МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ:

«Експертна система для технічного аналізу вартості криптовалюти»

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма **«Комп'ютерні науки»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи КН-41

Маурін А. А.

Керівник: Гайна Г. А.

Кандидат технічних наук

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 13 від 05.06.2023 р.
зав. кафедри_____доц. Іларіонов О. Є.

Київ – 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ:
Зав. кафедри інтелектуальних технологій

Іларіонов О.Є.
(звання, прізвище та ініціали)

(підпис)

«_____»_____2023 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Мауріну Андрію Андрійовичу

1. Тема проєкту (роботи)

«Експертна система для технічного аналізу вартості криптовалюти» затверджена протоколом засідання кафедри від «11» листопада 2022 р. протокол №4

2. Термін здачі студентом закінченої роботи:

31 травня 2023 року.

3. Вихідні дані до проєкту (роботи):

Дані про ціну за зазначений проміжок часу.

4. Зміст пояснювальної записки (перелік питань, що їх належить розробити):

Аналіз проблем в області фінансових ринків; Проєктування експертної ієрархічної системи; Інтеграція експертної ієрархічної системи з іншими системами;

Висновки.

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій):

Тема і мета дослідження, огляд відомих досліджень, вимоги, математична модель, діаграма класів, досліди і результати, висновки.

6. Консультанти з випускної кваліфікаційної роботи із зазначенням її розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання:

15 лютого 2023 року

Керівник _____ _____

(підпис) (ініціали та прізвище)

Завдання прийняв до виконання _____ Маурін А. А.

(підпис) (ініціали та прізвище)

КАЛЕДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1	Опрацювання літератури	15.02 – 27.02	
2	Робота над розділом 1. Аналіз проблем в області фінансових ринків	27.02 – 08.03	
3	Робота над розділом 2. Проєктування експертної ієрархічної системи	08.03 – 08.04	

4	Робота над розділом 3. Інтеграція експертної ієрархічної системи з іншими системами	08.04 – 13.05	
5	Робота над оформленням пояснювальної записки	13.05 – 25.05	
6	Робота над презентацією	25.05 – 29.05	

Студент  Маурін А. А.

(підпис) (ініціали та прізвище)

Керівник випускної кваліфікаційної роботи _____ _____

(підпис) (ініціали та прізвище)

Анотація

Маурін Андрій Андрійович виконав випускну кваліфікаційну роботу на тему «Експертна система для технічного аналізу вартості криптовалюти» за спеціальністю 122 – «Комп'ютерні науки».

У випускній кваліфікаційній роботі розглянуто функціонування фінансових ринків, основні принципи технічного аналізу, проведено аналіз сучасних методів алгоритмічного трейдингу, розроблено інформаційне та програмне забезпечення, що виконує аналіз ринкових даних і приймає автоматизовані рішення.

Ключові слова: інновація, аналіз, прогнозування, фінанси.

Summary

The graduation thesis: «Expert system for technical analysis of cryptocurrency price» authored by **Maurin Andrii**, specialty 122 - "Computer Science".

In this graduation thesis the function of financial markets is reviewed, the main principles of technical analysis, modern methods of algorithmic trading are analyzed, software, which is used to analyze market data and automate decision-making, is developed.

Keywords: innovation, analysis, forecasting, finance.

ЗМІСТ

ВСТУП

На сьогодні ми можемо бачити приклади торгівлі, яким тисячі років. Система або мережа, яка дозволяє виконувати обмін цінностями, називається ринком.

Вже протягом багатьох сторіч можна виділити окремий вид людської діяльності – спекуляція на ціні товарів, послуг, цінних паперів тощо. Матеріальна вигода людини, яка цим успішно займається, є очевидною. Людина, що може обміняти товар, який у майбутньому зросте у ціні, стає багатшою. Людина, що послідовно й успішно здійснює такі обміни, володіє все більшим грошовим еквівалентом.

Якщо зрозуміти яким саме чином працює ціноутворення, як саме працює ринок, можна опрацювати систему правил, яка допоможе отримувати прибуток, передбачаючи зростання чи падіння ціни.

Ця сфера була і залишається дуже цікавою для багатьох дослідників. На сьогодні відомо досить багато методів, публікацій, дослідницьких робіт, які мають на меті отримати систему правил, яка допоможе передбачати ціну на деякий товар.

Предметом цього дослідження будуть існуючи алгоритми, підходи, методи технічного аналізу. Також як предмет дослідження буде розглянуто функціонування ринку, конкуренція на ньому і процеси ціноутворення. Як цінність, що досліджується, була обрана криптовалюта.

Завдання теоретичного спрямування цього дослідження є вивчення відомих методів алгоритмічного трейдингу, а також розробка нового, покращеного методу, який буде брати до уваги особливості роботи вже відомих методів. Головною метою розробки такого методу буде можливість

практичного застосування його на фінансових ринках для отримання додаткової інформації про його стан. Така інформація може бути використана для отримання прибутку.

Щодо практичної частини дослідження, воно дозволяє автоматизувати торгові операції з використанням розроблених алгоритмів. Це забезпечує швидку реакцію на ринкові умови, уникнення емоційних впливів і зниження людського фактора, що часто може призвести до помилок.

В першому розділі курсової роботи наведено аналітичний огляд відомих підходів, методів технічного аналізу, та деяких алгоритмів машинного навчання, які були застосовані для завдання прогнозування ціни.

В другому розділі розглянуто практичні питання реалізації, алгоритми та структури, архітектура ієрархічної експерт-системи, допоміжні підсистеми, та усі інші аспекти практичної реалізації.

В третьому розділі розглянуто аспекти використання, тестування, налаштування і практичної інтеграції системи.

У висновках наведені результати дослідження, пояснення чим вони можуть бути спричинені, а також інші відповіді на завдання цього дослідження.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМ В ОБЛАСТІ ФІНАНСОВИХ РИНКІВ

1.1 Основні визначення

Для початку роботи необхідно ввести декілька визначень.

Фінансовий ринок (іноді маркет) — інфраструктура, яка дозволяє трейдерам досягати угоди й обмінюватися активами. Існує декілька розподілів фінансових ринків, в залежності від того, що саме на них торгується (цінні папери, облігації, валютні, деривативи тощо). Такі фінансові ринки критично важливі для ринкових капіталістичних економік [1].

Трейдер (торговець) — це особа, яка займається купівлею та продажем фінансових активів на будь-якому фінансовому ринку від себе чи від імені іншої особи, чи установи [2].

Інвестиції та трейдинг — це два дуже різні методи отримання прибутку на фінансових ринках. І інвестори, і трейдери прагнуть отримати прибуток через участь у ринку. Загалом інвестори прагнуть отримати більший прибуток протягом тривалого періоду шляхом купівлі та утримання. Трейдери, навпаки, користуються як зростанням, так і падінням ринків, щоб входити та виходити з позицій протягом коротшого періоду часу, отримуючи менші прибутки частіше [3].

Фундаментальний аналіз — вимірює внутрішню вартість цінних паперів шляхом вивчення відповідних економічних і фінансових факторів. Внутрішня вартість – це вартість інвестиції, яка базується на фінансовій ситуації компанії-емітента та поточних ринкових та економічних умовах [4].

Фундаментальні аналітики вивчають будь-що, що може вплинути на вартість цінних паперів, від макроекономічних факторів, таких як стан

економіки та умови галузі, до мікроекономічних факторів, таких як ефективність управління компанією.

Кінцева мета полягає в тому, щоб визначити число, яке інвестор може порівняти з поточною ціною цінного паперу, щоб побачити, недооцінений цей цінний папір іншими інвесторами.

Технічний аналіз — це аналіз історичної ціни деякого активу, з метою прогнозування майбутнього розвитку подій на фінансовому ринку [5]. Зазвичай трейдери збирають і аналізують статистичну інформацію, про останню поведінку ціни, обсягів, ринку, інших трейдерів. Така інформація оброблюється з метою знаходження нових торгових можливостей. В той час як фундаментальний аналіз зосереджується на метриках бізнесу, або активу, що торгується; технічний аналіз розглядає ціну, імпульс її руху, і обсяги торгів.

Технічні індикатори використовуються для обробки статистичної інформації з ринку [5]. Приклад найпростішого індикатора — середня ціна за добу. Таких індикаторів існує безліч, зазвичай вони оброблюють інформацію ціни й обсягу, і демонструють статистику ринку.

Деякі індикатори більш зосередженні на прогнозуванні тенденції, інші повідомляють імпульс (прискорення), також виділяють індикатори, що відстають, вони підбивають підсумки активності, що вже трапилась. Окремо виділяють рівні підтримки, лінії тенденції, рухомі індикатори, та інші. Такі індикатори використовуються разом, кожен трейдер може самостійно визначати для себе систему для прийняття рішень: які індикатори комбінувати і як інтерпретувати їх сигнали.

Свічкова діаграма (японські свічі) — основний вид графіку ціни, який наочно показує напрям і діапазон ціни за певний проміжок часу. Свіча складається з прямокутника і вертикальної лінії. Така свіча демонструє діапазон

від ціни відкриття, до ціни закриття деякого проміжку часу, а також мінімум і максимум, який було досягнуто. Трейдери використовують свічки для прийняття торгових рішень на основі регулярних моделей, які допомагають передбачити короткостроковий напрямок ціни.

1.2 Графічний інтерфейс трейдера

Розглянемо приклад і основні етапи технічного аналізу.

Зазвичай графік ціни кожен трейдер налаштовує під себе, але є мінімальний набір інформації, яка майже завжди наявна:

1. Графік ціни. Він зазвичай складається з японських свіч, де основна показує ціну відкриття та закриття, а тонша частина показує мінімальну і максимальну ціну за діапазон. Колір символізує напрям руху ціни.
2. Графік обсягів торгівлі. Зазвичай він зображається як гістограма під графіком ціни. Чим більший обсяг торгівлі, тим більш активно проходить конкуренція між попитом та пропозицією.
3. Інформація про час і ціну. Зображається позначками на осі X і Y. На прикладі, що розглядається (див. Рис. 1.1), одна свіча відповідає одному тижню.

Рисунок 1.1 — Графік валютної пари BTC/USD за 20-22 роки.

Основним завданням трейдера є визначення майбутнього руху ціни, і використання цієї інформації для отримання прибутку. Для прикладу, якщо трейдер у 2020 році вважає що ціна валютної пари BTC/USD буде зростати, він може придбати BTC за 4 000 USD і продати через рік за ціною 50 000 USD, таким чином примноживши свої інвестиції у 12.5 разів.

1.3 Приклад застосування технічного аналізу

Зазвичай для прийняття рішень, трейдери збирають і використовують статистичну інформацію, будують графіки, і використовують інші інструменти. Тому більшість сучасного програмного забезпечення надає змогу будувати

різноманітні графіки та аналізувати статистичну інформацію.

Далі ми розглянемо найпростіший приклад технічного індикатора і правила, яке використовує цей індикатор для генерації сигналів до покупки та продажу.

Технічний індикатор рухоме середнє — це середнє значення ціни за деякий період. На прикладі, що розглядається, обрані два індикатори із довжиною 26 і 52 тижні.

Правило для генерації сигналів — перетин двох рухомих середніх. Індикатор з коротшим діапазоном реагує на зміни в ціні швидше, ніж індикатор із довгим діапазоном. Тому якщо використовувати два рухомих середніх і зобразити їх на графіку, вони можуть перетинатися. Якщо коротше перетинає довше, це може розглядатися як сигнал до покупки, якщо навпаки, то сигнал до продажу (див. Рис. 1.2). Також можна розглядати обернене правило. Або замість звичайного середнього розглядати експоненційне середнє, де більша вага надається новим значенням.

Рисунок 1.2 — Застосування правила перетину двох рухомих середніх

Основні елементи, зображені на рисунку:

1. Рухоме середнє (26 тижнів, коротке);
2. Рухоме середнє (52 тижні, довге);
3. Коротке перетинає довге, сигнал до покупки;
4. Довге перетинає коротке, сигнал до продажу;
5. Результат трейду на цьому проміжку за цим правилом, 244% за 637 днів.

Таких правил технічного аналізу існує безліч, зазвичай кожне з них має декілька параметрів для налаштування (наприклад довжина рухомого середнього). Кожен трейдер дотримується своєї системи, і використовує правила, які вважає доречними.

1.4 Огляд відомих систем алгоритмічного трейдингу

Станом на 2012 рік звіт Моргана Стенлі показав, що 84% усіх торгів акціями на фондовому ринку США проводилися за допомогою комп'ютерних алгоритмів, тоді як лише 16% — інвесторами [6].

З 46 вивчених публікацій [7], які були опубліковані на тему застосування методів штучного інтелекту для прогнозування руху ціни, 21-на публікація використовує технологію на основі штучних нейронних мереж, 10 із них досліджують еволюційні методи або методи оптимізації, а решта поєднують різні алгоритми в гібридні системи. 31 з них використовують добову частоту, а 35 з них використовують лише ринковий індекс як ціль. Для методів оцінки більшість із них використовують лише помилки прогнозу як показник оцінки, і лише три статті перевіряють свої моделі в реальному або змодельованому торговому середовищі, де потрібно враховувати вимоги до маржі, торгові витрати та ризик.

Також окремо можна виділити застосування рекурентних нейронних мереж, які часто використовуються для передбачення і прогнозування часового ряду. Такі мережі використовуються для наближення майбутніх значень функції, прогнозу погоди та генерації музики. Внаслідок симуляції клітин пам'яті, такі мережі добре працюють із часовими патернами. Але застосування рекурентних мереж для завдання пронозу майбутньої ціни не є тривіальним, через велику кількість шуму, яка характерна для фінансових ринків.

Застосування штучного інтелекту дуже широке і за великим рахунком залежить більше від джерела зацікавленості дослідника. Частіше за все метою моделі є передбачення напрямку ціни, або наближення майбутніх значень, або прогнозування волатильності ринку. Тому вирішуються різноманітні задачі

класифікації, регресії, прогнозування.

Виходячи із мети дослідження обирають різні методи штучного інтелекту, але найчастіше застосовують штучні нейронні мережі, дерева рішень, метод опорних векторів, регресивні моделі, і класифікатори.

Найцікавіші дослідження з цього приводу:

- "Algorithmic trading of cryptocurrency based on Twitter sentiment analysis" [8]. Це дослідження описує підхід до алгоритмічного трейдингу на основі аналізу настроїв у соціальній мережі Twitter. Автори досліджували кореляцію між твітами та цінами криптовалют і розробили алгоритм, який використовує цю інформацію для трейдингу.
- "A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem" [9]. Розглядає підхід до трейдингу на основі глибинного підсиленого навчання. Автори використовували алгоритм Q-learning для оптимізації фінансового портфеля, що включає криптовалюти.
- "Bitcoin Trading Strategy: Statistical Arbitrage" [10]. Описує техніку статистичного арбітражу, яка використовується для трейдингу на криптовалютному ринку. Автори досліджували кореляцію між різними криптовалютами та розробили стратегію трейдингу на основі цієї інформації.
- "An algorithmic trading model for bitcoin" [11]. Це дослідження представляє особливу цікавість. Автори використовували технічний та фундаментальний аналіз для розробки стратегії трейдингу. Технічний аналіз містить в собі використання індикаторів, таких як середня ціна та індекс сили, для аналізу графіків цін біткоїна. Фундаментальний аналіз містить в собі аналіз новин та подій, що можуть вплинути на ціну

біткоїна. Дослідження також включало тестування алгоритму на історичних даних. Автори використовували дані про ціни біткоїна за останні 2 роки та перевірили ефективність алгоритму на цих даних. Результати дослідження показали, що алгоритм має високу точність в прийнятті рішень про купівлю та продаж біткоїна. Також було показано, що алгоритм працює краще, коли використовується комбінація технічного та фундаментального аналізу, порівняно з використанням лише одного з цих методів.

1.5 Тема, мета і методи дослідження

Темою цього дослідження є розробка експертної ієрархічної системи для алгоритмічного трейдингу криптовалюти [12].

Метою дослідження є застосування технічного аналізу для автоматизованого трейдингу на ринку криптовалют, для генерації прибутку, використовуючи алгоритми, що базуються на аналізі ринкових даних.

Методи розв'язання:

- Збір та аналіз ринкових даних: Для розробки системи для алгоритмічного трейдингу необхідно зібрати та обробити великі обсяги даних з криптовалютного ринку, такі як ціна, обсяги торгів тощо. Для цього можна використовувати програмне забезпечення для збору даних, таке як Bloomberg Terminal, Quandl або Binance API.
- Розробка стратегій трейдингу: Після збору даних, необхідно розробити алгоритми для відкриття та закриття позицій на основі аналізу цих даних. Ці стратегії можуть бути основаними на різних показниках, таких як технічний аналіз і машинне навчання.

- Розробка програмного забезпечення для трейдингу: Після розробки стратегій, необхідно реалізувати їх у вигляді програмного забезпечення, що автоматично здійснює купівлю та продаж на основі вищезгаданих стратегій.
- Тестування та підтвердження ефективності: Після розробки програмного забезпечення, необхідно виконати тестування на історичних даних.

1.6 Основні вимоги до системи

Функціональні вимоги:

- Онлайн робота з даними, що постійно оновлюються: система має обробляти інформацію в реальному часі та працювати з даними, які постійно оновлюються, такі як ціни на криптовалюти, новини про події, які можуть вплинути на ринок тощо.
- Можливість конфігурації: система має бути налаштована під конкретні потреби користувача, такі як встановлення параметрів ризику, часу інтервалів тощо.
- Можливість тестування системи на нових історичних даних дозволяє виявити ефективність розробленої стратегії та розуміти, як вона поводиться на ринку.

Нефункціональні вимоги:

- Швидкість прийняття рішень: система має бути достатньо швидкою, щоб забезпечити роботу у ситуаціях, що швидко змінюються.
- Надійність та стабільність: система повинна працювати без збоїв, адже вона працює з фінансами, і помилки можуть багато коштувати.

- Документація та підтримка: система повинна мати чітку документацію та підтримку.

Система взаємодіє із біржою Binance, має конфігурацію, алгоритм тренування і тестується на історичних даних.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ЕКСПЕРТНОЇ ІЄРАРХІЧНОЇ СИСТЕМИ

2.1 Аналіз бізнес-процесів

2.1.1 Ціноутворення

Процес ціноутворення на фондовому ринку відбувається відповідно до законів попиту та пропозиції. Ціноутворення - це процес визначення ринкової ціни для певного цінного паперу, який відбувається на основі попиту та пропозиції.

Попит на ринку визначається кількістю та готовністю інвесторів купувати цінні папери, тоді як пропозиція визначається кількістю та готовністю інвесторів продавати цінні папери. Коли попит на певний цінний папір перевищує пропозицію, ціна зазвичай зростає, оскільки інвестори більше зацікавлені в його придбанні, ніж продажу. Якщо пропозиція перевищує попит, ціна зазвичай падає, оскільки інвестори більше зацікавлені у продажу, ніж покупці.

Учасники ринку можуть впливати на процес ціноутворення, оскільки вони можуть бути здатні визначати як попит, так і пропозицію. Наприклад, якщо великий інституціональний інвестор робить велику покупку цінних паперів, він може сприяти зростанню цін на ці папери, оскільки його попит перевищує пропозицію. Зворотно, якщо великий інституціональний інвестор робить велику продаж цінних паперів, він може сприяти зниженню цін на ці папери, оскільки його пропозиція перевищує попит.

2.1.2 Маркетмейкери

Маркетмейкери — це фінансові інституції, які забезпечують ліквідність на фондовому ринку, шляхом створення попиту та пропозиції для певних цінних паперів. Вони забезпечують покупців і продавців цінних паперів з можливістю здійснення торгів, незалежно від того, чи є інші учасники ринку, готові купувати або продавати в той самий момент часу.

Маркетмейкери діють як посередники між покупцями та продавцями, створюючи власний ринок для певних цінних паперів. Вони готові купувати та продавати цінні папери за ціною, яка є прийнятною для них, в залежності від обсягу торгів та рівня попиту та пропозиції на ринку. Зазвичай, маркетмейкери надають ціну покупки та продажу для певного цінного паперу, що забезпечує зручність для інвесторів, які шукають кращу ціну для купівлі або продажу.

Один з основних принципів роботи маркетмейкерів полягає в тому, щоб забезпечити стабільну ліквідність на ринку, що знижує ризик для інвесторів при купівлі та продажу цінних паперів. Також, маркетмейкери можуть мати вплив на процес ціноутворення на ринку, оскільки вони можуть виступати як покупці та продавці, впливаючи на попит та пропозицію на ринку.

2.1.3 Ліквідність активів

Ліквідність - це міра, яка відображає можливість купівлі або продажу активів (цінних паперів, валюти, товарів тощо) на ринку без значних змін у їхній ціні. Іншими словами, ліквідність показує, наскільки легко можна обміняти актив на гроші.

Чим більша кількість купців та продавців на ринку, тим більша ймовірність того, що будь-який актив може бути легко куплений або проданий

за наявної ринкової ціни. З іншого боку, якщо на ринку є обмежений попит або пропозиція на певний актив, то можуть виникнути проблеми з ліквідністю.

2.1.4 Інвестування

Інвестування - це процес вкладання грошей з метою отримання доходу або збільшення вартості інвестованих активів в майбутньому. Інвестори мають різні цілі та стратегії інвестування, але в основі всіх інвестицій лежить бажання збільшити свої фінансові ресурси.

Для досягнення цілей інвестори можуть використовувати різні механізми, такі як:

1. Купівля акцій: Інвестори можуть купувати акції певних компаній, з метою збільшення їх вартості, отримання дивідендів або комбінації обох.
2. Фінансові інструменти: Інвестори можуть вкладати гроші в різні фінансові інструменти, такі як облігації, іпотечні цінні папери, пайові інвестиційні фонди, ETF та інші.
3. Нерухомість: Інвестори можуть інвестувати гроші в нерухомість з метою здобуття доходу від оренди, збільшення вартості нерухомості або комбінації обох.

Ризики інвестування можуть бути різними, залежно від виду інвестицій та ринку, на якому вони здійснюються. Деякі загальні ризики інвестування включають:

1. Ризик ринку: Ринкові умови можуть змінюватися, що може призвести до зменшення вартості інвестицій.
2. Ризик валюти: Інвестори, які інвестують у компанії за межами своєї країни, можуть зазнавати втрат від зміни курсу валют.

3. Ризик кредиту: Інвестори, які інвестують у боргові інструменти, можуть зазнавати втрат через неможливість повернення інвестицій.

2.1.5 Основні відмінності фондового і криптовалютного ринків

Регулювання: Фондовий ринок має більш жорстке регулювання з боку держави та регуляторних органів, що робить його менш вразливим до зловживань та маніпуляцій. У той же час, ринок криптовалют, зазвичай, має менше регулювання, що збільшує ризики для інвесторів.

Розмір ринку: Фондовий ринок є більшим та більш ліквідним, оскільки на ньому торгуються акції великих корпорацій, а також бонди та інші інструменти. Ринок криптовалют є меншим та менш ліквідним, оскільки на ньому торгуються лише криптовалюти.

Волатильність: Ринок криптовалют є більш волатильним, оскільки його ціни можуть значно змінюватися залежно від новин, технічних аспектів, а також різних інших факторів. Фондовий ринок, зазвичай, є менш волатильним та стабільнішим.

Валюти: На фондовому ринку торгуються акції та інші інструменти, які зазвичай торгуються в національній валюті. На ринку криптовалют, як правило, торгуються криптовалюти, які можуть торгуватися в різних валютах.

Технічні аспекти: Ринок криптовалют використовує технологію блокчейн, яка забезпечує децентралізацію та безпеку транзакцій. Фондовий ринок, зазвичай, використовує централізовану систему торгівлі та клірингу.

2.2 Проєктування системи

Розглянемо систему як чорну скриньку (див. Рис. 2.1).

Система на вхід приймає дані про ціну за деякий проміжок (наприклад, за останній місяць). На вихід видає один з трьох сигналів: купувати, продавати, очікувати.

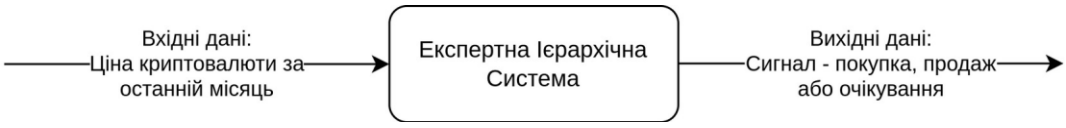

Рисунок 2.1 — Діаграма системи як чорної скриньки.

Система використовує дані, які отримує через Binance Api, яке має правила користування (див. Рис. 2.2). Системою управляє користувач, який може налаштовувати її під себе.

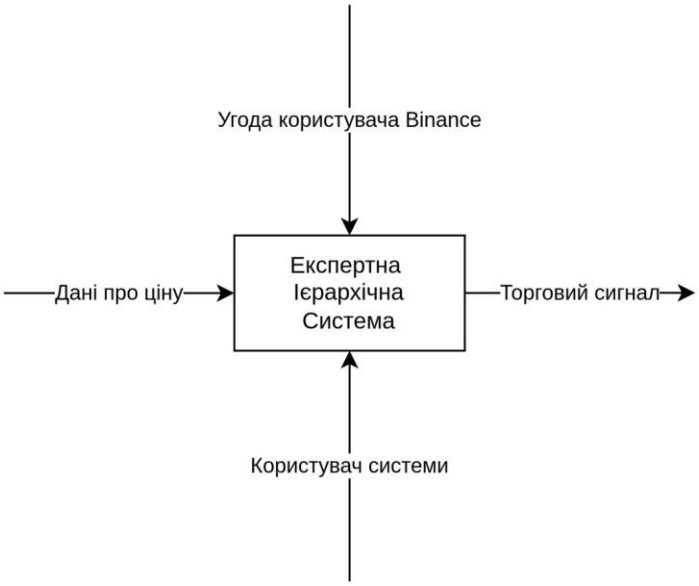

Рисунок 2.2 — IDEF0-діаграма системи.

Загальний процес складається із трьох етапів (див. Рис. 2.3):

1. Завантаження даних для тренування.
2. Тренування системи.
3. Прийняття рішень.

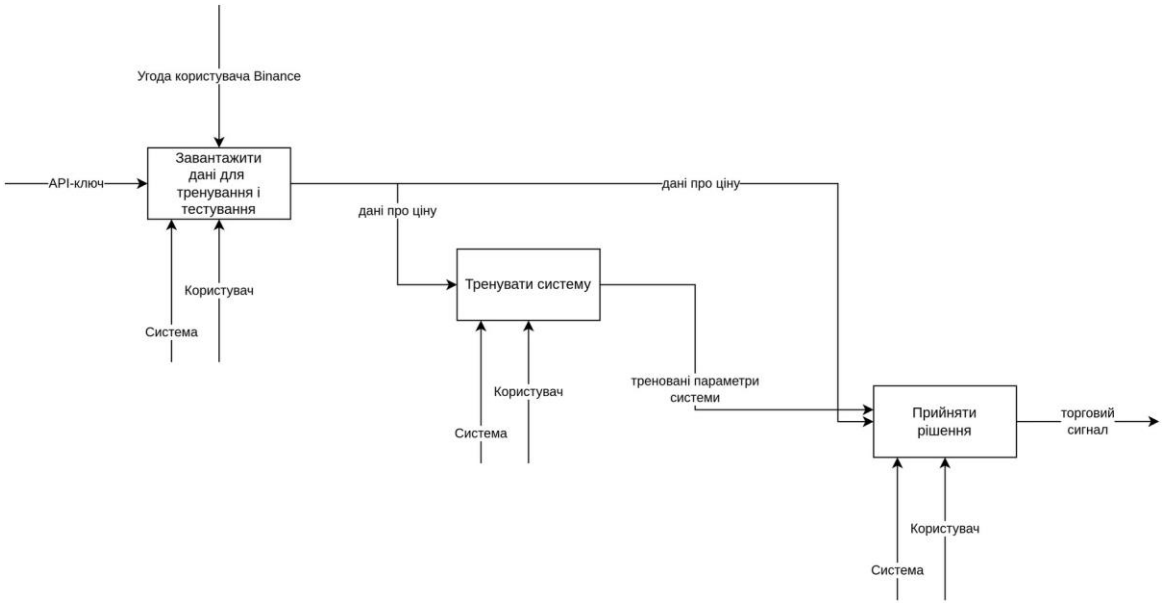

Рисунок 2.3 — IDEF0-діаграма системи, другий рівень декомпозиції.

Система складається із трьох основних компонент (див. Рис. 2.4):

1. Технічний індикатор. Він оброблює статистичну інформацію ринку за деякою ознакою.
2. Правило технічного аналізу. Застосовується до одного або більшої кількості технічних індикаторів. За цим правилом утворюється сигнал до покупки, продажу, або очікування.
3. Експерт. Може агрегувати декілька правил технічного аналізу або інших експертів. Зважує рішення, які від них надходять, і формує свою оцінку від -1 (сигнал до продажу) до +1 (сигнал до покупки).

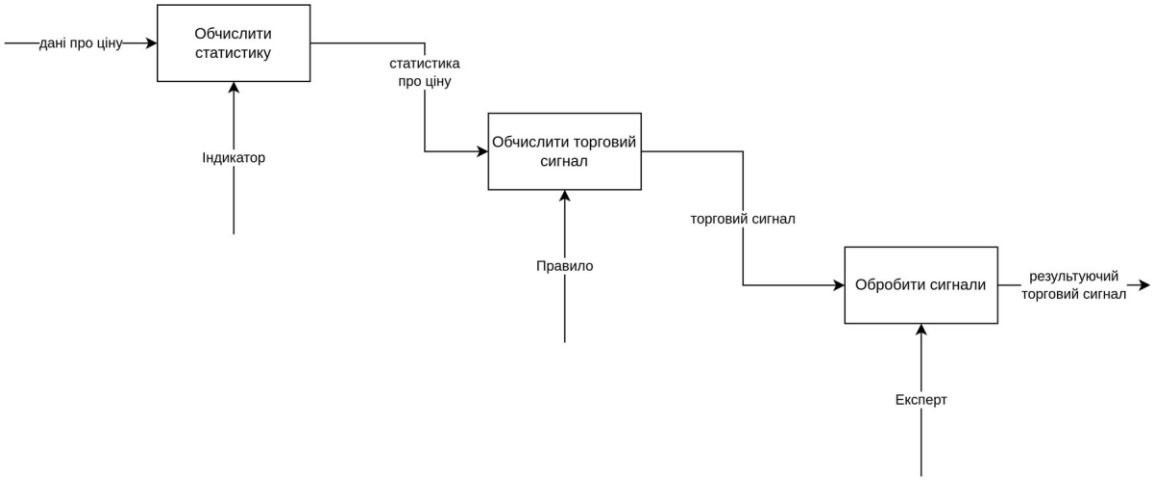

Рисунок 2.4 — IDEF0-діаграма системи, третій рівень декомпозиції.

Основний процес може бути розкладений на дерево функцій (див. Рис. 2.5).

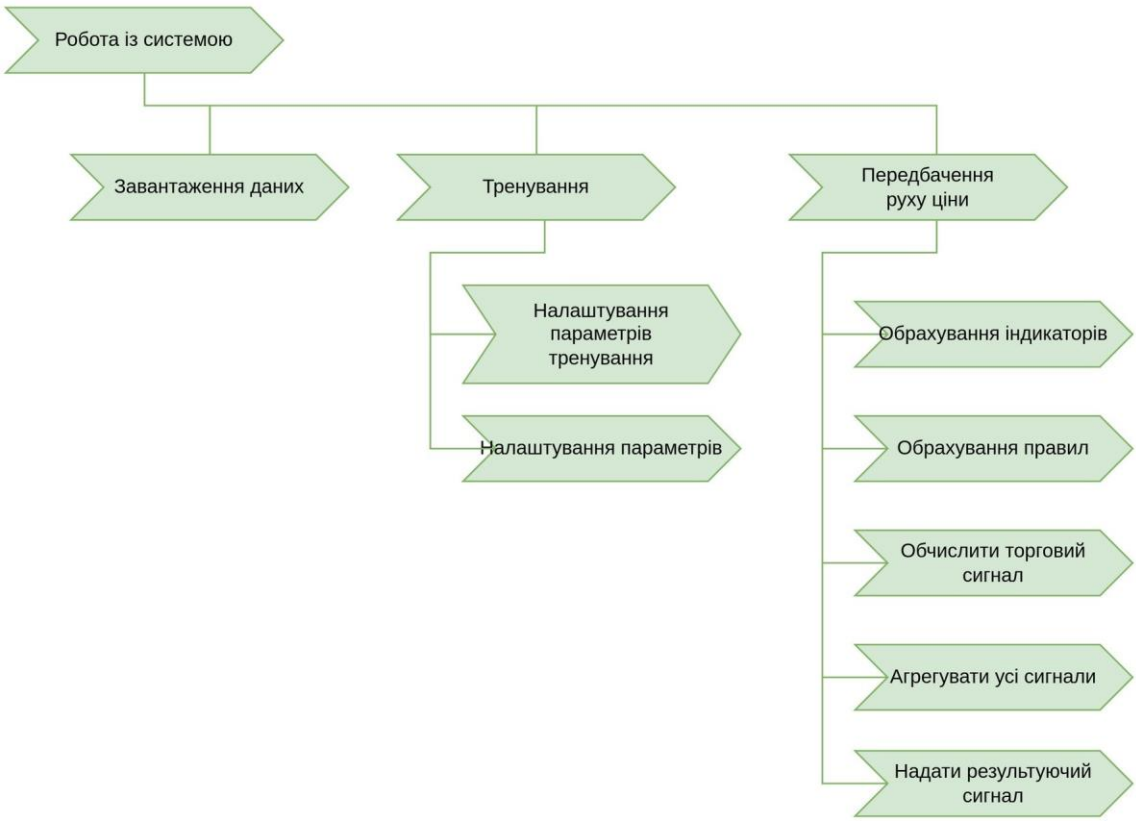

Рисунок 2.5 — Дерево функцій системи.

CRY складається із чотирьох рівнів ієрархії експертів (див. Рис. 2.6) (від найнижчого до найвищого):

- RuleExpert. Відповідальний за менеджмент технічного правила із конкретними параметрами (приклад: перетин двох рухомих середніх із довжиною 200 і 50). Такий експерт має одне правило й один або декілька індикаторів (це залежить від правила).
- RuleClassExpert. Відповідальний за менеджмент технічного правила. Агрегує рішення декількох експертів одного класу із різними параметрами (приклад: перетин двох рухомих).

- TimeFrameExpert. Відповідальний за менеджмент конкретного таймфрейму (частота оновлення свічок) (приклад: 1 день). Агрегує рішення декількох експертів RuleClassExpert.
- PairExpert. Відповідальний за менеджмент конкретної валютної пари (приклад BTC/USD). Агрегує рішення декількох експертів TimeFrameExpert.

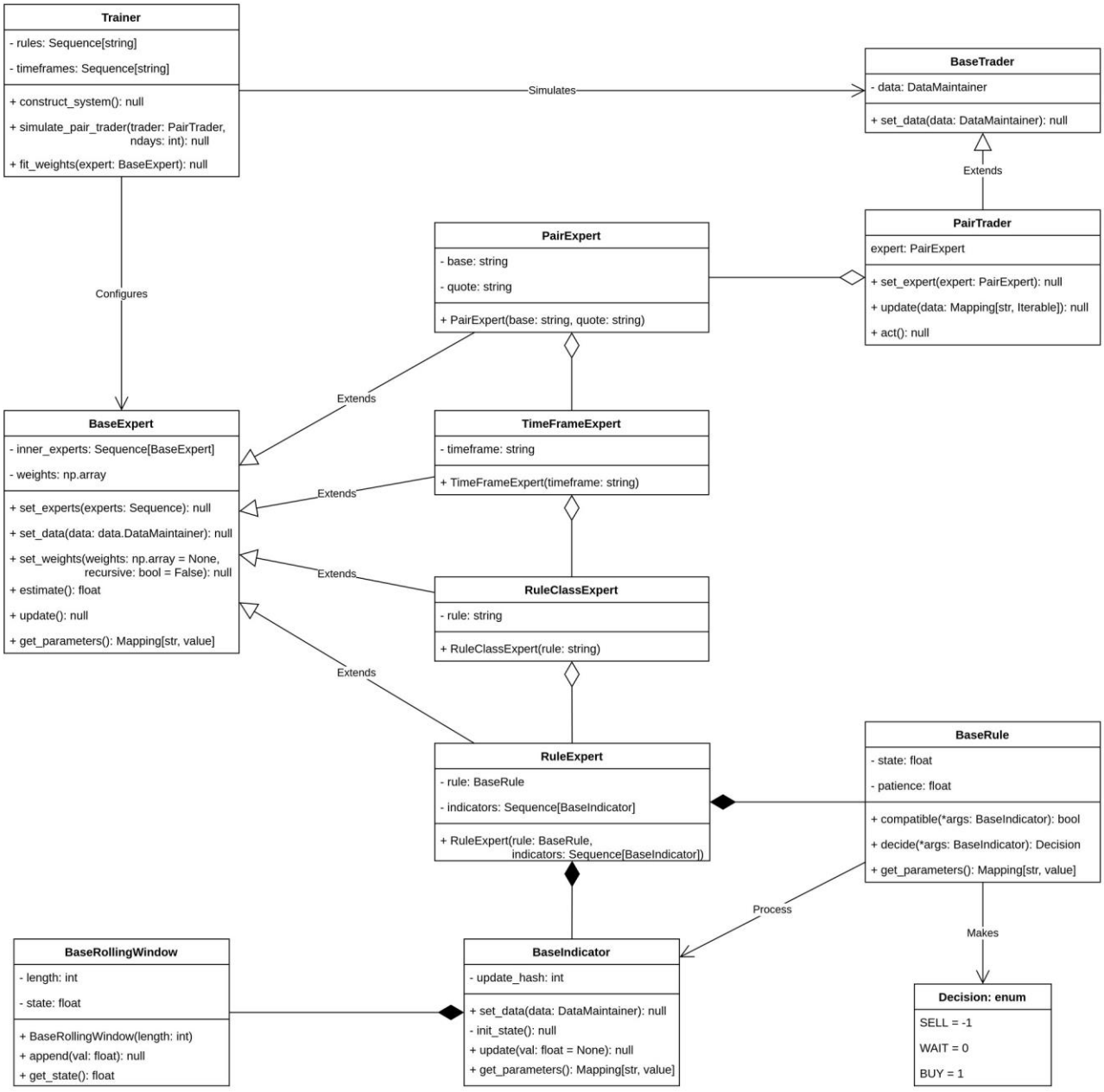

Рисунок 2.6 — UML-діаграма класів системи.

2.3 Математична модель системи

Система поступово формує технічні сигнали, низькорівневі оцінки, і нарешті високорівневі оцінки. Кожен наступний рівень базується на агрегації

оцінок попереднього рівня.

З математичної точки зору, можна бачити, що така оцінка буде лінійною комбінацією входів системи:

(1) $$f_{3,j} = \sum_{i=0}^{n} a_i f_{4,i}$$

де $f_{4,i}$ *– оцінка* i *–гоексперта* 4 *–го рівня і* a_i *– його ваговий коефіцієнт*

(2) $$f_{2,k} = \sum_{i=0}^{m} p_i f_{3,i}$$

де $f_{3,i}$ *– оцінка* i *–го експерта* 3 *–го рівня і* p_i *– його ваговий коефіцієнт*

(3) $$f_1 = \sum_{i=0}^{q} \gamma_i f_{2,i}$$

де $f_{2,i}$ *– оцінка* i *–го експерта* 2 *–го рівня і* γ_i *– його ваговий коефіцієнт*

Але саме така деревоподібна архітектура надає можливість значно зменшити кількість параметрів, розподілити обов’язки та навчати окремих експертів окремо. Таким чином можна збільшити точність і прискорити навчання.

Для роботи й ефективної обробки даних, що постійно надходять до системи. А також для розрахунку статистичної інформації, формування технічних індикаторів розроблено допоміжний модуль. Він включає класи, які дозволяють розраховувати статистики (середнє, мінімальне, максимальне значення, стандартне відхилення, та інші) на потоку даних за константний час.

2.4 Менеджмент даних

Також для розподілу доступу до даних на різних рівнях ієрархії розроблено допоміжний компонент, який надає доступ лише до необхідної інформації.

Таким чином індикатор має доступ до даних конкретного таймфрейму. Експерт рівня валютної пари має доступ до усіх таймфреймів цієї пари. Дані зберігаються в тій самій ієрархії, якою пов’язані експерти.

2.5 Інструментальні засоби для реалізації

Програмна реалізація знаходиться на ресурсі GitHub [13].

Для реалізації системи обрана мова програмування Python [14]. Ця мова програмування має багато корисних інструментів для розробки технологій, які пов’язані з аналітикою даних і штучним інтелектом.

Для роботи зі статистичними розподілами, математичними функціями, функціями активації, нормалізації були використані бібліотеки NumPy [15] і SciPy [16].

Для отримання історичних даних про ціну буде використано Binance Api [17]. Такий вибір було зроблено через безоплатний доступ і простоту використання.

Для зберігання та обробки табличних даних буде використано бібліотеку Pandas [18]. Ця бібліотека дозволяє ефективно зберігати й швидко опрацьовувати великі об’єми даних.

Для серіалізації експертів і їх параметрів буде застосовано бібліотеку Pickle [19].

Крім сторонніх бібліотек, ще буде використано функціонал, який надає стандартна бібліотека мови Python.

2.6 Схема навчання

Архітектура, яка розглянута вище, надає можливість абстракції та розподілення відповідальності за прийняття рішень. Тому зручніше за все, буде підібрати оптимальні параметри поступово, навчаючи одного експерта за раз, потім рухатись до наступного рівня ієрархії.

Із зазначеного поля допустимих параметрів, зіставляються усі можливі експерти RuleExpert, вони оцінюються на малому обсягу навчальних даних (100 днів), щоб отримати представлення наскільки кожен експерт ефективний. Таким чином можна одразу відкинути більшу частину експертів, які недостатньо ефективні.

З експертів, що залишились, зіставляється повний експерт, із необхідною конфігурацією.

Оптимізація параметрів відбувається на більшому обсягу даних (350 днів) знизу догори із застосуванням еволюційної стратегії. Таким чином в першу чергу налаштовуються нижні рівні ієрархії, далі верхні рівні налаштовуються виходячи із вже оптимальних нижніх рівнів.

2.7 План експериментальних досліджень

Дослідження будуть проводитись на основних таймфреймах: 5 хв, 1 год, 4 год, 1 день. Такий вибір параметрів для тестування надасть повне покриття темпів трейдингу (швидка торгівля, помірна торгівля, і торгівля на великих часових проміжках).

Валютна пара BTC/USD. Такий вибір було зроблено через популярність, це надасть дуже добре наближення ринку в цілому.

Кількість експертів найнижчого рівня: 3, 10, 30. Такий діапазон забезпечить повне покриття кількості параметрів системи (від легшої, до більш ускладненої).

Як вже зазначалось раніше, на практиці існує безліч технічних правил. Для дослідження ми обираємо найбільш популярні правила, дослідження яких нас найбільше цікавить:

- Moving Average Crossover — перетин двох рухомих середніх [20]. Існують модифікації: SMA, EMA, TRIX, та інші.
- Relative Strength Index — вимірює швидкість та імпульс руху ціни за діапазон [21].
- Moving average convergence/divergence — вимірює відношення між двома EMA, досліджує різницю між довшим і коротшим [22].
- Ichimoku Kinko Hyo — набір технічних правил, розроблений японським трейдером у 1930 році, і опублікований у 1960 році. Він агрегує декілька індикаторів у один, зосереджуючись на середній, мінімальній і максимальній цінах [23].
- Bollinger Bands — вимірює середню ціну і стандартне відхилення за діапазон [24].

У відкритому доступі є історичні дані за ціну цієї пари за останні 4 роки. Для тренування використаємо проміжок у 350 днів, для тестування — 1000 днів.

Цікаво дослідити вплив конфігурації системи на її результативність. Ми не зможемо покрити кожну можливу конфігурацію, тому зосередимось на двох:

- Повна система: усі доступні експерти, усі таймфрейми, 5 експертів останнього рівня.
- Спрощена система: усі доступні експерти, 1 год. таймфрейм, 3 експерти останнього рівня.

2.8 Протоколи навчання й оцінки точності

2.8.1 Повна система

Розглянемо результати тренування системи із різними параметрами. Для тестування була обрана наступна структура експерта (див. Рис. 2.7).

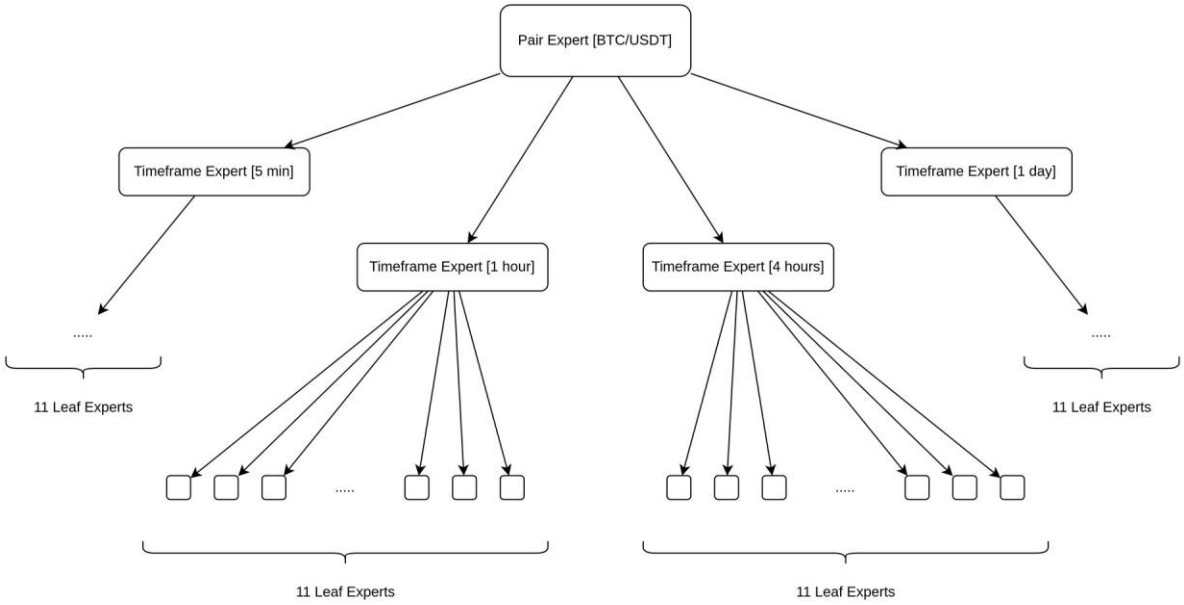

Рисунок 2.7 — UML-діаграма об’єктів повної системи.

Більш детально ієрархію можна описати текстом. Рівень відступу позначає рівень в ієрархії. Кожний рядок містить інформацію про кількість підпорядкованих експертів, а також назву самого експерта:

```
220] PairExpert [BTC/USDT]
  55] TimeFrameExpert [5m]
    5] RuleClassExpert [TripleExponentialDirectionChangeRule]
    5] RuleClassExpert [IchimokuKinkoHyoTenkanKijunCrossoverRule]
    5] RuleClassExpert [ExponentialMovingAverageCrossoverRule]
    5] RuleClassExpert [MovingAverageConvergenceDivergenceSignalLineCrossoverRule]
    5] RuleClassExpert [MovingAverageCrossoverRule]
    5] RuleClassExpert [IchimokuKinkoHyoChikouCrossoverRule]
    5] RuleClassExpert [IchimokuKinkoHyoSenkouASenkouBCrossoverRule]
```

```
    5] RuleClassExpert [BollingerBandsUpperMidCrossoverRule]
    5] RuleClassExpert [BollingerBandsLowerMidCrossoverRule]
    5] RuleClassExpert [BollingerBandsLowerUpperCrossoverRule]
    5] RuleClassExpert [RelativeStrengthIndexTrasholdRule]
  55] TimeFrameExpert [1h]
    5] RuleClassExpert [TripleExponentialDirectionChangeRule]
    5] RuleClassExpert [IchimokuKinkoHyoTenkanKijunCrossoverRule]
    5] RuleClassExpert [ExponentialMovingAverageCrossoverRule]
    5] RuleClassExpert [MovingAverageConvergenceDivergenceSignalLineCrossoverRule]
    5] RuleClassExpert [IchimokuKinkoHyoChikouCrossoverRule]
    5] RuleClassExpert [MovingAverageCrossoverRule]
    5] RuleClassExpert [IchimokuKinkoHyoSenkouASenkouBCrossoverRule]
    5] RuleClassExpert [BollingerBandsLowerMidCrossoverRule]
    5] RuleClassExpert [BollingerBandsUpperMidCrossoverRule]
    5] RuleClassExpert [BollingerBandsLowerUpperCrossoverRule]
    5] RuleClassExpert [RelativeStrengthIndexTrasholdRule]
  55] TimeFrameExpert [4h]
    5] RuleClassExpert [IchimokuKinkoHyoTenkanKijunCrossoverRule]
    5] RuleClassExpert [TripleExponentialDirectionChangeRule]
    5] RuleClassExpert [IchimokuKinkoHyoSenkouASenkouBCrossoverRule]
    5] RuleClassExpert [MovingAverageCrossoverRule]
    5] RuleClassExpert [ExponentialMovingAverageCrossoverRule]
    5] RuleClassExpert [MovingAverageConvergenceDivergenceSignalLineCrossoverRule]
    5] RuleClassExpert [IchimokuKinkoHyoChikouCrossoverRule]
    5] RuleClassExpert [BollingerBandsUpperMidCrossoverRule]
    5] RuleClassExpert [BollingerBandsLowerUpperCrossoverRule]
    5] RuleClassExpert [BollingerBandsLowerMidCrossoverRule]
    5] RuleClassExpert [RelativeStrengthIndexTrasholdRule]
  55] TimeFrameExpert [1d]
    5] RuleClassExpert [TripleExponentialDirectionChangeRule]
    5] RuleClassExpert [BollingerBandsLowerMidCrossoverRule]
    5] RuleClassExpert [BollingerBandsUpperMidCrossoverRule]
    5] RuleClassExpert [BollingerBandsLowerUpperCrossoverRule]
    5] RuleClassExpert [MovingAverageConvergenceDivergenceSignalLineCrossoverRule]
    5] RuleClassExpert [MovingAverageCrossoverRule]
    5] RuleClassExpert [ExponentialMovingAverageCrossoverRule]
    5] RuleClassExpert [IchimokuKinkoHyoTenkanKijunCrossoverRule]
    5] RuleClassExpert [IchimokuKinkoHyoSenkouASenkouBCrossoverRule]
    5] RuleClassExpert [IchimokuKinkoHyoChikouCrossoverRule]
    5] RuleClassExpert [RelativeStrengthIndexTrasholdRule]
```

Тренування було запущено із наступними параметрами (див. Рис. 2.8).

```
python main.py --all_rules --best_timeframes --nleaves 5 --nepochs 20 --fee 0
--reestimate
```

Такий експерт був тренований на проміжку у 350 днів. Таким чином ми можемо оцінити які таймфрейми були найбільш успішними на тренувальних даних:

- 5 хвилин — 17.08% прибутку.
- 1 година — 58.74% прибутку.
- 4 години — 22.47% прибутку.

- 1 день — 29.94% прибутку.

Для перегляду документації щодо цих параметрів:

```
python main.py --help
```

Лог тренування наведений у Додатку 2.

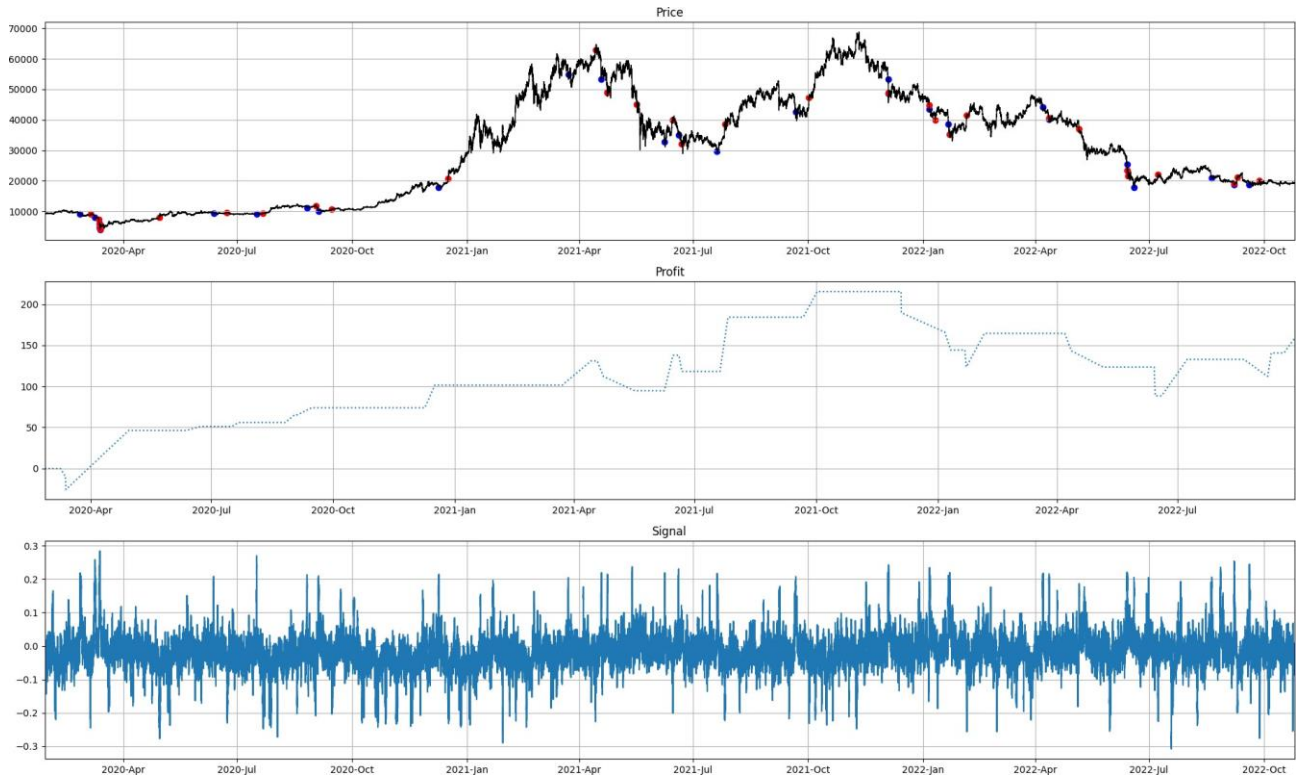

Рисунок 2.8 — Симуляція на тестових даних.

На тренувальних даних на проміжку 350 днів, система отримала -8% прибутку. Такий результат є допустимим, адже на тренувальних даних ціна валютної пари змінилася на -60%, результат трейдингу -8% є кращим аніж результат стратегії інвестування (buy and hold).

У результаті симуляції на 1000 днів на тестових даних, система провела 60 трейдів і отримала 158% прибутку. За той самий період ціна валютної пари

змінилась на +200%.

З логу тренування, можна зауважити, що найкращий результат показав таймфрейм 1 год.

2.8.1 Спрощена система

Експерт використовує 1 год. таймфрейм і 11 правил (див. Рис. 2.9).

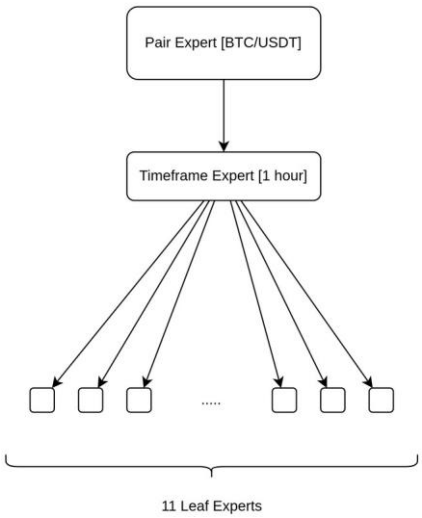

Рисунок 2.9 — UML-діаграма об’єктів спрощеної системи.

```
33] PairExpert [BTC/USDT]
  33] TimeFrameExpert [1h]
    3] RuleClassExpert [TripleExponentialDirectionChangeRule]
    3] RuleClassExpert [IchimokuKinkoHyoTenkanKijunCrossoverRule]
    3] RuleClassExpert [ExponentialMovingAverageCrossoverRule]
    3] RuleClassExpert [MovingAverageConvergenceDivergenceSignalLineCrossoverRule]
    3] RuleClassExpert [IchimokuKinkoHyoChikouCrossoverRule]
    3] RuleClassExpert [IchimokuKinkoHyoSenkouASenkouBCrossoverRule]
    3] RuleClassExpert [MovingAverageCrossoverRule]
    3] RuleClassExpert [BollingerBandsLowerMidCrossoverRule]
    3] RuleClassExpert [BollingerBandsUpperMidCrossoverRule]
    3] RuleClassExpert [BollingerBandsLowerUpperCrossoverRule]
    3] RuleClassExpert [RelativeStrengthIndexTrasholdRule]
```

Тренування було запущено із наступними параметрами (див. Рис. 2.10).

```
python main.py --all_rules --timeframe 1h --nleaves 3 --nepochs 20 --fee 0
```

Лог тренування наведений у Додатку З.

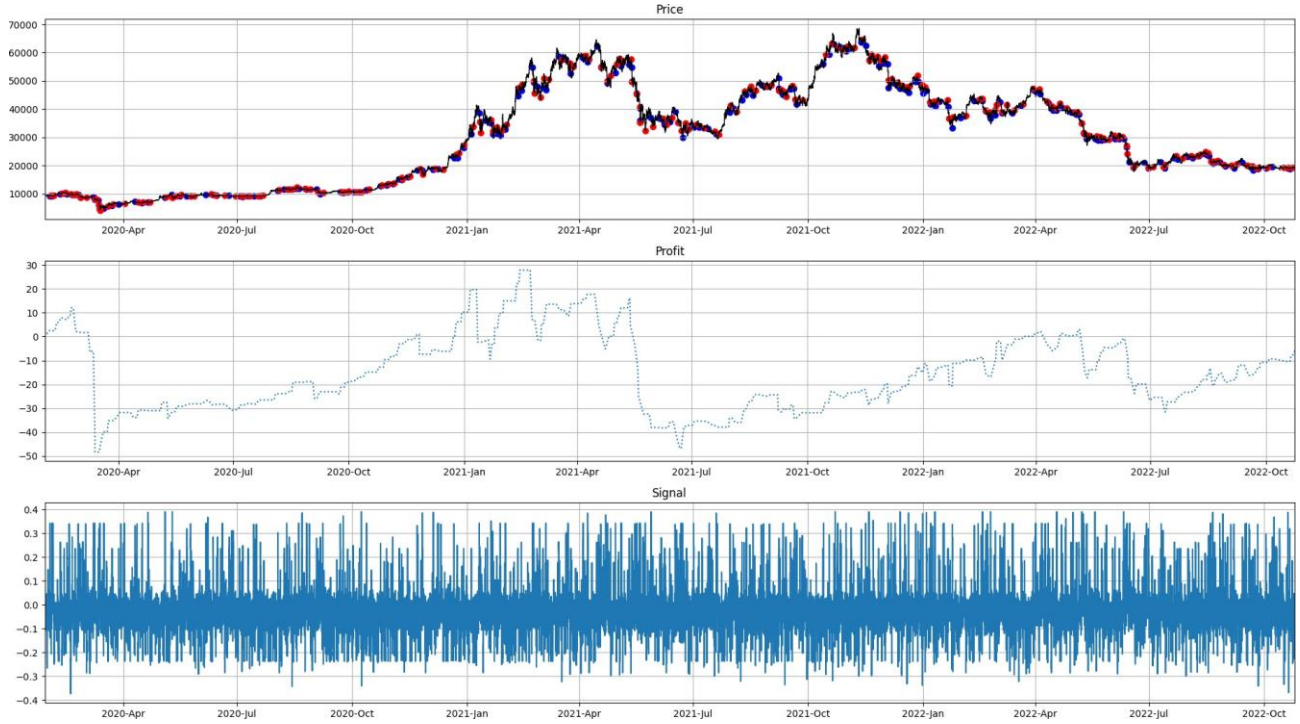

Рисунок 2.10 — Симуляція спрощеної системи на тестових даних.

На тренувальних даних на проміжку 350 днів, система отримала +22% прибутку. У результаті симуляції на 1000 днів на тестових даних, система провела 416 трейдів і отримала -6% прибутку. За той самий період ціна валютної пари змінилась на +200%.

Таким чином спрощена система показала гірший результат, якщо порівнювати із повною системою (-6% порівняно з +158%). Це може свідчити про те, що усі таймфрейми несуть корисну інформацію, комплексний аналіз якої може давати більш точні результати. Саме це і робить повна система, тому вона показує кращі результати.

2.9 Оцінка використання розробленої системи

Ми розглянули систему, яка базується на технічному аналізі, і приймає рішення, які утворюються як лінійна комбінація технічних сигналів. Така система показала свою прибутковість на історичних даних (за 1000 днів ріст ціни склав 200% і система показала 158% прибутку).

Для досягнення цього результату була обрана конфігурація із 11 правил технічного аналізу, на 4 таймфреймах одночасно. Такі параметри показали найкращий результат. Спрощення системи до 11 правил і 1 таймфрейму, погіршило прибутковість системи.

Час навчання та швидкість прогнозування більшою мірою залежить від розміру проміжку часу, за якого відбувається навчання, а також від топології самої системи — більше експертів — довше тренування.

Використання вже готових програмних засобів (наприклад, Binance Api [17]) значно спрощує розробку і зміцнює стабільність використання системи.

2.10 Область застосування і перспективність

Головною областю застосування такої системи може бути область прийняття рішень на фінансових ринках, для отримання допоміжної інформації про конкретний стан ринку, для оцінки стабільності цін. Система може надавати свою оцінку стану, і використовуватися як допоміжний індикатор у процесі прийняття рішень.

Одним із способів застосування такої системи може бути пряме підключення до інтерфейсу, який надає змогу здійснювати покупки та продажі активів, із метою отримання прибутку. Оцінка системи може використовуватися як прямий сигнал до дії на фінансовому ринку. Але це може нести серйозні

фінансові ризики, які пов’язані з усіма системами алгоритмічного трейдингу [25].

2.11 Загальні висновки

Фінансові ринки, з точки зору передбачення ціни, є дуже складним механізмом. Технічний аналіз заснований на аналітиці минулої поведінки на ринку, він не може передбачити майбутніх форс-мажорів на ринку, які дуже часто мають критичний вплив на поведінку ціни. Це є одною з основних причин, яка дуже ускладнює роботу такої системи.

За планом дослідження було дослідити результати аналізу 4-х таймфреймів (5 хв, 1 год, 4 год, 1 день); 11 технічних правил на валютній парі BTC/USDT.

Протестовано дві конфігурації системи (повна і спрощена) і отримані такі результати:

- Повна система надала 158% прибутку, і взагалі показала кращий результат. Одна з причин, що до цього призвела, це більш комплексний аналіз даних — система дивилась одразу на 4 таймфрейми і мала більше контексту.
- Спрощена система надала -6% прибутку, показала гірші результати. Це трапилось через недостатню кількість параметрів системи, і меншу кількість даних до аналізу.

Також перевірено систему на реальних історичних даних, і показано що за допомогою такого аналізу можна отримати дані про стан ринку, які корелюють із розподілом цін.

РОЗДІЛ 3. ІНТЕГРАЦІЯ ЕКСПЕРТНОЇ ІЄРАРХІЧНОЇ СИСТЕМИ З ІНШИМИ СИСТЕМАМИ

3.1 Інтерфейс користувача

На даному етапі розробки, уся система являє собою застосунок, який викликається з командного рядка.

Для налаштування системи є вхідні параметри які впливають на конфігурацію, навчання, серіалізацію і десеріалізацію системи, та ін. Ці параметри сприймаються і оброблюються як флаги застосунку.

Для перегляду можливих параметрів, їх значень, а також опису того, на що вони впливають:

```
python main.py --help
```

Документація формується автоматично у текстовому вигляді:

```
usage: main.py [-h] [--pair PAIR] [--timeframe TIMEFRAME] [--rule RULE] [--nleaves NLEAVES] [--reestimate] [--
load_expert LOAD_EXPERT] [--load_weights] [--nepochs NEPOCHS] [--all_rules] [--best_rules] [--best_timeframes]
[--threshold THRESHOLD]
[--fee FEE]

options:
-h, --help            show this help message and exit
--pair PAIR           Currency pair. (default: BTC/USDT)
--timeframe TIMEFRAME
Trading timeframe. (default: )
--rule RULE           Trading rule. (default: )
--nleaves NLEAVES     Maximum number of RuleExperts. (default: 3)
--reestimate          Optimize expert from scratch. (default: False)
--load_expert LOAD_EXPERT
                      Filename. Load expert from json file. (default: )
--load_weights        Load expert weights from file. (default: False)
--nepochs NEPOCHS     Fit weights for n epochs. (default: 3)
--all_rules           Use all available rules.
--best_rules          Use the rules:
--best_timeframes     Use Timeframes: (1h, 4h, 1d). (default: False)
--threshold THRESHOLD System sensativity. If confidence is above threshold the trade is made. Should be at
range (0, 1) (default: 0.2)
--fee FEE             Trading fee. Should be at range (0, 1) (default: 0.0075)
```

Опис параметрів:

- --pair PAIR — Визначає валютну пару, із якою буде працювати система. Для прикладу в роботі обрана пара за замовченням BTC/USDT.
- --timeframe TIMEFRAME — Визначає період оновлення свічок, можливі значення включають: 1h (1 година), 2h, 4h, 6h, 12h, 1d (1 день), та ін.
- --rule RULE — Правило технічного аналізу, яке буде використовуватися для аналізу і генерації сигналів.
- --nleaves NLEAVES — Максимальна кількість листових експертів, яких може мати система. Листовими експертами вважаються найнижчі експерти в ієрархії. Більша кількість експертів додає більшу кількість параметрів до усієї системи. Значення за замовченням nleaves = 3.
- --reestimate REESTIMATE — Якщо цей параметр зазначений, система почне перенавчання з нуля, намагаючись наново оптимізувати параметри. Після такого навчання можливо повторно використовувати вже знайдені оптимальні параметри. Для цього система серіалізує усі параметри і структуру всієї системи, і зберігає її у файл.
- --load_expert LOAD_EXPERT — Дозволяє завантажити збережену топологію системи, але не обов’язково ваги цієї системи. Це використовується для того, щоб після визначення оптимальної топології системи, можна було окремо оптимізувати лише ваги.
- --load_weights — Дозволяє завантажити серіалізовані ваги системи, щоб використовувати вже натреновану систему для розв'язання задач.
- --nepochs NEPOCHS — Параметр для тренування, визначає скільки разів датасет використовується для тренування кожного експерта.
- --all_rules — Один із готових пресетів правил, які можна використовувати. Цей параметр зазначає, що слід використовувати усі можливі правила технічного аналізу. Наразі їх 11.

- --best_rules — Один із готових пресетів, які можна використовувати. Цей параметр зазначає, що слід використовувати правила, що під час тренування і тестування показали найкращий результат. Наразі їх 6.
- --best_timeframes — Аналогічний пресет до таймфреймів. Містить наступні періоди: 1h (1 год), 4h (4 год), 1d (1 день).
- --threshold — Мінімальна впевненість (відхилення) у значенні сигналу, за якого може здійснюватись транзакція.
- --fee — Комісія за одну транзакцію. Взагалі різні біржі утримують різну комісію, цей параметр допомагає налаштувати симуляцію під конкретні умови.

Таким чином, як приклад, можна розглянути запуск системи, яка буде наново перенавчена, буде використовувати кращих експертів, таймфрейм 1 год, буде мати 3 листових експерти, тренуватися 10 епох, із комісією 0:

```
python main.py \
    --reestimate \
    --best_rules \
    --timeframe 1h \
    --nleaves 3 \
    --nepochs 10 \
    --fee 0
```

3.2 Тестування

Система має досить багато компонент. Для певності у правильності роботи системи для компонент були написані модульні тести. Основна робота системи, це обрахунок статистичної інформації, тому таких тестів цілком досить. Для прикладу можна взяти пакет Rolling, який використовуються для визначення статистик на потоці даних із фіксованим розміром вікна [26]:

Для прикладу декілька класів з цього пакету:

```
class BaseRollingWindow:
    def___init__(self, length: int = None, *, enqueueing: bool = True):
        self.length = length
        if enqueueing:
```

```
                self._queue = collections.deque(maxlen=self.length)
            self._state = 0

        def append(self, val: float):
            raise NotImplemented()

        def get_state(self):
            return self._state


    class Average(BaseRollingWindow):
        def __init__(self, length):
            super().__init__(length)

        def append(self, val: float):
            if len(self._queue) >= self.length:
                self._state -= self._queue.popleft() / self.length
            self._queue.append(val)
            self._state += val / self.length


    class Min(BaseRollingWindow):
        def __init__(self, length):
            super()._init_(length)
            self._queue.append((-self.length, None))
            self.time = 0

        def append(self, val: float):
            if self._queue[0][0] == self.time - self.length:
                self._queue.popleft()
            while self._queue and self._queue[-1][1] >= val:
                self._queue.pop()
            self._queue.append((self.time, val))
            self._state = self._queue[0][1]
            self.time += 1


    class Max(BaseRollingWindow):
        def __init__(self, length):
            super()._init_(length)
            self._queue.append((-self.length, None))
            self.time = 0

        def append(self, val: float):
            if self._queue[0][0] == self.time - self.length:
                self._queue.popleft()
            while self._queue and self._queue[-1][1] <= val:
                self._queue.pop()
            self._queue.append((self.time, val))
            self._state = self._queue[0][1]
            self.time += 1
```

Модульні тести для пакету Rolling:

```
class TestRolling(unittest.TestCase):
    def setUp(self) -> None:
        self.seq = [1, 3, 5, 4, 7]

    def test_max(self):
        mx = Max(length=3)
        expected = [1, 3, 5, 5, 7]
        for a, exp in zip(self.seq, expected):
            mx.append(a)
```

```
            self.assertEqual(mx.get_state(), exp)

    def test_min(self):
        mx = Min(length=3)
        expected = [1, 1, 1, 3, 4]
        for a, exp in zip(self.seq, expected):
            mx.append(a)
            self.assertEqual(mx.get_state(), exp)

    def test_avg(self):
        mx = Average(length=3)
        expected = [1/3, 4/3, 3, 4]
        for a, exp in zip(self.seq, expected):
            mx.append(a)
            self.assertEqual(mx.get_state(), exp)

    def test_sum(self):
        mx = Sum(length=3)
        expected = [1, 4, 9, 12, 16]
        for a, exp in zip(self.seq, expected):
            mx.append(a)
            self.assertEqual(mx.get_state(), exp)
```

Для того, щоб запустити автоматичні тести:

```
python -m unittest -vb
```

3.3 Процес впровадження системи

3.3.1 Налаштування віртуальної середи

Для кращої ізоляції проєкту, версії Python, а також бібліотек, що використовуються, використано інструмент ізоляції віртуальної середи Conda [27]. Такий підхід має декілька переваг:

- Здатність ефективно керувати залежностями між пакетами Python. Залежності - це бібліотеки або модулі, які проєкт використовує. Conda дозволяє встановлювати, оновлювати та видаляти різні версії пакетів, що робить керування залежностями простішим і безпечним.
- Можливість ізоляції проєкту. Це означає, що користувач може мати різні версії Python та пакетів для різних проєктів без взаємного конфлікту.

Ізоляція допомагає уникнути проблем зі сумісністю та зберігає проєкти чистими та організованими.

- Здатність створювати портативні середовища, які можна легко переносити між різними системами. Це надає можливість легко поділитися проєктом з іншими розробниками або виконувати його на різних системах без необхідності налаштовувати середовище наново.

Для початку роботи потрібно створити середу та встановити усі залежності:

```
conda env create --name myenv -f environment.yml
conda env update --name myenv--file local.yml --prune
```

Для того, щоб активувати або деактивувати середу:

```
conda activate —name myenv
conda deactivate
```

3.3.2 Налаштування змінних середи

Для того, щоб отримати доступ до інформації про ціну валютних пар, використовується Binance Api. Для його використання необхідно створити акаунт на цій біржі, і отримати Api ключ, який буде використовуватися для доступу до даних. Цей ключ надає змогу програмно отримувати доступ до акаунту, тому поводитись з ним варто обережно.

Для доступу до даних вистачить Read only ключа, він надасть змогу зчитувати дані, але не надасть змогу вносити зміни.

Щоб не записувати цей ключ у програмний код, він зберігається у файлі із змінними середи .env.

Для запуску тренування необхідно створити файл .env, у який будуть записані дві змінні:

- READONLY_API_KEY = ...

- READONLY_SECRET_KEY = ...

3.3.3 Завантаження даних для тренування

Для того, щоб отримати нові дані для тренування для конкретної валютної пари, можна скористуватись розробленим скриптом fetch_data.py. Для завантаження можна вказати усі необхідні параметри:

- Назва валютної пари.
- Список таймфреймів.
- Період за який завантажити дані.

Для доступу і завантаження цих даних використовується Binance Api.

Приклад завантаження даних:

```
python fetch_data.py

writing BTCUSDT | 1d
writing BTCUSDT | 4h
writing BTCUSDT | 1h
writing BTCUSDT | 5m
```

Дані завантажуються у директорію ./data у форматі csv (див. Рис. 3.1). Дані містять статистичну інформацію про ціну валютної пари:

- Час початку свічки.
- Ціна за свічку (відкриття, максимальна, мінімальна, закриття).
- Об’єм торгівлі.
- Час закриття свічки.
- Об’єм в залежній валюті.
- Кількість обмінів.
- Та ін.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Open time	Open	High	Low	Close	Volume	Close time	Quote asset volume	Number of trades	Taker buy base asset volume	Taker buy quote asset volume	Ignore
2	1559174400000	8646.5	9074.26	8005	8269.54	70379.9985211559260799999		603312537.380843	617341	34577.310598	296613619.219047	0
3	1559260800000	8267.1	8594	8108.5	8555	44727.491621559347199999		373629673.919014	416525	23391.68846	195418655.053399	0
4	1559347200000	8555	8626	8442.36	8544.07	31868.2341571559433599999		272098436.340696	348375	16922.662425	144486129.118262	0
5	1559433600000	8545.1	8814.78	8524	8725.98	27835.1332651559519999999		241414596.719505	299353	14585.886999	126530020.383437	0
6	1559520000000	8726	8800.95	8080.8	8115.82	45692.9651041559606399999		387155494.973049	431252	22514.316859	190882595.703443	0
7	1559606400000	8115.66	8115.66	7481.02	7687.03	74143.9489411559692799999		582173398.325797	645913	36776.597131	288822568.04322	0
8	1559692800000	7687.04	7896.7	7572.78	7776.5	48679.6564551559779199999		377635818.00441	426120	26296.533959	204082647.876733	0
9	1559779200000	7778.08	7868.13	7444.58	7786.7	36624.1187471559865599999		282156410.543789	371485	18877.986383	145537581.791608	0
10	1559865600000	7787.57	8100	7737.49	7980.53	33942.2256581559951999999		269125728.14165	323096	17193.575595	136343077.377057	0
11	1559952000000	7978.94	8044.65	7751	7893.62	22657.3296341560038399999		179047320.02388	250864	11795.097294	93237148.4745482	0
12	1560038400000	7895.28	7935	7506.66	7628.13	31568.4651571560124799999		243970971.024434	322175	16234.631275	125591062.039171	0
13	1560124800000	7627.57	8020	7511	7982.75	36756.0784681560211199999		287125339.970114	387179	19890.868561	155461282.831393	0
14	1560211200000	7981	8010	7692.23	7884.9	30334.9994271560297599999		238031771.517473	309059	15793.772451	124001190.82079	0
15	1560297600000	7884.9	8200	7788.99	8127.64	41597.0826221560383999999		333291726.646	361980	22826.901026	182925127.467565	0

Рисунок 3.1 — Формат даних для тренування.

3.3.4 Збереження параметрів системи

У моделях машинного навчання часто виникає задача зберігати натреновану модель для подальшого використання.

Серіалізація — це процес перетворення об'єктів або даних у послідовність байтів або даних іншого вигляду, який може бути збережений в файлі, переданий через мережу або збережений в пам'яті для подальшого відтворення. Цей процес дозволяє зберегти стан об'єкта та його дані, щоб їх можна було використовувати в майбутньому або передавати між різними системами.

Взагалі серіалізація використовується для декількох цілей:

- Для зберігання даних у структурованому форматі, наприклад, у файлі або у базі даних. Це дозволяє зберегти стан об'єкта для подальшого використання.
- Для передачі через даних через мережу або між різними системами, перетворюючи їх у формат, що може бути переданий як послідовність байтів. Це дозволяє обмінюватися даними між різними платформами та мовами програмування.
- Для кешування обчислених даних або результатів операцій, що дозволяє покращити продуктивність шляхом уникнення повторного обчислення.

Збережені серіалізовані дані можуть бути використані при наступних викликах програми без потреби повторного обчислення.

Для експертної системи виникає необхідність зберегти її структуру (ієрархію, кількість і типи експертів) а також її ваги (параметри).

Для цієї задачі доречно використати серіалізацію даних у файл, щоб при наступному запуску системи можна було відновити структуру і ваги системи. Для прикладу структура зберігається у JSON форматі:

```
{
    "name": "TimeFrameExpert",
    "parameters": {
        "timeframe": "4h"
    },
    "inner experts": [
        {
            "name": "RuleClassExpert",
            "parameters": {
                "rule": "MovingAverageCrossoverRule"
            },
            "inner experts": [
                {
                    "name": "RuleExpert",
                    "parameters": {
                        "rule": {
                            "name": "MovingAverageCrossoverRule",
                            "parameters": {
                                "patience": 1
                            }
                        },
                        "indicators": [
                            {
                                "name": "MovingAverageIndicator",
                                "parameters": {
                                    "length": 179,
                                    "source": "Close"
                                }
                            }, ...
                        ], ...
                    }
                }, ...
            ], ...
        }, ...
    ]
}
```

Для збереження результатів тренування і оцінки кожного окремого експерта додаються додаткові поля до кожного експерта. За цими даними, далі можна відрізати неефективних експертів, зменшити систему, і підвищити

результативність:


```
"estimation": {
    "profit": 2.8020770204706196,
    "ntrades": 4,
    "fitness": 2.8020770204706196
}
```


Ваги системи являють собою важливість і впевненість кожного експерта. Це параметри моделі які підлягають навчанню і оптимізації. Вони зберігаються і бінарному форматі за допомогою бібліотеки Pickle.

3.4 Загальні висновки

Загальні висновки щодо отриманих результатів:

- Система забезпечує онлайн роботу з постійно оновлюваними даними.
- Існує можливість конфігурування системи.
- Реалізована можливість тестування алгоритмів на історичних даних.
- Швидкість прийняття рішень в системі є задовільною.
- Система є надійною та стабільною.
- Забезпечена належна документація.

Отримані результати можуть бути успішно впроваджені на будь-яку систему. Результати роботи можуть бути використані для реального трейдингу на фінансових ринках, допомагаючи приймати рішення на основі аналізу живих та історичних даних, забезпечуючи швидкість, надійність та можливість конфігурації системи під конкретні стратегії.

ВИСНОВКИ

В роботі розглянуто основні процеси фінансового ринку, відомі системи алгоритмічного трейдингу, криптовалюта як цінність, що досліджується. Вивчено попередні підходи до аналізу ринку, а також розроблено нову систему, яка бере до уваги переваги і недоліки аналогів. За допомогою цієї системи автоматизовано процес прийняття рішень.

В першому розділі розглянуті методи технічного аналізу та деяких алгоритмів машинного навчання.

В другому розділі розглянуті питання практичної реалізації, алгоритми, архітектура, допоміжні підсистеми, і усі інші деталі реалізації.

В третьому розділі розглянуто аспекти використання, тестування, налаштування і практичної інтеграції системи.

Практична значущість результатів полягає у можливості використання розроблених алгоритмів трейдингу в реальних умовах фінансових ринків. Це може сприяти автоматизації торговельного процесу, зниженню впливу емоційних факторів та покращенню результативності трейдера.

На основі отриманих результатів можливо вдосконалення шляхом впровадження додаткових правил та стратегій для зменшення ризику. Також можна спробувати розширити систему, інтегрувати її з іншими методами аналізу для більш комплексного підходу.

Виявлено позитивні та негативні аспекти застосування алгоритмічного трейдингу. З одного боку, було досягнуто непогані результати та здатності до адаптації в ринкових умовах. З іншого боку, спостерігалося певне коливання результатів та ризик непередбачуваності в деяких ситуаціях.

Галузі використання алгоритмічного трейдингу можуть охоплювати фінансові установи, інвестиційні фонди, приватних трейдерів та індивідуальних інвесторів. Алгоритмічний трейдинг може бути застосований на різних ринках, включаючи фондовий ринок, валютний ринок та ринок товарів.

Інновація цієї роботи полягає в запровадженні ієрархічного аналізу даних для отримання зваженої картини подій, що відбуваються на фінансовому ринку.

Подальші перспективи досліджень та впровадження включають розширення аналізованих стратегій трейдингу, розробку алгоритмів з машинним навчанням та глибоким навчанням, а також вдосконалення методів оцінки ризику та виявлення ринкових залежностей. Крім того, рекомендовано проведення подальших досліджень з метою порівняння алгоритмічного трейдингу з традиційними методами торгівлі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Financial Markets: Role in the Economy, Importance, Types, and Examples [Електронний ресурс] — Режим доступу до ресурсу: https://www.investopedia.com/terms/f/financial-market.asp

2. What Is a Trader? [Електронний ресурс] — Режим доступу до ресурсу: https://www.investopedia.com/terms/t/trader.asp

3. Investing vs. Trading: What's the Difference? [Електронний ресурс] — Режим доступу до ресурсу: https://www.investopedia.com/ask/answers/12/difference-investing-trading.asp

4. Fundamental Analysis: Principles, Types, and How to Use It [Електронний ресурс] — Режим доступу до ресурсу: https://www.investopedia.com/terms/f/fundamentalanalysis.asp

5. Technical Analysis: What It Is and How to Use It in Investing [Електронний ресурс] — Режим доступу до ресурсу: https://www.investopedia.com/terms/t/technicalanalysis.asp

6. Ning Lu. A Machine Learning Approach to Automated Trading. — 2016

7. Krollner, Bjoern, Bruce Vanstone, Gavin Finnie. Financial time series forecasting with machine learningtechniques: A survey. — 2010

8. T. H. Nguyen, A. T. Nguyen. Algorithmic trading of cryptocurrency based on Twitter sentiment analysis. — 2020

9. J. Moody та M. Saffell. A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem. — 2017

10. K. Kim та J. Park. Bitcoin Trading Strategy: Statistical Arbitrage. — 2018

11. Y. Zhang, J. Jia, J. Dai. An algorithmic trading model for bitcoin. — 2017

12. Algorithmic trading [Електронний ресурс] — Режим доступу до ресурсу:

https://en.wikipedia.org/wiki/Algorithmic_trading

13. CRY: Cryptocurrency trading system [Електронний ресурс] — Режим доступу до ресурсу: https://github.com/9rib/graduation-qualification-work

14. Python [Електронний ресурс] — Режим доступу до ресурсу: https://www.python.org/

15. Numpy [Електронний ресурс] — Режим доступу до ресурсу: https://numpy.org/

16. SciPy [Електронний ресурс] — Режим доступу до ресурсу: https://scipy.org/

17. Binance API [Електронний ресурс] — Режим доступу до ресурсу: https://www.binance.com/en/binance-api

18. Pandas [Електронний ресурс] — Режим доступу до ресурсу: https://github.com/pandas-dev/pandas

19. Pickle [Електронний ресурс] — Режим доступу до ресурсу: https://docs.python.org/3/library/pickle.html

20. Moving average crossover [Електронний ресурс] — Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Moving_average_crossover

21. Relative Strength Index [Електронний ресурс] — Режим доступу до ресурсу: https://www.investopedia.com/terms/r/rsi.asp

22. Moving average convergence/divergence [Електронний ресурс] — Режим доступу до ресурсу: https://www.investopedia.com/terms/m/macd.asp

23. Ichimoku Charts in Forex Trading [Електронний ресурс] — Режим доступу до ресурсу: https://www.investopedia.com/articles/forex/06/ichimoku.asp

24. Bollinger Bands [Електронний ресурс] — Режим доступу до ресурсу: https://www.investopedia.com/trading/using-bollinger-bands-to-gauge-trends

25. Past Performance Is Not Indicative Of Future Results [Електронний ресурс] — Режим доступу до ресурсу:

https://www.forbes.com/sites/johnbrown/2016/09/29/past-performance-is-not-indicative-of-future-results

26. Moving average [Електронний ресурс] — Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Moving_average

27. Conda [Електронний ресурс] — Режим доступу до ресурсу: https://docs.conda.io/en/latest/

ДОДАТКИ

Додаток 1. Лістинг програми

main.py

```
from trainer import Trainer
from config import Config

if__name__ == "__main__":

    cfg = Config.from_args()
    trainer = Trainer()
    trader = trainer.construct_trader(cfg)
trainer.simulate pair trader(trader, 1000, display=True)
```

experts.py

```
# ==================== Omited ====================

class BaseExpert:
    """Base Expert class for decision making."""
    name = 'BaseExpert'

    def__init__(self):
        self._inner_experts = None
        self._weights = None
        self.estimation = {est: None for est in ('fitness', 'profit', 'ntrades')}

    def set_experts(self, experts: Sequence):
        self._inner_experts = experts

    def set_data(self, data: data.DataMaintainer):
        for expert in self._inner_experts:
            expert.set_data(data)

    def set_weights(self, weights: np.array = None, recursive=False):
        if hasattr(self, '_inner_experts'):
            if weights is not None:
                self._weights = weights
            else:
                self._weights = np.ones(len(self._inner_experts))
            if recursive:
                for expert in self._inner_experts:
                    expert.set_weights(None, recursive=True)

    def get_weights(self):
        return self._weights

    def save_weights(self, filename='weights', *, write=True):
        if hasattr(self, '_weights'):
            if isinstance(self, RuleClassExpert):
                inner = []
            else:
                inner = [exp.save_weights(write=False) for exp in self._inner_experts]
            weights = [self.get_weights(), inner]
```

```
            if write:
                pickle.dump(weights, open(filename, 'wb'))
            return weights

    def load_weights(self, filename='weights', *, weights=None):
        if weights is None:
            weights, inner = pickle.load(open(filename, 'rb'))
        else:
            weights, inner = weights
        self.set_weights(weights)
        for exp, weights in zip(self._inner_experts, inner):
            exp.load_weights(weights=weights)

    def estimate(self):
        estimations = np.array([expert.estimate() for expert in self._inner_experts])
        return estimations @ softmax(self._weights)

    def update(self):
        for expert in self._inner_experts:
            expert.update()

    def show(self, indentation=0, overview=True):
        total = self.count_total_inner_experts()
        if overview:
            print(f'{" " * indentation} {total:>5}] {self.name}')
            if not isinstance(self, RuleClassExpert):
                for expert in self._inner_experts:
                    expert.show(indentation + 10, overview=overview)
            return total
        else:
            print(f'{" " * indentation} {total:>5}] {self.name}')
            for expert in self._inner_experts:
                expert.show(indentation + 10, overview=overview)

    def count_total_inner_experts(self):
        if hasattr(self, '_inner_experts'):
            return sum(expert.count_total_inner_experts() for expert in self._inner_experts)
        else:
            return 1

# ==================== Omited ====================
```

rules.py

```
# ==================== Omited ====================

class MovingAverageCrossoverRule(BaseCrossoverRule):
    name = 'MACrossover'

    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self._cross = CrossoverState()

    def compatible(self, slow: indicators.MovingAverageIndicator,
                         fast: indicators.MovingAverageIndicator):
        return slow.length > fast.length

    def decide(self, slow: indicators.MovingAverageIndicator,
                     fast: indicators.MovingAverageIndicator):
        buy, sell = self._cross.update(fast.get_state(), slow.get_state())
        return self.signal(buy, sell)
```

```
class RelativeStrengthIndexTrasholdRule(BaseTrasholdRule):
    name = 'RSITrashold'

    def __init__(self, lower: float, upper: float, **kwargs):
        super()._init_(lower, upper, **kwargs)
        self._lower_cross = CrossoverState()
        self._upper_cross = CrossoverState()

    def decide(self, rsi: indicators.RelativeStrengthIndexIndicator):
        val = rsi.get_state()
        buy, _ = self._lower_cross.update(self._lower, val)
        sell, _ = self._upper_cross.update(val, self._upper)
        return self.signal(buy, sell, instant=False)

    def get_parameters(self):
        return {'lower': self._lower,
                'upper': self._upper,
                'patience': self._patience}

# ==================== Omited ====================
```

indicators.py

```
# ==================== Omited ====================

class BaseIndicator:
    name = 'Base Indicator'

    def ___init__(self):
        self.update_hash = None

    def set_data(self, data: data.DataMaintainer):
        self._data = data
        self.init_state()

    def get_parameters(self):
        raise NotImplementedError()

    def init_state(self):
        """Calculate initial state"""
        raise NotImplementedError()

    def is_updated(self):
        return self.update_hash == self._data.update_hash

    def update(self, val: float = None):
        raise NotImplementedError()

# ==================== Omited ====================
```

Додаток 2. Лог тренування повної системи

```
            RuleClassExpert [IchimokuKinkoHyoTenkanKijunCrossoverRule]
              [  20 / 20  ] fitness: -15.08 profit: -15.08 %
            RuleClassExpert [TripleExponentialDirectionChangeRule]
              [  20 / 20  ] fitness: -24.48 profit: -24.48 %
            RuleClassExpert [IchimokuKinkoHyoSenkouASenkouBCrossoverRule]
              [  20 / 20  ] fitness: -15.43 profit: -15.43 %
            RuleClassExpert [MovingAverageCrossoverRule]
              [ 20 / 20 ] fitness: -9.80 profit: -9.80 %
            RuleClassExpert [ExponentialMovingAverageCrossoverRule]
              [  20 / 20  ] fitness: -9.34 profit: -9.34 %
            RuleClassExpert [MovingAverageConvergenceDivergenceSignalLineCrossoverRule]
              [  20 / 20  ] fitness: -16.24 profit: -16.24 %
            RuleClassExpert [IchimokuKinkoHyoChikouCrossoverRule]
              [  20 / 20  ] fitness: -27.96 profit: -27.96 %
            RuleClassExpert [BollingerBandsUpperMidCrossoverRule]
              [  20 / 20  ] fitness: 6.57 profit: 6.57 %
            RuleClassExpert [BollingerBandsLowerUpperCrossoverRule]
              [  20 / 20  ] fitness: -24.44 profit: -24.44 %
            RuleClassExpert [BollingerBandsLowerMidCrossoverRule]
              [  20 / 20  ] fitness: 2.33 profit: 2.33 %
            RuleClassExpert [RelativeStrengthIndexTrasholdRule]
              [  20 / 20  ] fitness: -7.04 profit: -7.04 %
        TimeFrameExpert [4h]
          [ 20 / 20  ] fitness: 22.47 profit: 22.47 %
            RuleClassExpert [TripleExponentialDirectionChangeRule]
              [ 20 / 20 ] fitness: -27.08 profit: -27.08 %
            RuleClassExpert [IchimokuKinkoHyoTenkanKijunCrossoverRule]
              [ 20 / 20 ] fitness: -3.59 profit: -3.59 %
            RuleClassExpert [ExponentialMovingAverageCrossoverRule]
              [  20 / 20  ] fitness: -26.40 profit: -26.40 %
            RuleClassExpert [MovingAverageConvergenceDivergenceSignalLineCrossoverRule]
              [  20 / 20  ] fitness: -32.63 profit: -32.63 %
            RuleClassExpert [IchimokuKinkoHyoChikouCrossoverRule]
              [  20 / 20  ] fitness: -15.60 profit: -15.60 %
            RuleClassExpert [MovingAverageCrossoverRule]
              [ 20 / 20 ] fitness: -39.24 profit: -39.24 %
            RuleClassExpert [IchimokuKinkoHyoSenkouASenkouBCrossoverRule]
              [ 20 / 20 ] fitness: -25.96 profit: -25.96 %
            RuleClassExpert [BollingerBandsLowerMidCrossoverRule]
              [ 20 / 20 ] fitness: 0.62 profit: 0.62 %
            RuleClassExpert [BollingerBandsUpperMidCrossoverRule]
              [ 20 / 20 ] fitness: 13.13 profit: 13.13 %
            RuleClassExpert [BollingerBandsLowerUpperCrossoverRule]
              [ 20 / 20 ] fitness: 4.76 profit: 4.76 %
            RuleClassExpert [RelativeStrengthIndexTrasholdRule]
              [  20 / 20  ] fitness: -35.77 profit: -35.77 %
        TimeFrameExpert [1h]
          [ 20 / 20  ] fitness: 58.74 profit: 58.74 %
            RuleClassExpert [TripleExponentialDirectionChangeRule]
              [ 20 / 20 ] fitness: -3.99 profit: -3.99 %
            RuleClassExpert [IchimokuKinkoHyoTenkanKijunCrossoverRule]
              [ 20 / 20 ] fitness: -5.47 profit: -5.47 %
            RuleClassExpert [ExponentialMovingAverageCrossoverRule]
              [  20 / 20  ] fitness: -9.43 profit: -9.43 %
            RuleClassExpert [MovingAverageConvergenceDivergenceSignalLineCrossoverRule]
              [  20 / 20  ] fitness: -9.28 profit: -9.28 %
            RuleClassExpert [MovingAverageCrossoverRule]
              [ 20 / 20 ] fitness: -6.62 profit: -6.62 %
            RuleClassExpert [IchimokuKinkoHyoChikouCrossoverRule]
```

```
              [ 20 / 20 ] fitness: -14.95 profit: -14.95 %
            RuleClassExpert [IchimokuKinkoHyoSenkouASenkouBCrossoverRule]
              [ 20 / 20 ] fitness: -10.92 profit: -10.92 %
            RuleClassExpert [BollingerBandsUpperMidCrossoverRule]
              [ 20 / 20 ] fitness: -9.25 profit: -9.25 %
            RuleClassExpert [BollingerBandsLowerMidCrossoverRule]
              [ 20 / 20 ] fitness: -2.14 profit: -2.14 %
            RuleClassExpert [BollingerBandsLowerUpperCrossoverRule]
              [ 20 / 20 ] fitness: -6.86 profit: -6.86 %
            RuleClassExpert [RelativeStrengthIndexTrasholdRule]
              [  20 / 20  ] fitness: -7.71 profit: -7.71 %
      TimeFrameExpert [5m]
        [ 20 / 20  ] fitness: 17.08 profit: 17.08 %
            RuleClassExpert [TripleExponentialDirectionChangeRule]
              [ 20 / 20 ] fitness: -12.17 profit: -12.17 %
            RuleClassExpert [BollingerBandsLowerMidCrossoverRule]
              [ 20 / 20 ] fitness: 10.18 profit: 10.18 %
            RuleClassExpert [BollingerBandsUpperMidCrossoverRule]
              [ 20 / 20 ] fitness: -15.23 profit: -15.23 %
            RuleClassExpert [BollingerBandsLowerUpperCrossoverRule]
              [  20 / 20  ] fitness: -11.98 profit: -11.98 %
            RuleClassExpert [MovingAverageConvergenceDivergenceSignalLineCrossoverRule]
              [  20 / 20  ] fitness: -42.56 profit: -42.56 %
            RuleClassExpert [MovingAverageCrossoverRule]
              [ 20 / 20 ] fitness: -3.67 profit: -3.67 %
            RuleClassExpert [ExponentialMovingAverageCrossoverRule]
              [ 20 / 20 ] fitness: -20.60 profit: -20.60 %
            RuleClassExpert [IchimokuKinkoHyoTenkanKijunCrossoverRule]
              [  20 / 20  ] fitness: 15.30 profit: 15.30 %
            RuleClassExpert [IchimokuKinkoHyoSenkouASenkouBCrossoverRule]
              [ 20 / 20 ] fitness: -3.08 profit: -3.08 %
            RuleClassExpert [IchimokuKinkoHyoChikouCrossoverRule]
              [ 20 / 20 ] fitness: -32.72 profit: -32.72 %
            RuleClassExpert [RelativeStrengthIndexTrasholdRule]
              [  20 / 20  ] fitness: 4.65 profit: 4.65 %
      TimeFrameExpert [1d]
        [  20 / 20  ] fitness: 29.94 profit: 29.94 %
PairExpert [BTC/USDT]
  [  20 / 20  ] fitness: -8.19 profit: -8.19 %

Simulation:
  60 trades, 158% profit
```

Додаток 3. Лог тренування спрощеної системи

```
        RuleClassExpert [TripleExponentialDirectionChangeRule]
          [  20 / 20  ] fitness: -42.31 profit: -42.31 %
        RuleClassExpert [IchimokuKinkoHyoTenkanKijunCrossoverRule]
          [  20 / 20  ] fitness: -6.77 profit: -6.77 %
        RuleClassExpert [ExponentialMovingAverageCrossoverRule]
          [  20 / 20  ] fitness: -39.07 profit: -39.07 %
        RuleClassExpert [MovingAverageConvergenceDivergenceSignalLineCrossoverRule]
          [  20 / 20  ] fitness: -32.63 profit: -32.63 %
        RuleClassExpert [IchimokuKinkoHyoChikouCrossoverRule]
          [  20 / 20  ] fitness: -15.42 profit: -15.42 %
        RuleClassExpert [IchimokuKinkoHyoSenkouASenkouBCrossoverRule]
          [  20 / 20  ] fitness: -29.13 profit: -29.13 %
        RuleClassExpert [MovingAverageCrossoverRule]
          [ 20 / 20 ] fitness: -18.88 profit: -18.88 %
        RuleClassExpert [BollingerBandsLowerMidCrossoverRule]
          [ 20 / 20 ] fitness: -12.69 profit: -12.69 %
        RuleClassExpert [BollingerBandsUpperMidCrossoverRule]
          [ 20 / 20 ] fitness: 13.13 profit: 13.13 %
        RuleClassExpert [BollingerBandsLowerUpperCrossoverRule]
          [ 20 / 20 ] fitness: -1.96 profit: -1.96 %
        RuleClassExpert [RelativeStrengthIndexTrasholdRule]
          [  20 / 20  ] fitness: -41.08 profit: -41.08 %
    TimeFrameExpert [1h]
      [  20 / 20  ] fitness: 22.49 profit: 22.49 %

    416 trades, -6% profit
```