, а передаються через конфігураційний файл, який може бути зашифрований.

Інтеграція live-запуску забезпечується шляхом прямої підтримки архітектури Freqtrade: система сприймає AI-модуль як частину логіки populate_entry_trend() і populate_exit_trend(). Після проходження всіх етапів backtesting і dry-run модель може бути переведена в live-режим без зміни коду, виключно шляхом коригування налаштувань у config.json. Це

дозволяє розробнику швидко адаптувати систему до ринку, не вносячи технічних ризиків у програмну архітектуру.

Підтримка dry-run і live-режимів є критично важливою характеристикою сучасних трейдингових систем, і реалізована стратегія повністю відповідає цим вимогам. Dry-run забезпечує безпечне тестування й адаптацію до нових умов, live є доказом готовності до повноцінного практичного використання. Завдяки підтримці обох режимів, система IntelliBotStrategy здатна не лише демонструвати аналітичну ефективність, а й інтегруватися в реальні бізнес-процеси торгових платформ або приватних трейдерів.

3.4.3 Перспективи вдосконалення

Одним із найперспективніших напрямів розвитку є інтеграція моделей глибокого навчання (deep learning), зокрема рекурентних нейронних мереж (LSTM, GRU), які здатні ефективно обробляти часові ряди та враховувати контекст попередніх ринкових подій. Такі моделі мають потенціал у виявленні складних патернів, які класичні класифікатори або shallow-моделі не в змозі уловити. Їх використання дозволить не лише покращити точність рішень, але й створити передумови для побудови багаторівневих стратегій зі сценарним прогнозуванням.

Ще одним важливим кроком є впровадження explainable AI-механізмів (XAI), які надаватимуть інтерпретацію прийнятих рішень. Це особливо актуально для підвищення довіри користувачів до системи, а також для виявлення потенційних помилок у логіці або тренуваннях моделі. Використання алгоритмів SHAP або LIME дозволить проаналізувати, які саме ознаки вплинули на конкретне торгове рішення. Це відкриває шлях до глибшого контролю та аудиту дій стратегії.

На архітектурному рівні доцільним є впровадження механізмів автоматичного донавчання (online learning). Це дозволить моделі

адаптуватися до нових умов без повного перевчання з нуля, що є особливо корисним у випадку високочастотної торгівлі або різких ринкових змін. Крім того, існує можливість створення системи гібридного навчання, яка комбінуватиме історичні дані з поточними потоками у реальному часі. Це надасть трейдинговому агенту здатність прогнозувати не лише тенденції, але й реагувати на мікро-імпульси ринку.

З позиції розгортання системи у production-середовищі актуальним є створення dashboard-інтерфейсу з інтерактивною аналітикою. Такий інтерфейс дозволив би користувачу в режимі реального часу стежити за діями стратегії, змінювати параметри моделі, запускати перевірки якості та контролювати ризики. У поєднанні з автоматизованим звітуванням це суттєво розширює функціональність системи і робить її придатною для застосування не лише окремим трейдером, а й у корпоративних структурах.

Перспективи вдосконалення розробленої стратегії відкривають широке поле для подальших досліджень, інтеграції новітніх технологій та оптимізації існуючого функціоналу. Поступовий перехід від аналітичної моделі до самонавчального торгового агента зі здатністю до самоадаптації, пояснення рішень та безперервного розвитку є стратегічним напрямком еволюції проєкту у напрямку повноцінної автономної фінансової системи.

ВИСНОВКИ

У рамках дослідження було всебічно реалізовано як теоретичну, так і практичну частини, що дозволило не лише створити функціональну систему, але й перевірити її ефективність в умовах ринку за допомогою методу backtesting. Основна увага приділялася поєднанню архітектурної гнучкості із можливостями машинного навчання для підвищення точності торгових сигналів.

У ході виконання аналізу предметної галузі було досліджено структуру алгоритмічного трейдингу, проаналізовано існуючі фреймворки, зокрема Freqtrade, і сформульовано вимоги до системи. Особлива увага приділялася розмежуванню rule-based і AI-based стратегій, що дало змогу обґрунтувати доцільність використання штучного інтелекту в автоматизованих торгових системах. Сучасні торгові алгоритми вимагають адаптивності, прозорості та високої точності, а також здатності до самоудосконалення на основі історичних даних.

Формалізації задачі та структурування функціоналу, була розроблена загальна схема системи, що складається з ядра торгового бота, AI-модуля, модуля генерації звітів та бази даних. Використання шаблону Strategy дало змогу легко перемикатися між різними варіантами логіки: як rule-based, так і AI-залежної. Це рішення забезпечило масштабованість і зручність розширення системи в майбутньому без втрати стабільності.

У межах реалізації AI-модуля було проведено повний цикл машинного навчання: обрано релевантні технічні індикатори (RSI, EMA, Bollinger Width), сформовано вхідні ознаки, здійснено тренування моделі класифікації з подальшим збереженням у форматі .pkl. Модель було інтегровано в архітектуру стратегії, що дозволило отримувати рішення про buy/hold/sell у реальному часі. Результати backtesting показали, що AI-стратегія, порівняно з rule-based підходом, продемонструвала кращу стабільність і вищу точність під час спаду ринку. Це підтверджує

доцільність використання інтелектуального підходу до прогнозування фінансових ринків.

Створення модуля генерації звітів, у якому реалізовано автоматичне формування PDF-документа зі статистикою торгів, графіками кумулятивної прибутковості, щоденної дохідності, тепловими картами виходу з ринку та зведенням метрик (precision, recall, accuracy). Звіт виконує не лише інформативну, але й презентаційну функцію, підвищуючи довіру до прийнятих рішень.

У результаті виконання всіх завдань було розроблено стабільну, функціональну, інтелектуальну систему IntelliBot, що здатна адаптуватися до змін ринку, забезпечуючи баланс між ризиком і прибутковістю. Високий рівень гнучкості, забезпечений архітектурними патернами, дозволяє масштабувати та модифікувати систему без необхідності повного переписування коду. Розроблений програмний агент може розглядатися як прототип комерційного рішення, здатного ефективно функціонувати як у режимі симуляції (dry-run), так і в реальному середовищі (live trading).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Chan E. Algorithmic Trading: Winning Strategies and Their Rationale. Wiley & Sons, Incorporated, John, 2013. 224 p.

2. Forecasting Stock Prices from the Limit Order Book Using Convolutional Neural Networks / A. Tsantekidis et al. *2017 IEEE 19th Conference on Business Informatics (CBI)*, Thessaloniki, Greece, 24–27 July 2017. 2017. URL: https://doi.org/10.1109/cbi.2017.23 (date of access: 20.04.2025).

3. Gao W., Yan R., Wu X. Research Progress of Algorithmic Trading at Home and Abroad. *First International Conference Economic and Business Management 2016*, Qingdao, China, 15–17 November 2016. Paris, France, 2016. URL: https://doi.org/10.2991/febm-16.2016.7 (date of access: 25.04.2025).

4. Forecasting jump arrivals in stock prices: new attention-based network architecture using limit order book data / Y. Mäkinen et al. *Quantitative Finance*. 2019. Vol. 19, no. 12. P. 2033–2050. URL: https://doi.org/10.1080/14697688.2019.1634277 (date of access: 25.04.2025).

5. Sutton R. S., Barto A. G. Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks*. 1998. Vol. 9, no. 5. P. 1054. URL: https://doi.org/10.1109/tnn.1998.712192 (date of access: 25.04.2025).

6. Almahdi S., Yang S. Y. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*. 2017. Vol. 87. P. 267–279. URL: https://doi.org/10.1016/j.eswa.2017.06.023 (date of access: 25.04.2025).

7. Bahar M. H., Saad H. N. Adaptive Windowing (ADWIN3) to Learning from Time-Changing Data Stream. *Intelligent Systems Design and Applications*.

Cham, 2024. P. 150–163. URL: https://doi.org/10.1007/978-3-031-64850-2_14 (date of access: 25.04.2025).

8. Bengio Y., Courville A., Goodfellow I. Deep Learning. MIT Press, 2016. 800 p.

9. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns. Elements of Reusable Object-oriented Software. Addsion Wesley Longman, printed in India by Eastern Press, 1999.

10. Sierra K. Head First Design Patterns. O'Reilly Media, Incorporated, 2004.

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

12. Development of web application using model view controller (MVC) architecture. *International Journal of Progressive Research in Engineering Management and Science*. 2023. URL: https://doi.org/10.58257/ijprems30585 (date of access: 25.04.2025).

13. Larman C. Applying UML and patterns: An introduction to object-oriented analysis and design. Upper Saddle River, N.J : Prentice Hall PTR, 1998. 507 p.

14. Freqtrade. URL: https://www.freqtrade.io/en/stable/ (date of access: 25.04.2025).

15. Davey K. Building Algorithmic Trading Systems: A Trader's Journey from Data Mining to Monte Carlo Simulation to Live Trading. Wiley & Sons, Incorporated, John, 2014.

16. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter. O'Reilly Media, 2022. 550 p.

17. Kokane M. S. Algorithmic Stock Trading Using Python. *International Journal for Research in Applied Science and Engineering Technology*. 2023. Vol. 11, no. 6. P. 88–91. URL: https://doi.org/10.22214/ijraset.2023.53571 (date of access: 25.04.2025).

18. Prado M. L. d. Advances in Financial Machine Learning. Gildan Media, 2018.

19. Ozbayoglu A. M., Gudelek M. U., Sezer O. B. Deep learning for financial applications : a survey. *Applied soft computing*. 2020. Vol. 93. P. 106384. URL: https://doi.org/10.1016/j.asoc.2020.106384 (date of access: 06.05.2025).

20. Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques / J. Patel et al. *Expert systems with applications*. 2015. Vol. 42, no. 1. P. 259–268. URL: https://doi.org/10.1016/j.eswa.2014.07.040 (date of access: 06.05.2025).

21. Fischer T., Krauss C. Deep learning with long short-term memory networks for financial market predictions. *European journal of operational research*. 2018. Vol. 270, no. 2. P. 654–669. URL: https://doi.org/10.1016/j.ejor.2017.11.054 (date of access: 06.05.2025).

22. Dhanya R. K., Unnikrishnan V. S. Application of machine learning algorithms to improve stock prediction using technical indicators. *Recent advances in materials, mechanics and management*. 2019. P. 270–274. URL: https://doi.org/10.1201/9781351227544-46 (date of access: 06.05.2025).

23. 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. URL: https://doi.org/10.1016/j.asoc.2020.106181 (date of access: 06.05.2025).

24. Explainable AI in algorithmic trading: mitigating bias and improving regulatory compliance in finance. *International journal of computer applications technology and research*. 2025. URL: https://doi.org/10.7753/ijcatr1404.1006 (date of access: 06.05.2025).

25. Gudelek M. U., Boluk S. A., Ozbayoglu A. M. A deep learning based stock trading model with 2-D CNN trend detection. *2017 IEEE symposium series on computational intelligence (SSCI)*, Honolulu, HI, 27 November – 1 December

2017. 2017. URL: https://doi.org/10.1109/ssci.2017.8285188 (date of access: 06.05.2025).

26. Nisha Jenipher V., Radhika S. A study on early prediction of lung cancer using machine learning techniques. *2020 3rd international conference on intelligent sustainable systems (ICISS)*, Thoothukudi, India, 3–5 December 2020. 2020. URL: https://doi.org/10.1109/iciss49785.2020.9316064 (date of access: 07.05.2025).

27. Encyclopedia of machine learning and data mining / ed. by C. Sammut, G. I. Webb. Boston, MA : Springer US, 2016. URL: https://doi.org/10.1007/978-1-4899-7502-7 (date of access: 13.05.2025).

28. Model persistence. *scikit-learn*. URL: https://scikit-learn.org/stable/model_persistence.html (date of access: 07.05.2025).

29. Zhang C., Ma Y. Ensemble machine learning: methods and applications. Springer New York, 2014.

30. Jain M. Machine learning and deep learning approaches for cybersecurity: a review. *International journal of science and research (IJSR)*. 2023. Vol. 12, no. 10. P. 1706–1710. URL: https://doi.org/10.21275/sr231023115126 (date of access: 07.05.2025).

31. Zolotukhin, O., Filatov, V., Yerokhin, A., Lanovyy, O., Kudryavtseva, M., Semenets, V.: An approach to the selection of behavior patterns autonomous intelligent mobile systems. *In: Proceedings of the IEEE International Conference on Problems of Infocommunications Science and Technology* (PIC S&T), pp. 349–352. IEEE, Kyiv (2021). https://doi.org/10.1109/PICST54195.2021.9772110 (date of access: 02.06.2025).

32. Zolotukhin, O., Filatov, V., Yerokhin, A., Kudryavtseva, M.: The methods for the prediction of climate control indicators in the Internet of Things systems. *CEUR Workshop Proc.* (2021). https://doi.org/10.5281/zenodo.14526027 (date of access: 02.06.2025).

33. Filatov, V., Yerokhin, A., Zolotukhin, O., Kudryavtseva, M.: Methods of intellectual analysis of processes in medical information systems. Inf. Extr.

Process. 48(124), 92–98 (2020). https://doi.org/10.15407/vidbir2020.48.092 (date of access: 02.06.2025).

34. Filatov, V., Semenets, V., Zolotukhin, O.: Synthesis of semantic model of subject area at integration of relational databases. *In: Proceedings of the IEEE 8th International Conference on Advanced Optoelectronics and Lasers* (CAOL), pp. 598–601. IEEE, Sozopol (2019). https://doi.org/10.1109/CAOL46282.2019.9019532 (date of access: 02.06.2025).