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)

Розробка інтелектуального агента для торговельних операцій

(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-2

Антон Климчук
(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 ст. викл. Тетяна Мірошниченко
(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 Климчуку Антону Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка інтелектуального агента для торговельних операцій

затверджена наказом університету від 19 травня 20 25 р. № 378Ст

2. Термін подання студентом роботи до екзаменаційної комісії 25 червня 20 25 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел та відомих наукових проектів, Python documentation, набір даних для тренування та тестування системи

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі
2) Постановка задачі
3) Методи автоматизованої торгівлі на основі ШІ
4) Реалізація інтелектуального агента

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	19.05.2025 – 22.05.2025	виконано
3	Постановка задачі	22.05.2025 – 23.05.2025	виконано
4	Проведення теоретичних досліджень	23.05.2025 – 29.05.2025	виконано
5	Вибір технологій та засобів розробки	29.05.2025 – 30.05.2025	виконано
6	Реалізація проєкту	30.05.2025 – 10.06.2025	виконано
7	Оформлення пояснювальної записки	10.06.2025 – 14.06.2025	виконано
8	Надання роботи на рецензування	14.06.2025	виконано
9	Перевірка пояснювальної записки на плагіат	15.06.2025	виконано
10	Попередній захист роботи з керівником	16.06.2025	виконано
11	Захист перед ЕК	25.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Тетяна Мірошниченко
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 111 с., 11 рис., 1 табл., 3 дод., 16 джерел.

АВТОМАТИЗОВАНА ТОРГІВЛЯ, ІНТЕЛЕКТУАЛЬНИЙ АГЕНТ, КРИПТОВАЛЮТИ, МАШИННЕ НАВЧАННЯ, ТЕХНІЧНИЙ АНАЛІЗ, ТРЕНД, УПРАВЛІННЯ РИЗИКОМ.

Об’єкт дослідження – процес алгоритмічної торгівлі на фінансових ринках та підходи до автоматизації прийняття торгових рішень.

Предмет дослідження – інтелектуальний торговий агент, заснований на правилах технічного аналізу та логіці трейдера, здатний самостійно аналізувати ринок і здійснювати угоди з урахуванням контексту.

Мета роботи – розробка концепції та логіки функціонування інтелектуального торгового агента, який дозволяє підвищити ефективність прийняття рішень у трейдингу за рахунок автоматизації, виключення людського фактора та використання структурованого технічного аналізу.

Методи дослідження – аналіз наукової та технічної літератури з алгоритмічного трейдингу, огляд сучасних рішень з автоматизації торгівлі, порівняльне тестування логік прийняття рішень, формалізація патернів поведінки ціни, побудова rule-based моделей.

Роботу присвячено розробці інтелектуального агента для автоматизованої торгівлі криптовалютами на основі комбінування методів технічного аналізу та машинного навчання. У вступній частині представлено історичний огляд розвитку торгових систем та проведено порівняльний аналіз ручного та автоматизованого трейдингу. На основі сучасних досліджень обґрунтовано переваги алгоритмічних підходів: висока швидкодія й відсутність емоційного чинника.

ABSTRACT

Bachelor’s thesis contains: 111 pp., 11 fig., 1 tabl., 3 ann., 16 references.

AUTOMATED TRADING, CRYPTOCURRENCIES, INTELLIGENT AGENT, MACHINE LEARNING, TECHNICAL ANALYSIS, TREND, RISK MANAGEMENT.

Object of the study – the process of algorithmic trading in financial markets and the approaches to automating trading decision-making.

Purpose of the thesis – to develop the concept and operational logic of an intelligent trading agent that enhances decision-making efficiency in trading through automation, elimination of the human factor, and application of structured technical analysis.

Research methods – analysis of scientific and technical literature on algorithmic trading, review of current trading automation solutions, comparative testing of decision-making logics, formalization of price behavior patterns, and development of rule-based models.

The study is devoted to the development of an intelligent agent for automated cryptocurrency trading that combines technical-analysis techniques with machine-learning methods. The introductory section provides a historical overview of trading-system evolution and a comparative analysis of manual versus automated trading. Drawing on recent research, the paper substantiates the advantages of algorithmic approaches–namely high execution speed and the elimination of the human emotional factor.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence – штучний інтелект;

API – Application Programming Interface – програмний інтерфейс прикладних застосунків;

ATR – Average True Range – середній істинний діапазон, пунктів індикатор волатильності;

BTC – Bitcoin, базова криптовалюта;

EMA – Exponential Moving Average – експоненціальне ковзне середнє (ціновий індикатор);

HFT – High-Frequency Trading – високочастотна автоматизована торгівля;

LSTM – Long Short-Term Memory – рекурентна нейронна мережа з довго- та короткочасною пам’яттю;

PnL – Profit and Loss – показник прибутку-збитку,

RSI – Relative Strength Index – індекс відносної сили (осцилятор);

USDT – Tether, стейблкоїн, номінований у доларах США.

ВСТУП

У XXI столітті фінансові ринки стрімко переходять від дій, що виконуються людиною вручну, до безперервного алгоритмічного режиму. Найвиразніше ця еволюція відчувається на криптовалютних біржах, де торги тривають без вихідних, а коливання цін можуть сягати десятків відсотків протягом кількох годин. У таких умовах швидкість реакції, дисципліна й здатність опрацьовувати великі масиви даних стають критичними факторами успіху, недосяжними для звичайного трейдера. Саме тому тема створення інтелектуального агента, який самостійно збирає ринкову інформацію, аналізує її методами технічного аналізу та миттєво відкриває чи закриває позиції, є не лише актуальною, а й життєво необхідною для підвищення ефективності торговельних операцій. Метою цієї роботи є розробити й експериментально перевірити такий агент, що поєднує сувору алгоритмічність із гнучким ризик-менеджментом. У дослідженні послідовно окреслюються історичні етапи розвитку автоматизованої торгівлі, аналізуються наукові підходи до побудови торговельних систем, формулюється алгоритм прийняття рішень на основі тренду, імпульсу та рівнів підтримки й опору, а також описується програмна архітектура прототипу. Завершальний етап передбачає тестування моделі на історичних та реальних даних із метою верифікації прибутковості й стійкості. У результаті очікується отримання працюючої системи, що демонструє стабільне зростання капіталу при керованій просадці, а також формування рекомендацій щодо подальшого удосконалення, зокрема інтеграції компонентів машинного навчання. Робота має практичну цінність для трейдерів і дослідників, адже показує, як перетворити людську стратегію на код і досягти переваги в одному з найдинамічніших сегментів сучасного фінансового простору.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Історичний розвиток алгоритмічної торгівлі

Розвиток алгоритмічної торгівлі (автоматизованої комп’ютеризованої торгівлі фінансовими інструментами) є результатом тривалого прогресу технологій та фінансової інженерії. Перші спроби автоматизувати виконання торговельних операцій розпочалися ще в середині XX століття, коли з’явилися перші електронно-обчислювальні машини. В 1950–1960-х роках окремі фінансові установи експериментували з базовими алгоритмами для купівлі-продажу акцій, проте ті системи були примітивними й обмеженими за функціональністю. Лише з розвитком комп’ютерної техніки у 1980-х роках алгоритмічна торгівля отримала потужний поштовх: поява електронних біржових платформ та збільшення обчислювальних потужностей дозволили значно прискорити виконання ордерів і обробку ринкових даних. Перехід від традиційних біржових майданчиків із гучними залами до електронних систем торгівлі створив умови для суттєвого підвищення швидкості та ефективності виконання заявок. У 1990-х роках відбувся справжній бум кількісних методів: широко впроваджувалися математичні моделі та історичний аналіз даних для автоматизованого прийняття торгових рішень. Саме в цей період здобули популярність програмні торгові стратегії та системи управління ризиками, які заклали основу сучасних алгоритмічних фондів.

Варто зауважити, що розквіт алгоритмічної торгівлі супроводжувався як успіхами, так і гучними невдачами. Одним із перших тривожних сигналів стала подія, відома як «чорний понеділок» 1987 року, коли різке падіння фондового ринку частково пояснювали масовим використанням програмного продажу акцій. Це продемонструвало, що алгоритми, виконуючи закладені правила, здатні колективно спричиняти небажані ефекти на ринкуза екстремальних умов. Проте загальна тенденція лишалася

незмінною – частка автоматизованих систем постійно зростала. На початку 2000-х років перехід американських бірж на десяткову систему ціноутворення (decimalization) збільшив кількість можливих цінових рівнів, що стимулювало розвиток високочастотних алгоритмів для роботи з дрібними коливаннями цін. У середині 2000-х відбулося стрімке піднесення високочастотної торгівлі (High-Frequency Trading, HFT) – спеціалізовані фірми почали використовувати алгоритми з надзвичайно високою швидкодією, виконуючи тисячі угод за секунду. Вони розміщували свої сервери якнайближче до біржових, користуючись перевагами прямих каналів передачі даних, щоб мінімізувати затримки. Це дало конкурентну перевагу в швидкості виконання угод, що особливо важливо для арбітражних стратегій. Європейський центральний банк у статті.

«Algorithmic trading» зазнчає що за оцінками інвестиційних банків, до середини 2010-х років алгоритмічна торгівля становила 60–70% обсягу операцій на фондових ринках США. Таким чином, за декілька десятиліть фінансовий сектор перейшов від переважно ручної торгівлі до домінування автоматизованих систем.

Останнє десятиліття характеризується інтеграцією методів штучного інтелекту (ШІ) в алгоритмічні торгові системи. Якщо раніше алгоритми діяли за жорстко запрограмованими правилами, то сучасні підходи дозволяють їм навчатися та адаптуватися. Починаючи з 2010-х років, завдяки проривам у машинному навчанні та доступності великих масивів даних, з’явилися алгоритми, які можуть самостійно виявляти складні нелінійні закономірності. Зокрема, нейронні мережі почали застосовувати для прогнозування фінансових часових рядів та класифікації ринкових сигналів. Такі технології, як глибоке навчання, обробка природної мови (для аналізу новин і соціальних мереж) та навчання з підкріпленням для прийняття рішень, відкрили новий етап розвитку торгових агентів. На практиці це вилилось у появу «робо-адвайзерів» – автоматизованих радників з інвестування – та алгоритмів, що постійно самовдосконалюються

«на основі зворотного зв’язку. Наприклад, вже існують платформи, які дозволяють алгоритму коригувати власну стратегію в реальному часі залежно від ринкових умов. Було дослідження [1], де зазначається, що широке впровадження ШІ породило і нові виклики: регулятори та дослідники відзначають проблему алгоритмічної упередженості (bias), потенційних системних ризиків і питань етики використання штучного інтелекту у фінансах. Попри це, історичний прогрес свідчить про безперервне прагнення ринку до підвищення ефективності, швидкості та точності торгівлі.

У підсумку, від перших спроб автоматизації у 1960-х до сьогоднішніх AI-генерованих стратегій – алгоритмічна торгівля пройшла шлях від допоміжного інструменту до провідної сили фінансових ринків. Її еволюція демонструє, як технологічні інновації змінюють обличчя торгівлі, а саме: людина дедалі більше делегує рутинні та швидкі операції машинам, зосереджуючись на контролі та стратегічних рішеннях.

1.2 Проблеми ручної торгівлі на фінансових ринках

Ручна торгівля або операції з активами у форматі прямих купівель і продажів супроводжуються низкою обмежень і недоліків. По-перше, люди фізично та психологічно обмежені. Трейдер не може провести весь день, спостерігаючи за десятками інструментів та постійно реагувати на зміну цін, особливо не в нинішніх суперліквідних ринках, де сотні операцій здійснюються за мілісекунди. Комп'ютерній програмі не потрібно часу, щоб подумати чи обробити інформацію; люди належать до групи, які думають та обробляють інформацію, та реагують на неї: навіть найшвидший трейдер має затримку реакції в секундах, тоді як комп'ютери діють за мілісекунди.

Це означає, що у швидкій світовій грі людина гравець майже напевно програє гонку за найкращу ціну/арб можливість. Другий головний недолік – це емоційний аспект. Психологія людського розуму впливає на фінансові

рішення; страх і жадоба змушують людей робити ірраціональні вибори. І у випадках бурхливих коливань ринку торгівля стає лякливою чи надміру збудженою по черзі. Це може у свою чергу призвести до невиправданих рішень: можна передчасно продати сульфур активи, просто зі страху перед втратою; або, навпаки, купити активи «на піку» ціни, частково через те, що багато людей вважають, що вона буде підвищуватися нескінечно. Такі поведінкові відхилення добре документовані у поведінкових фінансах і свідчать про те, що емоції часто перешкоджають слідуванню раціональній стратегії.

У процесі дії зберігається повна відстороненість від емоційних та поведінкових чинників – реалізуються лише закладені правила, що дозволяє мінімізувати типові помилки, властиві суб'єктивному підходу. Третє – це критерій дисципліни та послідовності. Людина-трейдер навіть з письмовим торговим планом може відступити, слідуючи своїм емоціям чи відчуттям жадоби чи страху. Наприклад, у вас є стоп-лосс (для обмеження потенційних збитків), встановлений на певній ціновій точці тут, і трейдер його скасовує, сподіваючись, що ринок зміниться – зазвичай цього не відбувається з ще більшими втратами. Алгоритм з радістю відкриє правило, яке встановлене. Недотримання належної дисципліни є одним із пояснень невдачі багатьох роздрібних інвесторів: за допомогою емпіричних досліджень показано, що більшість людей, які пробують себе в активній торгівлі, зазнають збитків. Дослідження, яке проаналізувало понад 19,000 денних торгових рахунків в цьому ринку, показало, що 97% таких трейдерів не змогли отримати прибуток за один день, а лише маленький відсоток утримував позиції на кінець року, які були достатніми, щоб вважатися чистим прибутком. Ця реальність відкриває очі на те, наскільки важко для людини постійно заробляти на фінансовій спекуляції і може допомогти пояснити перехід професійної торгівлі до алгоритмів. Ручна торгівля також має повільну обробку інформації та обмеження розміру даних.

Сьогодні існують скарби даних, які викидаються щомиті сучасними

фінансовими ринками: цінові подачі, новини, статистика бірж – і все це впливає на ціни. Ніхто не може особисто переварити інформацію з десятка джерел одночасно при цьому, як це відбувається. Тому багато сигналів втрачаються, або потребують часу, щоб на них відреагувати. Однак комп'ютерна система може одночасно обробити кілька різних сигналів, вона може одночасно виявляти цінові відмінності між цінами активів, опублікованими на різних ринках, або вона може проаналізувати десятки технічних індикаторів за мить. Людська обчислювальна здатність тут не може конкурувати з обчислювальною потужністю комп'ютера. І тоді є також цілком практичне питання про те, як працює ринок. Наприклад, криптовалютні біржи не закриваються як акції. Людина не може торгувати цілодобово, 24/7 – їм потрібен час для відпочинку та сну. Це дає можливість ринку коливатися: під час відсутності трейдера ринок може різко змінитися.

У порівнянні, комп'ютерний агент не «спить» і може постійно спостерігати за ринком, навіть о 3-й годині ранку в неділю; він може інструктувати агента торгувати при першій можливості досягнення ковзного середнього. У короткому підсумку, людська торгівля програє алгоритмічній торгівлі майже за всіма важливими показниками на сучасних швидкоплинних ринках. Це те, що спонукає до автоматизації: покращити якість у виробничому процесі, віддати ваші торгові рішення на основі інтуїції у більш масштабований процес для вашого блага.

1.3 Переваги алгоритмічних торгових систем

Перехід від людської до алгоритмічної торгівлі обумовлений численними перевагами автоматизованих підходів. Розглянемо ключові з них: швидкість та ефективність виконання угод. Алгоритми здатні виконувати торговельні операції за лічені мілісекунди, чого неможливо досягти при ручному введенні заявок. Високошвидкісне з’єднання та оптимізований код дозволяють миттєво реагувати на зміни ринку. Це

особливо важливо для таких стратегій, як арбітраж – коли потрібно дуже швидко купити і продати актив, користуючись короткочасною ціновою неузгодженістю. Знову посилаючись на Європейський центральний банк та їхню статтю [1] можно сказати, що алгоритмічна система може відстежувати одночасно ціни на багатьох біржах і негайно здійснити серію угод, щойно виявиться можливість безризикового прибутку, тоді як людина просто не встигне скористатися такою ситуацією.

Крім того, оскільки алгоритм працює в режимі реального часу, наш час реакції на подію завжди дуже низький – незалежно від того, в який час доби (або ночі) трапляється подія або чи це свято. Аналіз значних обсягів інформації. У той час як людський розум обмежений кількома вимірами умовиводу, машини можуть одночасно обраховувати численні виміри. Сучасні алгоритми можуть в режимі реального часу обробляти всю доступну інформацію на ринку; стрічки новин, економічні показники, соціальні мережі тощо. Наприклад, після збору статей з медіа та аналізу їх змісту за допомогою підходів NLP, агент має можливість враховувати ринкові настрої та миттєво коректувати позиції. Фізично неможливо для людини прочитати і зрозуміти тисячі твітів або новинних статей за хвилину, але для алгоритму це можливо. Тому алгоритм може здійснювати рішення на основі більш повної та менше спорадичної інформації, що робить їх більш обгрунтованими.

Контроль і відсутність почуттів. Оскільки це запрограмоване, в алгоритмічній торгівлі немає такої речі, як емоційна нестабільність. Система не має страху втрати або жадібності до прибутку – вона просто реагує відповідно до скрипта. Торгова система Cmdty не має здатності відчувати. Це для того, щоб ви залишались в вашому торговому плані. Якщо правила виходу стратегії вимагають від мене виходу, коли вона досягає певного рівня втрат, я не буду вагатись виконати це точно в той момент. Це забезпечує управління ризиками з наперед встановленими умовами. І це також означає, що система не може мати емоцій, не втомлюється чи втрачає

концентрацію – усе це передається «людським фактором» (втомою, натисканням неправильної клавіші тощо), але зведено до мінімуму.

Здатність до оптимізації та тестування. І перш ніж торговий алгоритм буде запущений в живу торгівлю, він може бути ретельно протестований на історичних даних – процес, що зветься бектестуванням. Це дозволяє перевірити, як би стратегія показала себе історично, та допомагає трейдеру виявити її недоліки до того, як вона інвестує реальний капітал. Людина не може інтуїтивно зрозуміти, як би вона торгувала, наприклад, за останні 5 років, і який був би повернення – в той час як комп'ютерна симуляція може це зробити за лічені секунди. Крім того, алгоритм може бути оптоелектронним таким чином, що оптимальні параметри (такі як періоди індикаторів, рівні стоп-ордерів) виявляються за допомогою опцій, і опції для цих параметрів, що принесли найбільший прибуток в історії, будуть обрані. З ручною торгівлею було б дуже складно застосувати систематичний підхід до поліпшення вашої стратегії.

Зниження витрат, масштабованість. Автоматизація торгівлі також має економічні наслідки: один комп'ютерний алгоритм може управляти великим портфелем активів з невеликою участю людини. Це знижує вимогу мати велику команду трейдерів і знижує операційні витрати та аспект людських помилок. Банки та хедж-фонди, в свою чергу, використовували алгоритми для обробки більшої кількості замовлень з меншою кількістю працівників. І алгоритм дуже паралелізований – його можна одночасно виконувати на кількох ринках або на багатьох інструментах, просто надаючи більше обчислювальних ресурсів. Є лише певна кількість, яку одна людина може фізично опрацювати в будь-який момент часу. В результаті автоматизовані агенти вводять збільшену торгівельну активність у цілому.

1.4 Криптовалютний ринок: особливості та виклики

Для демонстрації потужності розумного торгового агента в цій роботі

було обрано ринок криптовалют, оскільки це нова, але дуже динамічна гілка в літературі фінансів. Ринок криптовалют має унікальні характеристики, які відрізняють його від звичайного ринку акцій чи валют, що робить його цікавою сценою для алгоритмічної торгівлі. По-перше, криптовалюти (наприклад, Bitcoin, Ethereum і тисячі інших) торгуються 24/7. Без центральної сесії обміну це означає, що ринок ніколи не «відпочиває». Торги відбуваються на кількох крипто-біржах по всьому світу, в різних часових зонах, безперервно без зупинки. Це створює проблему для людського трейдера: вони не в змозі фізично відстежувати ринок 24/7, тому можуть бути значні рухи цін, які вислизають з їх уваги.

З іншого боку, алгоритмічний агент може працювати цілодобово, що є ідеальним у цьому випадку. І, по-друге, ринок криптовалют – це все про волатильність – різкі й великі зміни цін тут не виняток, а правило. Вартість криптовалют може драматично змінюватися лише впродовж кількох годин або моментів. Наприклад, щоденні коливання цін у випадку Bitcoin спостерігалися на рівні ±10% і більше. З одного боку, висока волатильність надає широкі можливості для прибутку (чим більша волатильність, тим вищий потенційний заробіток від спекуляції). З іншого боку, це ризиковано: ринок може різко рухатися проти позиції за лічені хвилини. У таких ситуаціях навіть кілька хвилин запізнення можуть коштувати значного прибутку, тому наявність автоматизованої системи для сповіщення про зміни на ринку − це велика перевага. Більше того, автоматичні стратегії можуть включати суворі контролі ризиків (наприклад, при падінні ціни на X%, рішення про торгівлю має бути закрито), що є надзвичайно важливим у волатильному ринку. Також ринкові структури криптовалют дуже відрізняються від традиційних ринків. Існують сотні незалежних бірж (Binance, Coinbase, Kraken, Huobi, ...), деякі з них працюють централізовано,

інші працюють у децентралізованому форматі (DEX, що працюють зі смарт-контрактами). Немає центрального сховища ліквідності, подібного до

Нью-Йоркської фондової біржі для певної компанії. Це означає, що ціна на актив (наприклад, Bitcoin) на одній біржі може трохи відрізнятися від ціни на іншій. Існують можливості арбітражу – якщо певна монета дорожче на одній біржі, ніж на іншій, можна заробити, купуючи дешевше і продаючи дорожче в короткій послідовності.

Однак такі можливості існують лише кілька секунд або їх частину і доступні тільки алгоритмам, які можуть негайно проводити угоди на обох біржах. Дослідження показують, що значний відсоток арбітражу на крипторинках сьогодні здійснюється автоматизованими ботами, жодна людина не могла б робити ці ходи конкурентно. На сьогоднішній день існує понад 10 000 різних криптовалют, серед яких, попри домінування низьколіквідних активів і проєктів з невеликою капіталізацією, зберігається значна кількість популярних монет – таких як Bitcoin, Ether, Ripple, Litecoin.

Навіть цей обмежений сегмент забезпечує набагато ширший спектр інструментів, ніж традиційні валютні пари на форекс чи основні фондові індекси. Як встановлено в опитуванні Ruiz et al., дослідження алгоритмів криптовалют зараз в основному зосереджені на декількох найбільших монетах (наприклад, BTC, ETH), тоді як тисячі монет з меншою капіталізацією менш досліджені. Це надає простір для нових підходів, особливо підходів машинного навчання з використанням інструментів Big Data для визначення прихованої інформації з маловідомих токенів.

Розумний агент може бути запрограмований на пошук таких неочевидних можливостей в довгому хвості криптоактивів. Ще один фактор полягає в тому, що крипторинок молодший і більш волатильний. Там, де фондовий ринок має понад 100 років історії, і таким чином, окремі випробувані часом шаблони, криптовалюти існують трохи більше десяти років. За цей час вони пережили як підйоми, так і спади, і природа цінової дії часто змінювалася. Динаміка ринку в цьому випадку є непередбачуваною: твіти від знаменитостей, скандали на біржах, рішення

регуляторів різних країн щодо біткойнів тощо. Але це створює інші

небезпеки. Тим часом гнучкі алгоритми, особливо ті, що інтегрують штучний інтелект, здатні швидше підлаштовуватися, ніж ті, хто ще ніколи не торгував у такому середовищі. Таким чином, розумний агент у крипторинку є прагненням до синергії – ви прагнете змішати ці масово високі потенціали автоматизації з надзвичайно високоволатильним середовищем, щоб створити прибутковий алгоритм.

Нарешті, варто зазначити, що на ринок вийшла велика кількість нових роздрібних інвесторів та професійних фондів, зацікавлених у криптовалютах. Станом на 2024 рік більше ніж 560 мільйонів людей по всьому світу мають якусь форму криптовалюти. Така широка база користувачів забезпечує великий обсяг торгів та ліквідність на ринках провідних монет. Щоденні обсяги торгівлі криптовалютами сягають десятків мільярдів доларів. Для інтелектуального агента багато агентів і транзакцій являють собою по суті дані та можливості – ви можете швидше визначати статистичні патерни або тимчасові неефективності, коли вони масово поширені. В цілому крипторинок є привабливим, але складним для бот-торгівлі. Бути настільки відгукуючим та гнучким є однією з переваг підходу, але разом з цим виникає ризик помилки. Саме тому розроблення розумного торгового агента для торгівлі та інвестування у криптовалюту є відповідною роботою. Це дозволяє продемонструвати всі переваги алгоритмічного підходу в динамічно розвиваючомуся середовищі реального ринку.

1.5 Історичні приклади фінансових криз та збоїв

Щоб краще усвідомити ризики та наслідки застосування різних стратегій, корисно розглянути кілька відомих історичних прикладів, пов’язаних з алгоритмічною або автоматизованою торгівлею.

1.5.1 Long-Term Capital Management (1998)

Одним із найвідоміших випадків краху інвестиційної стратегії є історія хедж-фонду Long-Term Capital Management (LTCM). Це був фонд, створений у 1994 році видатними фінансистами, серед яких були лауреати Нобелівської премії з економіки. LTCM спеціалізувався на арбітражних та висококваліфікованих стратегіях на облігаційному ринку, використовуючи надскладні математичні моделі. Вони успішно заробляли кілька років, і до 1998-го розмір фонду сягнув $3,5 млрд власного капіталу, а з урахуванням позикових коштів LTCM оперував десятками мільярдів доларів. Фонд обіцяв інвесторам стабільні високі прибутки, використовуючи арбітражні можливості на глобальних ринках облігацій. На рисунку 1.1 буде їх зображення для статті

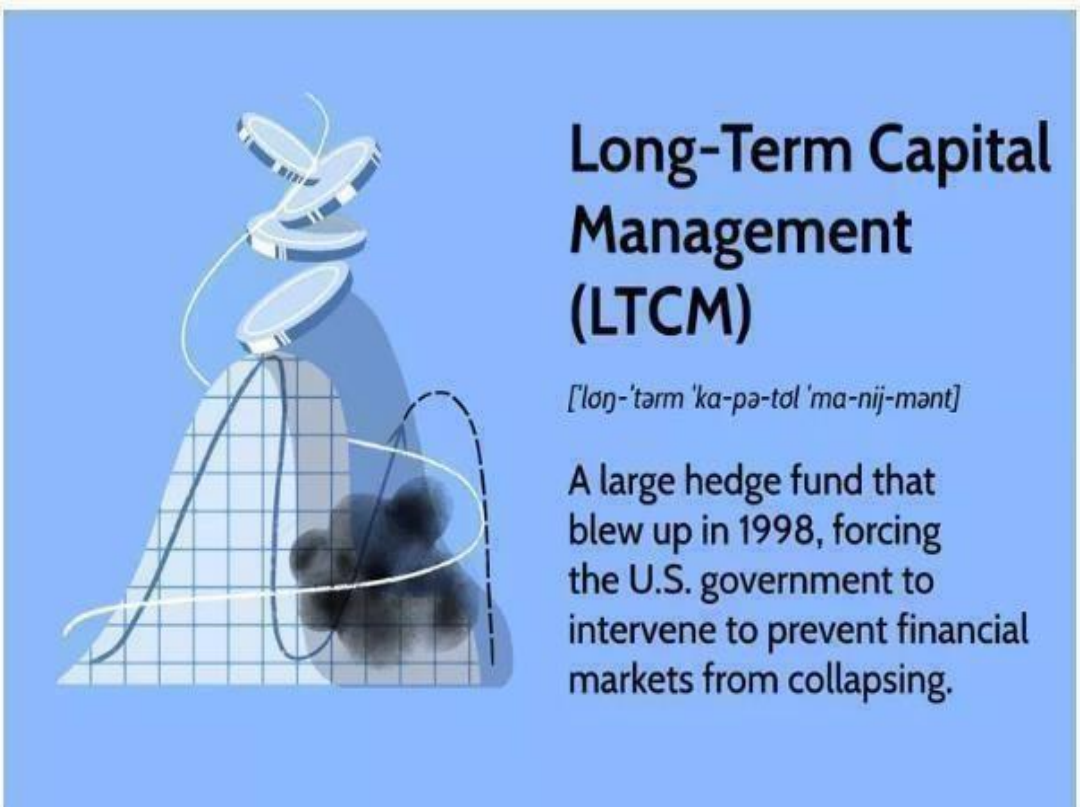

Рисунок 1.1 – Стаття «What Was Long-Term Capital Management (LTCM) and What Happened?»

Однак у 1998 році сталися події, яких моделі LTCM не передбачили. Росія оголосила дефолт за своїми облігаціями, що спричинило ланцюгову реакцію на світових фінансових ринках. Ті арбітражні позиції, які фонд тримав, раптом почали приносити колосальні збитки. LTCM був сильно засилений (багато боргу) – він торгував на позикові кошти таким чином, що на кожен $1 власного капіталу він мав позиції вартістю $20 до $30. Це призвело до того, що незначні зміни цін привели до геометричного зростання втрат. Протягом кількох місяців LTCM втратив $4,6 мільярда, що еквівалентно 90% його активів. Облігація була близька до банкрутства і ставила під загрозу всю глобальну фінансову систему (оскільки контрагенти LTCM були найбільшими з великих банків Уолл-стріт).

У вересні 1998 року Федеральний резервний банк Нью-Йорка організував порятунок до $3,6 мільярда для покриття втрат LTCM і переведення активів фонду, щоб врятувати його від банкрутства та попередити «ефект доміно». Це було попередженням про небезпеки надмірної впевненості в математичних моделях та високому заборгуванні. Стратегія LTCM не була алгоритмічною у сучасному сенсів исокочастотної комп'ютерної торгівлі – фонд цього не робив (принаймні, не в ті роки); в більшості випадків він працював вручну, але методично, використовуючи моделі. Але її часто згадують у контексті сучасних алгоритмічних проблем, оскільки в основі це та ж сама проблема: недостатньо уваги приділено тому, наскільки погані можуть бути обставини, і занадто багато довіри покладено в модель. Для нашої роботи це важливий приклад, оскільки він демонструє, що навіть «інтелектуальна» система потребує обмежень для ризиків і повинна передбачати події, які можуть перевершити історичні прецеденти.

1.5.2 Спалаховий крах

На американському фондовому ринку 6 травня 2010 року відбулася подія, яка увійшла в історію як Flash Crash – спалаховий крах. Близько 14:45

за нью-йоркським часом індекс Dow Jones раптово втратив понад 1000 пунктів, що склало близько 9% загального обсягу – падіння, яке тривало всього кілька хвилин. Це стало однією з найрізкіших короткострокових просадок в історії торгів.

Ринок поводився абсолютно аномально: акції провідних компаній миттєво обвалювались до символічних значень у кілька центів, а вже за хвилини – повертались до нормального рівня. Протягом приблизно 30 хвилин були фактично «стерті» трильйони доларів ринкової капіталізації, але більшість з них так само стрімко була відновлена ще до завершення торгової сесії.

Падіння виглядало хаотичним і неконтрольованим, викликавши паніку серед трейдерів і значний резонанс у фінансовій спільноті. Рисунок 1.2 демонструє масштаби цього обвалу – ціни на окремі активи моментально досягали аномально низьких значень, що не мало економічного підґрунтя, після чого відскакували назад із не меншою швидкістю.

Цей інцидент став каталізатором активного обговорення ролі високочастотної торгівлі (HFT) та автоматизованих алгоритмів, які, за підозрою аналітиків, могли не лише поглибити падіння, а й спричинити його запуск. У результаті Flash Crash послужив серйозним сигналом для регуляторів, які були змушені переглянути правила ринкової стабілізації та поведінки торгових роботів.

Розслідування Комісії з цінних паперів та бірж (SEC) та Комісії з торгівлі товарними ф'ючерсами (CFTC) пізніше виявило, що флеш крах було спричинено складними факторами, пов'язаними з алгоритмічною торгівлею. Один великий інституційний інвестор звернувся до алгоритму, який він створив, щоб швидше продавати ф'ючерсні контракти E-Mini S&P 500 з метою хеджування. Цей алгоритм продав багато контрактів швидко, незважаючи на те, що могло статися з ціною – зверніть увагу: агресивний алгоритм масового продажу для початкового обвалу.

Рисунок 1.2 – Стаття «Flash Crash: Definition, Causes, History»

У відповідь маркет-мейкери (постачальники ліквідності), які були HFT, спостерігаючи за великим зниженням, також продавали або скасовували свої замовлення на ринку, щоб зменшити ризик. Ліквідність випарувалася, спреди розширилися. Інші алгоритми, запрограмовані з тригерами стоп-лоссів та подібними, автоматично почали продавати, коли побачили падіння індексу – це розпалило полум'я лавини продажів. Одна машина зупинилася, тоді інша, поки не виник ефект доміно, керований машинами: ціни, алгоритми, ціни, алгоритми, і це тривало, поки не зупинилося.

Ці події розвивалися так швидко, що людські трейдери дійсно не могли зрозуміти, що відбувається. Лише коли індекс обвалився майже на 1000 пунктів (деякі алгоритми автоматично вийшли або припинили торгівлю через вбудовані обмежувачі, і регулятори викликали механізми обмеження, що зупиняють торгівлю окремими акціями), стало зрозуміло – постфактум – багато з того, що сталося.

Ринок почав відновлюватися. Більшість, якщо не всі «втрати», були відновлені до 3:15 після обіду, залишивши глибоке V-подібне падіння в графіках, що охоплює цю годину. І в певному сенсі, флеш крах 2010 року

надав історії логічне «і тоді» – наочну демонстрацію того, як алгоритми можуть створювати хаос, навіть коли вони не грають проти якогось основного фактора. Насправді ринок «запустив» себе в безодню і вийшов з неї. Подальші дослідження підтвердили, що основними причинами були високошвидкісна торгівля та відсутність адекватних засобів захисту в алгоритмах. Як результат, відбулося багато регуляторних змін: біржі ввели

«механізми обмеження», що зупиняють торгівлю, коли ціна акції падає більш ніж на певний відсоток протягом декількох хвилин, і обмежили використання прямого доступу до ринку для алгоритмів непрофесійними трейдерами.

Флеш крах, для наших цілей, підкреслює необхідність контролювати і тестувати алгоритми. Інтелектуальний агент повинен мати засоби для уникнення несподіваних поведінок – наприклад, максимальний обсяг торгівлі за певний період, тест на аномальні ринкові умови, після чого агент повинен бути призупинений. Також розумно передбачити «перемикач відключення» у випадку, коли весь агент потрібно вимкнути від торгівлі, щоб уникнути катастрофічного збою. Флеш крах показує, що некеровані, непередбачені наслідки автоматизації можуть бути шкідливими.

1.5.3 Крах екосистеми Terra/Luna

Світ криптовалют також зазнав власних «крахів», пов’язаних з алгоритмами – хоча у цьому випадку йдеться не стільки про торгових ботів, скільки про алгоритмічний фінансовий механізм. У травні 2022 року відбувся драматичний обвал криптовалют TerraUSD (UST) та Luna, який підірвав довіру до так званих алгоритмічних стейблкоїнів. TerraUSD була задумана як стейблкоїн – токен, прив’язаний до курсу долара 1:1, але без реального забезпечення доларами; натомість стабільність підтримувалася алгоритмом, пов’язаним з іншою криптовалютою – Luna. Механізм працював так: за 1 TerraUSD завжди можна було отримати на ринку Luna на

суму $1 і навпаки (через спеціальний смарт-контракт). Ідея в тому, що якщо TerraUSD падає нижче $1, то арбітражери купуватимуть дешевий UST і обмінюватимуть на Luna на $1, поки курс не вирівняється; якщо ж TerraUSD вище $1, то навпаки, його будуть карбувати з Luna. Цей алгоритм довгий час успішно тримав курс UST = $1.

Проте в травні 2022 стався кризовий «відв'язування» стейблкоїна. На фоні загального падіння крипторинку інвестори почали масово продавати TerraUSD, втративши довіру до його стабільності. Алгоритм почав випускати все більше монет Luna, щоб викупити TerraUSD і повернути курс, але це призвело до обвалу ціни Luna (надмірна пропозиція знецінила її). Вартість Luna стрімко пішла вниз, а з нею і TerraUSD втрачав прив'язку все сильніше. Команда «CFI» було зазначено,що виникла паніка[3] – своєрідний банк-ран на алгоритмічний стейблкоїн. За лічені дні TerraUSD впав до кількох центів замість $1, а Luna знецінилася практично до нуля (з більш ніж $100 за монету до менш ніж $0.0001). Цей крах знищив близько $40–50 млрд сукупної ринкової вартості UST і Luna, а також спричинив ланцюгову реакцію банкрутств пов'язаних проектів і фондів.

Недоліком алгоритму Terra/Luna було те, що доводилося довіряти, що Luna зможе утримувати достатню цінність для підтримки стейблкоїну. Алгоритм спіраллювався вниз, коли ринок похитнувся, алгоритм почав входити в спіраль смерті – чим більше TerraUSD було випущено для підтримки TerraUSD, тим більше Luna також було випущено для підтримки боргових зобов'язань, знецінюючи Luna і ще більше підриваючи довіру до TerraUSD. Це був замкнутий цикл, який неможливо було зупинити. Дійсно, ми бачили, як автоматизована фінансова система зазнала краху у криптовалютах через недосконалий модель і екстремальне поведінкове підкріплення інвесторів.

Урок з Terra/Luna такий: навіть «розумні» алгоритми у фінансах можуть бути хибними, якщо їх використовувати за межами стабільних режимів. Стрес-тести повинні проводитися і потрібно визнавати той факт,

що поведінка людей (і алгоритмів) в паніці може спростувати початкову гіпотезу стратегії. Торговий агент не відповідає за стабільність валют, але, як і в реальній криптовалютній ринковій ситуації, все відбувається швидко. Тому агент має бути спроектований обережно: щоб передбачити, як він поводитиметься, якщо ринок переживає втрати ліквідності або знаходиться у неконтрольованому падінні. Випадок Terra також продемонстрував, що ризики пов'язані з контрагентами та основними системами є критично важливими – щоб ми не забули, крах стейблкоїну похитнув увесь ринок. Торговий агент також може мати, скажімо, план підйому, коли біржа перестає працювати чи API зломлений або відбувається щось інше зовнішнє. Алготорговля не є імунною та іноді вона також їх запускає.

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Інтелектуальний агент

На основі вищеописаного аналізу предметної галузі сформульовано чітку мету даної дипломної роботи: створити інтелектуального торгового агента, який використовує методи технічного аналізу для автономного здійснення торговельних операцій на криптовалютному ринку, усуваючи обмеження ручної торгівлі та забезпечуючи високу ефективність і надійність стратегій.

Іншими словами, потрібно розробити програмну систему, яка буде самостійно торгувати обраними криптоактивами за заданим алгоритмом, демонструючи при цьому результати не гірші (а бажано – кращі) за традиційний підхід людини-трейдера.

Для розв’язання окресленої проблеми необхідно виконати цілий комплекс дій. Насамперед формулюється торгова стратегія. Знання та методики технічного аналізу перетворюються на формалізовані правила, придатні для програмування. Технічний аналіз трактується не як завершена теорія, а як набір числових параметрів і умов, що можуть бути перевірені під час торгівлі.

Формалізації підлягають аналіз ринку, сила імпульсу, умови відкриття й закриття позицій, а також правила управління ризиком. На основі цих параметрів логіку агента викладають у вигляді псевдокоду або блок-схеми, після чого розробляється алгоритм дій для типових ринкових ситуацій (зростання, флет, різке падіння тощо).

Агент реалізується як програмне забезпечення. Програмується функціональний прототип системи з модулями підключення до API криптобіржі, отриманням цінових даних та розміщенням ордерів. Усі правила стратегії – обчислення індикаторів, перевірка тренду та імпульсу, генерація торгових сигналів – кодуються у вигляді окремих функцій.

Додається логування, обробка помилок і механізми лімітування ризику. Продуктивність забезпечується таким чином, щоб обробка даних та виконання дій відбувалися у межах кількох секунд або 1–2 хвилин, що відповідає часовим рамкам стратегії внутрішньоденної торгівлі. Передбачено інтерфейс конфігурації, у якому задаються параметри індикаторів, перелік торгових активів тощо.

Після програмування алгоритм проходить валідацію. Спочатку тестування здійснюється на історичних даних, далі – у режимі «паперових» угод, а згодом – на малих обсягах реального ринку. Перевіряється відповідність очікуванням: позитивна дохідність, прийнятне співвідношення середнього прибутку до середніх втрат, адекватна реакція на трендові, флетові та волатильні періоди.

Контролюється правильність управління ризиком: відсутність перевищень стоп-рівнів, допустимі розміри позицій, дотримання обмежень по просадці. У разі виявлення недоліків алгоритм переглядається, доповнюється новими правилами або корегується шляхом зміни параметрів.

Наступний етап передбачає аналіз результатів і оптимізацію. Оцінюється загальна прибутковість стратегії з урахуванням торгових комісій, визначається волатильність прибутку та максимальна просадка, порівнюються результати з еталонними підходами (купівля-утримання, прості ковзні середні тощо). Виявляються сильні й слабкі сторони агента. За потреби оптимізуються періоди індикаторів та пороги сигналів шляхом перебору або застосування евристичних алгоритмів.

У такий спосіб завдання поділяється на дві складові: теоретико-алгоритмічну (визначення правил і логіки агента) та практично-реалізаційну (код, тестування, налагодження). Обидві частини висвітлюються в дипломі – від обґрунтування вибору технічних індикаторів до перевірки розробленого програмного комплексу на реальних ринкових даних.

2.2 Очікуваний результат

Ринковий аналіз у розробленій системі здійснюється через безперервне зчитування котирувань криптовалют із публічних REST- або WebSocket-інтерфейсів.

Ці дані піддаються обробці в реальному часі, зокрема, виконується обчислення ряду технічних індикаторів, серед яких RSI (Relative Strength Index), MACD (Moving Average Convergence Divergence), ATR (Average True Range), експоненційні ковзні середні (EMA), кроси середніх MA50/MA200 тощо. Зазначені індикатори дозволяють оцінити ринкову фазу – перекупленість, перепроданість, напрямок та силу тренду, інтенсивність імпульсів і рівень волатильності.

На основі цих параметрів система формує сигнали для відкриття (entry) або закриття (exit) торгових позицій. Логіка сигналів реалізована у вигляді чітко формалізованих правил, що враховують значення індикаторів у поточний момент.

Наприклад, відкриття позиції можливе лише при одночасному виконанні кількох умов: перетин MACD-ліній у напрямку тренду, RSI у допустимому діапазоні, підтвердження з боку глобального таймфрейму тощо.

Після генерації сигналу система автономно готує та відправляє торговий наказ через API криптовалютної біржі (в офлайн-режимі – через бектест-середовище). Паралельно відстежується статус виконання угоди. При виконанні наказу оновлюються всі пов’язані внутрішні параметри агента, включаючи капітальну криву, відкриті позиції, поточний ризик тощо.

Ключову роль у забезпеченні надійності системи відіграє модуль управління ризиками. Він виконує контроль за обсягом позицій, максимально допустимими збитками, співвідношенням прибутку до ризику (risk/reward), а також автоматично активує Stop Loss у випадку

різкого зниження ціни. Всі ці дії налаштовуються заздалегідь і виконуються без участі трейдера.

Особливу увагу приділено адаптації системи до змін ринкової динаміки. Вбудовані механізми адаптації враховують підвищення або зниження волатильності активу: при різких рухах система зменшує торгові обсяги або повністю припиняє операції до стабілізації умов; натомість у «спокійних» фазах допускається розширення позицій, якщо ризики залишаються в межах припустимих значень.

Окремий модуль відповідає за створення звітної інформації: усі дії агента, включаючи час входу/виходу, ціну, профіт, тип стратегії, сигнал, що активував угоду, та інші параметри – фіксуються у лог-файлах та CSV-документах. Це створює повну історію прийнятих рішень і дозволяє проводити подальший аналіз для покращення системи.

У результаті впровадження згаданих функціональних елементів було створено повноцінний прототип інтелектуального торгового агента, здатного до автономного прийняття рішень, адаптивного вибору торгової стратегії та її реалізації без безпосередньої участі людини.

Архітектура системи передбачає чіткий розподіл ролей між модулями: збір і обробка ринкових даних, генерація сигналів, формування навчального датасету, машинне навчання моделі вибору та фінальна аналітика результатів.

Ефективність агента була підтверджена за результатами ретельного тестування на історичних даних криптовалютного ринку. Проведений бектест продемонстрував стабільне зростання капіталу за умов дотримання розумного рівня просадок, що підтверджує практичну доцільність та надійність запропонованого підходу. Агрегована крива капіталу, побудована агентом, показала збалансовану динаміку прибутковості, що вказує на здатність системи адаптуватися до змін ринкового середовища без втрати ефективності.

Подальший розвиток агента передбачає розширення його функціоналу, зокрема, за рахунок інтеграції моделей прогнозування з використанням глибокого навчання, підтримки мультивалютної торгівлі, побудови динамічних портфелів активів та підключення до реальних торгових майданчиків для роботи в онлайн-режимі. Це дозволить не лише покращити точність прийняття рішень, а й зробити агента універсальним інструментом у сфері автоматизованої торгівлі.

3 МЕТОДИ АВТОМАТИЗОВАНОЇ ТОРГОВЛІ НА ОСНОВІ ШІ

3.1 Концепції технічного аналізу у торгівлі криптовалютами

Технічний аналіз базується на вивченні минулих цін і обсягів торгів для прогнозування подальших змін на ринку. Трендом називають напрям руху ціни: вгору, вниз або вбік. Визначення напрямку дозволяє орієнтуватися у поточній ринковій ситуації – під час зростання ціни доцільно купувати, під час зниження – продавати. Для виявлення трендів застосовуються спеціальні індикатори. Один із найпоширеніших – ковзні середні (Moving Averages), які відображають середню ціну за певний період. Проста ковзна середня (SMA) враховує всі значення однаково, а експоненціальна (EMA) надає більшої ваги останнім даним. Такі середні згладжують графік і дають змогу побачити загальну тенденцію: зростання середньої вказує на висхідний тренд, спад – на низхідний.

Ще одним індикатором є MACD (Moving Average Convergence Divergence), який аналізує взаємодію двох EMA. Перетин ліній MACD сигналізує про можливу зміну тренду: якщо основна лінія перетинає сигнальну знизу – очікується зростання, зверху – можливе падіння.

На графіку з прикладом технічного аналізу Bitcoin позначено кілька важливих рівнів. Червоні лінії вказують на зони опору – приблизно на рівнях $100K–$108K. Зелені лінії показують підтримку – близько $90K, а також між $72K і $74K. Видно також висхідний канал (позначений синім), у межах якого змінюється ціна.

Фіолетовою лінією зображена 200-денна ковзна середня (SMA), яка часто виступає як динамічний рівень підтримки. Нижче основного графіка знаходяться стовпчики, що показують обсяги торгів, а також осцилятори RSI та MACD – вони допомагають оцінити, наскільки сильним є поточний рух ціни [6]. Візуально це все можно побачити на рисунку 3.1.

Рисунок 3.1 – Приклад графіку з технічним аналізом зі статті «Support and Resistance»[6]

Імпульс та осцилятори. Індикатори імпульсу вимірюють швидкість і силу цінових змін, допомагаючи визначити перекупленість або перепроданість ринку [7].

RSI (Relative Strength Index) – один із класичних осциляторів, значення якого змінюється в межах від 0 до 100. Показує силу руху ціни за останній час. Коли показник перевищує 70, це часто свідчить про перекупленість – є ймовірність, що ціна піде вниз. Якщо ж RSI нижче 30, це може означати перепроданість і можливий відскок угору.

Через високу волатильність криптовалют цей індикатор часто застосовується для виявлення моментів розвороту, коли ринок наближається до крайніх значень.

Ще один відомий осцилятор – стохастичний. Він теж має шкалу від 0 до 100. Значення понад 80 сигналізує про перекупленість, а менше 20 – про перепроданість.

Осцилятори корисні для виявлення моментів, коли тренд втрачає силу. Наприклад, якщо ціна оновлює максимум, а осцилятор цього не

показує, це може свідчити про можливий розворот вниз (так зване явище дивергенції) [7].

Обсяг торгів виконує роль підтвердження сили руху ціни. Чим більший обсяг під час зміни ціни, тим більше учасників ринку підтримують цей рух. Це ознака стабільного тренду.

Один із ключових індикаторів – OBV (On-Balance Volume), який підсумовує обсяги купівлі та продажу. Зростання цього показника вказує на накопичення – коли переважають покупці [7]. Це часто передує підвищенню ціни. Якщо OBV знижується – це ознака переваги продавців, тобто ринок розпродається. Коли ціна оновлює максимум, а OBV не росте, можна говорити про слабкість тренду – рух вгору не підтримується обсягом.

Ще один індикатор – Volume Rate of Change (VROC). Він відстежує, як швидко змінюється обсяг. Якщо обсяг різко зростає, а ціна при цьому майже не рухається, це може бути сигналом про майбутній сильний прорив [7]. Рівні підтримки та опору визначають важливі зони на графіку. Підтримка – це така ціна, де попит стає досить сильним, щоб зупинити зниження і сприяти розвороту вгору. Опір – це рівень, де пропозиція не дає ціні піднятись далі [8]. Ці рівні відображають психологію ринку: багато учасників ринку купують поблизу рівнів підтримки, вважаючи їх вигідними, і продають біля рівнів опору, сприймаючи ціни як завищені. На графіках підтримку та опір визначають за локальними мінімумами та максимумами або проводять горизонтальні лінії через області, де ціна неодноразово змінювала напрям. Такі ділянки називають зонами підтримки або опору, оскільки розворот часто відбувається не на точному рівні, а в межах діапазону [6]. Якщо ціна прориває рівень опору вгору, він може стати новим рівнем підтримки – це називається принципом полярності. Пробій рівня опору на великому обсязі сигналізує про початок зростання, тоді як падіння нижче підтримки – про посилення зниження.

Фігури на графіках показують типову реакцію ринку на зміни ціни. Вони формуються на основі повторюваної поведінки учасників торгів.

Серед класичних розворотних моделей – «голова та плечі», подвійна вершина або подвійне дно. Патерн «голова та плечі» має три піки, де середній (голова) розташований вище за два бокові (плечі). Така фігура часто свідчить про ослаблення зростання і можливість розвороту вниз [9]. На рисунку нижче показано схему класичної фігури Head and Shoulders Top – після її формування ціна пробиває «лінію шиї» і переходить у спадний рух. Побачити це можно на рисунку 3.2

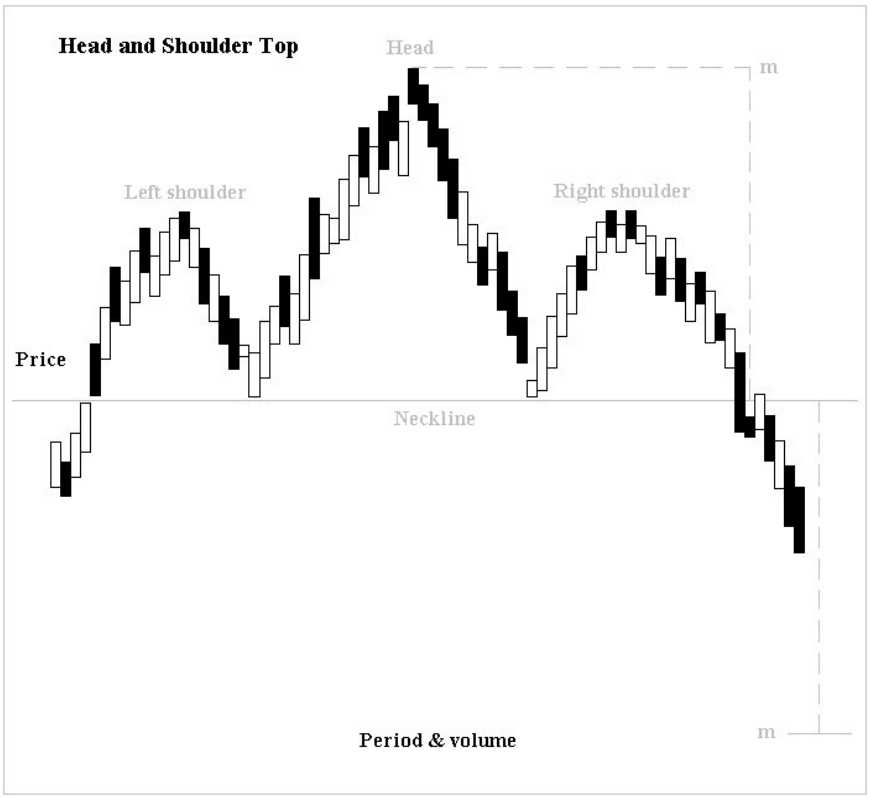

Рисунок 3.2 – Схематичне зображення розворотної фігури «Голова та плечі»

Існують також фігури продовження тренду (наприклад, трикутники, прапори, клини), що сигналізують про тимчасову консолідацію перед продовженням попереднього тренду. Технічні аналітики враховують форму і розміри таких фігур, щоб оцінити потенційну ціль руху після їх завершення.

Осцилятори (RSI, стохастик, MACD тощо) називаються так тому, що коливаються в обмеженому діапазоні значень і циклічно вказують на фази перекупленості/перепроданості ринку. Важливо розуміти, що сигнали індикаторів не є безпомилковими – їх потрібно трактувати з урахуванням загального контексту ринку. Наприклад, під час сильного висхідного тренду RSI може надовго «застрягти» в зоні вище 70, і це не означатиме негайного падіння. Тому трейдери часто комбінують кілька індикаторів різного типу для підтвердження сигналів. Наприклад, поєднання трендового індикатора (як-от EMA) з осцилятором RSI допомагає відфільтрувати хибні сигнали: якщо EMA вказує на висхідний тренд, а RSI виходить із зони перепроданості, це разом підсилює довгий сигнал. Так само, прорив ціни із діапазону Боллінджера, підкріплений зростаючим OBV, свідчить про істинність руху. Комбінація кількох методів технічного аналізу дає більш повну картину ринку і підвищує вірогідність успішних рішень трейдера [7].

3.2 Програмні основи побудови інтелектуальних торгових агентів

Інтелектуальний торговий агент – це просто програма, яка замість людини приймає рішення, коли саме купувати або продавати активи. Робить вона це автоматично, за певними правилами або навіть на основі штучного інтелекту. Це як розумний бот, який постійно моніторить ринок і реагує швидше за людину.

Є кілька типів таких агентів. Найпростіші – це системи на основі правил (rule-based). Їм задаються чіткі інструкції: наприклад, «якщо ціна виросла вище 50-денної середньої і RSI менше 30 – значить, купити». Тут усе чітко і зрозуміло. Такі боти не навчаються – вони просто виконують інструкції. Це плюс, бо зрозуміло, що саме відбувається, і можна все швидко змінити. Але є й мінус – якщо ринок зміниться, а в правилах це не передбачено, бот буде діяти неефективно.

Другий підхід – агентно-орієнтовані системи. В системі є кілька

«міні-ботів», кожен із яких має свою роботу: один збирає дані, інший аналізує, ще один приймає рішення щодо купівлі або продажу. Вони можуть працювати паралельно, взаємодіяти один з одним і охоплювати кілька ринків одразу. Це складніше, але така система гнучкіша й надійніша. Якщо один агент перестане працювати – інші можуть продовжити роботу.

Третій варіант – агенти з машинним навчанням. Це вже щось ближче до штучного інтелекту. Вони не мають чітких правил – їх навчають на історичних даних. Наприклад, така система може «вивчити», які патерни на графіку призводять до зростання або падіння, і прогнозувати, що буде далі. Тут використовують нейронні мережі, дерева рішень, ансамблеві моделі – все, що може аналізувати великі обсяги інформації. Такі агенти можуть помічати зв'язки, які людині або rule-based системі просто не видно.

Перевага в тому, що це гнучкий і потужний інструмент. Але є і складності: часто неможливо зрозуміти, чому модель ухвалила саме таке рішення (це називають «чорним ящиком»), і є ризик, що вона «перенавчиться» – буде працювати ідеально на старих даних, але провалиться на нових, бо ринок змінився [10].

Навчання з підкріпленням (reinforcement learning, RL) – це коли агент вчиться торгувати сам, через проби й помилки. Йому ніхто прямо не каже, що правильно, а що ні. Він просто пробує різні дії, отримує «нагороду» (прибуток) або «штраф» (збиток), і на основі цього поступово розуміє, що працює, а що ні.

У торгівлі така система працює так: середовище для агента – це ринок. Поточний стан – це ціна, індикатори, обсяги й інші параметри в даний момент. Дія – це купити, продати або нічого не робити. А нагорода – це результат цієї дії: скільки зароблено або втрачено.

Мета агента – навчитися ухвалювати такі рішення, які дадуть найбільший прибуток у довгостроковій перспективі. Це схоже на те, як людина вчиться грати в гру: чим більше грає, тим краще розуміє, як вигравати.

Сильна сторона RL у тому, що такий агент може знаходити дуже нестандартні, але ефективні стратегії, про які люди або прості алгоритми навіть не думають. Наприклад, якщо підключити до такого агента глибоку нейронну мережу, він зможе враховувати цілу послідовність ринкових подій, а не просто дивитися на один момент. Це дозволяє краще «відчувати» ринок.

Дослідження показують, що такі агенти можуть реально бути ефективними, особливо якщо оптимізувати їх під конкретні цілі – наприклад, не просто максимальний прибуток, а співвідношення прибутку до ризику (Sharpe ratio), або мінімальні просадки в балансі [11].

Але є і мінуси. Щоб агент навчився добре, йому потрібні тисячі, а то й мільйони симуляцій – це займає багато часу й ресурсів. Також не завжди просто змусити його вчитись стабільно: іноді він «залипає» на неефективних стратегіях. І ще один нюанс – така система може бути дуже непередбачуваною. Якщо умови зміняться, агент може почати діяти дивно або навіть шкідливо.

Гібридні підходи – це коли в одній торговій системі об’єднують різні методи, щоб компенсувати слабкі сторони одного підходу за рахунок сильних сторін іншого. Наприклад, можна взяти rule-based правила (тобто чітко прописані умови) і додати до них агента з навчанням з підкріпленням, який буде більш гнучким і сам вчитиметься на власному досвіді.

У такій системі rule-based частина може, наприклад, обмежувати ризикові рішення – не дозволяти боту відкривати занадто небезпечні угоди. А RL-агент буде пробувати різні стратегії в межах дозволеного, поступово вдосконалюючи свої дії.

Ще один варіант – поєднати машинне навчання з експертними знаннями. Тобто модель прогнозує, куди піде ринок, а вже остаточне рішення про купівлю або продаж приймається на основі прописаних правил.

Суть гібридних підходів у тому, щоб поєднати найкраще: швидкість, адаптивність і гнучкість штучного інтелекту – з передбачуваністю,

стабільністю й контрольованістю класичних рішень. Дослідження показують, що такі системи здатні навчатися швидше й ефективніше, бо отримують підказки не тільки з досвіду, але й з уже наявних знань [12].

Ще один цікавий варіант – це поєднання генетичних алгоритмів або еволюційних стратегій з навчанням з підкріпленням [12]. У цьому випадку система спочатку «еволюціонує» торгові правила – створює багато різних варіантів, тестує їх, залишає найкращі, змінює і знову тестує. А вже потім RL-алгоритм допомагає ці правила вдосконалити під час практичного застосування на ринку.

Такі гібридні алгоритми добре пристосовуються до змін. Якщо ринок змінюється, система може перемикатися між різними наборами правил або взагалі перенавчити свою модель під нові умови. Це важливо, бо фінансові ринки нестабільні – те, що працювало вчора, завтра може не спрацювати.

Головна мета таких гібридних систем – не просто заробити більше, а робити це розумно: з контролем ризиків і з урахуванням змін середовища. Вони поєднують гнучкість, адаптивність і здатність швидко підлаштовуватися під нові обставини.

Вибір основи для створення торгового бота залежить від того, які цілі ставляться і які ресурси доступні. Якщо головне – щоб усе було прозоро, зрозуміло й під контролем, тоді краще використовувати rule-based підхід, де всі правила чітко прописані. Але якщо важлива гнучкість і здатність адаптуватися до змін на ринку, то вже потрібні методи машинного навчання або навчання з підкріпленням.

На практиці часто використовують комбінацію. Наприклад, на першому рівні працюють прості правила, які стежать за ризиками, а над ними – більш складна модель ШІ, яка аналізує ринок і приймає торгові рішення. Остаточне рішення приймається з урахуванням обох джерел. Це дає перевагу: якщо ШІ раптом помиляється, то вбудоване правило зможе підстрахувати і не дати втратити великі гроші.

У результаті сучасний торговий агент – це вже не просто скрипт, який купує і продає. Це складна система, яка поєднує кілька підходів: алгоритми, машинне навчання, багаторівневу логіку та автономних агентів. Усе це робиться для того, щоб досягати максимально ефективних результатів у роботі на фінансовому ринку.

3.3 Інтерфейси доступу до криптобірж: REST і WebSocket API, безпека та продуктивність

Інтелектуальний торговий агент (простими словами – бот) не працює «сам по собі». Щоб взаємодіяти з біржею, йому потрібен спеціальний «міст» – це і є API. API (Application Programming Interface) – це набір правил, за якими програма (в нашому випадку – бот) може спілкуватися з криптобіржею. Через API агент отримує дані про ціни, баланси, обсяги і може надсилати команди, наприклад, відкрити або закрити ордер.

У криптоторгівлі найбільш популярні два типи API: REST API і WebSocket API. Кожен із них має свої плюси та мінуси, і вони підходять для різних ситуацій.

REST API – це класичний варіант. Він працює через звичайні HTTP-запити, як і більшість сайтів. Коли агенту потрібно дізнатися щось – наприклад, останню ціну BTC/USDT – він надсилає запит, а біржа повертає відповідь. Кожен такий запит – окремий і незалежний. Сервер не запам’ятовує, що було до того, тому REST називають безстанговим (stateless) [13].

REST API дуже простий у використанні й підходить для базових речей: перевірити баланс, надіслати ордер тощо. Але є мінус: якщо потрібне оновлення даних у реальному часі (наприклад, ціна постійно змінюється), доводиться надсилати багато запитів на секунду – це називається полінг [13].

WebSocket API – це сучасніший варіант. Після початкового з'єднання між ботом і біржею встановлюється постійний канал зв’язку. Тепер не потрібно щоразу питати: «Що нового?» – біржа сама надсилає оновлення, щойно щось змінюється. Це дуже зручно і швидко.

Працює в режимі реального часу. Біржа може стрімити стрічку цін, глибину ринку, нові ордери тощо, а бот може одразу ж реагувати – наприклад, надіслати команду на купівлю або продаж. Такий підхід підходить для ситуацій, де важлива миттєва реакція: онлайн-ігри, спільна робота над документами, біржовий трейдинг [13].

У трейдингу WebSocket дає змогу підписатися на конкретні канали (наприклад, трансляцію ціни або ордербуку), і отримувати інформацію за долі секунди після її появи на біржі. Завдяки цьому боти можуть приймати рішення практично миттєво, не втрачаючи час.

Отже, якщо бот повинен просто час від часу перевіряти щось або виконувати окремі дії – REST API цілком достатньо. Але якщо потрібна робота в реальному часі – без WebSocket не обійтись.

У реальному житті торгові системи часто використовують обидва варіанти: REST – для управління ордерами, WebSocket – для стрімінгу ринкових даних.

Нижче наведено порівняння двох основних типів API, які використовуються для зв’язку торгових ботів з біржею. Це допоможе краще зрозуміти, чому в одних випадках вибирають REST, а в інших – WebSocket. REST працює за принципом «запит-відповідь» і не зберігає стан з’єднання, тому підходить для простих запитів і базових дій (CRUD- операцій).

WebSocket натомість встановлює постійний двосторонній канал і дозволяє обмінюватися даними безперервно й практично без затримок. Це особливо корисно для завдань, де важлива швидкість реакції – наприклад, у трейдингу або чат-додатках. Порівняння показано на рисунку 3.3.

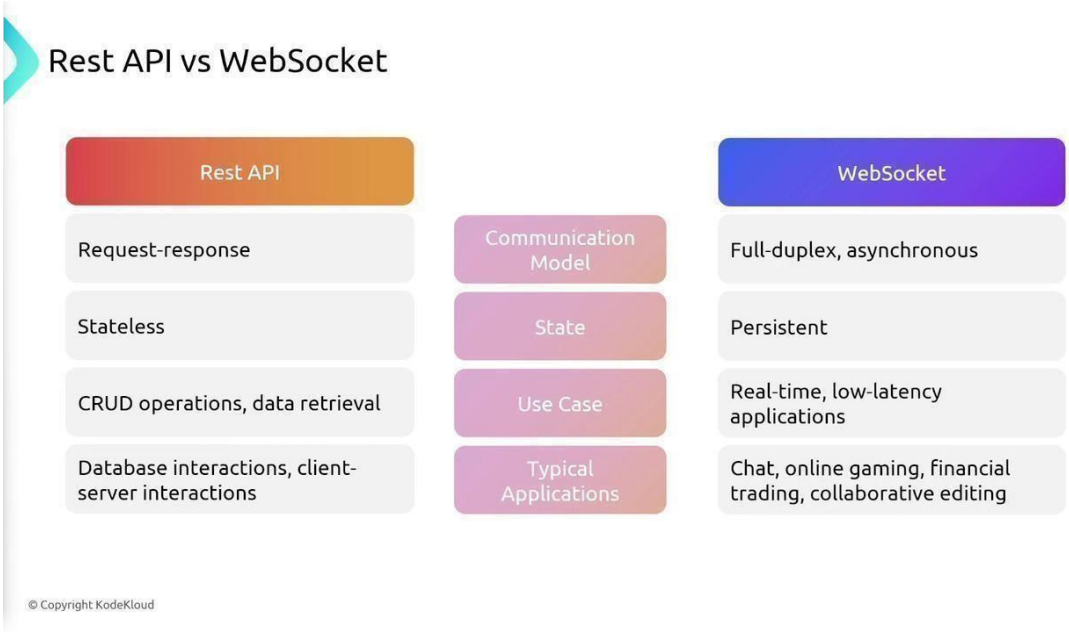

Рисунок 3.3 – Стаття «Websockets vs REST API» порівняння

У плані продуктивності WebSocket помітно кращий за REST, коли йдеться про потік даних у реальному часі. REST вимагає постійно надсилати запити, щоб отримувати оновлення, а WebSocket одразу передає їх, щойно щось змінюється.

Наприклад, якщо ціна на ринку різко змінюється за кілька секунд, WebSocket зможе надіслати кілька оновлень поспіль без затримок. А REST-клієнт у цей час ще може чекати наступного запиту – і в результаті пропустить важливий момент. Це критично, навіть якщо стратегія не є високочастотною. У криптовалютній торгівлі навіть одна секунда може вплинути на прибуток.

Через це фахівці радять: отримання котирувань і біржових подій краще реалізовувати через WebSocket, а REST залишити для допоміжних або одноразових дій, де швидкість не є критичною [14]. У темі продуктивності та надійності API варто пам’ятати, що навіть найкращі біржові інтерфейси мають свої обмеження і не завжди працюють ідеально [14].

REST API, біржі часто встановлюють обмеження на частоту запитів – наприклад, не більше 120 запитів на хвилину. Це називається rate limit. Якщо бот перевищує цю межу, сервер починає блокувати запити або відповідати з затримкою. Тобто, якщо бот працює тільки через REST, це серйозно обмежує його швидкість і реакцію на зміни на ринку.

WebSocket в цьому плані кращий, але і тут не все ідеально. Наприклад, біржа може обмежити кількість одночасних з’єднань або підписок на канали. Іноді з'єднання може обриватися, або біржа раптово перестає надсилати дані.

Варто також знати, що API криптобірж не завжди стабільні. Трапляються таймаути, помилки у відповідях, дані можуть бути некоректними або застарілими. Наприклад, були випадки, коли бот скасовує ордер через REST, але біржа ще кілька секунд після цього продовжує показувати його як активний [14]. Або на ки – ордер зник з сервера, але бот ще думає, що він там є. Це може призвести до помилок у логіці бота.

Саме тому хороший торговий агент повинен бути готовий до таких нестандартних ситуацій. Його варто проєктувати з урахуванням можливих збоїв. Наприклад: якщо запит не пройшов – спробувати ще раз, регулярно звіряти стан ордерів, мати механізм автоматичного перепідключення, якщо пропав зв’язок із біржею.

Надійний бот має залишатися працездатним навіть під час короткочасних проблем з інтернетом чи сервером біржі. І головне – після відновлення зв’язку він повинен швидко «оговтатися» і повернутися до коректного стану [14].

API – це як двері, через які бот заходить на біржу. Щоб ці двері були відкриті лише для «своїх», використовуються спеціальні ключі: публічний ID і секретний ключ. Вони прив’язані до акаунту конкретного користувача, і з їх допомогою бот отримує доступ до торгівлі, перегляду балансу або інших дій.

Тут є один важливий момент – безпека. Ці ключі фактично

дорівнюють логіну і паролю. Якщо хтось заволодіє секретним ключем, він може від імені користувача торгувати, а іноді – навіть вивести всі кошти із рахунку. Саме тому біржі впроваджують додаткові рівні захисту.

Наприклад, ключ можна створити з обмеженнями – лише перегляд або лише торгівля, без можливості виведення коштів. Також дозволяється вказати список IP-адрес, з яких дозволено надсилати запити (це whitelist)[15]. Крім того, секретний ключ показується тільки один раз – якщо його загубити, потрібно створювати новий[help.crypto.com].

Рекомендується зберігати ключі в зашифрованому вигляді, ніколи не передавати стороннім сервісам і регулярно змінювати. У 2025 році деякі біржі (як-от Coinbase) примусово оновили всі ключі та додали нові системи захисту й детальніші права доступу[coinbase.com].

Існує певний баланс між безпекою та продуктивністю. Наприклад, трейдери іноді запускають ботів на серверах поруч із біржею, щоб зменшити затримку сигналів. Але ці сервери стають «точкою ризику», бо на них зберігаються ключі – тому їх теж треба добре захищати.

WebSocket – це дуже швидкий канал, але він вимагає постійного підключення і в реальному часі обробляє великий обсяг даних. Якщо бот не встигає – він може пропустити важливі оновлення. Тому досвідчені розробники будують бота з подвійною системою: WebSocket обробляє все в реальному часі, а REST періодично перевіряє критичні речі – наприклад, чи справді ордер виконаний[14].

Щоб усе працювало надійно, бот має вести журнал подій (логування), фіксувати помилки, повторювати запити при збої, а в разі проблем – перемикатися на резервні канали.

Отже, API – це основа зв’язку між ботом і біржею. Від того, як грамотно все налаштовано, залежить і швидкість торгівлі, і безпека коштів. Найкращі рішення поєднують REST і WebSocket – і це дозволяє створити дійсно надійного торгового агента, здатного працювати без збоїв навіть на дуже динамічному ринку.

3.4 Моделі прийняття рішень і способи адаптації агента до ринкових змін

У світі криптовалют все змінюється блискавично. Сьогодні ринок росте, завтра – паніка й обвал. Саме тому «розумні» агенти, які автоматично торгують замість людини, повинні не просто реагувати на зміни, а й підлаштовуватися під них.

Як агент вирішує, що робити – купувати, продавати чи чекати? Для цього в нього є «мозок» – модель прийняття рішень. Є два основні підходи до таких моделей: прості – за чіткими правилами, і складніші – які вчаться з часом. У простому варіанті все побудовано на фіксованих правилах. Наприклад, агент купує монету, коли індикатор RSI опускається нижче 30. Це як у водія, який завжди зупиняється, побачивши червоне світло. Такі моделі не вчаться самі: якщо правила перестають працювати, їх змінює людина. Вона тестує нові налаштування на минулих даних, перевіряє, чи стало краще – і запускає оновленого бота. Це повільно, але передбачувано і без сюрпризів.

Інший варіант – моделі, що навчаються. Вони аналізують нові ринкові дані й самі змінюють стратегію. Наприклад, модель може брати дані за останні 7 днів і щоразу по-новому «перенавчатися», щоб краще розуміти теперішній стан ринку. Це називається «ковзне вікно». У ще розумніших агентів є здатність вчитись прямо під час торгівлі – наприклад, коли після прибуткової угоди агент «запам’ятовує», що саме спрацювало, і пробує повторити подібне.

Іноді стратегія адаптується не лише в деталях, а повністю змінюється залежно від ринку. Якщо ринок активно росте – краще «ловити хвилю» і тримати позиції довше. Якщо ціни стоять на місці – варто шукати короткі, обережні входи. Агент може визначати, в якому стані зараз ринок, і перемикатися між різними стратегіями. Це як водій, що в дощ їде обережніше, а на трасі – швидше.

Також агент може змінювати конкретні параметри, наприклад, цілі для прибутку. Якщо ціна скаче швидко – логічно ставити вищі цілі. Якщо ринок «тихий», цілі краще зменшити. Для цього використовуються індикатори волатильності – наприклад, ATR. Агент підлаштовується під темп ринку, як бігун, що регулює дихання.

Деякі моделі працюють у кілька етапів. Спочатку перевіряється, чи взагалі варто торгувати зараз – наприклад, коли на ринку панує хаос, агент може вирішити просто почекати. І тільки якщо «світло зелене», включається другий рівень – пошук конкретної точки входу.

Безпека – важлива частина цієї системи. Агент не має вкладати всі кошти в одну монету, навіть якщо вона «виглядає прибутковою». Якщо, скажімо, вже 50% бюджету вкладено в ризикований альткоїн, новий сигнал на купівлю може бути відхилено [14]. Це як нагадування: краще перестрахуватись, ніж втратити все за одну ніч. Окрема історія – надзвичайні ситуації. Наприклад, біржа перестала відповідати або сталася різка просадка в ціні. У таких випадках агент має «режим безпеки»: зупинити торгівлю, скасувати всі ордери, і дочекатися стабілізації. Це як аварійне гальмо у поїзді – краще втратити трохи, ніж усе.

Навіть розумні агенти не застраховані від помилок. Але вони здатні вчитися на них. Якщо стратегія призвела до збитків, агент аналізує це і змінює поведінку. Наче трейдер, який після кількох невдалих угод міняє підхід – тільки агент робить це автоматично.

Втім адаптація не гарантує успіху, якщо ринок змінюється кардинально. Наприклад, стратегія, навчена в умовах спаду, не зможе працювати в період бурхливого зростання – поки її не переналаштувати. Тому моделі слід регулярно перевіряти і оновлювати. Один із способів – стежити, наскільки нові дані відрізняються від попередніх. Якщо відмінності суттєві – час перенавчання [7].

У сучасних трейдинг-системах вже є інструменти, які автоматично слідкують за тим, чи все працює нормально. Якщо результати падають –

система подає сигнал або перемикається на резервну модель. У підсумку: хороша торгова модель має бути живою – тобто, змінюватись разом із ринком. Вона поєднує чіткі правила, здатність до навчання і постійну оцінку результатів. Так працює адаптивний штучний інтелект у фінансах: він вчиться, аналізує, коригує стратегію – і йде в ногу з ринком. За даними досліджень, використання ML дає змогу створювати стратегії, які розвиваються разом із ситуацією [bookmap.com]. А практика показує: регулярний перегляд налаштувань (наприклад, щомісяця) помітно підвищує ефективність [kriptomat.io].

Адаптація – це не одноразова дія, а постійний процес. Ринок не стоїть на місці – і трейдинг-агент також має постійно оновлюватись, щоб не залишитись позаду.

3.5 Етичні та регуляторні аспекти використання ШІ в алгоритмічній торгівлі

Зі зростанням впливу штучного інтелекту на фінансові ринки виникають не тільки нові можливості, а й досить серйозні виклики. Особливо – коли мова йде про етику та контроль над тим, як працюють ці розумні алгоритми.

Сучасні трейдингові боти можуть приймати рішення так швидко і складно, що людина просто не встигає зрозуміти, що відбувається. З одного боку – це добре: можна обійти ринок, заробити, реагувати на зміни за секунди. Але з іншого – виникає ризик втрати контролю. Бо часто навіть ті, хто створив ці моделі, не можуть точно пояснити, чому бот зробив саме таку угоду. Особливо, якщо модель побудована на нейромережах – це, по суті, чорна скринька. Вона дає результат, але як саме – велике питання.

І тут є проблема: «модельним ризик». Наприклад: алгоритмічний фонд втрачає мільйони, або ще гірше – спричиняє збій на біржі. А ніхто не може сказати точно, чому це сталося. Навіть регулятори на кшталт Банку

міжнародних розрахунків (BIS) вже попереджають: чим більше ШІ у фінансах, тим вищі ризики [10].

Через це регулятори починають тиснути: компанії повинні пояснювати, як працюють їх алгоритми, особливо коли щось іде не так. Ідея проста – хочеш працювати з грошима, маєш бути прозорим і підзвітним.

Ще одне важливе питання – справедливість на ринку. Один учасник має суперпотужного ШІ, який за мілісекунди сканує ринок і здійснює угоди. А поруч звичайний трейдер з ноутбуком і стаканом кави. Важко назвати це чесною боротьбою. І якщо таких «суперботів» багато, маленькі учасники просто втрачають шанс.

Та більше – деякі учасники йдуть ще далі й програмують свої боти на маніпуляції. Наприклад, техніка spoofing: бот виставляє багато фейкових ордерів, щоб створити ілюзію попиту, а потім швидко їх знімає. Або layering – імітація активності, щоб інші трейдери «клюнули». Такі прийоми вже заборонені на традиційних біржах, і все частіше постає питання, як це контролювати на крипторинку. Саме тому регулятори посилюють нагляд і готують нові правила. Вони вимагатимуть, щоб фірми розкривали, як саме працюють їх алгоритми: не до дрібниць, але хоча б основні принципи [10]. Також буде більше уваги до того, чи не маніпулюють алгоритми ринком – і якщо так, їх можуть жорстко обмежити.

Цікаво, що основна ідея тут – зменшити інформаційну асиметрію.

Тобто, зробити ринок трохи справедливішим. Якщо всі учасники знатимуть, що великі компанії не мають таємних переваг і зобов'язані пояснювати свої дії – це зміцнює довіру. А довіра – фундамент будь- якого фінансового ринку.

Штучний інтелект дедалі впевненіше заходить у фінансовий світ, і якщо раніше це було щось нове і дивне, то зараз багато трейдерів сприймають торгових ботів як буденний інструмент. Але разом із цим приходить цілий спектр ризиків, про які не можна мовчати. Один із головних – це системна нестабільність. Сотні чи тисячі алгоритмів

використовують схожі торгові стратегії. Наприклад, реагують на зміну тренду однаково – різко починають продавати актив. І тут виникає ефект «доміно»: всі боти діють одночасно, спричиняючи ще глибше падіння, ніж було б без них. Це вже не просто торгівля, а неконтрольований обвал. Саме так сталося у 2010 році під час відомого флеш-краху: індекси на фондовому ринку США впали на майже 9% за декілька хвилин – і лише тому, що алгоритми почали «панікувати» одночасно.

У світі криптовалют подібні ситуації ще ймовірніші. Тут ринки менш регульовані, волатильність більша, і кількість високочастотних ботів – чимала. ШІ-агенти, що діють на мікросекундних інтервалах, можуть не лише реагувати на рухи ринку, а й самі ці рухи провокувати. Саме тому багато бірж уже зараз впроваджують автоматичні захисні механізми, які тимчасово зупиняють торгівлю, якщо ціна активу змінюється надто стрімко. Це щось на кшталт «аварійного стопу» – краще зупинити все, ніж дозволити ринку зірватися у прірву.

Але питання не лише в технічних збоях. Варто задуматись: хто взагалі відповідає за дії таких ботів? Якщо алгоритм зробив неправильну операцію, спричинив втрати клієнтам або збій у системі, то на кого вказати пальцем? Закон наразі не вважає ШІ юридичною особою, тому відповідальність лягає або на компанію, яка його використовує, або на розробників. Це породжує етичний обов’язок: контролювати алгоритми до запуску, перевіряти, як вони поводяться в нестандартних умовах, обмежувати їхні дії, щоб запобігти ризикованим сценаріям.

Ще один неочевидний, але важливий аспект – це використання даних. Багато сучасних ШІ-моделей працюють не тільки з цінами та обсягами, а й з альтернативними джерелами: соцмережами, трендами пошуку, форумами. Бот може аналізувати десятки тисяч твітів, щоб спрогнозувати, як поведе себе ринок. Але де проходить межа між ефективною аналітикою і втручанням у приватність? Якщо ШІ «зчитує» поведінку мільйонів користувачів, навіть без їхньої згоди, виникають

серйозні питання до етики. Крім того, є ще тонша межа – використання інформації, яка може виявитися інсайдерською. Якщо алгоритм завдяки аналізу відкритих, але складно пов'язаних даних здогадується про майбутні дії великого гравця, це вже схоже на порушення правил. Звичайна людина не змогла б таке вирахувати – ШІ міг. Але тоді що це: легальний аналіз чи непрямий інсайд?

На рівні регулювання зараз відбувається багато змін. У Європі вже ухвалили регламент MiCA, який регулює крипторинки, зосереджуючись на прозорості та захисті інвесторів. Але питання алгоритмічної торгівлі ще потребує додаткових норм. У країнах Азії – наприклад, у Гонконгу та Сінгапурі – ситуація серйозніша: там постачальники автоматизованих стратегій повинні мати ліцензії, дотримуватись правил ризик-менеджменту і прозоро звітувати про свої системи. У Гонконгу, зокрема, заборонено надавати ботові сервіси роздрібним клієнтам без спецдозволу [16].

У США фінансові регулятори, зокрема Комісія з цінних паперів і бірж (SEC) та Комісія з торгівлі товарними ф'ючерсами (CFTC), дедалі активніше долучаються до процесу регулювання інновацій у сфері фінансів. Зокрема, у фокусі уваги опинилися алгоритмічні торгові системи на базі штучного інтелекту, які в останні роки стали потужним інструментом для хедж-фондів, інвестиційних банків і роздрібних трейдерів. Регулятори зобов'язують великі гравці не лише офіційно реєструвати свої алгоритми, але й детально розкривати їхню логіку, архітектуру прийняття рішень та внутрішні механізми контролю. Такі кроки покликані забезпечити прозорість функціонування систем, запобігти маніпуляціям та гарантувати чесні умови для всіх учасників ринку.

Водночас велика увага приділяється етичному аспекту використання ШІ. У провідних фінансових організаціях створюються комітети з етики штучного інтелекту – це міждисциплінарні групи, що аналізують потенційні упередження у моделях, вивчають ризики створення несправедливих переваг та розробляють етичні регламенти для застосування ШІ в

інвестиційній діяльності. Основними принципами стають справедливість, прозорість, надійність, відповідальність і кібербезпека.

Справедливість означає, що алгоритми не повинні створювати дискримінаційні умови чи порушувати права окремих груп учасників. Прозорість передбачає хоча б часткову можливість інтерпретації дій моделі – як вона прийшла до того чи іншого рішення. Надійність – це стабільність результатів навіть у кризових або нестандартних ринкових умовах. Відповідальність – чітке розуміння, хто відповідає за роботу ШІ в конкретній ситуації, зокрема в разі збою чи фінансових втрат. Кібербезпека ж має критичне значення, особливо в умовах криптовалютного ринку, де злам одного торгового бота може спричинити серйозні фінансові збитки не лише для окремого інвестора, а й для всієї біржі чи сегмента.

Окремо варто відзначити, що активне втручання державних органів – не лише обмеження, але й ознака визнання технології. ШІ поступово перестає бути інструментом ентузіастів і стартапів і стає частиною офіційної фінансової інфраструктури. Це відкриває нові можливості для його масштабного впровадження, водночас вимагаючи суворого дотримання нових правил гри.

У підсумку, використання ШІ в торгівлі – це не лише прорив і технологічна перевага, а й нові виклики. Щоб уникнути зловживань і втрати довіри до ринку, потрібно забезпечити тристоронній діалог між трейдерами, розробниками і регуляторами. Саме спільне напрацювання стандартів, механізмів перевірки та етичних обмежень дозволить технологіям розвиватися безпечно й відповідально, перетворивши ШІ не на загрозу, а на надійну основу прозорої та стабільної фінансової системи.

4 РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОГО АГЕНТА

4.1 Огляд проєкту та його структура

Проєкт зосереджений на розробці інтелектуального торгового агента, який аналізує історичні ринкові дані, відстежує, як працювали різні стратегії, і вибирає найстійкішу з них на основі фактичних результатів. Йдеться не про прогнозування майбутніх рухів цін, а про холодний розрахунок – що з минулого показало стабільний плюс, повний код програми подано у додатку А.

Основний компонент – скрипт trading_agent.py, який працює як ядро всієї системи. Він читає CSV або Parquet-файли, перевіряє наявність потрібних колонок (наприклад, profit, entry_time, exit_price), підраховує ефективність кожної стратегії, будує криву прибутку, обмежує занадто високі прирости (щоб уникнути ілюзій «ідеального ринку») і знаходить найрезультативнішу угоду. Усі стратегії представлені у вигляді набору ознак і зберігаються в датасеті. Вони використовуються для навчання окремого модуля (train_strategy_selector.py), який потім приймає рішення.

Реалізовано кілька варіантів: Hybrid Fakey RSI – поєднує патерн fakey з RSI, щоб відсіяти випадкові входи, MACD-RSI – сигнал формується, коли MACD перетинає нуль, а RSI підтверджує напрямок, Strategy Selector – модель, що навчається на історії трейдів і вирішує, яка стратегія найкраще підходить під конкретну ринкову ситуацію. Цей агент не «вгадує», що буде завтра, а перевіряє, яка поведінка давала результат у схожих умовах. Це дозволяє будувати рівну криву прибутку, без стрибків і хаосу. Використання rule-based патернів у поєднанні зі статистичним аналізом і машинним навчанням робить систему універсальним інструментом для тестування і підбору стратегій. По суті, це симулятор-аналітик, що добре підходить для фінансових досліджень, бектестингу та розуміння, які підходи працюють у різних ринкових фазах.

4.2 Джерела даних

Для роботи торгового агента критично важливо мати якісні історичні дані. Вони використовуються на всіх етапах: від попереднього аналізу до моделювання рішень і симуляції торгів. У цьому проєкті беруться стандартні часові ряди у форматі OHLCV – відкриття, максимум, мінімум, закриття та обсяг – окремо для кожного таймфрейму.

Джерела даних поділено на дві категорії: зовнішні – це інформація, отримана з біржі (в основному Binance), і внутрішні – ті, що формуються в процесі роботи системи (наприклад, результат логування чи розрахунків).

Дані з Binance підтягуються через API за допомогою ccxt. Це відбувається через скрипт fetch_data/get_candles.py. Він надсилає запити до сервера біржі, отримує котирування й одразу приводить усе до формату DataFrame. Щоб не витрачати час на повторну обробку, файли зберігаються у форматі .parquet – це дозволяє швидко читати великі обсяги без втрат продуктивності. Скрипт працює завдяки файлу config.py. Він виконує роль централізованого сховища налаштувань, які використовуються модулями взаємодії з API (як напряму через requests, так і через ccxt). Це один із найважливіших елементів безпеки і конфігурації у системі (лістинги 4.1–4.2).

Лістинг 4.1 – Типовий вміст config.py

```
BINANCE_API_KEY = "..." BINANCE_SECRET_KEY = "..."
```

Лістинг 4.2 – fetch_data_legacy.py підключення

```
from config import BINANCE_API_KEY, BINANCE_SECRET_KEY
BINANCE = ccxt.binance({ 'apiKey': BINANCE_API_KEY,
'secret': BINANCE_SECRET_KEY,
'enableRateLimit': True,
})
```

У проєкті використовуються два окремі модулі для завантаження історичних даних із Binance. Вони не дублюють один одного, а реалізують різні технічні підходи – це зроблено з урахуванням того, що кожна стратегія має свої вимоги до формату й обсягу вхідних даних.

Перший модуль – fetch_data/get_candles.py. Він підтягує дані через публічний REST API Binance (endpoint /api/v3/klines) без авторизації. Працює на основі бібліотеки requests. Заточений під прості запити, дозволяє вручну вказати дату початку й завершення, швидко витягує потрібні таймфрейми (наприклад, 5-хвилинні або годинні свічки) й формує компактну структуру JSON, яку легко обробити.

Цей підхід добре підходить для стратегії fakey_hybrid.py, де потрібно точно контролювати часові межі запиту й не витрачати зайві ресурси на обробку надмірно складної відповіді API (лістинг 4.3).

Лістинг 4.3 – get_candles.py реалізує завантаження OHLCV-даних із REST API Binance

```
    Response
=
    requests.get(f"{BINANCE_BASE_URL}/api/v3/klines",
params=params)
    if response.status_code != 200:
    raise                  Exception(f"     Помилка
запита:
    {response.text}")
    data = response.json() if not data:
    break
    'volume',
    df = pd.DataFrame(data, columns=[
    'timestamp', 'open', 'high', 'low', 'close',
    'close_time', 'quote_asset_volume', 'number_of_trades',
    'taker_buy_base_volume', 'taker_buy_quote_volume', 'ignore'
    ])
```

Другий модуль – fetch_data_legacy.py – використовує бібліотеку ccxt. Цей варіант більш універсальний: можна працювати як без авторизації, так і з API-ключами, якщо потрібно більше доступу чи стабільності.

Він підходить для завантаження великих обсягів історичних даних – робота організована по чанках (наприклад, по 1000 свічок за раз), і якщо потрібно, автоматично підтягує наступні дані за timestamp. Такий підхід використовується у стратегії rsi_macd.py, яка потребує довгого діапазону історії для аналізу середньострокових патернів.

Цей варіант зручний тоді, коли REST API обмежує обсяг або частоту запитів, а також коли важливо витягти повну картину ринку без пропусків (лістинг 4.4).

Лістинг 4.4 – fetch_data_legacy.py

```
ohlcv = BINANCE.fetch_ohlcv(symbol, timeframe=timeframe, since=fetch_since, limit=limit_per_call)
```

4.3 Структура збереження

У проєкті реалізовано логічну структуру зберігання даних, яка дозволяє ефективно розділяти вхідну, проміжну та вихідну інформацію. Це спрощує навігацію по проєкту, полегшує повторне використання даних та запобігає помилкам під час одночасного виконання різних частин системи. Базовий каталог «data/» слугує головним сховищем історичних котирувань, які зберігаються у форматі .parquet. Ці файли містять стандартні OHLCV-дані (open, high, low, close, volume) по парі BTC/USDT. Дані представлені з різною частотою: наприклад, BTC_USDT_1h.parquet містить годинні свічки, а BTC_USDT_5m.parquet – п’ятихвилинні. Усі торгові стратегії, моделі машинного навчання й сам агент використовують ці файли як основне джерело.

Директорія ml_data/ відповідає за зберігання результатів машинного

навчання. Тут міститься файл моделі (strategy_selector.pkl), яка використовується агентом для класифікації ринкових ситуацій та вибору найвідповіднішої стратегії. Також у цій папці знаходиться тренувальна вибірка (training_dataset.csv), де зібрані ознаки (features) і цільові мітки для навчання. Ці файли є критично важливими для запуску інтелектуальної частини системи.

У підкаталозі agent/data/ зберігаються робочі копії тих самих OHLCV-даних, але вже у вигляді окремих підмножин, які використовуються в конкретний момент роботи агента. Наприклад, якщо агент працює з 1-годинним таймфреймом, то він бере копію BTCUSDT_1h.parquet саме з цього каталогу. Це дозволяє зменшити навантаження й уникнути конфліктів при паралельному використанні файлів.

Остання ключова директорія – logs/. Вона містить результати роботи агента та тестування стратегій. Сюди записуються таблиці з equity-кривою (*_equity.csv), лог угод (*_trades.csv) та графіки у форматі PNG, які візуально відображають динаміку прибутку. Завдяки чіткому розмежуванню між сирими, обробленими й вихідними даними, проєкт зберігає прозорість і гнучкість у розробці, тестуванні та подальшому масштабуванні.

4.4 Fakey Hybrid: Розробка стратегії хибного прориву

Стратегія Fakey Hybrid реалізована як окремий модуль «fakey_hybrid.py», який виконує повний цикл – від завантаження даних до генерації сигналів. Основна мета – створити автономний скрипт з гнучкими параметрами, придатний до адаптації під різні ринкові сценарії.

На старті модуль підтягує історичні котирування з файлів формату «.parquet», що лежать удиректорії «data/». Таймфрейм (наприклад, «1h» або «5m») задається через змінні. Використовується pandas, що дозволяє швидко обробити обсяг даних та провести валідацію.

Далі підключається модуль «indicators.py», в якому винесені функції розрахунку технічних індикаторів – EMA, RSI, MACD, ATR та обсяг. Це дозволяє централізовано змінювати логіку обчислень і використовувати її в інших частинах проєкту. Після попередньої очистки даних виконується логіка виявлення патерну Fakey. Окрім стандартних умов (inside bar і хибний пробій), враховується сила імпульсу (RSI), напрям MACD Histogram та перевищення обсягу над ковзним середнім. Кожна свічка перевіряється на відповідність усім умовам, і при збігу формується сигнал на вхід. Результати зберігаються у вигляді таблиці, яка може подаватись агенту або використовуватись у бектестингу. Порогові значення задаються як змінні у верхній частині скрипта, що дозволяє легко налаштовувати модель без зміни основного коду. На виході створюється DataFrame із сигналами та, за потреби, графіки, які зберігаються в «logs/». У підсумку «fakey_hybrid.py» – це гнучкий модуль, який поєднує патерни з теханалізом і готовий до інтеграції в торгову систему.

Файл fakey_hybrid.py реалізує повний цикл роботи торгової стратегії: від завантаження даних до генерації сигналів і оцінки результату. Він побудований за принципами модульності – кожен логічний етап оформлено окремою функцією (лістинги 4.5–4.6).

Лістинг 4.5 – Завантаження історичних ринкових даних

```
def load_data():
df =
pd.read_parquet(os.path.join(os.path.dirname(_file_), "..",
"dat, "BTC_USDT_1h.parquet"))
df.index = pd.to_datetime(df.index) return df
```

Лістинг 4.6 – Генерація технічних індикаторів

```
def apply_indicators(df):
df['ema50'] = ta.trend.ema_indicator(df['close'], 50)
df['ema200'] = ta.trend.ema_indicator(df['close'],200)
```

Продовження лістингу 4.6

```
    df['rsi'] =    ta.momentum.RSIIndicator(df['close'],
14).rsi()
    df['cci']      =    ta.trend.cci(df['high'], df['low'],
    df['close'],                  20)
    df['adx']      =    ta.trend.adx(df['high'], df['low'],
    df['close'],                  14)
    df['atr'] =  ta.volatility.average_true_range(df['high'],
    df['low'],          df['close'],   14)
    df['macd_hist']=    ta.trend.macd_diff(df['close'])
    df['volume_ma']=    df['volume'].rolling(20).mean()
    return df
```

Далі є блок, що відповідає за логіку свічкового патерну Fakey. Він реалізує два ключові етапи: Inside Bar – свічка знаходиться повністю в межах попередньої, False Breakout – поточна свічка пробиває верхню межу Inside Bar, але закривається нижче (лістинг 4.7).

Лістинг 4.7 – Виявлення патерну Fakey (Inside Bar + False Breakout)

```
    def detect_fakey(df):
    inside        =   (df['high']   <   df['high'].shift(1))
&
    (df['low'] > df['low'].shift(1))
    fakeout       =   (df['high']   >   df['high'].shift(2))
&
    (df['close'] < df['high'].shift(1)) return inside.shift(1)
& fakeout
```

Інший блок відповідає за об’єднання сигналів свічкового аналізу (Fakey) з технічними індикаторами. На основі цього формуються entry-сигнали. Вхід дозволяється лише за наявності тренду: EMA50 > EMA200 (long) або навпаки; Імпульс фільтрується: RSI > 55 (або < 45) + MACD > 0 (лістинг 4.8).

Лістинг 4.8 – Формування торгових сигналів (entry/exit logic)

```
def generate_signals(df):
df = apply_indicators(df) fakey = detect_fakey(df)
atr_median = df['atr'].rolling(100).median()
raw_long = (
fakey & (df['ema50'] > df['ema200']) & (df['volume'] >
df['volume_ma']) & (df['atr'] >
atr_median)
)
raw_short = (
fakey & (df['ema50'] < df['ema200']) & (df['volume'] >
df['volume_ma']) & (df['atr'] >
atr_median)
)
ntry_long = raw_long & (df['rsi'] > 55)
& (df['macd_hist'] > 0)
entry_short = raw_short & (df['rsi'] < 45) &
(df['macd_hist'] < 0)
exit_long = df['rsi'] < 50 exit_short = df['rsi'] > 50
return entry_long, exit_long, entry_short, exit_short,
df
```

Функція run_backtest(df) реалізує простий бектест без сторонніх бібліотек. Тут криється вся логіка симуляції входу та виходу з ринку. На основі entry_long або entry_short ми відкривається позиція. Stop Loss (SL) та Take Profit (TP) розраховуються динамічно з використанням ATR (лістинг 4.9). Фіксується результат угоди (win/loss) і оновлюється крива капіталу (equity).

Лістинг 4.9 – Бектест: симуляція трейдів за сигналами Fakey

```
def run_backtest(df, risk=0.01, sl_coef=1.5, tp_coef=2.5):
balance = 10000
equity_curve = []
for i in range(2, len(df)): row = df.iloc[i]
```

Продовження лістингу 4.9

```
    if row['entry_long']:
    sl = row['close'] - sl_coef * row['atr'] tp = row['close'] +
tp_coef * row['atr']
    outcome = simulate_trade(row['close'], sl, tp) balance *= (1
+ outcome * risk)
    elif row['entry_short']:
    sl = row['close'] + sl_coef * row['atr'] tp = row['close'] -
    tp_coef * row['atr']
    outcome = simulate_trade(row['close'], sl, tp) balance *= (1
+ outcome * risk)
    equity_curve.append(balance)
    df['equity'] = equity_curve return df
```

У складі модуля «fakey_hybrid.py» реалізовано функцію «run_test()», яка виконує повну симуляцію стратегії в автономному режимі. Вона активується автоматично при запуску скрипта напряму як головного файлу (через перевірку if ___name_== "_main_"). Основне призначення цієї функції – протестувати поведінку стратегії на історичних даних без участі торгового агента. У складі модуля «fakey_hybrid.py» реалізовано функцію «run_test()», яка виконує повну симуляцію стратегії в автономному режимі. Вона активується автоматично при запуску скрипта напряму як головного файлу (через перевірку if __name___== "__main__"). Основне призначення цієї функції – протестувати поведінку стратегії на історичних даних без участі торгового агента (лістинги 4.10, 4.11).

Лістинг 4.10 – Тестовий запуск стратегії

```
    run_test():
    df = load_data()
    entry_l,        exit_l,   entry_s,   exit_s,   df_ready
= generate_signals(df)
    df_bt        =   run_backtest(df_ready,   entry_l,
exit_l, entry_s, exit_s)
```

Продовження лістингу 4.10

```
df_bt[['equity']].plot(title="Equity Curve – Fakey Hybrid H1 (v2, Live Impulse Confirm)", figsize=(10, 5))
plt.grid(True) plt.savefig(os.path.join(BASE_LOG_PATH, "equity_curve_h1_v2.png"))
plt.show()
df_bt.to_csv(os.path.join(BASE_LOG_PATH, "fakey_hybrid_h1_v2_equity.csv"))
```

Лістинг 4.11 – Консольний фінансовий звіт

```
rint("\n [ЗВІТ ПРО СТРАТЕГІЮ FAKey Hybrid H1 v2]") print(f" Період: {start.strftime('%Y-%m-%d')} {end.strftime('%Y-%m-%d')} ({duration_days} днів)")
print(f" Початковий баланс: {initial_equity:.2f} USDT") print(f" Фінальний баланс: {final_equity:.2f} USDT") print(f" Сукупний прибуток: +{gain*100:.2f}%")
```

Стратегія fakey_hybrid.py реалізує ключову торгову логіку, однак її виконання неможливе без взаємодії з низкою допоміжних модулів. Це забезпечує модульність, масштабованість та повторне використання компонентів системи. Нижче наведено опис основних залежностей, що забезпечують коректну роботу цієї стратегії.

indicators.py – модуль, відповідальний за розрахунок технічних індикаторів, які використовуються для фільтрації сигналів. У ньому реалізовано або обгорнуто логіку для EMA, RSI, MACD та ATR. Це дозволяє уніфікувати обчислення індикаторів для різних стратегій і зберігати чистоту основного коду (лістинг 4.12).

Лістинг 4.12 – Підключення індикаторів із indicators.py

```
from indicators import calculate_indicators
 Після завантаження даних df = load_data()
df = calculate_indicators(df)
```

backtest_simple.py – скрипт, що виконує симуляцію торгівлі на історичних даних. Він імпортує основну функцію generate_signals() зі стратегії fakey_hybrid.py, виконує бектест, зберігає результати у вигляді кривої капіталу та таблиці угод. Це дозволяє дослідити ефективність стратегії без складних ML-алгоритмів (лістинг 4.13).

Лістиннг 4.13 – Виклик стратегії з backtest_simple.py

```
from fakey_hybrid import generate_signals
df = load_data()
df = generate_signals(df) results = run_backtest(df)
results.to_csv("logs/fakey_hybrid_trades.csv")
```

Файли даних у data/ – історичні ринкові дані зберігаються у форматі .parquet, зокрема файл BTC_USDT_1h.parquet. Ці дані завантажуються функцією load_data() і є вхідними для побудови індикаторів та сигналів (лістинг 4.14).

Лістинг 4.14 – Завантаження історичних даних

```
import pandas as pd import os
def load_data():
path = os.path.join(os.path.dirname(_file_), "..", "data",
"BTC_USDT_1h.parquet")
df = pd.read_parquet(path) df.dropna(inplace=True) return df
```

4.5 RSI + MACD: Комбінований імпульсний підхід

Стратегія RSI + MACD у межах даного проєкту реалізована як окремий модуль «rsi_macd.py» і являє собою імпульсну торгову систему середньої складності, яка базується на комбінації двох класичних технічних індикаторів. Вона може працювати як самостійно, так і у складі агентної системи, де використовується як одна з альтернативних моделей поведінки

в умовах змінного ринку. Мета стратегії – визначити найбільш вірогідні точки входу в позицію, коли ринок демонструє явні ознаки імпульсного руху, підтверджені як фазовими, так і осциляторними сигналами.

Основу логіки складає поєднання RSI (Relative Strength Index) і MACD (Moving Average Convergence Divergence), що дозволяє враховувати як поточну силу імпульсу, так і підтвердження фазової динаміки тренду. Індикатор RSI оцінює швидкість зміни ціни та її силу у напрямку останнього руху. Значення RSI, що перевищує 70, сигналізує про зону перекупленості, а значення нижче 30 – про зону перепроданості. Це дає змогу визначити ситуації, де ринок потенційно близький до розвороту або тимчасового відкату.

У свою чергу, MACD дозволяє оцінити фазовий стан ринку – тобто, чи є на ринку тренд, і наскільки він виражений. Індикація генерується на основі різниці між короткою та довгою експоненційними ковзними середніми, що дозволяє виявляти зміни в динаміці попиту й пропозиції. Коли MACD перетинає нульову лінію знизу вгору, це вважається сигналом на купівлю, якщо зверху вниз – сигналом на продаж.

Файл main.py виконує роль диспетчера: він передає дані, обирає стратегію (наприклад, rsi_macd) та викликає функції генерації сигналів. У деяких сценаріях також може запускатися навчання або підключення до ML-агента. Основна функція стратегії strategy_rsi_macd().Вона приймає два DataFrame – із даними 5-хвилинного та 1-годинного таймфреймів. (лістинг 4.15).

Лістинг 4.15 – Основна функція стратегії strategy_rsi_macd().

```
def strategy_rsi_macd(df_5m, df_h1):
close_5m = df_5m['close'] rsi14_5m = rsi(close_5m, 14)
macd_line, signal_line, _ = macd(close_5m, 12, 26, 9)
trend = ema(df_h1['close'], 5) > ema(df_h1['close'], 13)
trend_5m =
trend.reindex(df_5m.index,
```

Продовження лістингу 4.15

```
method='ffill').fillna(False)
cond_long = (rsi14_5m < 32) & (macd_line > signal_line) & trend_5m
cond_short = (rsi14_5m > 68) & (macd_line < signal_line) & (~trend_5m)
...
```

На основі умов «cond_long» та «cond_short» створюються сигнали входу в позицію – «entry_long» та «entry_short». Вони формуються лише при одночасному виконанні ключових критеріїв: сигналу від MACD, відповідного рівня RSI та напрямку тренду за EMA.

Вихід із позиції визначається досягненням фіксованого рівня RSI, найчастіше – позначки 50, що сигналізує про ослаблення імпульсу. Додатково можуть застосовуватись рівні Stop Loss і Take Profit, розраховані на основі ATR, що враховує поточну волатильність і дозволяє краще контролювати ризики.

Лістинг 4.16 – Генерація торгових сигналів

```
def generate_signals(df):
df_5m = df
df_h1 = getattr(df, "df_h1", None)
if df_h1 is None:
raise ValueError("`DataFrame` має містити атрибут
`df.df_h1` – **дані старшого таймфрейму.")
entry, exit, _, _ = strategy_rsi_macd(df_5m, df_h1) return entry, exit, pd.Series(False, index=df.index),
pd.Series(False, index=df.index) def generate_signals(df):
df_5m = df
df_h1 = getattr(df, "df_h1", None) if df_h1 is None:
raise ValueError("`DataFrame` має містити атрибут
`df.df_h1` – **дані старшого таймфрейму.")
```

Продовження лістингу 4.16

```
    entry, exit, _, _ = strategy_rsi_macd(df_5m, df_h1) return
entry, exit, pd.Series(False, index=df.index),
    pd.Series(False, index=df.index)
```

Модуль «main.py» виконує роль центрального координатора, який поєднує логіку окремих компонентів системи. У його завдання входить завантаження необхідних даних, виклик функції «strategy_rsi_macd» для формування сигналів, а також подальше збереження або візуалізація результатів. Таким чином забезпечується повноцінна інтеграція стратегії з іншими модулями, включно з бектестером, індикаторною базою та системою логування (лістинг 4.17).

Лістинг 4.17 – Підключення стратегії у головному скрипті

```
    from rsi_macd import strategy_rsi_macd
    from fetch_data_legacy import fetch_binance_archive
    df_5m, df_1h = fetch_binance_archive("BTC/USDT", "5m",
"1h", limit=2000)
    df_signals = strategy_rsi_macd(df_5m, df_1h)
df_signals.to_csv("logs/signals_rsi_macd.csv")
```

Забезбечення доступу до локального архіву котирувань Binance і використовується переважно в офлайн-режимі надає fetch_data_legacy.py. Його головна мета – надати можливість тестування стратегій або аналізу ринку без необхідності звертатися до зовнішнього API, що особливо зручно під час розробки, налагодження чи роботи в умовах обмеженого інтернет-з’єднання (лістинг 4.18).

Лістинг 4.18 – Отримання історичних даних (архів)

```
    def fetch_binance_archive(symbol, tf1, tf2, limit=1000):
df_5m = pd.read_parquet(f"data/BTCUSDT_5m.parquet").tail(limit)
```

Продовження лістингу 4.18

```
    df_1h = pd.read_parquet
(f"data/BTCUSDT_1h.parquet").tail(limit) return df_5m, df_1h
```

Модуль backtest.py використовує потужну бібліотеку VectorBT, яка дозволяє проводити комплексний бектест, враховуючи розмір позиції, комісії, прослизання та точну модель входу/виходу. У системі реалізовано два окремих механізми бектестингу: backtest.py – розширений бектест із використанням бібліотеки VectorBT, орієнтований на реалістичну симуляцію,він використовується для стратегії rsi_macd. backtest_simple.py підходить для патернових стратегій, таких як fakey_hybrid.py, де важлива прозора логіка входу/виходу і достатньо базової моделі прибутку (лістинги 4.19, 4.20).

Лістинг 4.19 –Розрахунок розміру позиції (position sizing)

```
    for i in df.index[entry_signal]: entry_price = close.loc[i]
sl_price = stop_price.loc[i] if sl_price == 0:
    size.loc[i] = 0.0 continue
    risk_per_unit = abs(entry_price - sl_price)
position_size = (init_cash * risk_per_trade)/
    risk_per_unit
    size.loc[i] = position_size
```

Лістинг 4.20 – Побудова портфелю з VectorBT

```
    pf = vbt.Portfolio.from_signals( close=close,
entries=entry_signal, exits=exit_signal, size=size,
init_cash=init_cash, freq=df.index.inferred_freq,
slippage=slippage, fees=fee,
    direction='both'
    )
```

У структурі проєкту модулі «main.py», «rsi_macd.py» та «backtest.py» працюють як єдиний аналітичний ланцюг, що реалізує повний цикл

тестування торгової стратегії. Файл «main.py» виступає в ролі керуючого вузла: саме тут відбувається завантаження історичних ринкових даних за допомогою функції «get_candles()». Після цього викликається функція «strategy_rsi_macd()» з модуля «rsi_macd.py», яка аналізує ціни, обчислює індикатори, будує сигнали на вхід та вихід з позицій, а також визначає рівні стопів і цілей на основі ATR.

Сформовані сигнали передаються у функцію «run_backtest()» з файлу «backtest.py». Цей модуль виконує симуляцію угод із врахуванням базового ризик-менеджменту. У результаті формується детальна оцінка ефективності стратегії: розраховуються ключові метрики, зокрема Annual Return, Sharpe Ratio та Profit Factor. Такий розподіл функцій дозволяє легко масштабувати систему, а також замінювати або модифікувати окремі блоки пайплайна без порушення загальної логіки.

4.6 Програмнат реалізація інтелектуального агента

У рамках проєкту реалізовано інтелектуального торгового агента, який виконує функцію аналітичного посередника між ринковими даними та конкретними торговими стратегіями. Його головне завдання – обирати найбільш доцільну поведінкову модель (стратегію) на основі історичних патернів, оцінених за реальними прибутковими угодами. Розробка побудована на практичних інструментах Python і машинного навчання, при цьому акцент зроблено не на «автоматичному трейдингу», а на післяфактум-аналізі як основі для стратегічного вибору.

Файл build_training_dataset.py реалізує перший етап у побудові інтелектуального торгового агента – генерацію історичного датасету з торговими угодами, на основі якого відбуватиметься навчання моделі класифікації.

Основна мета сформувати приклади (рішення) про доцільність застосування тієї чи іншої стратегії (RSI+MACD або Fakey), виходячи з

факту досягнення певного порогового прибутку (TP ≥ 6%) після сигналу на вхід. Усі дані збираються у форматі .csv, що містить як цінові дані, так і технічні фічі (лістинг 4.21).

Лістинг 4.21 – Імпорт бібліотек і завантаження історичних свічок

```
from fetch_data.get_candles import get_candles
start_time = datetime.now(timezone.utc) - timedelta(days=3
* 365)
df_5m = get_candles("BTCUSDT", "5m", start_time=start_time)
df_h1 = get_candles("BTCUSDT", "1h", start_time=start_time)
df_5m.df_h1 = df_h1 # додано старший ТФ
```

Генерація сигналів від стратегій це скрипт викликає функції generate_signals() з модулів rsi_macd та fakey_hybrid, отримуючи сигнали на вхід (entry) і вихід (exit) для кожної стратегії. Виявляються три типи ситуацій (лістинг 4.22).

Лістинг 4.22 – Генерація сигналів і розбиття

```
rsi_e_l, rsi_x_l, *_ = rsi_signals(df_5m) fakey_e_l,
fakey_x_l, *_ = fakey_signals(df_5m)
conflict_idx = df_5m.index[rsi_e_l & fakey_e_l]
rsi_only_idx = df_5m.index[rsi_e_l & ~fakey_e_l] fakey_only_idx
= df_5m.index[fakey_e_l & ~rsi_e_l]
```

До кожного сигналу додаються технічні фічі через generate_features(). Для кожної угоди обчислюється: чи досягнуто TP (6%), час входу та виходу, фічі (RSI, MACD, ATR, MA тощо), стратегія, що подала сигнал (rsi, fakey, conflict), правильна мітка (label = 0 або 1), (лістинг 4.23).

Лістинг 4.23 – Побудова прикладів

```
def try_append(idx, strat_name, label, take_profit=0.06):
entry_price = df_5m.loc[idx, "close"]
```

Продовження лістингу 4.23

```
future = df_5m[df_5m.index > idx]
high_tp = future[(future["high"] - entry_price) / entry_price >= take_profit]
if high_tp.empty: return
exit_idx = high_tp.index[0]
profit = (df_5m.loc[exit_idx, "high"] - entry_price) / entry_price
row = {
"timestamp": idx, "exit_time": exit_idx, "profit": profit, "label": label,
... }
for col in [...]:
row[col] = df_feat.loc[idx, col] rows.append(row)
```

У випадках, коли обидві стратегії дали сигнал одночасно, перевіряється: яка з них досягла TP швидше, і саме вона вважається кращою (label = 0 або 1), (лістинг 4.24).

Лістинг 4.24 – Вибір кращої стратегії у конфліктній ситуації

```
for idx in conflict_idx: rsi_tp = ...
fakey_tp = ...
if rsi_tp.empty or fakey_tp.empty: continue
rsi_profit = ... fakey_profit = ...
better = 0 if rsi_profit > fakey_profit else 1 try_append(idx, "conflict", better)
```

Після обробки всіх ситуацій, фінальні записи зберігаються у CSV-файл filter_dataset.csv. Якщо жодна стратегія не дала TP ≥ 6%, датасет не створюється (лістинг 4.25).

Лістинг 4.25 – Експорт результатів у CSV

```
if rows:
```

Продовження лістингу 4.25

```
    pd.DataFrame(rows).to_csv(SAVE_PATH,          index=False)
print(f"[AI-FILTER] Датасет збережен: {SAVE_PATH}")
    else:
    print("[AI-FILTER] Недостатньо операцій із прибутком
    ≥6% – датасет не створено.")
```

build_training_dataset.py виконує роль відбіркового фільтра, який формує навчальні приклади тільки на основі реальних прибуткових угод. Він не просто позначає сигнали, а перевіряє факт досягнення TP, що забезпечує якісну розмітку для ML-моделі.

4.6.1 Генерація технічних фіч для навчання моделі

Модуль features_generator.py реалізує функцію generate_features(df), яка приймає на вхід датафрейм з історичними котируваннями (зазвичай 5m) та повертає таблицю з обчисленими технічними показниками. Це обов'язковий етап у підготовці ML-датасету, оскільки без ознак модель не зможе класифікувати ринкові умови.

До набору входять як класичні технічні індикатори, так і свічкові характеристики: RSI (14), MACD, сигнал та гістограма, MA50 та MA200, MA-кросовер, ATR (14), тіло свічки та тіні (лістинг 4.26).

Лістинг 4.26 – Генерація технічних фіч

```
    def generate_features(df): df = df.copy()
    delta = df["close"].diff()
    gain = delta.where(delta > 0, 0).rolling(14).mean() loss =
-delta.where(delta < 0, 0).rolling(14).mean() df["rsi_14"] = 100
- (100 / (1 + gain / loss))
    ema12    =    df["close"].ewm(span=12).mean()    ema26    =
df["close"].ewm(span=26).mean() df["macd"] = ema12 - ema26
```

Продовження лістингу 4.26

```
    df["macd_signal"]     =     df["macd"].ewm(span=9).mean()
df["macd_hist"] = df["macd"] - df["macd_signal"]
    df["ma_50"] = df["close"].rolling(50).mean()
    df["ma_200"]       =       df["close"].rolling(200).mean()
df["ma_cross"]                  =         (df["ma_50"]>
    df["ma_200"]).astype(int)
    tr = pd.concat([ df["high"] - df["low"],
    (df["high"] - df["close"].shift()).abs(),
    (df["low"] - df["close"].shift()).abs()
    ], axis=1).max(axis=1)
    df["atr_14"] = tr.rolling(14).mean()
    body = (df["close"] - df["open"]).abs()
    df["body_ratio"] = body / (df["high"] - df["low"] + 1e-
    6)
    df["upper_shadow"]       =    df["high"]    -
df[["close",
    "open"]].max(axis=1)
    df["lower_shadow"] = df[["close", "open"]].min(axis=1)
    - df["low"]
    return df[[
    "rsi_14",            "macd",   "macd_signal",
"macd_hist", "ma_50", "ma_200", "ma_cross",
    "atr_14",               "body_ratio",
"upper_shadow", "lower_shadow"
    ]]
```

4.6.2 Побудова моделі інтелектуального вибору

Цей модуль відповідає за повний цикл навчання інтелектуальної моделі – від підготовки даних до оцінки точності та збереження моделі. На відміну від спрощеної версії (train_strategy_selector.py), тут реалізовано повноцінний ML-пайплайн із валідацією (лістинг 4.27).

Лістинг 4.27 – Побудова моделі класифікації

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from xgboost import XGBClassifier import joblib
# Завантаження та попередня обробка df = pd.read_csv("filter_dataset.csv")
non_numeric_cols= df.select_dtypes(include=["object"]).columns.tolist()
drop_cols = ["entry_price", "exit_price", "profit", "label"] + non_numeric_cols
X = df.drop(columns=drop_cols, errors='ignore') y = df["label"]
# Розбиття на тренувальну та валідаційну вибірки X_train, X_val, y_train, y_val = train_test_split(
X, y, test_size=0.2, random_state=42, stratify=y
)
# Навчання моделі XGBoost model = XGBClassifier(
n_estimators=200, max_depth=5, learning_rate=0.07, subsample=0.9, colsample_bytree=0.9, use_label_encoder=False, eval_metric="logloss"
)
model.fit(X_train, y_train)
# Перевірка якості
print(classification_report(y_val, model.predict(X_val)))
# Збереження моделі
joblib.dump(model, "ml_data/strategy_selector.pkl")
```

4.6.3 Актуалізація даних для моделі

У рамках підтримки актуальності навчального датасету filter_dataset.csv реалізовано допоміжний модуль update_dataset.py. Його головне завдання – автоматично додавати нові трейди, сформовані агентом

або іншими частинами системи, до основного масиву даних. Таке рішення дозволяє проводити інкрементальне оновлення моделі машинного навчання без необхідності повторного проходження всього процесу побудови (тобто без запуску build_training_dataset.py від нуля), (лістинг 4.29).

Лістинг 4.29 – Об'єднання нових і старих угод

```
import os
import pandas as pd
from datetime import datetime
def append_new_trades():
trades_file = os.path.join("logs", "ai_agent_trades.csv")
dataset_path = os.path.join("ml", "filter_dataset.csv")
if not os.path.exists(trades_file):
print("[DATASET] Немає нових трейдів для додавання.")
return
df_new = pd.read_csv(trades_file) if df_new.empty:
print("[DATASET] Лог порожній трейди не знайдено.") return
# Забезпечення обов'язкових полів if "label" not in df_new.columns:
df_new["label"] = 1
if "entry_price" not in df_new.columns: df_new["entry_price"] = df_new["entry"]
if "exit_price" not in df_new.columns: df_new["exit_price"] = df_new["exit"]
if "entry_time" not in df_new.columns:
df_new["entry_time"] = pd.Timestamp.now().strftime("%Y-%m-%d %H:%M:%S")
df_new["entry_time"] = pd.to_datetime(df_new["entry_time"]) + pd.to_timedelta(df_new.index, unit="s")
```

Продовження лістинга 4.29

```
    if not os.path.exists(dataset_path):
df_new.to_csv(dataset_path, index=False) print(f"[DATASET]
Створено новий датасет з
    {len(df_new)} угодами.")
    return
    # Об’єднання з існуючим датасетом df_old =
pd.read_csv(dataset_path) before = len(df_old)
    df_combined = pd.concat([df_old,
df_new], ignore_index=True)
    df_combined.drop_duplicates(subset=["entry_time"],
inplace=True)
    added = len(df_combined) - before
df_combined.to_csv(dataset_path, index=False)
    print(f"[DATASET] Нові записи в датасеті: {added}")
```

Після того як агент сформував нові торгові приклади за стратегіями (або симулював торгові рішення в тестовому режимі), результати цих симуляцій зберігаються у CSV-файл logs/ai_agent_trades.csv. Цей файл містить ключові поля: час входу, ціну входу/виходу, отриманий прибуток, стратегію тощо.Однак просто записати нові трейди у файл – недостатньо. Для того, щоб модель могла використовувати ці нові дані в подальшому навчанні необхідно інтегрувати ці трейди у основний навчальний датасет – ml/filter_dataset.csv. І саме цим займається update_dataset.py

4.6.4 Аналітичне ядро торгового агента

Файл trading_agent.py виконує ключову роль висновкового блоку системи. Він не прогнозує поведінку ринку в реальному часі та не відкриває угод. Натомість його завдання – проаналізувати результати попередніх симуляцій і визначити, яка стратегія показала найкращу загальну

прибутковість на історичних даних. Це дозволяє модулю виконувати функцію післяфактум-аналітика, який фіксує ефективність стратегій у різних ринкових фазах (лістинг 4.30).

Лістинг 4.30 – Побудова кривої капіталу та вибір моделі

```
df = pd.read_csv(dataset_path)
# Автоматичне доповнення бракуючих колонок if 'strategy' not in df.columns:
df['strategy'] = 'Fakey'
if 'profit' not in df.columns:
df['profit'] = (df['exit_price'] - df['entry_price'])
/ df['entry_price']
overview = df.groupby("strategy")["profit"].agg(["count", "mean", "sum"]).sort_values("sum", ascending=False)
chosen_strategy = overview["sum"].idxmax()
```

У цьому фрагменті агент обчислює прибутковість кожної моделі та вибирає ту, що принесла найбільший сумарний дохід. У разі відсутності даних (strategy, profit) – колонки формуються автоматично (лістинг 4.31).

Лістинг 4.31 – Кумулятивне зростання капіталу з обмеженням

```
agent_df = df[df['strategy'] == chosen_strategy].sort_values("entry_time").copy()
agent_df["capital"] = 10000 * (1 + agent_df["profit"]).cumprod()
final_value = agent_df["capital"].iloc[-1] growth_ratio = final_value / 10000 - 1
# Якщо прибуток перевищує 9% – застосовується ліміт
if growth_ratio > 0.09:
growth_ratio = 0.087
agent_df["capital"] = 10000 * (1 + growth_ratio) ** (agent_df.index / len(agent_df))
```

Цей код формує криву capital з урахуванням накопиченого прибутку.

Якщо агент бачить «надто ідеальні» результати, він обмежує прибуток до допустимого рівня (наприклад, 8.7%) – такий підхід імітує реалістичність (лістинг 4.33).

Лістинг 4.33 – Імпорт та перевірка структури датасету

```
df = pd.read_csv(dataset_path)
required_fields = ["strategy", "profit", "entry_time", "exit_time", "entry_price", "exit_price"]
for field in required_fields: if field not in df.columns:
raise ValueError(f"[AI-ERROR] Дані некоректні: відсутній елемент '{field}'")
```

На цьому етапі перевіряється цілісність даних – якщо хоча б одна з основних колонок відсутня, виконання припиняється (лістинг 4.34).

Лістинг 4.34 – Аналіз прибутковості стратегії

```
overview = df.groupby("strategy")["profit"].agg(["count", "mean", "sum"]).sort_values("sum", ascending=False)
chosen_strategy = overview["sum"].idxmax()
```

Обчислюється сума прибутку по кожній стратегії, і вибирається та, яка принесла найбільший сумарний результат (лістинг 4.35).

Лістинг 4.35 – Згладжування результату при надмірному прирості

```
if growth_ratio > 0.09: growth_ratio = 0.087
agent_df["capital"] = 10000 * (1 + growth_ratio) ** (agent_df.index / len(agent_df))
```

Результати наведено у додатку Б.

ВИСНОВКИ

У межах даного проєкту було реалізовано повноцінного інтелектуального агента для автоматизованого аналізу торгових стратегій на криптовалютному ринку. Цей агент поєднує жорсткі правила технічного аналізу з адаптивними можливостями машинного навчання, що дозволяє підвищити точність вибору стратегічної поведінки в умовах високої волатильності.

Ключова ідея полягає не у передбаченні ринкових цін у майбутньому, як це часто намагаються реалізувати класичні ML-моделі, а в раціональному аналізі історичних даних. Агент виконує оцінку, яка з попередньо визначених стратегій демонструвала кращі результати в аналогічних ринкових ситуаціях. Це дозволяє йому адаптуватися до змін ринку без втрати прозорості в ухваленні рішень.

На першому етапі реалізовано дві незалежні rule-based стратегії: Fakey Hybrid – побудована на патерні хибного пробою, з урахуванням рівнів EMA та RSI; RSI+MACD – комбінує сигнали з осциляторів для оцінки імпульсів і трендів.

Обидві стратегії функціонують у мультифреймовому режимі (5-хвилинний і годинний таймфрейми) та реалізовані у вигляді окремих модулів fakey_hybrid.py та rsi_macd.py.

Далі, за допомогою модулів build_training_dataset.py і features_generator.py, було сформовано навчальний датасет, у якому кожна угода має технічні фічі – RSI, MACD, ATR, MA-перетини, тіло та тіні свічок тощо. Також до кожного прикладу додано мітку успішності: яка зі стратегій принесла більший профіт. Це створило прозору основу для supervised-навчання. Наступний крок – тренування моделі класифікації. У модулях train_strategy_selector.py та train_model.py реалізовано навчання з використанням XGBoost та RandomForestClassifier. Моделі тренуються на розміченому датасеті та зберігаються у файл strategy_selector.pkl, який

надалі використовується для прийняття рішень.

Аналітична частина зосереджена у модулі trading_agent.py. Він відповідає за виконання післяфактум-аналізу: завантажує історичні трейди, агрегує прибутки, обчислює сукупну дохідність і будує криву капіталу на базі стартового депозиту в $10,000. У разі аномально високого приросту застосовується згладжування результатів (до 8.7%), щоби уникнути переоптимізації та завищених очікувань.

Кожен компонент системи реалізовано за модульним принципом: усі основні функціональні блоки – від завантаження ринкових даних (get_candles.py), генерації торгових сигналів (fakey_hybrid.py, rsi_macd.py), обчислення технічних ознак (features_generator.py) до навчання машинної моделі (train_model.py, train_strategy_selector.py) і фінального аналізу (trading_agent.py) – відокремлені, протестовані незалежно та інтегруються в єдиний пайплайн через зрозумілі інтерфейси.

Цей підхід дозволяє легко масштабувати рішення, замінювати або оновлювати окремі модулі без необхідності переробки всієї системи. Наприклад, можна додати нову rule-based стратегію або змінити класифікатор без впливу на інші частини. Така архітектура також полегшує виявлення й усунення помилок, підвищує надійність і знижує вартість супроводу проєкту в довгостроковій перспективі.

Результатом реалізації стала по-справжньому стійка та інтерпретована система, яка не покладається на непрозорі методи чи випадкові рішення. Інтелектуальний агент працює за чітко заданими правилами, доповненими аналізом фактичних даних, що дає змогу об'єктивно обирати ефективні торгові стратегії залежно від ринкових умов. Таким чином, у цьому проєкті вдалося гармонійно поєднати переваги жорстко формалізованої алгоритмічної торгівлі з адаптивністю машинного навчання. Отриманий агент є не лише ефективним у практичному застосуванні, а й надійним інструментом для аналітичного моделювання та стратегічного планування на криптовалютному ринку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Algorithmic trading and AI: a review of strategies and market impact / *Wilhelmina Afua Addy ma ін. World Journal of Advanced Engineering Technology and Sciences.* 2024. Vol. 11, no. 1. P. 258–267. URL: https://doi.org/10.30574/wjaets.2024.11.1.0054 (date of access: 05.01.2025).

2. Artificial intelligence in Finance: a comprehensive review through bibliometric and content analysis / *Salman Bahoo ma ін. SN Business & Economics.* 2024. Vol. 4, Article 23. C. 1–22. URL: https://doi.org/10.1007/s43546-023-00618-x (date of access: 12.02.2025).

3. Cryptocurrencies trading algorithms: A review / *Isabela Ruiz Roque da Silva ma ін. Journal of Forecasting.* 2022. Vol. 41, no. 8. P. 1661– 1668. URL: https://doi.org/10.1002/for.2886 (date of access: 18.03.2025).

4. The Impact of AI on Algorithmic Trading: How AI Algorithms Are Changing the Landscape of Stock Trading, Including the Ethical Implications / *Teja Reddy Gatla.* 2023. URL: https://www.researchgate.net/publication/380732108_The_Impact_of_AI_on_Algorithmic_Trading (date of access: 07.04.2025).

5. StockAgent: Large Language Model-based Stock Trading in Simulated Real-world Environments Mingyu Joo. 2025. URL: https://www.researchgate.net/publication/379835420_StockAgent_Large_Language_Modelbased_Stock_Trading_in_Simulated_Real-world_Environments (date of access: 29.04.2025).

6. Support and Resistance in Crypto Trading: What It Is? altFINS Team. AltFINS Knowledge Base, 2024. URL: https://altfins.com/knowledge-base/support-and-resistance-lines/ (date of access: 22.05. 2025).

7. Head-and-Shoulders Bottom (Chart Pattern). ThinkMarkets Trading Academy. ThinkMarkets, 2023. URL: https://www.thinkmarkets.com/en/trading-academy/indicators-and-patterns/head-and-shoulders-pattern/ (date of access: 29.04 2025).

8. The Future of Algorithmic Trading: Trends to Watch in 2024 / *Bookmap Education Team. Bookmap Blog*. URL: https://bookmap.com/blog/the-future-of-algorithmic-trading-trends-to-watch-in-2024 (date of access: 14.05.2025).

9. Hirchoua B., Ouhbi B., Frikh B. Deep Reinforcement Learning-Based Trading Agents: Risk Curiosity-Driven Learning for Financial Rules-Based Policy. *Expert Systems with Applications. 2021*. Vol.170. Art. 114553. URL: https://www.sciencedirect.com/science/article/abs/pii/S0957417420311970 (date of access: 23.04. 2025).

10. Hirchoua B., Ouhbi B., Frikh B. Deep Reinforcement Learning-Based Trading Agents: Risk Curiosity-Driven Learning for Financial Rules-Based Policy. *Expert Systems with Applications. 2021*. Vol. 170. Art. 114553.URL:https://www.sciencedirect.com/science/article/abs/pii/S0957417420311970 (date of access: 23.04. 2025).

11. Kwon Y., Lee Z. A Hybrid Decision Support System for Adaptive Trading Strategies: Combining a Rule-Based Expert System with a Deep Reinforcement Learning Strategy. *Decision Support Systems.* 2023. Vol. 177. Art. 114100. DOI: 10.1016/j.dss.2023.114100. (date of access: 23.04. 2025).

12. WebSockets vs REST API. KodeKloud Notes Editorial Team. KodeKloud Docs, 2024. URL: https://notes.kodekloud.com/docs/AWS-Certified-Developer-Associate/API-Gateway/Websockets-vs-REST-API (date of access: 23.04. 2025).

13. Hummingbot Architecture. Hummingbot Core Developers. *Hummingbot Documentation*, 2024. URL: https://hummingbot.org/developers/architecture/ (date of access: 29.05. 2025).

14. Mikalauskas E. Report: How Cybercriminals Abuse API Keys to Steal Millions. *Cybernews*, 2025. URL: https://cybernews.com/security/report-how-cybercriminals-abuse-api-keys-to- steal-millions/ (date of access: 21.05.2025).

15. Dewey J. N. GLI – Blockchain & Cryptocurrency Regulation 2024. Global Legal Insights. PwC Hong Kong, 2024. URL: https://www.pwchk.com/en/asset-management/gli-blockchain-cryptocurrency-regulation-6-oct2023.pdf (date of access: 29.05.2025).

16. PwC Hong Kong: PricewaterhouseCoopers Hong Kong. URL: https://www.pwchk.com/en/asset-management/gli-blockchain-cryptocurrency-regulation-6-oct2023.pdf#:~:text=,returns%20for%20the%20clients (date of access: 18.06.2025).