

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет водного господарства та
природокористування Навчально-науковий інститут кібернетики,
інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики**

Допущено до захисту:

**Завідувач кафедри
комп'ютерних технологій та
економічної кібернетики
д. е. н., проф. П. М. Грицюк**

«_____» _____ 2026 р.

**КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬО-КВАЛІФІКАЦІЙНОГО РІВНЯ
«БАКАЛАВР»**

**Інформаційна система для трейдингу на основі методів
інтелектуального аналізу фінансових даних**

Виконав:

**здобувач вищої освіти за ОПП
«Інформаційні системи та технології»
спеціальності 126 «Інформаційні системи та
технології», групи ІСТ-41
Березніцький Євгеній Олегович**

Керівник:

д.т.н., доцент Барановський С. В.

Рецензент:

д. е. н., проф. П. М. Грицюк

Національний університет водного господарства та природокористування
ННІ кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

Освітньо-кваліфікаційний рівень – *бакалавр*
Освітньо-професійна програма «*Інформаційні системи і технології*»
Спеціальність 126 «*Інформаційні системи і технології*»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ д. е. н., проф. П.М. Грицюк

« _____ » _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачу _____ *Березницькому Євгенію Олеговичу*
(прізвище, ім'я, по-батькові)

1. Тема роботи _____ *Інформаційна система для трейдингу на основі інтелектуального аналізу фінансових даних*

керівник роботи _____ *Барановський Сергій Віталійович, докт. техн. наук, доцент*
(прізвище, ім'я, по-батькові, науковий ступінь, вчене звання)

затверджена наказом по університету від “ _____ ” _____ 202__ року № _____

2. Термін здачі студентом закінченої роботи

3. Вихідні дані до роботи _____ *Важливою проблемою трейдингу є необхідність урахування*

численних факторів: технічних індикаторів, новинного фону, статистичних закономірностей, поведінки інших гравців на ринку тощо. Завданням кваліфікаційної роботи є розробка інформаційної системи для підтримки рішень трейдера на основі методів інтелектуального аналізу даних, що дають можливість здійснювати аналіз історичних даних, оцінювати ризики, знаходити приховані закономірності та пропонувати найбільш раціональні дії, а також мінімізувати вплив людського чинника та емоційних рішень.

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

1. *Сучасний стан, перспективи та тенденції розвитку трейдингу*

2. *Огляд та аналіз інформаційних технологій трейдингу*

3. *Аналіз експертних методів та системи аналізу даних*

4. *Особливості застосування технологій нейронних мереж для обробки і аналізу даних*

5. *Програмна реалізація методів збору даних*

6. *Програмна реалізація аналітичного інструментарію обробки даних*

7. *Опис функціональних можливостей та інтерфейсу ІС*

Висновки

5. Перелік графічного матеріалу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання отримав
1.	<i>Барановський С.В., доцент</i>		
2.	<i>Барановський С.В., доцент</i>		
3.	<i>Барановський С.В., доцент</i>		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Терміни виконання етапів роботи	Примітка
1.	<i>Аналіз сучасного стану та перспектив розвитку трейдингу</i>		
		<i>до 20.01.2026</i>	
2.	<i>Аналіз інформаційних технологій підтримки рішень у сфері трейдингу</i>		
		<i>до 20.02.2026</i>	
3.	<i>Аналіз аналітичних методів та інформаційних технологій аналізу даних у сфері трейдингу</i>		
		<i>до 20.03.2026</i>	
4.	<i>Програмна реалізація методів збору даних</i>		
		<i>до 10.04.2026</i>	
5.	<i>Програмна реалізація інструментарію обробки даних</i>		
		<i>до 25.04.2026</i>	
6.	<i>Опис функціональних можливостей та інтерфейсу IT проекту</i>		
		<i>до 25.05.2026</i>	
7.	<i>Оформлення та представлення до захисту бакалаврської роботи</i>		
		<i>до 10.06.2026</i>	

Здобувач вищої освіти _____
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи _____
(підпис) (прізвище та ініціали)

ЗАЯВА
щодо самостійності виконання випускної кваліфікаційної роботи

Я, _____ (ПІБ),
студент(ка) _____ курсу _____ групи _____ ННІ _____

ЗАЯВЛЯЮ:

моя випускна кваліфікаційна робота на тему «_____»
_____»,
яка надається в екзаменаційну комісію із захисту кваліфікаційних робіт зі
спеціальності
«_____»
для захисту, виконана самостійно і не містить плагіату.

Всі запозичення з друкованих та електронних джерел, у тому числі із захищених раніше випускових кваліфікаційних робіт мають відповідні посилання.

Я не використовував(ла) шахрайські методи маніпуляції з текстом (заміна букв, інтервали, мікропробіли, білі знаки, парафрази та ін.).

Інструменти штучного інтелекту використовував(ла) без порушення академічної доброчесності.

Я ознайомлений(а) з чинним Порядком перевірки навчальних, кваліфікаційних, навчально-методичних та наукових робіт на наявність ознак академічного плагіату в НУВГП та Положенням про академічну доброчесність в НУВГП, за яким виявлення плагіату є підставою для відмови в допуску моєї роботи до захисту та застосування відповідних санкцій (академічної відповідальності).

Метадані

ДОКУМЕНТ

Заголовок

Кваліфікаційна робота.docx

Автор

Березніцький Євгеній Олегович

Науковий керівник / Експерт

Березніцький Євгеній Олегович

ІД документа

334334614

ОРГАНІЗАЦІЯ

Назва організації

National University of Water and Environmental Engineering

підрозділ

National University of Water and Environmental Engineering

ЗВІТ

Дата звіту

6/16/2026

Дата редагування

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



10569

Кількість слів



83678

Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		0
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)		69

Джерела

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз

Колір тексту

#	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
---	--	---

АКТ
перевірки випускної кваліфікаційної роботи
автора _____
на рівень запозичень та можливих маніпуляцій з текстом

Відповідно до даних системи StrikePlagiarism файл
« _____ »

містить: ____ % коефіцієнта подібності; ____ % коефіцієнта цитованості;
____ замінених букв; ____ інтервалів; ____ мікропробілів; ____ білих знаків;
____ % ймовірність використання ІІІ.

За результатами перевірки засвідчую, що:
автор кваліфікаційної роботи _____ :

- самостійно виконав(ла) кваліфікаційну роботу;
- коректно посилався(лась) на використані інформаційні джерела;
- в роботі відсутні ознаки академічної недоброчесності.

Керівник кваліфікаційної роботи

підпис

(ПІП)

АНОТАЦІЯ

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ТРЕЙДИНГУ НА ОСНОВІ МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ФІНАНСОВИХ ДАНИХ.

Предметом цієї роботи є розробка інформаційної системи, що допомагає трейдерам приймати рішення за допомогою методів інтелектуального аналізу даних. Об'єктом дослідження обрано процес формування торгових рішень на фінансових ринках. Метою є створення робочого прототипу системи, який підтримує трейдера під час укладання угод завдяки інтеграції аналітичних методів.

Для реалізації цього завдання застосовано комплексний підхід. Було інтегровано експертні системи, що діють на основі чітких правил із використанням індикаторів RSI, MACD, EMA. Додатково, для виявлення історичних закономірностей у динаміці котирувань, використано нейронні мережі архітектур LSTM, GRU та MLP.

Актуальність такої системи зумовлена ускладненням фінансових ринків та зростанням обсягів даних. Використання розробленої системи дозволяє нівелювати вплив емоцій та підвищити результативність торгівлі.

Результатом роботи став створений на мові Python прототип, у якому експертні правила поєднані з можливостями нейронних мереж.

У подальшому планується масштабування цієї системи. Зокрема, розглядається додавання нових ринків (акції, Forex) та впровадження сучасних нейромережових архітектур, таких як Transformer або TCN. Також передбачається інтеграція аналізу новин та соціальних сигналів, що дозволить наблизитися до повної автоматизації торговельного процесу.

ЗМІСТ

ВСТУП.....	9
РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ	13
1.1. Огляд сфери трейдингу	13
1.1.1. Оглядова інформація	13
1.1.2. Види ринків та інструментів.....	15
1.1.3. Типи угод	18
1.1.4. Основні методи аналізу та проблеми прийняття рішень у трейдингу	20
1.2. Інформаційна система для трейдингу	24
1.2.1. Оглядова інформація	25
1.2.2. Види інформаційних систем для трейдингу у трейдингу ..	27
1.2.3. Порівняльний аналіз існуючих інформаційних систем для трейдингу	28
Висновки до розділу 1	31
РОЗДІЛ 2 ОГЛЯД МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ.....	32
2.1. Оглядова інформація	32
2.2 Використання експертних систем	34
2.2.1. Індeksi RSI та MACD.....	35
2.2.2. Експоненційне ковзне середнє (EMA)	37
2.2.3 Об'єм торгів та відкритий інтерес	38
2.2.4. Індекс страху та жадібності	39
2.3. Використання нейронних мереж	39
2.3.1. Мережа типу Long Short-Term Memory.....	41
2.3.2. Мережа типу Gated Recurrent Unit	43
2.3.3. Мережа типу Multi-Layer Perceptron.....	45
2.3.4. Порівняльний аналіз нейронних мереж та вибір оптимальної моделі.....	48
Висновки до розділу 2	49
РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ	51
3.1. Архітектура системи.....	51
3.3. Реалізація аналітичної системи	53
3.4. Практичне використання інформаційної системи.....	55
Висновки до розділу 3	57
ВИСНОВКИ	59
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....	62
ДОДАТОК А. ЛІСТИНГ ПРОГРАМНОГО ДОДАТКУ	64

ВСТУП

Сучасний етап розвитку суспільства характеризується масштабною цифровою трансформацією, що охоплює всі ключові сектори, включаючи охорону здоров'я, логістику та фінансову індустрію. Стрімке зростання обсягів доступних даних ускладнює завдання їх ефективного накопичення, систематизації та аналізу. Перед фахівцями постає проблема своєчасного ухвалення виважених рішень в умовах обмеженого часу та надмірного інформаційного навантаження, що робить методи опрацювання даних ключовим предметом наукових досліджень.

Фінансовий ринок є середовищем із високим рівнем відповідальності за кожен крок, де ефективність операцій напряду залежить від точності аналітичних інструментів. Успішна торговельна діяльність вимагає комплексного врахування технічних показників, статистичних тенденцій, новинних факторів та психології інших ринкових учасників. Оскільки людські можливості в аналізі даних у режимі реального часу є обмеженими, виникає потреба у впровадженні спеціалізованих інформаційних систем, які забезпечують автоматизовану підтримку ухвалення торговельних рішень.

Актуальність кваліфікаційної роботи обґрунтована постійним підвищенням потреби фінансового ринку в інтелектуальних інструментах, що сприяють ухваленню обґрунтованих рішень. Через стрімке ускладнення ринкових механізмів та значне розширення масивів аналітичної інформації, навіть професійні трейдери часто неспроможні своєчасно ідентифікувати всі актуальні тенденції. У зв'язку з цим, високу практичну цінність мають програмні комплекси, здатні до комплексного опрацювання історичних даних, оцінки ризикових факторів та виявлення прихованих закономірностей для формування ефективних торгових стратегій. Експлуатація таких систем мінімізує негативний вплив суб'єктивних психологічних чинників, емоційної нестабільності та неперевіраних інтуїтивних припущень, що зазвичай мінімізує ризики фінансових втрат. Застосування методів інтелектуального

аналізу даних дозволяє сформувати надійну систему підтримки, яка оптимізує аналітичні процеси та забезпечує оперативну реакцію на динамічні зміни ринкового середовища.

Глобальний ринок фінансових технологій (FinTech) демонструє стабільне зростання: за даними аналітичних агентств, обсяг ринку алгоритмічної торгівлі перевищив 15 мільярдів доларів США і продовжує збільшуватися темпами понад 10% щороку. Це свідчить про зростаючий попит на автоматизовані рішення для управління інвестиціями та підтримки торгових рішень. У цьому контексті розробка спеціалізованих інформаційних систем, орієнтованих на потреби індивідуального трейдера, набуває особливої актуальності.

Важливо зазначити, що сучасні інформаційні системи для трейдингу не замінюють людину, а виступають інструментом підтримки прийняття рішень. Вони дозволяють систематизувати аналітичний процес, зменшити час реакції на ринкові зміни та підвищити дисципліну торгівлі через стандартизацію підходів до оцінки ринкової ситуації.

Головною метою кваліфікаційної роботи є проектування та створення інформаційної системи, призначеної для аналітичної підтримки трейдерів шляхом впровадження методів інтелектуального аналізу даних. Функціонал системи передбачає моніторинг ринкових показників, ідентифікацію цінових тенденцій та надання рекомендацій, що підвищують обґрунтованість ухвалення торговельних рішень. Процес прийняття рішень під час здійснення операцій на фінансових ринках є об'єктом даного дослідження, тоді як предметом виступають алгоритми та методи інтелектуальної обробки даних, використані для побудови відповідних програмних комплексів.

До переліку ключових завдань, реалізація яких передбачена у межах кваліфікаційної роботи, належать:

1. Огляд та критичний аналіз сучасних методів і технологій, що застосовуються при проектуванні торговельних інформаційних систем.
2. Аналіз специфіки отримання фінансової інформації з різних джерел та

опрацювання цих даних.

3. Вибір та адаптація моделей машинного навчання, здатних генерувати аналітичні прогнози.
4. Проведення серії тестувань розроблених моделей із використанням ретроспективних даних фінансових ринків.
5. Оцінювання якісних показників роботи системи з використанням математичних метрик ефективності.
6. Методологічну основу дослідження становлять методи системного аналізу, математичної статистики та машинного навчання. У роботі використовуються бібліотеки мови Python (pandas, numpy, TensorFlow/Keras), що забезпечують ефективну обробку великих масивів фінансових даних та побудову прогностичних моделей. Верифікація розроблених моделей проводилася на реальних ринкових даних, отриманих через загальнодоступні API провідних фінансових платформ.

Ключовим підсумком кваліфікаційної роботи є розробка програмного комплексу для трейдингу, що інтегрує сучасні моделі машинного навчання для аналітичного прогнозування та оптимізації процесу ухвалення рішень.

Розроблена система наділена такими характеристиками:

1. Адаптивність до динамічних трансформацій ринкової кон'юнктури.
2. Масштабованість, що дозволяє обробляти значні масиви даних з різних джерел.
3. Формування індивідуалізованих та аргументованих рекомендацій.
4. Технічна готовність до інтеграції з фінансовими API або профільними торговими платформами.

Практична цінність кваліфікаційної роботи підтверджується можливістю впровадження системи у реальні умови фінансового ринку. Програмний продукт є придатним для використання не лише кваліфікованими фахівцями, а й трейдерами-початківцями, забезпечуючи зниження ризиків та підвищення результативності інвестиційних операцій. Отримані результати

створюють підґрунтя для подальших науково-технічних розробок у галузі FinTech, зокрема стосовно вдосконалення автономних торгових алгоритмів, створення систем прогнозування фінансових ризиків та розвитку інтелектуальних засобів підтримки користувачів.

Структура кваліфікаційної роботи відповідає логіці дослідження: перший розділ присвячено аналізу предметної області трейдингу та огляду існуючих систем; другий — теоретичним основам методів інтелектуального аналізу даних; третій — практичній реалізації програмного забезпечення; четвертий — функціонально-вартісному аналізу розробленого продукту.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Огляд сфери трейдингу

1.1.1. Оглядова інформація

Термін «трейдинг» (від англ. *trading* — торгівля) визначає процес систематичного здійснення операцій купівлі-продажу фінансових інструментів, метою якого є отримання доходу від коливань ринкової вартості цих активів. До об'єктів торгівлі належить широкий спектр активів: від корпоративних акцій та валютних пар до криптовалют і товарних деривативів. Трейдинг виступає важливим елементом світової фінансової системи, оскільки він сприяє перерозподілу капіталу, забезпечує ринкову ліквідність та механізми ціноутворення через баланс попиту й пропозиції. Завдяки стрімкому розвитку цифрових технологій та доступності спеціалізованого програмного забезпечення, останніми роками спостерігається значне розширення кола учасників фінансових ринків.

Фундаментальна мета трейдера — придбати актив дешевше та продати дорожче або відкрити коротку позицію і заробити на зниженні ціни. Принципова відмінність трейдингу від довгострокового інвестування полягає у горизонті утримання позиції: трейдери орієнтуються на значно коротші часові рамки — від лічених хвилин до кількох тижнів, а не на роки чи десятиліття.

Залежно від частоти укладання угод та тривалості перебування у позиції, виділяють наступні підходи до торгівлі:

1. Скальпінг (Scalping): стратегія, орієнтована на отримання прибутку від мінімальних цінових змін протягом дуже коротких проміжків часу, зазвичай від декількох секунд до хвилин.
2. Денна торгівля (Intraday): передбачає відкриття та закриття всіх торгових позицій протягом одного робочого дня без перенесення їх на наступну добу.
3. Свінг-трейдинг (Swing trading): стиль середньострокової торгівлі,

розрахований на утримання позиції протягом періоду від кількох днів до тижнів.

4. Інвестування (Investing): стратегія довгострокового володіння активом з метою отримання прибутку від його зростання у тривалій перспективі, що може вимірюватися місяцями або роками.

Кожен із зазначених стилів передбачає специфічний підхід до управління ризиками, різні вимоги до технічного оснащення та психологічної готовності трейдера. Скальпінг потребує блискавичної реакції й мінімальних транзакційних витрат, тоді як інвестиційний стиль вимагає глибокого розуміння фундаментальних чинників і терпіння. Успішна практика в будь-якому з цих стилів неможлива без міцних знань у галузі фінансового аналізу, управління капіталом і психологічної стійкості в умовах постійної ринкової невизначеності.

Серед ключових компетентностей успішного трейдера особливе місце займають: здатність швидко аналізувати великі масиви ринкових даних, строге дотримання правил управління капіталом, психологічна витримка під час збиткових серій і системна торгова дисципліна. Висока динаміка і непередбачуваність фінансових ринків перетворюють трейдинг на середовище підвищеного ризику, де навіть досвідчені учасники можуть зазнавати значних збитків через одне невчасне або емоційно обумовлене рішення.

Ефективність торгової діяльності нерозривно пов'язана з якістю і швидкістю прийняття рішень, які безпосередньо залежать від повноти та своєчасності обробки ринкової інформації. В умовах, коли потоки ринкових даних генеруються щосекунди, людина фізично не здатна охопити і систематизувати весь обсяг значущих сигналів. Саме тому дедалі більшого поширення набувають спеціалізовані програмні рішення, що поєднують алгоритми обробки великих даних, методи машинного навчання та інтелектуальний аналіз ринкових закономірностей. Розробка і застосування подібних інформаційних систем є одними з провідних напрямів сучасних

фінансових технологій.

1.1.2. Види ринків та інструментів

Структуризація фінансових ринків здійснюється на основі наступних критеріїв:

1. Вид базового активу, який виступає об'єктом торгівлі (табл. 1.1).
2. Формат організаційної структури ринку (табл. 1.2).
3. Рівень державного або інституційного регулювання (табл. 1.3).
4. Стадія обігу фінансового інструменту (табл. 1.4).

Найбільш поширеними категоріями є фондовий ринок, валютний ринок, товарний ринок, криптовалютний ринок, ф'ючерсний ринок та опціонний ринок.

Таблиця 1.1

Види ринків, за типом активів, що торгується

Тип ринку	Об'єкти торгівлі
Фондовий	Корпоративні акції, боргові облігації, біржові фонди (ETF)
Валютний	Валютні пари (Forex)
Товарний	Сировинні ресурси, сільськогосподарська продукція
Криптовалютний	Цифрові активи та криптовалюти

Таблиця 1.2

Види ринків, за формою організації

Тип ринку	Характеристика
Біржовий	Торговельні операції централізовано проходять через єдину біржову платформу

Позабіржовий (OTC)	Укладання угод відбувається безпосередньо між контрагентами, обминаючи біржу
--------------------	--

Таблиця 1.3

Види ринків, за ступенем регуляції

Тип ринку	Особливості функціонування
Регульований	Діяльність суворо контролюється державними установами (наприклад, NASDAQ, LSE)
Нерегульований	Функціонує поза офіційною правовою юрисдикцією або децентралізовано (DEX, DeFi)

Таблиця 1.4

Види ринків, за етапом обігу активів

Тип ринку	Специфіка обігу
Первинний	Емісія та первинне розміщення нових активів (процеси IPO, ICO)
Вторинний	Здійснення торгових операцій із раніше випущеними активами між учасниками ринку

Також сучасний трейдинг оперує широким спектром фінансових інструментів, що відрізняються механізмом ціноутворення, рівнем ризику і функціональним призначенням. Ключовим критерієм класифікації є характер виконання угод і природа активу: виокремлюють базові (первинні) інструменти — ті, вартість яких визначається безпосередньо ринком, — та похідні (деривативи), чия вартість залежить від ціни базового активу.

Базові інструменти — це активи, що функціонують автономно і є самостійним об'єктом торгівлі. Їхня ринкова вартість визначається виключно

попитом і пропозицією на них самих, без прив'язки до інших продуктів. До них належать:

1. Акції — частки власності в компаніях.
2. Облігації — боргові цінні папери.
3. Валюта — грошові одиниці різних країн.
4. Сировина — нафта, золото, агропродукція тощо.
5. Криптовалюти — цифрові активи.
6. Фондові індекси — показники динаміки кошика акцій.

Похідні інструменти (деривативи) — це контракти, ціна яких є похідною від вартості базового активу. Вони використовуються для хеджування ризиків, спекулятивних операцій або отримання прибутку від зміни ціни активу без необхідності володіти ним фізично.

Основні види деривативів:

1. Ф'ючерс — стандартизований біржовий контракт, який зобов'язує сторони здійснити купівлю або продаж активу за фіксованою ціною у визначений момент у майбутньому. Основні характеристики: стандартизація умов (обсяг, дата, вид активу), обов'язкова маржа (забезпечення) та висока ліквідність на спеціалізованих біржах (CME, Eurex, ICE).
2. Форвард — індивідуалізований позабіржовий контракт. Умови угоди сторони узгоджують самостійно, тому форварди не є стандартизованими, що підвищує ризик контрагента порівняно з ф'ючерсами.
3. Опціон — контракт, що надає покупцю право (але не зобов'язання) купити (call-опціон) або продати (put-опціон) актив за заздалегідь встановленою ціною. За типом виконання поділяються на європейські (виконання лише в дату експірації) та американські (виконання у будь-який час до дати експірації).
4. Своп — угода про обмін грошовими потоками на основі різних умов (наприклад, обмін плаваючої відсоткової ставки на фіксовану). Це

складні інструменти, які переважно застосовуються для професійного управління валютними та кредитними ризиками.

Кожен із розглянутих класів інструментів виконує свою функціональну роль в екосистемі фінансових ринків. Базові інструменти формують первинний об'єкт торгівлі, тоді як деривативи дозволяють трейдерам гнучко управляти ризиком і реалізовувати складніші стратегії без необхідності фізичного придбання активу. Правильний вибір інструменту відповідно до торгового стилю й ризик-профілю є важливою складовою ефективного трейдингу.

1.1.3. Типи угод

Аналіз функціонування фінансових ринків неможливий без вивчення механізму функціонування «книги заявок» (Order Book). Це електронний реєстр, у якому в режимі реального часу накопичуються та систематизуються всі актуальні лімітні заявки на придбання (біди) та продаж (аски) певного фінансового активу .

Order book виступає фундаментальним інструментом біржової торгівлі, що відображає поточний стан попиту та пропозиції. Саме цей реєстр забезпечує механізм автоматичного зведення контрагентів: коли ціна заявки на купівлю збігається з ціною заявки на продаж, відбувається виконання угоди. Відповідно, книга заявок є ключовим індикатором ринкової ліквідності та визначає динаміку ціноутворення.

Цей механізм візуалізує глибину ринку, дозволяючи трейдерам оцінювати обсяги, наявні на кожному ціновому рівні. Чи переходимо до детального розгляду основних видів торгових ордерів, які взаємодіють з цією книгою заявок?

Структура книги заявок (Order Book) передбачає поділ на дві ключові складові:

1. Ask Orders - заявки на продаж активу, що відображаються у порядку зростання ціни (від найнижчої до найвищої).

2. Bid Orders - заявки на купівлю активу, які впорядковані від найвищої ціни до найнижчої.

Поточна ринкова ціна (last price) формується на перетині цих частин як вартість останньої угоди між покупцем і продавцем. Аналіз книги заявок надає трейдерам можливості для оцінки ліквідності, визначення рівнів підтримки та опору, планування точок входу і виходу з ринку, а також формування торгових стратегій.

Таблиця 1.5

Приклад книги заявок

Ask Price	Ask Size	Bid Price	Bid Size
120.5	1.4 BTC	111	3.2 BTC
121	0.9 BTC	105	2.7 BTC
121.5	2.1 BTC	110	1.9 BTC

Завдяки моніторингу книги заявок трейдери отримують можливість виконувати такі аналітичні дії:

1. Визначати рівень ліквідності фінансового активу.
2. Виявляти потенційні зони підтримки або супротиву.
3. Здійснювати планування оптимальних точок для входу в позицію або виходу з неї.
4. Обирати та реалізовувати відповідні торгові стратегії.

У контексті трейдингу під терміном «заявка» (order) розуміють розпорядження, яке користувач передає брокеру чи торговій системі для виконання операції купівлі або продажу. Оскільки механізм виконання угоди безпосередньо залежить від обраного типу заявки, глибоке розуміння цих інструментів є обов'язковим для ефективної торгівлі. Загалом виділяють дві великі категорії заявок:

1. Market orders (ринкові заявки).
2. Pending orders (відкладені заявки).

Ринкові заявки передбачають негайну реалізацію купівлі або продажу за найкращою ціною, наявною на ринку в даний момент. Вони гарантують швидкість проведення операції, але не фіксують ціну, що при високій ринковій волатильності може викликати прослизання (squeeze), при якому фактична вартість угоди відхиляється від очікуваної.

Види ринкових заявок:

1. Market sell - реалізація продажу за поточною ринковою ціною.
2. Market buy - реалізація купівлі за поточною ринковою ціною.

Відкладені заявки активуються лише тоді, коли ринкова ціна активу досягає заздалегідь встановленого рівня, що дає змогу автоматизувати процес входу в ринок.

Види відкладених заявок:

1. Buy limit - встановлення ціни купівлі нижче поточної ринкової вартості.
2. Sell limit - встановлення ціни продажу вище поточної ринкової вартості.
3. Buy stop - встановлення ціни купівлі вище поточної ринкової вартості.
4. Sell stop - встановлення ціни продажу нижче поточної ринкової вартості.

Відкладені заявки безпосередньо формують рівень глибини ринку, тоді як ринкові заявки впливають на ціну, поглинаючи наявну ліквідність. Автоматичне зведення заявок виконується системою matching engine, що керується принципом пріоритету «ціна/час», де в першу чергу обробляються найвигідніші цінові пропозиції, подані раніше за інші.

Тип заявки - це фундаментальний елемент операційного контролю в трейдингу. Правильне застосування ринкових і відкладених ордерів дозволяє реалізовувати торгову стратегію з точністю, мінімізувати прослизання, захищати відкриті позиції та автоматизувати входи в ринок на заздалегідь визначених рівнях цін. Для автоматизованих систем знання механіки ордерів є обов'язковою умовою коректної реалізації алгоритму управління позицією.

1.1.4. Основні методи аналізу та проблеми прийняття рішень у трейдингу

Якість торгових рішень значною мірою визначається аналітичним

підходом, який обирає трейдер. На сьогодні в практиці трейдингу сформувалося кілька усталених методологічних напрямів, кожен з яких має власну концептуальну основу, інструментальний апарат і типові сфери застосування. Ці підходи не є взаємовиключними: більшість досвідчених трейдерів практикує їхнє поєднання залежно від ситуації на ринку і горизонту торгівлі.

Технічний аналіз ґрунтується на трьох базових постулатах: ціна враховує всю доступну ринкову інформацію; рух цін підпорядкований стійким тенденціям (трендам); ринкові патерни мають властивість повторюватися [4]. Спираючись на аналіз графіків і математичні індикатори, технічний аналіз є основним інструментом трейдерів, що працюють на коротких і середніх часових горизонтах.

Основні інструменти технічного аналізу:

1. Графіки.
2. Індикатори.
3. Фігури графічного аналізу.
4. Рівні підтримки та опору.
5. Лінії тренду.
6. Канали.

Фундаментальний аналіз досліджує внутрішню (справедливу) вартість активу через вивчення зовнішніх чинників: макроекономічного середовища, монетарної політики, фінансових показників емітента та галузевих тенденцій. Основна ідея полягає в тому, що ринкова ціна з часом прагне до справедливої вартості, тому активи, що торгуються нижче або вище цієї вартості, є потенційно привабливими для відповідних позицій. Фундаментальний аналіз особливо затребуваний при довгострокових стратегіях.

Фундаментальний аналіз базується на оцінці загальних ринкових умов та внутрішніх факторів, що впливають на вартість активів. До основних методів належать наступні:

1. Макроекономічні показники, до яких зараховують рівень ВВП, динаміку

інфляції, показники безробіття та встановлені процентні ставки.

2. Звітність фінансових установ, що містить відомості про прибутковість, загальні доходи, рівень рентабельності та показники кредитного навантаження.
3. Інформаційні події, що охоплюють політичну ситуацію, корпоративні зміни та коригування в законодавчій базі.
4. Стратегії центральних банків, включаючи ключові рішення ФРС, ЄЦБ та НБУ щодо монетарного регулювання.
5. Ончейн-метрики, що використовуються для аналізу блокчейн-мереж, зокрема підрахунок активних адрес, загальний обсяг транзакцій та параметри емісії.

Окремим підходом є концепція Smart Money, яка базується на відстеженні операцій інституційних гравців, таких як інвестиційні фонди та великі банківські структури. Теорія стверджує, що такі учасники володіють інформаційною перевагою, тому ринкові рухи часто сліднують за їхньою капітальною активністю. Прихильники цього методу надають перевагу аналізу чистого графіка, ціновим коливанням та обсягам, ігноруючи стандартні технічні індикатори.

Основні інструменти реалізації цього підходу:

1. Аналіз об'ємів торгів.
2. Детальний розгляд стрічки угод.
3. Ідентифікація зон ліквідності.
4. Вивчення інституційних шаблонів.

Market Profile є спеціалізованою технікою аналізу, яка візуалізує розподіл обсягів торгівлі залежно від ціни та часу, формуючи профіль у вигляді кривої «дзвоника». Такий інструмент дозволяє ідентифікувати зони ринкової цінності. Хоча Market Profile не є прямим інструментом прогнозування напрямку ціни, він дає розуміння комфортних цінових рівнів для учасників ринку та потенційних точок виникнення імпульсів.

Ключові компоненти аналізу в системі Market Profile:

1. Точка контролю, що відповідає ціновому рівню з максимальним обсягом торгів.
2. Зона ціни, що охоплює діапазон, у якому було здійснено 70% усього торгового обсягу.
3. Об'ємні вузли з високими та низькими значеннями, що виконують функцію рівнів підтримки або опору.
4. Показник Time Price Opportunity, який враховує часовий проміжок, протягом якого ціна активу перебувала на певному рівні.

На практиці більшість трейдерів не обмежується одним методом, а застосовує їхню комбінацію, що дозволяє зменшити кількість хибних сигналів і підвищити обґрунтованість рішень. Поєднання кількох підходів є особливо продуктивним у контексті побудови автоматизованих аналітичних систем, де кожен метод може бути формалізований і реалізований у вигляді незалежного модуля.

Важливо підкреслити, що сфера трейдингу включає велику кількість додаткових аналітичних методик. Аналізуючи відповідні наукові та практичні джерела, загальну сукупність підходів до вивчення ринків доцільно класифікувати на три категорії:

1. Предметний аналіз.
2. Математичний аналіз.
3. Гібридний аналіз.

Предметний аналіз охоплює методи, сфокусовані на вивченні цінової динаміки через призму сутності трейдингових процесів. До цієї групи належать усі раніше обговорені підходи, оскільки кожен із них пропонує власну специфічну інтерпретацію механізмів ринкової поведінки.

Математичний аналіз, на відміну від попереднього, зосереджується на абстракції від ринкової реальності. Його методологія базується виключно на опрацюванні числових даних та застосуванні кількісних методів до них.

Гібридний підхід поєднує переваги предметного і математичного

аналізу: формалізовані ринкові знання задають структуру і інтерпретацію, тоді як математичні методи забезпечують об'єктивну кількісну оцінку та здатність до адаптації на нових даних. Саме такий гібридний підхід покладено в основу розроблюваної в рамках цієї роботи інформаційної системи.

Прийняття торгових рішень — один із найскладніших і найвідповідальніших аспектів трейдингу. По суті, це багатовимірне завдання оптимізації, де трейдер змушений одночасно враховувати технічний стан активу, макроекономічний контекст, власний фінансовий стан і психологічний настрій. Дослідники виділяють чотири групи бар'єрів, що перешкоджають якісному прийняттю рішень: психологічні (страх, жадібність, упередженість підтвердження), технічні (затримки виконання ордерів, збої платформ), інформаційні (перевантаження даними, суперечливі сигнали) та ринкові (різка зміна волатильності, дії великих гравців).

Незважаючи на різноманіття описаних вище ускладнень, безпосередньо перед відкриттям конкретної торгової позиції перед трейдером постають два ключових аналітичних питання, вирішення яких є першочерговою функцією розроблюваної інформаційної системи:

1. Визначити рух ціни.
2. Визначити межі виходу та входу.

1.2. Інформаційна система для трейдингу

Стрімкий розвиток цифрових технологій та зростання обсягів доступних ринкових даних зумовили появу нового покоління програмних інструментів — інформаційних систем для трейдингу. Ці системи функціонують на перетині фінансового аналізу й обчислювальних методів, поєднуючи традиційний технічний та фундаментальний аналіз із сучасними підходами до обробки даних і машинного навчання. Вони виконують роль інтелектуального посередника між безперервним потоком ринкової інформації та конкретним торговим рішенням.

Концепція автоматизованого аналізу ринку не є принципово новою:

перші алгоритмічні торгові системи з'явилися ще у 1970–80-х роках з розвитком комп'ютерних технологій та кількісних фінансів. Однак масове поширення Інтернету, поява загальнодоступних API бірж і кількісне зростання крипторинку зумовили якісний стрибок у можливостях таких систем. Сучасні рішення здатні опрацьовувати ринкові дані в режимі реального часу, навчатися на нових прикладах і адаптуватися до мінливого характеру ринку без необхідності ручного перепрограмування правил.

Важливою відмінністю інформаційних систем для трейдингу від простих торгових терміналів або індикаторних платформ є наявність аналітичного ядра — компонента, що не лише відображає ринкові дані, але й інтерпретує їх, виявляє закономірності і формує структуровані аналітичні висновки. Таке ядро може бути реалізовано у вигляді продукційної експертної системи (на основі явно сформульованих правил «якщо–то»), нейронної мережі (що виявляє неявні залежності у часових рядах) або їхньої комбінації — гібридного підходу, що поєднує переваги обох парадигм.

Інформаційна система для трейдингу — це програмна система, що реалізує повний цикл роботи з ринковими даними: від їх збору з різномірних джерел (торгові майданчики, макроекономічні бази, агрегатори ринкового сентименту) через структуроване зберігання і первинну обробку до аналізу засобами інтелектуального аналізу даних і подання структурованих висновків кінцевому користувачеві. Такі системи поєднують методи математичного моделювання, статистики, технічного аналізу та машинного навчання, утворюючи єдине аналітичне середовище для підтримки торгових рішень.

1.2.1. Оглядова інформація

Інформаційні аналітичні системи у трейдингу — це програмні або інтелектуальні модулі, що опрацьовують великі обсяги даних (цінових, фундаментальних, сентиментальних) і генерують структуровані аналітичні результати: прогнозний напрям руху ціни, рівні потенційного входу та виходу з позиції, оцінку ринкових ризиків або рекомендації щодо торгової стратегії. У практичних реалізаціях такі системи можуть бути вбудовані у торгові

платформи, мобільні додатки, боти в месенджерах або надавати доступ через програмний API.

Основні функції інформаційних аналітичних систем у трейдингу:

1. Визначення оптимальних точок виходу та входу з позицій.
2. Розпізнавання торгових шаблонів.
3. Прогноз руху ціни.
4. Формування торгових сигналів за обраною стратегією.
5. Аналіз ризиків і рекомендація щодо розміру позицій.

Архітектура інформаційних аналітичних систем зазвичай включає кілька функціональних підсистем. Підсистема збору даних реалізує підключення до зовнішніх API та потоків ринкових даних. Підсистема зберігання відповідає за структуроване накопичення даних у реляційній базі з можливістю ефективного доступу до часових рядів. Аналітичне ядро реалізує методи виявлення закономірностей і формування аналітичних висновків. Підсистема інтерфейсу доставляє результати аналізу до кінцевого користувача через обраний канал: веб-платформу, мобільний додаток, месенджер або програмний REST API [2].

Суттєве значення у таких системах мають джерела та якість вхідних даних. Сучасні рішення, як правило, оперують щонайменше трьома типами: ціновими (OHLCV — ціни відкриття, закриття, максимуму, мінімуму та обсяг торгів), фундаментальними (макроекономічні показники: процентні ставки, індекс споживчих цін, дані ринку праці тощо) та сентиментальними (індекси страху і жадібності, ончейн-метрики, показники соціальних медіа). Поєднання різномірних джерел дозволяє формувати контекстно-залежну аналітику, що враховує одночасно технічний стан активу й макроекономічне тло.

Для функціонування таких систем застосовуються алгоритмічні методи обробки інформації, що базуються на застосуванні однієї або сукупності кількох парадигм:

1. Предметний аналіз, що охоплює технічні методи, фундаментальні показники, концепцію Smart Money та інші подібні підходи.

2. Математичний аналіз, до якого належать статистичні методи опрацювання даних, аналіз часових послідовностей, алгоритми машинного навчання та споріднені дисципліни.
3. Гібридний аналіз, який передбачає інтеграцію математичних методів із підходами предметного аналізу.

Важливо підкреслити, що жодна аналітична система не є безпомилковою і не може гарантувати прибутковість торгівлі: фінансові ринки є складними динамічними системами з елементами непередбачуваності. Тому інформаційні системи для трейдингу розглядаються як інструмент підтримки рішень, а не як заміна людського судження. Проте їхнє застосування суттєво підвищує системність торгового процесу, зменшує вплив емоційних і інтуїтивних чинників та дозволяє значно скоротити час, витрачений на рутинний аналіз.

1.2.2. Види інформаційних систем для трейдингу у трейдингу

Різноманіття підходів до побудови аналітичних систем для трейдингу зумовлює існування кількох незалежних класифікаційних схем. Системи різняться як за технічними характеристиками (методи реалізації, джерела даних), так і за функціональними можливостями (тип та рівень деталізації результатів) і способом взаємодії з кінцевим користувачем (канал доставки, рівень персоналізації).

Відповідно до методів реалізації алгоритмів:

1. Системи, що функціонують на основі чітко визначених правил (rule-based).
2. Системи, що базуються на технологіях машинного навчання (ML-based).
3. Гібридні системи, які поєднують обидва підходи.

Відповідно до джерел та типів вхідних даних:

1. Технічні системи (аналіз цінових графіків).
2. Фундаментальні системи (аналіз економічних показників).
3. Поведінкові системи (аналіз дій учасників ринку).
4. Гібридні системи, що інтегрують різні типи даних.

Відповідно до рівня персоналізації налаштувань:

1. Універсальні інструменти, розраховані на широке коло користувачів.
2. Персоналізовані системи, адаптовані під індивідуальні запити трейдера.

Відповідно до характеру видачі рекомендацій:

1. Сигнальні системи, що сповіщають про конкретні точки входу або виходу.
2. Аналітичні системи, що забезпечують опрацювання та візуалізацію ринкової інформації.
3. Пояснювальні системи, що надають обґрунтування прийнятих рішень.

Відповідно до каналів взаємодії з користувачем:

1. Мобільні додатки для оперативного доступу.
2. Веб-платформи, що забезпечують роботу через браузер.
3. Десктопні версії для персональних комп'ютерів.

1.2.3. Порівняльний аналіз існуючих інформаційних систем для трейдингу

Ринок аналітичних систем для трейдингу є достатньо зрілим: існують як комерційні рішення з широкою аудиторією, так і відкриті платформи, орієнтовані на технічно підготовленого користувача. Для розуміння конкурентного оточення і особливостей розробленої системи доцільно порівняти її з найбільш поширеними аналогами. У таблиці 1.6 наведено порівняльний аналіз провідних платформ за ключовими характеристиками: методологією, переважними перевагами і обмеженнями.

Аналіз таблиці 1.6 показує, що жодна з розглянутих систем не забезпечує комплексного охоплення аналітичних задач у єдиному рішенні: TradingView зосереджена переважно на візуалізації, Cryptohopper — на автоматизації виконання торгів, MetaTrader — на backtesting-середовищі, а QuantConnect — на повноцінному алгоритмічному трейдингу зі значними вимогами до кваліфікації розробника. Сучасні ML-орієнтовані підходи, зокрема архітектури LSTM і GRU, демонструють стабільно вищу точність

прогнозування на фінансових часових рядах порівняно з детерміністичними індикаторними методами — особливо на криптовалютному ринку з його характерною волатильністю.

Водночас системи на основі штучного інтелекту пов'язані зі суттєвими технічними і ресурсними вимогами: необхідністю розгортання обчислювальної інфраструктури, складністю підбору гіперпараметрів і ризиком перенавчання моделей на обмежених вибірках. Для більшості роздрібних трейдерів це є реальним бар'єром доступу. Розробка системи, що поєднала б переваги ML-підходів зі зручним низькопороговим інтерфейсом, є актуальним практичним завданням, на вирішення якого спрямована ця робота.

На підставі проведеного аналізу для розроблюваної в межах цієї роботи інформаційної системи для трейдингу обрані такі характеристики:

1.Метод реалізації: гібридний тип.

2.Джерело даних: гібридний тип.

3.Рівень персоналізації: універсальний формат.

4.Тип видачі рекомендацій: сигнальний механізм.

5.Канал доставки: месенджер.

6.Обраний набір характеристик зумовлений кількома практичними міркуваннями. Гібридний метод реалізації дозволяє охопити ширший спектр ринкових ситуацій: детерміновані правила ефективні при чітко вираженому тренді або характерних індикаторних сигналах, тоді як нейронні мережі краще виявляють тонкі часові залежності, недоступні для формульних методів. Гібридне джерело даних — поєднання технічних ринкових і фундаментальних макроекономічних показників — забезпечує ширший аналітичний контекст, ніж при використанні виключно цінових графіків. Доставка результатів через Telegram-месенджер з одночасно доступним REST API дозволяє охопити дві категорії споживачів:

трейдерів-практиків, які потребують зручного мобільного інтерфейсу, і розробників, що інтегрують аналітику в автоматизовані торгові стратегії.

Таблиця 1.6

Порівняльний аналіз інформаційних систем для трейдингу

Назва	Аналітична методологія	Ключові переваги	Основні недоліки
TradingView	Заснована на правилах, індикативний аналіз	Висока якість візуалізації, інтеграція інструментів з графіком	Обмеженість адаптації, залежність від внутрішніх скриптів
Cryptohopper	Поєднання правил та елементів машинного навчання	Широка автоматизація, зручний інтерфейс	Необхідність платної підписки, уніфікованість сигналів
MetaTrader	Робота за правилами, опціональне машинне навчання	Розвинені можливості автоматичної торгівлі	Складність процесу тестування, ризик надмірної переоптимізації
QuantConnect	Повноцінні алгоритми машинного навчання	Висока гнучкість, підтримка мов програмування Python та C#	Вимагає наявності фахових навичок у сфері розробки

Висновки до розділу 1

Отже, у першому розділі проведено комплексне дослідження предметної області, яке охопило ключові аспекти трейдингу як фінансової практики — стилі торгівлі, типи фінансових інструментів, механіку книги заявок, а також провідні методи ринкового аналізу. Детально розглянуто технічний і фундаментальний аналіз, концепції Smart Money і Market Profile, визначено їхні переваги та обмеження у контексті практичного застосування.

Особливу увагу приділено обґрунтуванню необхідності інформаційних систем для трейдингу: встановлено, що складність сучасних ринків, швидкість надходження даних і когнітивні обмеження людини роблять інтелектуальні аналітичні інструменти практично необхідними. У розділі 1.2 розкрито концепцію інформаційної системи для трейдингу: сформульовано її визначення, охарактеризовано архітектурні компоненти, описано основні функції та класифікацію за методами реалізації, джерелами даних, рівнем персоналізації, типом результатів і каналом доставки.

На підставі порівняльного аналізу існуючих систем (TradingView, Cryptohopper, MetaTrader, QuantConnect) визначено їхні ключові обмеження та обґрунтовано характеристики розроблюваного рішення: гібридний метод реалізації (правила + нейронні мережі), гібридне джерело даних (технічні + фундаментальні), сигнальний тип результатів і доставка через Telegram-месенджер з програмним REST API. Сформована теоретична база є підґрунтям для поглибленого вивчення методів інтелектуального аналізу даних, що є предметом другого розділу.

Проведений аналіз засвідчив, що ключовими чинниками ефективності сучасної інформаційної системи для трейдингу є: здатність інтегрувати різноманітні джерела даних, застосовувати адаптивні алгоритми прогнозування, що враховують поточний стан ринку, та надавати рекомендації у зручному для трейдера форматі. Ці висновки стали основою для формування архітектурної концепції розроблюваної системи та визначення набору методів інтелектуального аналізу даних, що будуть досліджені в наступному розділі.

РОЗДІЛ 2 ОГЛЯД МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

2.1. Оглядова інформація

Оскільки стрімкий прогрес технологій фіксації та архівації відомостей призвів до накопичення масивів інформації критичних масштабів, її опрацювання без застосування спеціальних технічних засобів стало нездійсненним. Класичні статистичні методи виявилися недієвими для роботи з даними такого обсягу, що стало передумовою для виникнення та впровадження новітніх методологій, серед яких ключове місце посідає інтелектуальний аналіз даних.

Інтелектуальний аналіз даних (ІАД) визначається як процедура видобутку з первинних масивів інформації раніше прихованих, нетривіальних, практично цінних та придатних для подальшої інтерпретації відомостей, які є необхідними для формування обґрунтованих управлінських рішень у різноманітних галузях. Наукова база ІАД ґрунтується на концепції.

ІАД базується на концепції шаблонів, які відображають фрагменти багатоаспектних взаємин у даних. Ці шаблони — це закономірності, властиві підвибіркам даних, які можуть бути компактно виражені у зрозумілій людині формі. Пошук шаблонів проводиться методами, не обмеженими апріорними припущеннями про структуру вибірки та види розподілів значень аналізованих показників.

Ключовим компонентом інтелектуального аналізу даних є виявлення нетривіальних шаблонів, що означає визначення прихованих, раніше невідомих закономірностей, які складають суть прихованих знань. Сучасні ІАД-системи впроваджують такі базові аналітичні методи:

1. Методи класифікації (серед них нейронні мережі, дерева рішень, а також наївний байєсівський класифікатор), які забезпечують віднесення об'єктів до конкретних категорій з огляду на їхні параметри.
2. Методи кластеризації (наприклад, ієрархічна кластеризація або алгоритм k-середніх), спрямовані на об'єднання об'єктів у групи за

ознакою їхньої подібності.

3. Методи асоціативного аналізу (включаючи алгоритм Apriori), що фокусуються на ідентифікації взаємозв'язків та регулярностей у масивних наборах даних.
4. Методи прогнозування (такі як часові ряди чи регресійний аналіз), які дають можливість моделювати майбутні показники, спираючись на аналіз історичної динаміки.
5. Методи виявлення аномалій, що застосовуються для ідентифікації нестандартних шаблонів або відхилень у даних, які можуть бути ознакою помилок чи виняткових ситуацій.
6. Застосування методів ІАД у фінансовій сфері має свою специфіку, зумовлену особливостями фінансових даних. Фінансові часові ряди характеризуються нестационарністю, наявністю шуму, нелінійними залежностями та залежністю від зовнішніх (макроекономічних) факторів. Це зумовлює необхідність комбінувати різні методи ІАД для досягнення прийнятної точності прогнозування: жоден із методів у ізольованому застосуванні не є достатнім для надійного аналізу фінансових ринків у всіх ринкових умовах.
7. Одним із найактуальніших напрямків застосування ІАД у трейдингу є побудова гібридних систем прийняття рішень, що поєднують правила, сформульовані експертами ринку, з адаптивними алгоритмами машинного навчання. Правила-засновані підходи (rule-based) забезпечують прозорість і зрозумілість рекомендацій, тоді як нейронні мережі дозволяють виявляти неочевидні закономірності в даних. Саме такий гібридний підхід реалізовано в межах цього дослідження: поєднання продукційної експертної системи з нейромережевими моделями прогнозування забезпечує як інтерпретованість, так і адаптивність торгових сигналів.

У межах даного дослідження було проведено комплексний аналіз доцільності поєднання експертних систем із нейронними мережами з метою

розробки алгоритмів формування рекомендацій в межах інформаційної системи для трейдингу.

2.2 Використання експертних систем

Експертні системи належать до ключових напрямків інтелектуального аналізу даних, методологія яких базується на імітації алгоритмів прийняття рішень, що їх використовують фахівці у конкретних предметних сферах. Типова структура експертної системи охоплює такі елементи:

1. База знань.
2. Механізм логічного виведення.
3. Користувацький інтерфейс.

Основне завдання цих систем полягає у генерації експертних висновків або рекомендацій, що базуються на обробці наявного масиву знань та вхідних даних.

Класифікація експертних систем за типами:

1. Продукційні системи, функціонування яких ґрунтується на логічних конструкціях «якщо-то».
2. Фреймові системи, що базуються на застосуванні структурованих об'єктів та визначенні їхніх властивостей.
3. Байєсівські мережі, робота яких враховує імовірнісні характеристики.
4. Нечіткі експертні системи.

У межах поточного дослідження розроблено продукційну експертну систему, яка функціонує згідно з формалізованими правилами. У процесі її роботи кожен окремий індикатор (зокрема RSI, MACD, EMA, об'єм торгів, індекс страху та жадібності) генерує локальний сигнал, після чого система застосовує механізм голосування між отриманими сигналами для формування фінального рішення. Варто також зауважити, що на роботу механізму логічного виведення суттєво впливає обраний користувачем параметр торгового стилю (зокрема, часовий таймфрейм для проведення операцій), оскільки від цього показника залежать налаштування алгоритмів виведення.

Зазначена методологія сприяє ідентифікації напрямку цінового руху через застосування чітко визначених правил.

Ще одним важливим аспектом реалізованої продукційної системи є механізм агрегування сигналів. Замість того, щоб приймати рішення на основі одного індикатора, система збирає голоси від усіх індикаторів (RSI, MACD, ЕМА, об'єм, відкритий інтерес і за потреби — індекс страху/жадібності) та визначає підсумковий напрямок більшістю голосів. Такий підхід підвищує стійкість системи до хибних сигналів, що є важливою перевагою порівняно з однофакторними методами аналізу, оскільки хибний сигнал одного індикатора нівелюється правильними сигналами решти.

Принципово важливим є те, що налаштування індикаторів у системі не є фіксованими — вони автоматично адаптуються залежно від обраного трейдером торгового стилю. Наприклад, для скальпінгу використовуються коротші періоди та ширші пороги RSI, що дозволяє реагувати на короткострокові зміни цін. Натомість для довгострокового інвестування застосовуються тривалі горизонти ковзних середніх та вужчі пороги, що фільтрують короткостроковий шум і фокусуються на стійких трендах. Ця адаптивність є ключовою перевагою системи з точки зору практичного застосування для трейдерів із різними інвестиційними горизонтами і схильністю до ризику.

Розглянемо логічні висловлювання, які використовувались в межах кваліфікаційної роботи.

2.2.1. Індекси RSI та MACD

Індекс відносної сили (Relative Strength Index, RSI) являє собою ключовий осцилятор у технічному аналізі, призначений для вимірювання швидкості та динаміки цінових змін з метою ідентифікації станів перекупленості або перепроданості торгового активу. Показник RSI змінюється в діапазоні від 0 до 100 одиниць.

Математичне вираження для обчислення RSI визначається за такою формулою:

$$RSI = 100 - \left(\frac{100}{1 + RS} \right),$$

де: $RS = \frac{EMA_{gain}}{EMA_{loss}}$ - це відношення середнього приросту ціни за визначений період до середнього падіння ціни за той самий період. У цьому контексті:

- Average Gain — це значення експоненційної ковзної середньої (ЕМА), що інтегрує всі позитивні динаміки вартості активу за заданий період. Якщо торговий день завершується зростанням, різниця відображається у показнику gain; у протилежному випадку враховується нуль.
- Average Loss — це експоненційна ковзна середня (ЕМА) всіх від'ємних змін ціни. У разі падіння котирувань зміна реєструється як додатна величина втрати; якщо падіння відсутнє, значення прирівнюється до нуля.

Практичне застосування індикатора RSI передбачає наступні правила прийняття рішень:

1. Якщо значення RSI менше нижнього критичного рівня, актив вважається перепроданим, що формує сигнал до відкриття довгих позицій (купівля).
2. Якщо значення RSI перевищує верхню межу, фіксується перекупленість, що є сигналом до продажу.
3. Якщо показник знаходиться у межах між нижнім та верхнім порогами, ринок перебуває в стані нейтральності, що передбачає утримання поточної позиції.

Параметри індикатора залежно від обраної торгової стратегії:

1. Скальпінг: пороги 20 та 80, період 7.
2. Інтрадей: пороги 20 та 80, період 14.
3. Свінг-трейдинг: пороги 30 та 70, період 21.
4. Інвестування: пороги 30 та 70, період 28.

Індикатор MACD (Moving Average Convergence Divergence) дозволяє оцінювати динаміку зближення та розбіжності між короткостроковими та довгостроковими експоненційними ковзними середніми. Розрахунок MACD здійснюється за наступним математичним виразом:

$$MACD_{(a,b)} = EMA_a - EMA_b,$$

$$Signal_c = EMA_c(MACD),$$

де: EMA_a - короткострокова експоненційна ковзна середня; EMA_b - довгострокова експоненційна ковзна середня; EMA_c - сигнальна лінія, що являє собою ЕМА від значень MACD..

Алгоритм застосування сигналами індикатора:

1. $MACD > Signal$ (на купівлю).
2. $MACD < Signal$ (на продаж).
3. $MACD \approx Signal$ (тримати позицію).

Налаштування індикатора для різних торгових стилів:

1. Скальпінг ($a = 6, b = 13, c = 5$).
2. Інтрадей ($a = 8, b = 21, c = 7$).
3. Свінг ($a = 12, b = 26, c = 9$).
4. Інвестування ($a = 19, b = 39, c = 12$).

2.2.2. Експоненційне ковзне середнє (ЕМА)

ЕМА (Exponential Moving Average) — це інструмент технічного аналізу, різновид ковзної середньої, який надає пріоритетну вагу найбільш актуальним ціновим даним. Завдяки цьому індикатор демонструє вищу чутливість до поточних змін ринкової кон'юнктури порівняно з класичними методами усереднення.

Математична модель для розрахунку ЕМА базується на рекурсивному співвідношенні:

$$EMA_t = \alpha * P_t + (1 - \alpha),$$

де: $\alpha = \frac{2}{n+1}$ – коефіцієнт згладження, P_t - ціна (наприклад, закриття) на момент t , n – період.

Застосування сигналу індикатора на:

1. Ціна > ЕМА, купівлю.
2. Ціна < ЕМА, продаж.
3. Ціна $\approx$ ЕМА, тримати позицію.

Налаштування індикатора для різних торгових стилів:

1. Скальпінг: ЕМА(9), ЕМА(20), ЕМА(50).
2. Інтрадей: ЕМА(12), ЕМА(26), ЕМА(100).
3. Свінг: ЕМА(20), ЕМА(50), ЕМА(100).
4. Інвестування: ЕМА(50), ЕМА(100), ЕМА(200).

2.2.3 Об'єм торгів та відкритий інтерес

Обсяги торгів та відкритий інтерес (OI) виступають основними інструментами, за допомогою яких оцінюється поточна ринкова активність та загальний настрій трейдерів. Сукупний обсяг торгів за визначений часовий інтервал розраховується шляхом підсумовування кількості всіх активів, що були предметом угод купівлі або продажу протягом цього періоду.

$$Volume_t = n_t,$$

де: n_t – кількість угод, укладених за період t .

Відкритий інтерес (OI) визначається як сукупна кількість активних контрактів (наприклад, ф'ючерсних), що не були ліквідовані, погашені або компенсовані шляхом здійснення протилежних операцій. Математичний вираз для визначення відкритого інтересу має такий вигляд:

$$OI_t = OI_{t-1} + C_t - D_t,$$

де: OI_t - відкритий інтерес у момент часу t , OI_{t-1} – відкритий інтерес у попередній момент часу, C_t – кількість нових контрактів у момент t , D_t – кількість ліквідованих контрактів у момент t .

2.2.4. Індекс страху та жадібності

Індекс страху та жадібності є зведеним показником, що відображає панівні настрої на фондовому ринку або ринку криптовалют. Процедура розрахунку даного показника забезпечується компанією Alternative.me, дані якої надходять через API з періодичністю один раз на добу. Розрахунок базується на таких даних, як рівень волатильності, обсяг торгів, соціальна активність, домінування біткоїна, запити у Google Trends та інші параметри; діапазон значень індексу становить від 0 до 100.

Правила застосування індикатора:

1. Якщо показник становить менше 25, фіксується стан страху, що вказує на потенціал для здійснення купівлі.
2. Якщо значення перевищує 75, ринок перебуває у стані жадібності, що свідчить про високий ризик корекції.
3. Якщо значення індексу коливається в межах від 25 до 75, ринковий стан класифікується як нейтральний.

Адаптація залежно від стилю торгівлі:

1. Для стратегій свінг-трейдингу та інвестування — індикатор береться до уваги.
2. Для стратегій скальпінгу та інтрадей-торгівлі — індикатор не враховується.

2.3. Використання нейронних мереж

Штучні нейронні мережі (ШНМ) є одним із найдієвіших інструментів машинного навчання, що забезпечують моделювання складних нелінійних залежностей у масивах даних значного обсягу. Ці архітектури базуються на принципах функціонування біологічних нейронних структур і демонструють високу ефективність у випадках, коли традиційні алгоритми виявляються недостатньо точними, зокрема при вирішенні задач класифікації, прогнозування та ідентифікації шаблонів.

Структура нейронної мережі охоплює три фундаментальні елементи :

1. Вхідний шар, що здійснює прийом вхідних даних .
2. Приховані шари, де виконується опрацювання інформації із застосуванням активаційних функцій .
3. Вихідний шар, що формує прогноз або визначає клас об'єкта .

У сфері трейдингу нейронні мережі мають високу цінність завдяки таким можливостям:

1. Автоматична ідентифікація прихованих закономірностей у часових рядах котирувань .
2. Врахування історичного контексту (особливо при використанні рекурентних архітектур) .
3. Адаптація до динамічних змін ринкових умов .

У межах поточного дослідження було проаналізовано та впроваджено три типи нейронних мереж:

1. LSTM (Long Short-Term Memory) — архітектура, що вирізняється здатністю запам'ятовувати довгострокові залежності в часових рядах завдяки використанню спеціальних гейтів керування пам'яттю .
2. GRU (Gated Recurrent Unit) — оптимізований варіант LSTM, що характеризується меншою кількістю параметрів та прискореним процесом навчання, що робить її ефективною за умови обмеженості вхідних даних .
3. MLP (Multi-Layer Perceptron) — класична повнозв'язна нейронна мережа, яка демонструє високі показники в задачах регресії за умови належної підготовки структурованих вхідних даних .

Кожна з моделей володіє власним набором переваг і недоліків, а проведення їхнього порівняльного аналізу на основі метрик MAE та RMSE дозволяє встановити найоптимальніший підхід для побудови інформаційної системи для трейдингу

2.3.1. Мережа типу Long Short-Term Memory

LSTM (Long Short-Term Memory) — це архітектура рекурентних нейронних мереж (RNN), спеціалізована для аналізу послідовних даних, зокрема часових рядів [11]. На відміну від стандартних мереж прямого поширення, RNN використовують прихований внутрішній стан, що забезпечує збереження інформації про попередні елементи послідовності. Архітектура LSTM була створена для усунення проблеми «затухання градієнта», що дозволяє ефективно моделювати довгострокові залежності. Ця особливість є ключовою для прогнозування фінансових ринків, де історичний контекст безпосередньо впливає на майбутню динаміку цін.

Переваги:

1. Ефективна фіксація довгострокових залежностей та утримання контексту впродовж тривалого періоду.
2. Гнучке керування потоками даних за допомогою системи гейтів (забуття, вхід, вихід).
3. Висока результативність у вирішенні завдань обробки послідовностей.

Недоліки:

1. Значна обчислювальна складність, що вимагає тривалого часу для навчання.
2. Схильність до перенавчання на невеликих вибірках даних, що зумовлює необхідність застосування методів регуляризації.
3. Трудомісткість підбору оптимальних гіперпараметрів (кількість шарів, розмірність прихованих станів тощо).

Класична архітектура LSTM включає вхідний рівень, один або декілька прихованих LSTM-шарів та вихідний шар регресії. Процес функціонування, описаний на рисунку 2.1, включає:

1. Вхідний шар, розрахований на обробку 50 попередніх цінових значень.
2. LSTM-шари з визначеною кількістю нейронів, де для фільтрації інформації застосовуються сигмоїдальні та тангенсоїдні активації.
3. «Забуваючий» гейт (forget gate), який через сигмоїдальні ваги регулює збереження попередньої пам'яті C_{t-1} .
4. «Вхідний» гейт, що формує нову кандидатну пам'ять на основі σ та $\tanh$.
5. «Вихідний» гейт, який через сигмоїдальне кодування визначає ступінь передачі оновленої пам'яті C_t на вихід мережі.
6. Фінальний Dense-шар з лінійною активацією, що генерує прогноз наступного значення.

Навчання моделей зазвичай триває від 50 до 100 епох із використанням пакетів (batch size) розміром 16–64. Оптимізація проводиться через алгоритм Adam із застосуванням регресійних функцій втрат (MSE або MAE). Для оцінки точності прогнозування використовуються метрики MAE (середня абсолютна помилка) та RMSE (корінь середньоквадратичної помилки), що кількісно відображають відхилення прогнозних значень від фактичних ринкових цін.

Вибір оптимальної моделі здійснюється за мінімальними показниками обраної метрики на валідаційних даних. Для запобігання надмірному навчанню та оптимізації архітектури застосовують механізми ранньої зупинки (early stopping) та крос-валідацію. Еталонною вважається модель, що демонструє найнижчі значення помилок MAE/RMSE на тестовому наборі даних.

Практика застосування LSTM у задачах фінансового прогнозування показує, що якість моделі суттєво залежить від тривалості вхідної послідовності (lookback window) та кількості шарів мережі. Вибір вікна у 50 кроків у даному дослідженні обґрунтований компромісом між достатнім контекстом для виявлення середньострокових залежностей і обчислювальною ефективністю. Регуляризація через механізм early stopping є важливою

складовою процесу навчання, оскільки фінансові дані схильні до перенавчання моделей через відносно малий обсяг навчальної вибірки порівняно з кількістю параметрів мережі.

2.3.2. Мережа типу Gated Recurrent Unit

Мережі типу Gated Recurrent Unit (GRU) є ще одним різновидом рекурентних нейронних мереж, що базуються на механізмах «гейтування» інформаційних потоків. На відміну від LSTM, архітектура GRU вирізняється меншою кількістю гейтів та загальною кількістю параметрів: вона об'єднує функції «вхідного» та «забуваючого» гейтів у єдиний гейт оновлення (update gate), а також використовує гейт скидання (reset gate). За своєю структурою GRU є спрощеним аналогом LSTM, де відсутні окремий вихідний гейт та спеціалізований блок пам'яті, оскільки функцію пам'яті виконує безпосередньо прихований стан мережі.

Переваги:

1. Спрощена структура з меншою кількістю параметрів порівняно з LSTM, що забезпечує прискорення процесу навчання.
2. Експериментальні дані свідчать про те, що за рівнем продуктивності у багатьох задачах GRU не поступається LSTM.

Недоліки:

1. Теоретично менша ефективність при обробці наддовгих послідовностей через поєднання «вхідного» та «забуваючого» гейтів (хоча на практиці результативність залежить від специфіки конкретних даних).
2. Чутливість до обсягу навчальної вибірки та налаштувань гіперпараметрів (кількість шарів, розмір прихованого стану), аналогічна до LSTM.

Як ілюструє архітектура, представлена на рисунку 2.2, блок GRU базується на двох ключових механізмах:

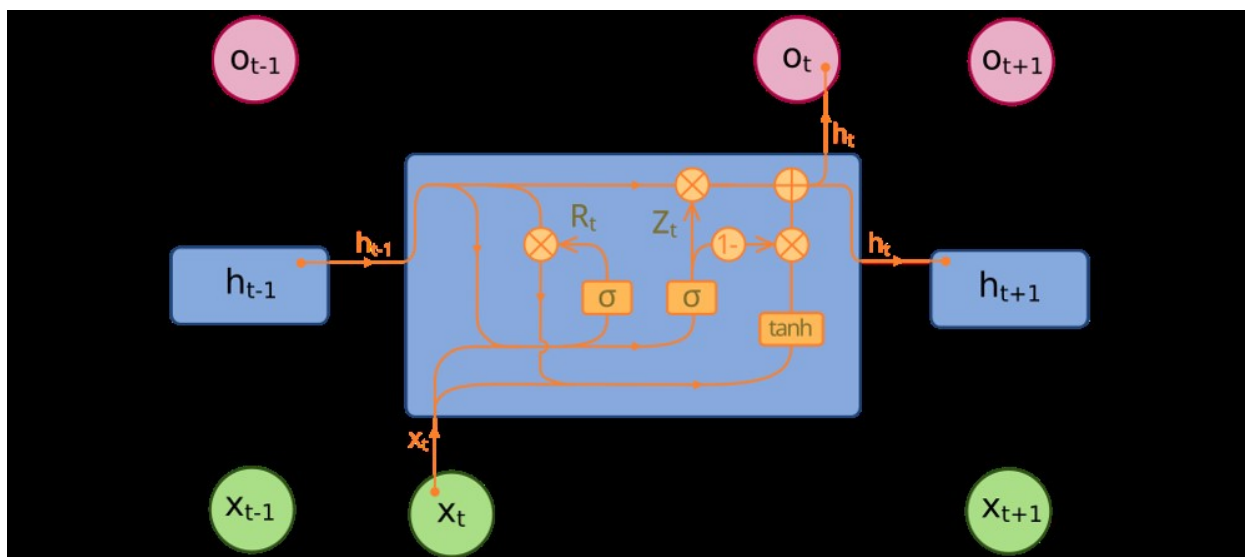


Рис. 2.2. Архітектура GRU

Параметри навчання GRU ідентичні до LSTM: тренування триває протягом десятків або сотень епох при розмірі пакету (batch size) у межах 16–64. Для оптимізації застосовується алгоритм Adam, а для оцінки якості прогнозів — стандартні метрики MAE та RMSE [12]. Вибір найкращої конфігурації мережі базується на досягненні мінімальних значень помилок на валідаційному наборі даних. При порівнянні моделей LSTM та GRU пріоритет надається архітектурі, яка демонструє нижчий показник MAE/RMSE за ідентичних умов навчання.

Порівняльні дослідження показують, що GRU та LSTM демонструють схожу точність прогнозування на більшості наборів фінансових даних при суттєво меншій обчислювальній вартості GRU. З практичної точки зору це означає, що для застосувань із обмеженими обчислювальними ресурсами або великою кількістю активів для аналізу GRU може бути кращим вибором, ніж LSTM. У даному дослідженні обидві архітектури навчаються паралельно, що дозволяє визначити найкращу для конкретних ринкових умов без апіорних припущень щодо переваги тієї чи іншої архітектури.

У межах поточного дослідження було розроблено модель нейронної мережі зі специфічними конфігураційними параметрами:

- a. Архітектура: послідовна модель, що включає два приховані шари GRU та один вихідний шар.
- b. Вхідні дані: набір із 50 попередніх значень ціни закриття, що пройшли процедуру нормалізації. Розмірність вхідного тензора становить $(50, 1)$.
- c. Цільовий показник (вихід): прогнозована ціна на наступний часовий період.
- d. Структура шарів:
- e. GRU(64): перший шар, що опрацьовує послідовність та повертає вихід для кожного часового кроку.
- f. GRU(32): другий шар, що повертає виключно фінальний прихований стан після обробки всієї послідовності.
- g. Dense(1): повнозв'язний вихідний шар, призначений для отримання одиничного числового прогнозу.

2.3.3. Мережа типу Multi-Layer Perceptron

Мережа MLP (Multi-Layer Perceptron) — це класична багатошарова нейронна мережа прямого поширення. Вона складається з одного вхідного шару, одного або кількох прихованих шарів і одного вихідного шару. При цьому кожен нейрон попереднього шару з'єднаний з усіма нейронами наступного. На відміну від RNN, MLP не має внутрішньої пам'яті і не розроблена спеціально для роботи з послідовними даними. Проте для прогнозування часових рядів можна використовувати підхід «ковзного вікна»: як вхід подають останні 50 значень ряду (пакетом ознак), а вихід має одне передбачуване значення.

Переваги:

1. Простота архітектури.
2. Відносно швидке тренування.
3. MLP добре підходить для апроксимації нелінійних функцій при достатній кількості нейронів.

4. Завдяки прямому поширенню і відсутності гейтів налаштування моделі й інтерпретація її роботи простіші, ніж у RNN.

Недоліки:

1. Немає вбудованого механізму «пам'яті» — мережа не враховує порядок і залежність між входами, якщо їх просто подати як незалежні ознаки. Це може знизити точність на чисто часових даних.
2. MLP потребує ретельного відбору й масштабування вхідних ознак та може перенавчатися через велику кількість параметрів.

У межах даної роботи щодо архітектури MLP-мережі використано вхідний шар з 50 нейронами (по одному на кожен попередній ціновий відлік), один-два приховані шари з нелінійними активаціями (наприклад, ReLU або tanh) і один вихідний нейрон з лінійною активацією для прогнозування наступної ціни. Загальна форма: Input (50) → Hidden (ReLU/tanh) → Output (1). На рисунку 2.3 подано спрощену схему такої MLP.

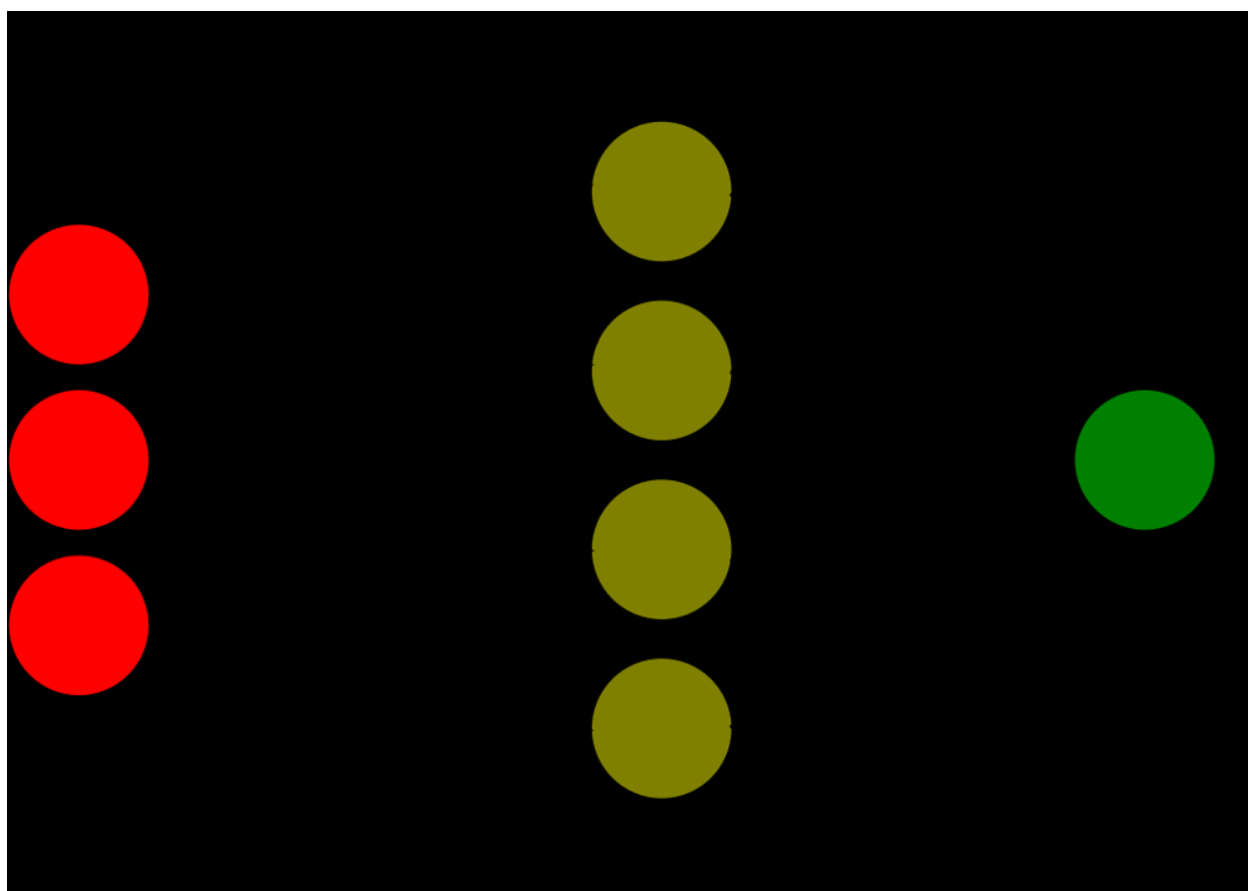


Рис. 2.3. Спрощена архітектура MLP. Червоні вузли – вхід (50 значень), жовті – приховані (наприклад, 4 нейрони в прихованому шарі), зелений – вихід (1 прогностичне значення).

Зазвичай для MLP-моделей обирають відносно невелику кількість прихованих шарів (1–3). Мережу тренують протягом 100+ епох із використанням розміру пакету (batch size) ≈ 32 [13]. Як функцію втрат застосовують MSE або MAE, а як оптимізатор — алгоритм Adam. Точність прогнозування оцінюють за допомогою метрик MAE та RMSE.

Серед конфігурацій із різною кількістю нейронів та шарів обирають ту MLP-модель, яка демонструє найменші показники MAE/RMSE на валідаційному (перевірочному) наборі даних. Часто результати MLP порівнюють із результатами RNN-підходів, віддаючи перевагу моделі з вищою прогностичною точністю, тобто тій, що забезпечує найнижчі значення похибок.

Незважаючи на відсутність механізму послідовної пам'яті, MLP демонструє конкурентоспроможні результати у задачах прогнозування цін, особливо коли вхідні дані нормалізовані і відібрані ознаки добре корелюють з цільовою змінною. У контексті даного дослідження MLP слугує базовою моделлю (baseline), з якою порівнюються рекурентні архітектури LSTM та GRU. Якщо різниця в точності між MLP та рекурентними мережами є незначною, це може свідчити про те, що довгострокові залежності не є домінуючим фактором для конкретного активу у вибраному часовому вікні.

У межах цього дослідження було розроблено модель із такими параметрами:

1. Архітектура: послідовна модель типу feed-forward, що включає етап Flatten, два приховані шари та один вихідний шар.
2. Вхідні дані: набір із 50 нормалізованих значень ціни. Вхідна форма становить (50, 1), яка за допомогою шару Flatten() трансформується у вектор розмірністю (50,).

3. Цільовий показник (вихід): прогнозована ціна на наступний часовий крок у вигляді одного числового значення.

4. Структура шарів:

Flatten(): шар, що перетворює вхідний тензор (50, 1) у вектор (50,).

Dense(128): перший прихований шар.

Dense(64): другий прихований шар.

Dense(1): вихідний шар регресії.

5. Функції активації:

Для прихованих шарів застосовується ReLU (Rectified Linear Unit).

Для вихідного шару використовується лінійна активація (за замовчуванням).

Варто зазначити, що навчання всіх моделей проводилося на послідовностях довжиною 50 значень. Результатом роботи кожної моделі є одне числове значення, яке інтерпретується як прогноз ціни. Інформаційна система для трейдингу здійснює автоматичне порівняння метрик MAE та RMSE для трьох моделей, обирає найефективнішу з них і генерує для користувача рекомендації з визначенням точок входу та виходу.

2.3.4. Порівняльний аналіз нейронних мереж та вибір оптимальної моделі

Для порівняння ефективності трьох архітектур нейронних мереж (LSTM, GRU, MLP) у межах даного дослідження використовуються три ключові метрики якості прогнозування. MAE (Mean Absolute Error) — середня абсолютна похибка — показує середнє відхилення прогнозованих значень від фактичних і є стійкою до викидів. RMSE (Root Mean Squared Error) — корінь із середньоквадратичної помилки — надає більшу вагу великим похибкам і є чутливою до аномальних значень у даних. MAPE (Mean Absolute Percentage Error) — середня абсолютна відсоткова похибка — дозволяє оцінити відносну точність прогнозу незалежно від масштабу цін активу.

Вибір найкращої моделі здійснюється автоматично системою на основі мінімізації суми нормалізованих значень MAE, RMSE і MAPE. Такий підхід

дозволяє уникнути суб'єктивності при виборі моделі та забезпечує автоматичну адаптацію до специфіки конкретного активу й поточних ринкових умов. Обрана модель використовується для прогнозу наступного цінового значення та розрахунку рівнів цілі (take-profit) і стоп-лосу відповідно до обраного торгового стилю.

Важливим практичним аспектом є те, що всі три моделі перенавчаються при кожному новому запиті користувача на основі останніх 1000 хвилинних свічок. Це забезпечує актуальність прогнозу у швидкоплинних ринкових умовах. Хоча така архітектура призводить до певної затримки у відповіді системи (час навчання моделей), вона виключає старіння моделі та не потребує регулярного ручного перенавчання. Для скорочення часу відгуку у майбутніх версіях системи може бути впроваджений кеш моделей з інкрементальним донавчанням на нових даних.

Висновки до розділу 2

Підсумовуючи, у розділі було розглянуто методи інтелектуального аналізу даних (ІАД), які складають фундамент для розробки інформаційних систем для трейдингу. Було проаналізовано ключові підходи до інтелектуального аналізу, зокрема класифікацію, кластеризацію, асоціативний аналіз, прогнозування та виявлення аномалій. Також було приділено увагу експертним системам і нейронним мережам як інструментам підтримки прийняття рішень.

Окремий акцент було зроблено на детальному вивченні архітектур нейронних мереж, таких як LSTM, GRU та MLP, їхніх переваг, недоліків та специфіки використання для прогнозування котирувань активів. Крім того, були висвітлені методологічні аспекти побудови моделей, процеси їх навчання та критерії оцінки якості прогнозів. Таким чином, сформовано необхідну теоретичну базу для обґрунтованого вибору алгоритмів ІАД, що є критично важливими компонентами для реалізації інформаційної системи для трейдингу.

Вибір саме трьох архітектур нейронних мереж — LSTM, GRU та MLP — обумовлений принципом комплементарного покриття різних аспектів задачі прогнозування: рекурентні мережі (LSTM, GRU) здатні фіксувати довгострокові залежності у часових рядах, тоді як MLP, незважаючи на відсутність механізму внутрішньої пам'яті, демонструє ефективність при добре підготовлених вхідних даних. Паралельне навчання та порівняння цих трьох моделей за метриками MAE, RMSE та MAPE дозволяє автоматично обирати оптимальний інструмент прогнозування для конкретного активу та ринкових умов.

Розроблена продукційна експертна система на базі RSI, MACD, EMA, об'єму торгів, відкритого інтересу та індексу страху і жадібності забезпечує прозорість та інтерпретованість торгових рекомендацій. Завдяки механізму голосування між сигналами різних індикаторів система мінімізує хибні сигнали окремих показників і надає більш збалансовану картину стану ринку. Адаптація налаштувань індикаторів до різних торгових стилів (scalping, intraday, swing, investing) суттєво підвищує практичну цінність системи для трейдерів із різними інвестиційними горизонтами.

Отже, теоретичні засади, сформовані у другому розділі, є достатньою основою для практичної реалізації інформаційної системи для трейдингу. Поєднання інтерпретованих правил експертної системи з адаптивними нейромережевими прогностичними моделями відповідає меті роботи — створити систему, яка одночасно забезпечує обґрунтованість рекомендацій та здатність до навчання на нових ринкових паттернах.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Архітектура системи

Система спроектована як асинхронний багаторівневий Python-додаток, що забезпечує аналітичну підтримку трейдерів через два незалежні інтерфейси: Telegram-бот для безпосередньої взаємодії з користувачем та REST API на базі FastAPI для інтеграції з іншим програмним забезпеченням. Архітектура відзначається модульністю, гнучкістю та високою масштабованістю.

Основні компоненти системи:

1. Telegram-бот (на базі aiogram 3): реалізує інтерфейс комунікації з користувачем, надаючи можливість налаштування параметрів аналізу (вибір ринку, тип прогнозу, торговий стиль) та виведення результатів.
2. Контролер (бізнес-логіка): координує обробку користувацьких запитів, викликаючи необхідні аналітичні функції, включаючи модулі експертної системи або нейронних мереж.
3. Модуль збору даних: асинхронний процес, що здійснює збір:
 - Ринкових даних через Binance API (котирування, обсяги, відкритий інтерес).
 - Фундаментальних показників через FRED API та сервіс alternative.me.
4. Моделі прогнозування (нейронні мережі): включають реалізацію архітектур LSTM, GRU та MLP на базі бібліотеки PyTorch для прогнозування руху ціни.
5. Експертна система: базується на наборі аналітичних правил, що використовують індикатори EMA, RSI, MACD, обсяги торгів, відкритий інтерес (OI) та індекс страху і жадібності для оцінки вектора цінового руху.

6. База даних (MySQL): забезпечує асинхронне зберігання історичних ринкових та фундаментальних даних, а також профілів користувачів Telegram.

Процес збору даних у системі реалізовано на базі асинхронних функцій, які активуються під час старту програми. Алгоритм передбачає первинне завантаження історичних даних (або відновлення пропущених значень), після чого система перемикається на моніторинг у режимі реального часу: щохвилинне оновлення ринкових даних та щоденне оновлення фундаментальних показників. Усі зібрані дані зберігаються у базі MySQL згідно зі структурованою схемою [14].

Структура бази даних

1. Таблиця `crypto_tickers_data` Призначена для збереження котирувань, обсягів та відкритого інтересу криптовалютних активів із щохвилинним інтервалом (джерело: Binance Futures).

Поля: `id`, `symbol`, `timestamp` (unix-мітка), `open`, `close`, `high`, `low`, `quote_asset_volume` (USDT), `number_of_trades`, `sum_open_interest`, `sum_open_interest_value`.

2. Таблиця `fundamental_crypto_data` Використовується для агрегації фундаментальних показників ринку криптовалют з періодичністю один раз на добу (джерело: `alternative.me`).

Поля: `id`, `date`, `fear_greed_index`, `btc_market_cap`, `eth_market_cap`, `total1` (загальна капіталізація), `total2` (капіталізація без BTC), `total3` (капіталізація без BTC та ETH).

3. Таблиця `fundamental_data` Слугує для зберігання макроекономічних даних, що отримуються через API FRED (Federal Reserve Economic Data) з добовим інтервалом. Таблиця включає понад 30 індикаторів, серед яких:

Інфляція: CPI, Core CPI, PPI.

ВВП та ринок праці: GDP, Unemployment Rate.

Грошова маса та ставки: M1, M2, Fed Rate.

Ринок облігацій: дохідності 2Y, 5Y, 10Y, Yield Spread.

Індекси: S&P500, VIX, VXO.

3.3. Реалізація аналітичної системи

Експертна система виконує прогнозування подальшого руху вартості активу та формує один із трьох торгових сигналів: LONG, SHORT або HOLD. Рішення приймається на основі набору правил, що враховують технічні індикатори, а також окремі фундаментальні показники.

Послідовність роботи системи:

1. Користувач вводить назву фінансового активу на момент часу t .
2. Із бази даних завантажується історична інформація за попередні n годин або днів.
3. Виконується аналіз відповідно до правила індикатора RSI (опис наведено в розділі 2.2.1).
4. Проводиться перевірка умов, пов'язаних з індикатором MACD (розділ 2.2.2).
5. Здійснюється оцінка сигналів на основі експоненціальної ковзної середньої ЕМА (розділ 2.2.3).
6. Аналізується торговий обсяг:

якщо ціна в момент $t-1$ перевищує значення ціни в момент $t-2$ та одночасно обсяг торгів у момент $t-1$ більший за обсяг у момент $t-2$, формується сигнал LONG;

якщо ціна в момент $t-1$ нижча за ціну в момент $t-2$ або обсяг торгів у момент $t-1$ менший за відповідний показник у момент $t-2$, генерується сигнал SHORT.
7. Аналізується показник відкритого інтересу:

якщо ціна активу в момент $t-1$ є вищою за ціну в момент $t-2$ та значення відкритого інтересу в момент $t-1$ також перевищує його значення в момент $t-2$, система формує сигнал LONG;

якщо ціна активу в момент $t-1$ є нижчою за ціну в момент $t-2$ або відкритий інтерес у момент $t-1$ менший, ніж у момент $t-2$, формується сигнал SHORT.

8. У випадку, коли користувач обирає стиль торгівлі Swing або Investing, додатково враховується індекс страху та жадібності:

значення індексу понад 75 інтерпретується як сигнал LONG;

значення індексу нижче 25 інтерпретується як сигнал SHORT.

Для підвищення точності прогнозування система застосовує моделі нейронних мереж, які дозволяють визначати не лише ймовірний напрямок зміни ціни активу, а й орієнтовні рівні входу та виходу з торгової позиції [15]. Завдяки цьому користувач отримує додаткову інформацію для прийняття торгових рішень.

Послідовність роботи модуля прогнозування:

1. Із бази даних завантажуються значення цін закриття (*Close*) за останні 1000 хвилин.
2. Отримані дані приводяться до єдиного масштабу за допомогою методу нормалізації MinMaxScaler.
3. На основі підготовлених даних формуються часові послідовності з використанням ковзного вікна розміром 50 спостережень.
4. Створюються та навчаються три моделі машинного навчання:
 - мережа LSTM (Long Short-Term Memory);
 - мережа GRU (Gated Recurrent Unit);
 - багатошаровий перцептрон MLP (Multi-Layer Perceptron).
5. Після завершення навчання виконується порівняння якості моделей: для оцінювання використовуються показники R^2 , MAE, RMSE та MAPE;

оптимальною вважається модель, яка демонструє найменшу сумарну похибку за метриками MAE, RMSE та MAPE.

6. За допомогою обраної моделі розраховується прогнозне значення ціни на наступний період.
7. На основі отриманого прогнозу визначається цільовий рівень залежно від обраного стилю торгівлі:
 - Scalping – відхилення $\pm 0,2$ %;
 - Intraday – відхилення ± 1 %;
 - Swing – відхилення ± 3 %;
 - Investing – відхилення ± 5 %.

3.4. Практичне використання інформаційної системи

На рисунку 5.1. наведено процес запуску інформаційної системи для трейдингу для користувача, який вперше взаємодіє з системою.

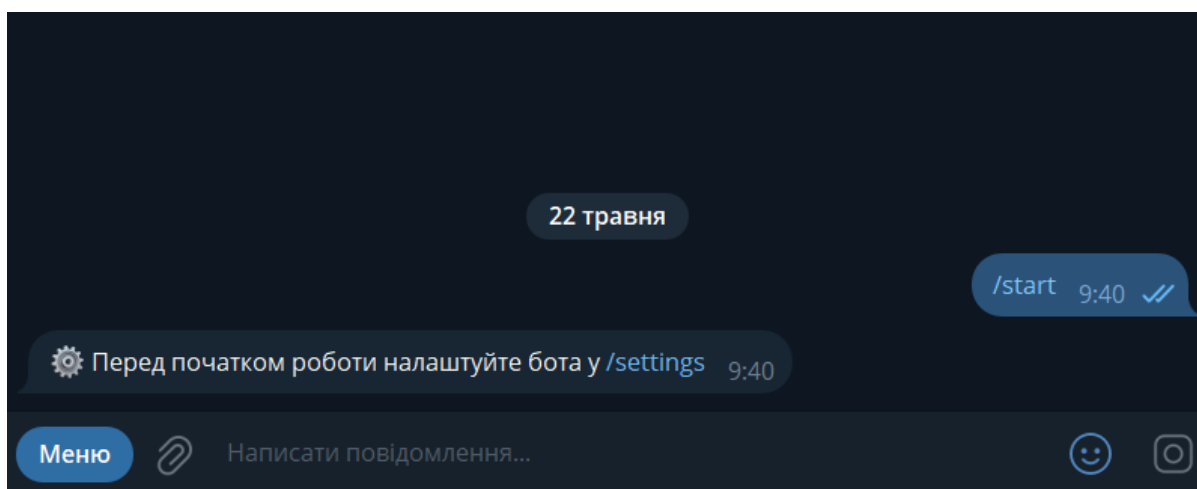


Рис. 5.1. Перший запуск системи

На рисунку 5.2. наведено налаштування системи.



Рис. 5.2. Налаштування системи

На рисунку 5.3 наведено результати роботи експертної системи, що демонструють процес визначення напрямку руху ціни активу на основі механізму голосування між різними аналітичними індикаторами.

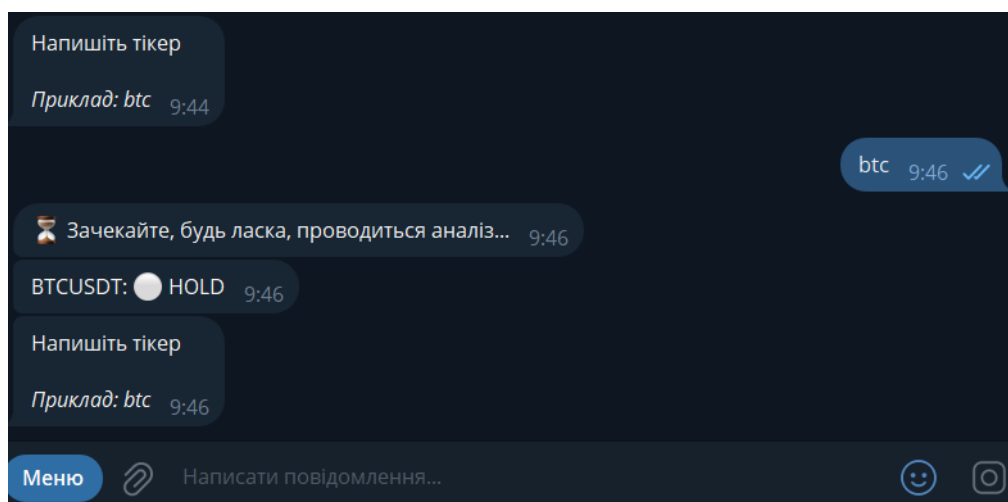


Рис. 5.3. Результат аналізу напрямку руху ціни

На рисунку 5.4 наведено результати роботи нейронних мереж, де візуалізовано конкретні точки входу та виходу з торгової позиції.

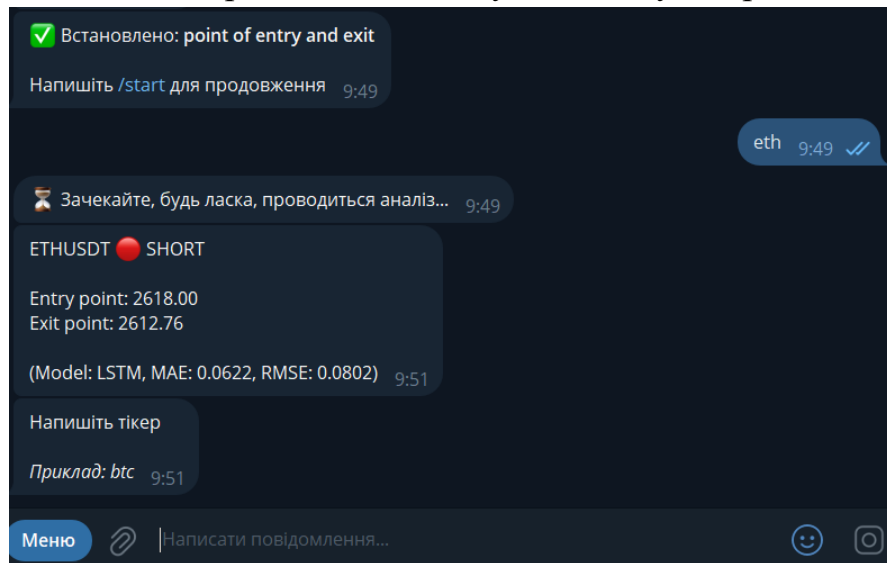


Рис. 5.4. Результат аналізу точки входу та виходу

Висновки до розділу 3

У цьому розділі було розглянуто практичну реалізацію інформаційної системи підтримки прийняття рішень у сфері трейдингу. Проаналізовано структуру програмного забезпечення, до складу якого входять Telegram-бот, модуль керування бізнес-логікою, підсистема отримання даних, моделі штучного інтелекту та система зберігання даних на основі MySQL.

Окрему увагу приділено механізму асинхронного отримання ринкової та фундаментальної інформації з використанням зовнішніх API. Такий підхід забезпечує оперативне оновлення даних та підвищує продуктивність системи під час обробки великої кількості запитів. Також розглянуто реалізацію експертних модулів, що формують торгові рекомендації на основі набору правил, адаптованих до різних стратегій торгівлі, зокрема scalping, intraday, swing та investing.

Крім того, описано застосування нейронних мереж для прогнозування майбутніх значень цінових показників на підставі історичних даних. Використання методів машинного навчання дозволяє визначати потенційні

точки входу та виходу з позицій, що підвищує інформативність торгових сигналів.

Інтеграція асинхронного збору даних, експертних правил, моделей штучного інтелекту та Telegram-інтерфейсу дала змогу створити комплексну систему, здатну надавати актуальні рекомендації в режимі реального часу. Отримані результати підтверджують доцільність поєднання експертних методів та нейромережових алгоритмів для підвищення ефективності процесу аналізу фінансових ринків.

Реалізований модуль збору даних забезпечує повноту та актуальність інформаційної бази системи: щохвилинне оновлення котирувань через Binance Futures API та щоденне поповнення фундаментальних показників через FRED та Alternative.me дозволяють формувати багатовимірний контекст для прийняття торгових рішень. Зберігання даних у реляційній базі MySQL з чіткою схемою таблиць забезпечує надійність, послідовність і масштабованість сховища даних, що є важливою основою для подальшого розширення функціональності системи.

Тестування системи на реальних ринкових даних підтвердило коректність роботи обох аналітичних підсистем. Експертна система коректно формує сигнали на основі технічних індикаторів, адаптуючи пороги й налаштування до обраного торгового стилю. Нейронні мережі (LSTM, GRU, MLP) успішно навчаються на послідовностях із 50 кроків і генерують прогноз наступного значення ціни, а автоматичний вибір кращої моделі за сукупністю метрик MAE, RMSE, MAPE підвищує надійність прогнозу і знижує ризик застосування неоптимальної моделі.

Таким чином, третій розділ продемонстрував технічну реалізованість та ефективність розробленої архітектури. Асинхронна архітектура Python-застосунку, модульність компонентів та інтеграція через Telegram забезпечують зручність використання системи в реальних умовах торгівлі. Отримані результати слугують підставою для проведення функціонально-вартісного аналізу розробленого програмного продукту в наступному розділі.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи було спроектовано та реалізовано інформаційну систему підтримки прийняття рішень у сфері трейдингу. Розроблене програмне рішення поєднує методи технічного аналізу, засновані на використанні ринкових індикаторів, із сучасними технологіями машинного навчання та нейромережевого прогнозування. Завдяки такій інтеграції система здатна здійснювати комплексний аналіз значних обсягів фінансової інформації та генерувати рекомендації щодо торгових дій відповідно до поточної ринкової ситуації.

На початковому етапі дослідження було виконано аналіз предметної області, пов'язаної з біржовою торгівлею. Розглянуто основні підходи до здійснення торговельних операцій, особливості різних стилів трейдингу та характеристики фінансових активів. Також проведено огляд існуючих програмних рішень, призначених для аналізу ринків і підтримки торгових рішень. Отримані результати стали основою для вибору архітектури майбутньої системи та визначення її функціональних можливостей.

Теоретичною основою роботи стали методи інтелектуального аналізу даних та прогнозування часових рядів. Було досліджено принципи функціонування експертних систем, побудованих на індикаторах RSI, MACD та ЕМА, а також проаналізовано особливості використання нейронних мереж типів LSTM, GRU і MLP для прогнозування фінансових показників. Проведений аналіз підтвердив доцільність поєднання правил експертних систем із механізмами глибинного навчання для підвищення якості прогнозів та зменшення впливу ринкової невизначеності.

Практична складова роботи передбачала створення програмного прототипу у форматі Telegram-бота з використанням мови програмування Python та спеціалізованих бібліотек для роботи з даними і машинним навчанням. Реалізована система забезпечує автоматизований збір інформації з зовнішніх джерел, зокрема API Binance, FRED та Alternative.me, виконує її

попередню обробку, аналіз та формування торгових рекомендацій. Для оцінювання ефективності моделей прогнозування було проведено їх тестування на історичних даних. Порівняння результатів за метриками MAE та RMSE дозволило автоматично визначати модель, яка демонструє найкращу якість прогнозування за поточних умов.

У результаті виконання роботи поставлену мету було досягнуто: створено інформаційну систему для трейдингу, яка враховує особливості обраного користувачем стилю торгівлі, аналізує поточний стан ринку та адаптується до змін ринкового середовища. Запропоноване рішення має практичне значення для учасників фінансових ринків, оскільки сприяє підвищенню обґрунтованості торгових рішень та зменшенню часових витрат на аналіз даних.

Перспективами подальшого розвитку системи є розширення набору використовуваних джерел інформації, впровадження механізмів аналізу новинного фону, застосування більш складних та інтерпретованих моделей штучного інтелекту, а також реалізація функцій автоматизованого виконання торгових операцій. Це дозволить підвищити точність прогнозування та розширити сферу практичного застосування розробленої системи.

Результати тестування нейронних мереж на реальних ринкових даних підтвердили їх прогностичний потенціал: серед трьох досліджених архітектур (LSTM, GRU, MLP) кожна продемонструвала здатність відтворювати загальні тенденції цінового руху з прийнятним рівнем похибки. Автоматичний вибір найкращої моделі за метриками MAE, RMSE та MAPE забезпечує адаптивність системи до різних активів та ринкових режимів, що підвищує надійність генерованих торгових рекомендацій.

Наукова новизна роботи полягає у розробці гібридного підходу до побудови інформаційної системи для трейдингу, що поєднує прозору інтерпретовану логіку продукційної експертної системи з адаптивними нейромережевими моделями прогнозування. Запропонована архітектура системи є розширюваною та може бути адаптована для інших фінансових

ринків і торгових інструментів поза межами криптовалютного ринку, що підвищує її наукову та практичну цінність.

Серед обмежень розробленої системи слід відзначити залежність від якості та доступності зовнішніх API, а також те, що нейронні мережі тренуються виключно на цінових рядах без урахування текстових фундаментальних даних (новин, звітів компаній). Подальші дослідження можуть бути спрямовані на інтеграцію аналізу настроїв (sentiment analysis) на основі обробки природної мови, що дозволить врахувати ширший спектр ринкових чинників та підвищити точність прогнозів.

Система реалізує повний цикл аналітичної підтримки: від збору та зберігання ринкових і фундаментальних даних до формування конкретних торгових рекомендацій із зазначенням точок входу та виходу з позиції. Її архітектурна гнучкість та модульність уможливають поетапне впровадження нових функцій — зокрема автоматичного виконання угод, розширеного управління ризиками та персоналізованих сповіщень — без необхідності повного переписування системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. John J. Murphy. Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications. New York: New York Institute of Finance, 1999. 576 p.
2. John C. Hull. Options, Futures, and Other Derivatives. 10th ed. Hoboken, NJ: Pearson Education, 2017. 896 p.
3. Larry Harris. Trading and Exchanges: Market Microstructure for Practitioners. New York: Oxford University Press, 2003. 646 p.
4. Martin J. Pring. Technical Analysis Explained: The Successful Investor's Guide to Spotting Investment Trends and Turning Points. 5th ed. New York: McGraw-Hill, 2014. 684 p.
5. Richard D. Wyckoff. Studies in Tape Reading. Burlington, VT: Traders Press, 2005. 210 p.
6. Peter Steidlmayer, Steve Hawkins. Steidlmayer on Markets: Trading with Market Profile. 2nd ed. Hoboken, NJ: Wiley, 2003. 304 p.
7. Ian H. Witten, Eibe Frank, Mark A. Hall. Data Mining: Practical Machine Learning Tools and Techniques. 3rd ed. Burlington, MA: Morgan Kaufmann, 2011. 664 p.
8. Peter Jackson. Introduction to Expert Systems. 3rd ed. Harlow: Addison-Wesley, 1998. 558 p.
9. Joseph C. Giarratano, Gary D. Riley. Expert Systems: Principles and Programming. 4th ed. Boston: Thomson Course Technology, 2005. 868 p.
10. Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning. Cambridge, MA: MIT Press, 2016. 775 p.
11. Sepp Hochreiter, Jürgen Schmidhuber. Long Short-Term Memory. Neural Computation, 1997, 9(8). P. 1735–1780. URL: <https://doi.org/10.1162/neco.1997.9.8.1735>.
12. Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau et al. Learning Phrase Representations using RNN Encoder–Decoder for

Statistical Machine Translation. Proceedings of EMNLP 2014. P. 1724–1734.

URL: <https://doi.org/10.3115/v1/D14-1179>.

13. Angelo Casolaro, Vincenzo Capone, Gennaro Iannuzzo, Francesco Camastra. Deep Learning for Time Series Forecasting: Advances and Open Problems. Information 2023, 14(11). 598 p. URL: <https://doi.org/10.3390/info14110598>.

14. Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems 32. NeurIPS 2019. P. 8026–8037. URL: <https://arxiv.org/abs/1912.01703>.

15. Yongfeng Wang, Guofeng Yan. Survey on the application of deep learning in algorithmic trading. Data Science in Finance and Economics 2021, 1(4). P. 345–361. URL: <https://doi.org/10.3934/DSFE.2021019>.

16. Ebikella Mienye, Nobert Jere, George Obaido, Ibomoiye Domor Mienye, Kehinde Aruleba. Deep Learning in Finance: A Survey of Applications and Techniques. AI 2024, 5(4). P. 2066–2091. URL: <https://doi.org/10.3390/ai5040101>.

17. Л.О. Шумова, О.І. Рязанцев, С.А. Покришка. Моделі машинного навчання для формування рекомендацій. Вісник Східноукраїнського національного університету імені Володимира Даля 2023, 2(278). С. 96–105. URL: <https://doi.org/10.33216/1998-7927-2023-278-2-96-105>.

18. В.Б. Мокін, М.В. Дратовий. Наука про дані: машинне навчання та інтелектуальний аналіз даних. Вінниця: ВНТУ, 2024. 263 с.

ДОДАТОК А. ЛІСТИНГ ПРОГРАМНОГО ДОДАТКУ

Файл: app/main.py

(Головний модуль запуску додатку)

```
import asyncio
import uvicorn
from app.api.app import create_app
from app.data.crypto_market.base import start_data_collection, assets
from app.data.crypto_market.crypto_fundamental_data import
collect_fundamental_crypto_data
from app.data.fundamental_data import collect_fundamental_data
from app.db.config import settings
from app.db.queries.general import create_tables
from app.bot.telegram_bot import start_bot

async def main():
    await create_tables()

    api_config = uvicorn.Config(
        app=create_app(),
        host=settings.API_HOST,
        port=settings.API_PORT,
        loop="none",
        log_level="info",
    )
    server = uvicorn.Server(api_config)

    await asyncio.gather(
        server.serve(),
        collect_fundamental_data(),
        collect_fundamental_crypto_data(),
        start_bot(),
        *[start_data_collection(asset, "1m") for asset in assets],
    )

if __name__ == "__main__":
    asyncio.run(main())
```

Файл: app/config.py

(Конфігурація та константи)

```
# — Crypto tickers —————

ASSETS: list[str] = [
    "BTCUSDT",
    "ETHUSDT",
]

# — Binance API —————

BINANCE_KLINES_URL = "https://api.binance.com/api/v3/klines"
BINANCE_FUTURES_OI_URL = "https://fapi.binance.com/fapi/v1/openInterest"
BINANCE_WS_URL = "wss://stream.binance.com:9443/ws"

# Скільки секунд кешувати open interest перед повторним запитом до API
OI_CACHE_TTL_SECONDS = 30

# — Alternative.me API —————

FEAR_GREED_URL = "https://api.alternative.me/fng/?limit=0&format=json"
MARKET_METRICS_URL = "https://api.alternative.me/v2/ticker/?structure=array"

# — FRED API —————

FRED_BASE_URL = "https://api.stlouisfed.org/fred/series/observations"
```

```

# Макроекономічні індикатори: назва поля в БД → series_id у FRED
FRED_INDICATORS: dict[str, str] = {
    # Базові
    "cpi": "CPIAUCSL", # Consumer Price Index (1982-1984 =
100)
    "fed_rate": "FEDFUNDS", # Effective Federal Funds Rate (%)
    "m2_global_money_supply": "M2SL", # M2 Money Stock (Billions of $)
    "us_debt": "GFDEBTN", # Total Public Debt (Millions of $)
    "vix_index": "VIXCLS", # CBOE Volatility Index
    "treasury_yield_10y": "GS10", # 10-Year Treasury Yield (%)

    # Економічна активність
    "real_gdp": "GDPC1", # Real GDP (Billions of chained 2012
$)
    "unemployment_rate": "UNRATE", # Unemployment Rate (%)
    "nonfarm_payrolls": "PAYEMS", # Nonfarm Payrolls (Thousands)
    "industrial_production": "INDPRO", # Industrial Production Index
(2017=100)

    # Грошово-кредитна політика
    "m1_money_stock": "M1SL", # M1 Money Stock (Billions of $)
    "fed_target_rate_upper": "DFEDTARU", # Fed Target Rate - Upper Bound (%)
    "total_consumer_credit": "TOTCI", # Total Consumer Credit (Billions of
$)
    "mortgage_rate_30y": "MORTGAGE30US", # 30-Year Fixed Mortgage Rate (%)

    # Бюджет
    "federal_debt_to_gdp": "GFDGDP188S", # Federal Debt as % of GDP

    # Ринки
    "sp500_index": "SP500", # S&P 500 Index
    "vxo_index": "VXOCLS", # VXO Volatility Index (legacy VIX)
    "treasury_yield_2y": "GS2", # 2-Year Treasury Yield (%)
    "treasury_yield_5y": "GS5", # 5-Year Treasury Yield (%)
    "yield_curve_10y_2y": "T10Y2Y", # Yield Curve: 10Y - 2Y Spread (%)

    # Інфляція
    "core_cpi": "CPILFESL", # Core CPI (Excludes Food & Energy)
    "ppi_all_commodities": "PPIACO", # Producer Price Index: All
Commodities
    "import_price_index": "IR", # Import Price Index

    # Торівля
    "trade_balance": "BOPGSTB", # Trade Balance (Millions of $)
    "exports": "EXPGS", # Exports of Goods & Services
(Billions of $)
    "imports": "IMPGS", # Imports of Goods & Services
(Billions of $)

    # Недухомість
    "home_price_index": "CSUSHPINSA", # S&P/Case-Shiller Home Price Index
    "new_home_sales": "HSN1F", # New Home Sales (Thousands)
    "housing_starts": "HOUST", # Housing Starts (Thousands)
}

# — ML моделі (point of entry & exit) —————

ML_SEQ_LENGTH = 50 # кількість свічок у вхідній послідовності
ML_MAX_CANDLES = 1_000 # скільки останніх свічок брати для навчання

# Множник для розрахунку exit point залежно від торгового стилю
ML_EXIT_MULTIPLIERS: dict[str, float] = {
    "scalping": 0.002,
    "intraday": 0.01,
    "swing": 0.03,
    "investing": 0.05,
}

```

Файл: app/api/app.py

(FastAPI застосунок)

```

from fastapi import FastAPI
from app.api.routes import health, tickers, analysis

```

```
def create_app() -> FastAPI:
    app = FastAPI(
        title="Trading Bot API",
        description="REST API для аналізу криптовалютного ринку",
        version="1.0.0",
    )

    app.include_router(health.router)
    app.include_router(tickers.router)
    app.include_router(analysis.router)

    return app
```

Файл: app/api/dependencies.py

(Залежності FastAPI)

```
from typing import AsyncGenerator
from sqlalchemy.ext.asyncio import AsyncSession
from app.db.database import async_session_maker

async def get_db() -> AsyncGenerator[AsyncSession, None]:
    async with async_session_maker() as session:
        yield session
```

Файл: app/api/routes/health.py

(Маршрут перевірки стану)

```
from fastapi import APIRouter

router = APIRouter()

@router.get("/health", tags=["system"])
async def health_check():
    return {"status": "ok"}
```

Файл: app/api/routes/tickers.py

(Маршрут списку тікерів)

```
from fastapi import APIRouter
from app.data.crypto_market.base import assets
from app.schemas.tickers import TickerListResponse

router = APIRouter(prefix="/tickers", tags=["tickers"])

@router.get("", response_model=TickerListResponse)
async def list_tickers():
    return TickerListResponse(tickers=[a.replace("USDT", "") for a in assets])
```

Файл: app/api/routes/analysis.py

(Маршрути аналізу)

```
from fastapi import APIRouter, Depends, HTTPException, Query
from sqlalchemy.ext.asyncio import AsyncSession

from app.api.dependencies import get_db
from app.data.crypto_market.base import assets
from app.schemas.analysis import AnalysisResponse, TradingStyle
```

```

from app.services.crypto_market.price_movement_direction.analysis import
price_movement_direction_analysis
from app.services.crypto_market.point_of_entry_and_exit.analysis import
point_of_entry_and_exit

router = APIRouter(prefix="/analysis", tags=["analysis"])

def _resolve_symbol(symbol: str) -> str:
    upper = symbol.upper()
    return upper if upper.endswith("USDT") else upper + "USDT"

@router.get(
   ("/{symbol}/price-direction",
    response_model=AnalysisResponse,
    summary="Аналіз напрямку руху ціни",
)
)
async def get_price_direction(
    symbol: str,
    trading_style: TradingStyle = Query(..., description="scalping | intraday | swing
| investing"),
    session: AsyncSession = Depends(get_db),
):
    ticker = _resolve_symbol(symbol)
    if ticker not in assets:
        raise HTTPException(status_code=404, detail=f"Ticker '{symbol}' not found")

    result = await price_movement_direction_analysis(ticker, trading_style, session)
    return AnalysisResponse(
        symbol=ticker,
        trading_style=trading_style,
        analysis_type="price_movement_direction",
        result=result,
    )

@router.get(
   ("/{symbol}/entry-exit",
    response_model=AnalysisResponse,
    summary="Визначення точок входу та виходу",
)
)
async def get_entry_exit(
    symbol: str,
    trading_style: TradingStyle = Query(..., description="scalping | intraday | swing
| investing"),
    session: AsyncSession = Depends(get_db),
):
    ticker = _resolve_symbol(symbol)
    if ticker not in assets:
        raise HTTPException(status_code=404, detail=f"Ticker '{symbol}' not found")

    result = await point_of_entry_and_exit(ticker, trading_style, session)
    return AnalysisResponse(
        symbol=ticker,
        trading_style=trading_style,
        analysis_type="point_of_entry_and_exit",
        result=result,
    )

```

Файл: app/bot/telegram_bot.py

(Telegram-бот)

```

from aiogram import Bot, Dispatcher, F
from aiogram.client.default import DefaultBotProperties
from aiogram.filters import CommandStart, Command
from aiogram.types import Message, CallbackQuery, InlineKeyboardMarkup,
InlineKeyboardButton
from app.db.config import settings
from app.db.database import async_session_maker
from app.services.crypto_market.price_movement_direction.analysis import
price_movement_direction_analysis
from app.services.crypto_market.point_of_entry_and_exit.analysis import
point_of_entry_and_exit
from app.data.crypto_market.base import assets
from app.db.queries.users import create_user, get_user, user_update

```

```
bot = Bot(token=settings.TELEGRAM_BOT_TOKEN,
default=DefaultBotProperties(parse_mode="HTML"))
dp = Dispatcher()
```

```
_INFO_TEXT = """
```

```
-----
Ринок - фінансовий ринок, на якому бот буде здійснювати аналіз:
```

- ☐ `<i>crypto</i>` (ринок криптовалют)
- ☐ `<i>stocks</i>` (ринок акцій)
- ☐ `<i>bonds</i>` (ринок облігацій)

```
-----
Тип результату - це параметр, який вказує, який тип аналізу буде проводитися:
```

- ☐ `<i>price movement direction</i>` (аналіз напрямку руху ціни)
- ☐ `<i>point of entry and exit</i>` (визначення точок входу та виходу з ринку з позиції)

```
-----
Торговий стиль - це параметр, який визначає стратегію торгівлі:
```

- ☒ `<i>scalping</i>` (короткострокова торгівля з відкриттям і закриттям угод у межах кількох хвилин)
- ☐ `<i>intraday</i>` (торгівля в межах одного дня, угоди відкриваються та закриваються в той же день)
- ☐ `<i>swing</i>` (середньострокова торгівля, де угоди тривають від кількох днів до кількох тижнів)
- ☐ `<i>investing</i>` (довгострокове інвестування, угоди можуть тривати місяцями або навіть роками)

```
-----
Режим - це параметр, що визначає, режим роботи з ботом:
```

- ☐ `<i>manual</i>` (ви надсилаєте назву тікера, і отримуєте аналіз на свій запит)
- ☐ `<i>auto</i>` (бот автоматично 24/7 надсилає аналіз, коли на ринку з'являється відповідна ситуація)

```
-----
Напишіть /start для продовження
"""
```

```
_REQUIRED_FIELDS = ("market", "result_type", "trading_style", "mode")
```

```
_FIELD_MAP = {
    "crypto": "market",
    "price movement direction": "result_type",
    "point of entry and exit": "result_type",
    "scalping": "trading_style",
    "intraday": "trading_style",
    "swing": "trading_style",
    "investing": "trading_style",
    "manual": "mode",
    "auto": "mode",
}
```

```
def is_configured(user: dict | None) -> bool:
    return bool(user) and all(user.get(f) is not None for f in _REQUIRED_FIELDS)
```

```
@dp.message(F.content_type.in({"photo", "video", "audio", "document", "sticker",
"voice", "location", "contact"}))
async def handle_non_text(message: Message):
    await message.answer("❑ Ви не можете надсилати інформацію цього типу")
```

```

await handle_start(message)

@dp.message(CommandStart())
async def handle_start(message: Message):
    async with async_session_maker() as session:
        await create_user(session, message.from_user.id, message.from_user.username)
        user = await get_user(session, message.from_user.id)

    if not is_configured(user):
        await message.answer("❑ Перед початком роботи налаштуйте бота у /settings")
    else:
        await message.answer("Напишіть тікер \n\n<em>Приклад: btc</em>")

@dp.message(Command("list"))
async def handle_list(message: Message):
    tickers = "\n".join(f"- {a.replace('USDT', '')}" for a in assets)
    await message.answer(f"Список доступних тікерів:\n\n{tickers}")

@dp.message(Command("settings"))
async def handle_settings(message: Message):
    markup = InlineKeyboardMarkup(inline_keyboard=[
        [InlineKeyboardButton(text="❑ Інформація про налаштування",
            callback_data="info")],
        [InlineKeyboardButton(text="❑ Ринок", callback_data="market"),
         InlineKeyboardButton(text="❑ Тип результату", callback_data="result_type")],
        [InlineKeyboardButton(text="❑ Торговий стиль", callback_data="style"),
         InlineKeyboardButton(text="❑ Режим", callback_data="mode")],
    ])

    async with async_session_maker() as session:
        user = await get_user(session, message.from_user.id) or {}

    def safe(val):
        return val if val is not None else "не задано"

    await message.answer(
        f"Поточні налаштування:\n\n"
        f"Ринок: <em>{safe(user.get('market'))}</em>\n"
        f"Тип результату: <em>{safe(user.get('result_type'))}</em>\n"
        f"Торговий стиль: <em>{safe(user.get('trading_style'))}</em>\n"
        f"Режим: <em>{safe(user.get('mode'))}</em>",
        reply_markup=markup,
    )

@dp.callback_query()
async def handle_settings_callback(callback: CallbackQuery):
    data = callback.data
    markup = InlineKeyboardMarkup(inline_keyboard=[])

    if data == "info":
        await callback.message.answer(_INFO_TEXT, reply_markup=markup)

    elif data == "market":
        markup.inline_keyboard.append(
            [InlineKeyboardButton(text="❑ crypto", callback_data="crypto")]
        )
        await callback.message.answer("Оберіть ринок:", reply_markup=markup)

    elif data == "result_type":
        markup.inline_keyboard.extend([
            [InlineKeyboardButton(text="❑ price movement direction",
                callback_data="price movement direction")],
            [InlineKeyboardButton(text="❑ point of entry and exit",
                callback_data="point of entry and exit")],
        ])
        await callback.message.answer("Оберіть тип результату:", reply_markup=markup)

    elif data == "style":
        markup.inline_keyboard.extend([
            [InlineKeyboardButton(text="❌ scalping", callback_data="scalping"),
             InlineKeyboardButton(text="❑ intraday", callback_data="intraday")],
            [InlineKeyboardButton(text="❑ swing", callback_data="swing"),

```

```

        InlineKeyboardButton(text="📈 investing", callback_data="investing")],
    ])
    await callback.message.answer("Оберіть торговий стиль:", reply_markup=markup)

elif data == "mode":
    markup.inline_keyboard.append(
        [InlineKeyboardButton(text="📉 manual", callback_data="manual")]
    )
    await callback.message.answer("Оберіть режим:", reply_markup=markup)

if data in _FIELD_MAP:
    async with async_session_maker() as session:
        await user_update(session, callback.from_user.id, _FIELD_MAP[data], data)
        await callback.message.answer(f"✔️ Встановлено: <b>{data}</b>\n\nНапишіть
/start для продовження")
        await callback.answer()

@dp.message(F.text)
async def handle_text(message: Message):
    ticker_input = message.text.upper()
    ticker = ticker_input if ticker_input.endswith("USDT") else ticker_input + "USDT"

    async with async_session_maker() as session:
        user = await get_user(session, message.from_user.id)

        if not is_configured(user):
            await message.answer("📌 Перед початком роботи налаштуйте бота у
/settings")
            return

        if ticker not in assets:
            await message.answer("📌 Такого тікера не існує")
            return

        await message.answer("📌 Зачекайте, будь ласка, проводиться аналіз...")

        if user["result_type"] == "price movement direction":
            analysis = await price_movement_direction_analysis(ticker,
user.get("trading_style"), session)
        else:
            analysis = await point_of_entry_and_exit(ticker,
user.get("trading_style"), session)

        await message.answer(analysis)
        await handle_start(message)

async def start_bot():
    print("Launching the bot...")
    await dp.start_polling(bot)

```

Файл: app/db/config.py

(Конфігурація бази даних)

```

from pydantic_settings import BaseSettings, SettingsConfigDict

class Settings(BaseSettings):
    TELEGRAM_BOT_TOKEN: str
    FRED_API_KEY: str

    DB_HOST: str = "localhost"
    DB_PORT: int = 3306
    DB_USER: str = "root"
    DB_PASSWORD: str = "root"
    DB_NAME: str = "trading_bot"
    DB_ECHO: bool = False

    API_HOST: str = "0.0.0.0"
    API_PORT: int = 8000

```

```

@property
def DATABASE_URL_aiomysql(self) -> str:
    return (
        f"mysql+aiomysql://{self.DB_USER}:{self.DB_PASSWORD}"
        f"@{self.DB_HOST}:{self.DB_PORT}/{self.DB_NAME}"
    )

model_config = SettingsConfigDict(env_file=".env", extra="ignore")

settings = Settings()

```

Файл: app/db/database.py

(Підключення до бази даних)

```

from sqlalchemy.ext.asyncio import create_async_engine, async_sessionmaker,
AsyncSession
from app.db.config import settings

async_engine = create_async_engine(
    url=settings.DATABASE_URL_aiomysql,
    echo=settings.DB_ECHO,
    future=True,
    pool_pre_ping=True,
)

async_session_maker = async_sessionmaker(
    bind=async_engine,
    expire_on_commit=False,
    class_=AsyncSession,
)

```

Файл: app/db/models.py

(Моделі бази даних)

```

from sqlalchemy import Table, Column, Integer, MetaData, VARCHAR, String, Float,
BigInteger, Date

metadata_obj = MetaData()

users_table = Table(
    "users",
    metadata_obj,
    Column("user_id", Integer, primary_key=True, unique=True, autoincrement=True),
    Column("telegram_user_id", BigInteger),
    Column("telegram_user_name", VARCHAR(64)),
    Column("market", VARCHAR(64)),
    Column("result_type", VARCHAR(64)),
    Column("trading_style", VARCHAR(64)),
    Column("mode", VARCHAR(64)),
)

crypto_tickers_data_table = Table(
    "crypto_tickers_data",
    metadata_obj,
    Column("id", Integer, primary_key=True, autoincrement=True),
    Column("symbol", String(16)),
    Column("timestamp", BigInteger),
    Column("open", Float),
    Column("close", Float),
    Column("high", Float),
    Column("low", Float),
    Column("quote_asset_volume", Float),
    Column("number_of_trades", Integer),
    Column("sum_open_interest", Float),
    Column("sum_open_interest_value", Float),
)

fundamental_data_table = Table(
    "fundamental_data",
    metadata_obj,
    Column("id", Integer, primary_key=True, autoincrement=True),

```

```

Column("date", Date, unique=True),

Column("cpi", Float),
Column("fed_rate", Float),
Column("m2_global_money_supply", Float),
Column("us_debt", Float),
Column("vix_index", Float),
Column("treasury_yield_10y", Float),

Column("real_gdp", Float),
Column("unemployment_rate", Float),
Column("nonfarm_payrolls", Float),
Column("industrial_production", Float),

Column("m1_money_stock", Float),
Column("fed_target_rate_upper", Float),
Column("total_consumer_credit", Float),
Column("mortgage_rate_30y", Float),

Column("federal_debt_to_gdp", Float),

Column("sp500_index", Float),
Column("vxo_index", Float),
Column("treasury_yield_2y", Float),
Column("treasury_yield_5y", Float),
Column("yield_curve_10y_2y", Float),

Column("core_cpi", Float),
Column("ppi_all_commodities", Float),
Column("import_price_index", Float),

Column("trade_balance", Float),
Column("exports", Float),
Column("imports", Float),

Column("home_price_index", Float),
Column("new_home_sales", Float),
Column("housing_starts", Float),
)

fundamental_crypto_data_table = Table(
    "fundamental_crypto_data",
    metadata_obj,
    Column("id", Integer, primary_key=True, autoincrement=True),
    Column("date", Date, unique=True),
    Column("fear_greed_index", Float),
    Column("btc_market_cap", Float),
    Column("eth_market_cap", Float),
    Column("total1", Float),
    Column("total2", Float),
    Column("total3", Float),
)

```

Файл: app/db/queries/general.py

(Загальні запити до БД)

```

from app.db.database import async_engine
from app.db.models import metadata_obj

async def create_tables() -> None:
    async with async_engine.begin() as conn:
        await conn.run_sync(metadata_obj.create_all)

```

Файл: app/db/queries/users.py

(Запити користувачів)

```

from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy import select, insert, update, delete
from app.db.models import users_table

```

```

async def create_user(session: AsyncSession, user_id: int, user_name: str) -> None:
    stmt = select(users_table.c.user_id).where(users_table.c.telegram_user_id ==
user_id)
    result = await session.execute(stmt)
    if result.first() is None:
        await session.execute(
            insert(users_table).values(telegram_user_id=user_id,
telegram_user_name=user_name)
        )
        await session.commit()

async def get_user(session: AsyncSession, user_id: int) -> dict | None:
    stmt = select(users_table).where(users_table.c.telegram_user_id == user_id)
    result = await session.execute(stmt)
    return result.mappings().first()

async def user_update(session: AsyncSession, user_id: int, field: str, value: str) ->
None:
    if field not in {"market", "result_type", "trading_style", "mode"}:
        raise ValueError("Invalid field name")
    stmt = (
        update(users_table)
        .where(users_table.c.telegram_user_id == user_id)
        .values({field: value})
    )
    await session.execute(stmt)
    await session.commit()

async def delete_user(session: AsyncSession, user_id: int) -> None:
    stmt = delete(users_table).where(users_table.c.telegram_user_id == user_id)
    await session.execute(stmt)
    await session.commit()

```

Файл: app/db/queries/crypto_tickers.py

(Запити криптовалютних даних)

```

from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy import select, desc
from app.db.models import crypto_tickers_data_table

async def get_all_crypto_tickers(session: AsyncSession):
    result = await session.execute(select(crypto_tickers_data_table))
    return result.mappings().all()

async def get_all_crypto_tickers_by_symbol(session: AsyncSession, symbol: str):
    stmt = select(crypto_tickers_data_table).where(crypto_tickers_data_table.c.symbol
== symbol)
    result = await session.execute(stmt)
    return result.mappings().all()

async def get_all_close_prices_by_symbol(session: AsyncSession, symbol: str):
    stmt = (
        select(crypto_tickers_data_table.c.close)
        .where(crypto_tickers_data_table.c.symbol == symbol)
        .order_by(crypto_tickers_data_table.c.timestamp)
    )
    result = await session.execute(stmt)
    return result

async def get_all_ohlc_by_symbol(session: AsyncSession, symbol: str):
    stmt = (
        select(
            crypto_tickers_data_table.c.open,

```

```

        crypto_tickers_data_table.c.close,
        crypto_tickers_data_table.c.high,
        crypto_tickers_data_table.c.low,
    )
    .where(crypto_tickers_data_table.c.symbol == symbol)
    .order_by(crypto_tickers_data_table.c.timestamp)
)
result = await session.execute(stmt)
return result

async def get_all_crypto_tickers_by_symbol_and_from_timestamp(
    session: AsyncSession, symbol: str, start_ts: int
):
    stmt = (
        select(crypto_tickers_data_table)
        .where(crypto_tickers_data_table.c.symbol == symbol)
        .where(crypto_tickers_data_table.c.timestamp >= start_ts)
        .order_by(desc(crypto_tickers_data_table.c.timestamp))
    )
    result = await session.execute(stmt)
    return result.mappings().all()

```

Файл: app/db/queries/fundamental_crypto_data.py

(Запити фундаментальних крипто-даних)

```

from datetime import date
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy import select
from app.db.models import fundamental_crypto_data_table

async def get_all_fundamental_crypto_data(session: AsyncSession):
    result = await session.execute(select(fundamental_crypto_data_table))
    return result.mappings().all()

async def get_fear_and_greed_index_by_date(session: AsyncSession, date_input: date):
    stmt = (
        select(fundamental_crypto_data_table.c.fear_greed_index)
        .where(fundamental_crypto_data_table.c.date == date_input)
    )
    result = await session.execute(stmt)
    return result.scalar_one_or_none()

```

Файл: app/schemas/analysis.py

(Схеми аналізу)

```

from typing import Literal
from pydantic import BaseModel

TradingStyle = Literal["scalping", "intraday", "swing", "investing"]
AnalysisType = Literal["price_movement_direction", "point_of_entry_and_exit"]

class AnalysisRequest(BaseModel):
    symbol: str
    trading_style: TradingStyle
    analysis_type: AnalysisType

class AnalysisResponse(BaseModel):
    symbol: str
    trading_style: TradingStyle
    analysis_type: AnalysisType
    result: str

```

Файл: app/schemas/tickers.py*(Схеми тикерів)*

```
from pydantic import BaseModel
```

```
class TickerListResponse(BaseModel):
    tickers: list[str]
```

Файл: app/data/fundamental_data.py*(Збір фундаментальних макроданих)*

```
from collections import defaultdict
from datetime import date, datetime, timedelta
import aiohttp
from sqlalchemy import select, func, text
from sqlalchemy.dialects.mysql import insert
from app.config import FRED_BASE_URL, FRED_INDICATORS
from app.db.config import settings
from app.db.database import async_engine
from app.db.models import fundamental_data_table

async def _fetch_indicator(session, series_id: str, start_date: date, end_date: date)
-> dict:
    params = {
        "series_id": series_id,
        "api_key": settings.FRED_API_KEY,
        "file_type": "json",
        "observation_start": start_date.isoformat(),
        "observation_end": end_date.isoformat(),
    }
    try:
        async with session.get(FRED_BASE_URL, params=params) as resp:
            data = await resp.json()
            if "observations" not in data:
                print(f"Error fetching {series_id}: {data}")
                return {}
            return {
                obs["date"]: float(obs["value"]) if obs["value"] not in ("", ".") else
                None
                for obs in data["observations"]
            }
    except Exception as e:
        print(f"Exception while fetching {series_id}: {e}")
        return {}

async def collect_fundamental_data():
    today = date.today()

    async with async_engine.begin() as conn:
        result = await conn.execute(select(func.max(fundamental_data_table.c.date)))
        last_date = result.scalar()

    start_date = date(2017, 1, 1) if last_date is None else last_date +
    timedelta(days=1)
    if start_date > today:
        print("Fundamental data is already up to date.")
        return

    print(f"Collecting fundamental data from {start_date} to {today}...")

    async with aiohttp.ClientSession() as session:
        indicator_data = {}
        for name, series_id in FRED_INDICATORS.items():
            data = await _fetch_indicator(session, series_id, start_date, today)
            if data:
                indicator_data[name] = data
            else:
                print(f"Indicator '{name}' returned no data.")

    if not indicator_data:
        print("No indicator data received. Skipping save.")
```

```

        return

    try:
        min_date = datetime.strptime(min(min(d) for d in indicator_data.values()),
"%Y-%m-%d").date()
        max_date = datetime.strptime(max(max(d) for d in indicator_data.values()),
"%Y-%m-%d").date()
    except ValueError:
        print("Cannot determine date range. Skipping save.")
        return

    effective_start = max(min_date, start_date)
    date_range = [
        effective_start + timedelta(days=i)
        for i in range((max_date - effective_start).days + 1)
    ]
    last_values: dict = defaultdict(lambda: None)

    # Пре-заповнюємо last_values значеннями до effective_start
    for current_date in [min_date + timedelta(days=i) for i in range((effective_start
- min_date).days)]:
        for name in FRED_INDICATORS:
            value = indicator_data.get(name, {}).get(current_date.isoformat())
            if value is not None:
                last_values[name] = value

    rows = []
    for current_date in date_range:
        row: dict = {"date": current_date}
        for name in FRED_INDICATORS:
            value = indicator_data.get(name, {}).get(current_date.isoformat())
            if value is not None:
                last_values[name] = value
            row[name] = last_values[name]
        rows.append(row)

    if not rows:
        print("No new fundamental data to save.")
        return

    update_dict = {
        col.key: text(f"VALUES(`{col.key}`)")
        for col in fundamental_data_table.c
        if col.key not in ("id", "date")
    }
    stmt = insert(fundamental_data_table).on_duplicate_key_update(**update_dict)
    async with async_engine.begin() as conn:
        await conn.execute(stmt, rows)

    print(f"Upserted {len(rows)} rows of fundamental data.")

```

Файл: app/data/crypto_market/base.py

(Збір OHLC даних з Binance)

```

import asyncio
import aiohttp
import json
import websockets
from datetime import datetime, timedelta
from sqlalchemy import insert
from app.config import ASSETS, BINANCE_KLINES_URL, BINANCE_FUTURES_OI_URL,
BINANCE_WS_URL, OI_CACHE_TTL_SECONDS
from app.db.database import async_engine
from app.db.models import crypto_tickers_data_table

assets = ASSETS

_oi_cache: dict[str, tuple[float, datetime]] = {}

async def fetch_open_interest(symbol: str) -> float:
    now = datetime.utcnow()
    if symbol in _oi_cache:
        cached_value, ts = _oi_cache[symbol]
        if now - ts < timedelta(seconds=OI_CACHE_TTL_SECONDS):

```

```

        return cached_value

    async with aiohttp.ClientSession() as session:
        try:
            async with session.get(BINANCE_FUTURES_OI_URL, params={"symbol": symbol})
as resp:
            data = await resp.json()
            oi = float(data["openInterest"])
            _oi_cache[symbol] = (oi, now)
            return oi
        except Exception as e:
            print(f"[{symbol}] OI fetch error: {e}")
            return 0.0

async def collect_historical_data(symbol: str, interval: str = "1m"):
    print(f"[{symbol}] Collecting historical data...")

    async with async_engine.begin() as conn:
        result = await conn.execute(
            crypto_tickers_data_table.select()
            .where(crypto_tickers_data_table.c.symbol == symbol)
            .order_by(crypto_tickers_data_table.c.timestamp.desc())
            .limit(1)
        )
        row = result.fetchone()
        start_time = row.timestamp + 1 if row else 0

    while True:
        url =
f"{BINANCE_KLINES_URL}?symbol={symbol}&interval={interval}&startTime={start_time}&limi
t=1000"
        async with aiohttp.ClientSession() as session:
            async with session.get(url) as resp:
                data = await resp.json()

        if not data or "code" in data:
            break

        rows = []
        for d in data:
            try:
                oi = await fetch_open_interest(symbol)
            except Exception as e:
                print(f"[{symbol}] OI error: {e}")
                oi = 0.0

            close_price = float(d[4])
            rows.append({
                "symbol": symbol,
                "timestamp": d[0],
                "open": float(d[1]),
                "high": float(d[2]),
                "low": float(d[3]),
                "close": close_price,
                "quote_asset_volume": float(d[7]),
                "number_of_trades": int(d[8]),
                "sum_open_interest": oi,
                "sum_open_interest_value": oi * close_price,
            })

        async with async_engine.begin() as conn:
            await conn.execute(insert(crypto_tickers_data_table), rows)
        print(f"[{symbol}] Inserted {len(rows)} rows")

        start_time = data[-1][0] + 60_000
        await asyncio.sleep(1)

    print(f"[{symbol}] Historical data collection complete.")

async def collect_live_data(symbol: str, interval: str):
    stream_name = f"{symbol.lower()}@kline_{interval}"
    url = f"{BINANCE_WS_URL}/{stream_name}"
    retry_delay = 5

    while True:

```

```

try:
    async with websockets.connect(url) as websocket:
        print(f"[{symbol}] WebSocket connected")
        retry_delay = 5
        while True:
            message = await websocket.recv()
            data = json.loads(message)
            k = data["k"]
            if not k["x"]:
                continue

            try:
                oi = await fetch_open_interest(symbol)
            except Exception as e:
                print(f"[{symbol}] Live OI error: {e}")
                oi = 0.0

            close_price = float(k["c"])
            async with async_engine.begin() as conn:
                await conn.execute(
                    insert(crypto_tickers_data_table).values(
                        symbol=symbol,
                        timestamp=k["t"],
                        open=float(k["o"]),
                        high=float(k["h"]),
                        low=float(k["l"]),
                        close=close_price,
                        quote_asset_volume=float(k["q"]),
                        number_of_trades=int(k["n"]),
                        sum_open_interest=oi,
                        sum_open_interest_value=oi * close_price,
                    )
                )
            except Exception as e:
                print(f"[{symbol}] WebSocket error: {e}. Reconnecting in
{retry_delay}s...")
                await asyncio.sleep(retry_delay)
                retry_delay = min(retry_delay * 2, 60)

async def start_data_collection(symbol: str, interval: str):
    print(f"[{symbol}] Starting data collection...")
    await collect_historical_data(symbol, interval)
    await collect_live_data(symbol, interval)

```

Файл: app/data/crypto_market/crypto_fundamental_data.py

(Збір Fear & Greed Index та ринкової капіталізації)

```

import asyncio
from datetime import datetime, date
import aiohttp
from sqlalchemy import select
from sqlalchemy.dialects.mysql import insert
from app.config import FEAR_GREED_URL, MARKET_METRICS_URL
from app.db.database import async_session_maker
from app.db.models import fundamental_crypto_data_table

async def fetch_fear_greed_data():
    async with aiohttp.ClientSession() as session:
        async with session.get(FEAR_GREED_URL) as response:
            data = await response.json()
            return data["data"]

async def fetch_market_metrics():
    async with aiohttp.ClientSession() as session:
        async with session.get(MARKET_METRICS_URL) as response:
            data = await response.json()
            return data["data"]

async def _upsert_entry(session, record date: date, fg_value: float,
                        btc_cap=None, eth_cap=None,
                        total1=None, total2=None, total3=None):

```

```

stmt = insert(fundamental_crypto_data_table).values(
    date=record_date,
    fear_greed_index=fg_value,
    btc_market_cap=btc_cap,
    eth_market_cap=eth_cap,
    total1=total1,
    total2=total2,
    total3=total3,
).on duplicate key update(
    fear_greed_index=fg_value,
    btc_market_cap=btc_cap,
    eth_market_cap=eth_cap,
    total1=total1,
    total2=total2,
    total3=total3,
)
await session.execute(stmt)
await session.commit()

async def collect_historical_fundamental_crypto_data():
    fg_data = await fetch_fear_greed_data()
    async with async_session_maker() as session:
        for record in reversed(fg_data):
            record_date = datetime.fromtimestamp(int(record["timestamp"])).date()
            value = float(record["value"])
            stmt = select(fundamental_crypto_data_table.c.id).where(
                fundamental_crypto_data_table.c.date == record_date
            )
            if not (await session.execute(stmt)).scalar():
                await _upsert_entry(session, record_date, fg_value=value)

async def collect_live_fundamental_crypto_data():
    while True:
        print("[LIVE] Collecting fundamental crypto data")
        fg_data = await fetch_fear_greed_data()
        latest = fg_data[0]
        record_date = datetime.fromtimestamp(int(latest["timestamp"])).date()
        fg_value = float(latest["value"])

        market_data = await fetch_market_metrics()
        btc = next((x for x in market_data if x["name"] == "Bitcoin"), None)
        eth = next((x for x in market_data if x["name"] == "Ethereum"), None)

        btc_cap = btc.get("quotes", {}).get("USD", {}).get("market_cap") if btc else None
        eth_cap = eth.get("quotes", {}).get("USD", {}).get("market_cap") if eth else None
        total1 = sum(float(c.get("quotes", {}).get("USD", {}).get("market_cap", 0))
            for c in market_data)
        total2 = total1 - btc_cap if btc_cap else None
        total3 = total2 - eth_cap if total2 and eth_cap else None

        async with async_session_maker() as session:
            await _upsert_entry(session, record_date, fg_value,
                btc_cap=btc_cap, eth_cap=eth_cap,
                total1=total1, total2=total2, total3=total3)

        await asyncio.sleep(86400)

async def collect_fundamental_crypto_data():
    await collect_historical_fundamental_crypto_data()
    asyncio.create_task(collect_live_fundamental_crypto_data())

```

Файл: app/services/crypto_market/price_movement_direction/indicators.py

(Технічні індикатори RSI, MACD, EMA)

```
import pandas as pd
```

```
def calculate_ema(series: pd.Series, period: int = 20) -> pd.Series:
    return series.ewm(span=period, adjust=False).mean()
```

```
def calculate_macd(
    close_prices: pd.Series,
    fast_ema_period: int = 12,
    slow_ema_period: int = 26,
    signal_ema_period: int = 9,
):
    fast_ema = calculate_ema(close_prices, fast_ema_period)
    slow_ema = calculate_ema(close_prices, slow_ema_period)
    macd_line = fast_ema - slow_ema
    signal_line = calculate_ema(macd_line, signal_ema_period)
    histogram = macd_line - signal_line
    return macd_line, signal_line, histogram

def calculate_rsi(close_prices: pd.Series, period: int = 14) -> pd.Series:
    delta = close_prices.diff()
    gain = delta.where(delta > 0, 0).rolling(window=period).mean()
    loss = (-delta).where(delta < 0, 0).rolling(window=period).mean()
    rs = gain / loss
    return 100 - (100 / (1 + rs))
```

Файл: app/services/crypto_market/price_movement_direction/base.py

(Базовий аналіз напрямку руху ціни)

```
from dataclasses import dataclass
from datetime import datetime, timedelta, date
import pandas as pd
from sqlalchemy.ext.asyncio import AsyncSession

from app.services.crypto_market.price_movement_direction.indicators import (
    calculate_rsi, calculate_macd, calculate_ema,
)
from app.db.queries.crypto_tickers import
get_all_crypto_tickers_by_symbol_and_from_timestamp
from app.db.queries.fundamental_crypto_data import get_fear_and_greed_index_by_date

@dataclass
class AnalysisConfig:
    window_hours: int
    rsi_period: int
    rsi_oversold: float
    rsi_overbought: float
    macd_fast: int
    macd_slow: int
    macd_signal: int
    ema_periods: list[int]
    use_fear_greed: bool = False
    fear_greed_extreme: float = 25.0
    fear_greed_greed: float = 75.0

async def run_analysis(symbol: str, session: AsyncSession, cfg: AnalysisConfig) ->
str:
    start_ts = int((datetime.now() - timedelta(hours=cfg.window_hours)).timestamp() *
1000)
    rows = await get_all_crypto_tickers_by_symbol_and_from_timestamp(session, symbol,
start_ts)

    if len(rows) < 2:
        return f"❑ Недостатньо даних для аналізу {symbol}"

    df = pd.DataFrame(rows)
    df["timestamp"] = pd.to_datetime(df["timestamp"], unit="ms")
    df.set_index("timestamp", inplace=True)
    close = df["close"]

    latest, previous = rows[0], rows[1]
    votes = {"LONG": 0, "SHORT": 0, "HOLD": 0}
    explanation = "❑ Пояснення:\n\n"

    rsi = calculate_rsi(close, cfg.rsi_period).iloc[-1]
```

```

if rsi < cfg.rsi_oversold:
    votes["LONG"] += 1
    explanation += f"✓ RSI ({rsi:.2f}) вказує на перепроданість – сигнал LONG\n"
elif rsi > cfg.rsi_overbought:
    votes["SHORT"] += 1
    explanation += f"□ RSI ({rsi:.2f}) вказує на перекупленість – сигнал SHORT\n"
else:
    votes["HOLD"] += 1
    explanation += f"□ RSI ({rsi:.2f}) в нейтральній зоні – сигнал HOLD\n"

macd_line, signal_line, _ = calculate_macd(close, cfg.macd_fast, cfg.macd_slow,
cfg.macd_signal)
m, s = macd_line.iloc[-1], signal_line.iloc[-1]
if m > s:
    votes["LONG"] += 1
    explanation += f"✓ MACD ({m:.2f}) вище сигнальної ({s:.2f}) – сигнал LONG\n"
elif m < s:
    votes["SHORT"] += 1
    explanation += f"□ MACD ({m:.2f}) нижче сигнальної ({s:.2f}) – сигнал SHORT\n"
else:
    votes["HOLD"] += 1
    explanation += f"□ MACD ≈ сигнальна лінія – сигнал HOLD\n"

for period in cfg.ema_periods:
    ema = calculate_ema(close, period)
    price, ema_val = close.iloc[-1], ema.iloc[-1]
    if price > ema_val:
        votes["LONG"] += 1
        explanation += f"✓ Ціна ({price:.2f}) вище EMA{period} ({ema_val:.2f}) –
сигнал LONG\n"
    elif price < ema_val:
        votes["SHORT"] += 1
        explanation += f"□ Ціна ({price:.2f}) нижче EMA{period} ({ema_val:.2f}) –
сигнал SHORT\n"
    else:
        votes["HOLD"] += 1
        explanation += f"□ Ціна ≈ EMA{period} – сигнал HOLD\n"

if latest["close"] > previous["close"]:
    if latest["quote_asset_volume"] > previous["quote_asset_volume"]:
        votes["LONG"] += 1
        explanation += f"✓ Ціна зросла разом із об'ємом – сигнал LONG\n"
    else:
        votes["SHORT"] += 1
        explanation += f"□ Ціна зросла, але об'єм впав – сигнал SHORT\n"

if latest["close"] > previous["close"]:
    if latest["sum_open_interest"] > previous["sum_open_interest"]:
        votes["LONG"] += 1
        explanation += f"✓ Ціна зросла разом із відкритим інтересом – сигнал
LONG\n"
    else:
        votes["SHORT"] += 1
        explanation += f"□ Ціна зросла, але відкритий інтерес впав – сигнал
SHORT\n"

if cfg.use_fear_greed:
    fg = await get_fear_and_greed_index_by_date(session, date.today())
    if fg is not None:
        if fg > cfg.fear_greed_greed:
            votes["SHORT"] += 1
            explanation += f"□ Fear & Greed Index = {fg}, ринок жадібний – сигнал
SHORT\n"
        elif fg < cfg.fear_greed_extreme:
            votes["LONG"] += 1
            explanation += f"✓ Fear & Greed Index = {fg}, ринок боязкий – сигнал
LONG\n"
        else:
            votes["HOLD"] += 1
            explanation += f"□ Fear & Greed Index = {fg}, ринок нейтральний –
сигнал HOLD\n"

if votes["LONG"] > votes["SHORT"]:
    verdict = f"{symbol.upper()} □ LONG\n\n"
elif votes["SHORT"] > votes["LONG"]:
    verdict = f"{symbol.upper()} □ SHORT\n\n"
else:
    verdict = f"{symbol.upper()} □ HOLD\n\n"

return verdict + explanation

```

Файл: app/services/crypto_market/price_movement_direction/analysis.py*(Диспетчер аналізу за стилем торгівлі)*

```

from sqlalchemy.ext.asyncio import AsyncSession
from app.services.crypto_market.price_movement_direction.scalping import
scalping_analysis
from app.services.crypto_market.price_movement_direction.intraday import
intraday_analysis
from app.services.crypto_market.price_movement_direction.swing import swing_analysis
from app.services.crypto_market.price_movement_direction.investing import
investing_analysis

_HANDLERS = {
    "scalping": scalping_analysis,
    "intraday": intraday_analysis,
    "swing": swing_analysis,
    "investing": investing_analysis,
}

async def price_movement_direction_analysis(
    symbol: str, trading_style: str, session: AsyncSession
) -> str:
    handler = _HANDLERS.get(trading_style)
    if handler is None:
        return f"❑ Невідомий торговий стиль: {trading_style}"
    return await handler(symbol, session)

```

Файл: app/services/crypto_market/price_movement_direction/scalping.py*(Аналіз для скальпінгу)*

```

from sqlalchemy.ext.asyncio import AsyncSession
from app.services.crypto_market.price_movement_direction.base import run_analysis,
AnalysisConfig

_CFG = AnalysisConfig(
    window_hours=6,
    rsi_period=7,
    rsi_oversold=20,
    rsi_overbought=80,
    macd_fast=6,
    macd_slow=13,
    macd_signal=5,
    ema_periods=[9, 20, 50],
    use_fear_greed=False,
)

async def scalping_analysis(symbol: str, session: AsyncSession) -> str:
    return await run_analysis(symbol, session, _CFG)

```

Файл: app/services/crypto_market/price_movement_direction/intraday.py*(Аналіз для внутрішньоденної торгівлі)*

```

from sqlalchemy.ext.asyncio import AsyncSession
from app.services.crypto_market.price_movement_direction.base import run_analysis,
AnalysisConfig

_CFG = AnalysisConfig(
    window_hours=7 * 24,
    rsi_period=14,
    rsi_oversold=20,
    rsi_overbought=80,
    macd_fast=8,
    macd_slow=21,
    macd_signal=7,
    ema_periods=[12, 26, 100],
    use_fear_greed=False,
)

```

)

```
async def intraday_analysis(symbol: str, session: AsyncSession) -> str:
    return await run_analysis(symbol, session, _CFG)
```

Файл: app/services/crypto_market/price_movement_direction/swing.py

(Аналіз для свінг-торгівлі)

```
from sqlalchemy.ext.asyncio import AsyncSession
from app.services.crypto_market.price_movement_direction.base import run_analysis,
AnalysisConfig
```

```
_CFG = AnalysisConfig(
    window_hours=30 * 24,
    rsi_period=21,
    rsi_oversold=30,
    rsi_overbought=70,
    macd_fast=12,
    macd_slow=26,
    macd_signal=9,
    ema_periods=[20, 50, 100],
    use_fear_greed=True,
)
```

```
async def swing_analysis(symbol: str, session: AsyncSession) -> str:
    return await run_analysis(symbol, session, _CFG)
```

Файл: app/services/crypto_market/price_movement_direction/investing.py

(Аналіз для інвестування)

```
from sqlalchemy.ext.asyncio import AsyncSession
from app.services.crypto_market.price_movement_direction.base import run_analysis,
AnalysisConfig
```

```
_CFG = AnalysisConfig(
    window_hours=180 * 24,
    rsi_period=28,
    rsi_oversold=30,
    rsi_overbought=70,
    macd_fast=19,
    macd_slow=39,
    macd_signal=12,
    ema_periods=[50, 100, 200],
    use_fear_greed=True,
)
```

```
async def investing_analysis(symbol: str, session: AsyncSession) -> str:
    return await run_analysis(symbol, session, _CFG)
```

Файл: app/services/crypto_market/point_of_entry_and_exit/models_build.py

(Архітектури нейронних мереж LSTM, GRU, MLP)

```
import torch
import torch.nn as nn
```

```
class LSTMModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.lstm1 = nn.LSTM(1, 64, batch_first=True)
        self.lstm2 = nn.LSTM(64, 32, batch_first=True)
        self.fc = nn.Linear(32, 1)
```

```

def forward(self, x):
    out, _ = self.lstm1(x)
    out, _ = self.lstm2(out)
    return self.fc(out[:, -1, :])

class GRUModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.gru1 = nn.GRU(1, 64, batch_first=True)
        self.gru2 = nn.GRU(64, 32, batch_first=True)
        self.fc = nn.Linear(32, 1)

    def forward(self, x):
        out, _ = self.gru1(x)
        out, _ = self.gru2(out)
        return self.fc(out[:, -1, :])

class MLPModel(nn.Module):
    def __init__(self, seq_length: int = 50):
        super().__init__()
        self.net = nn.Sequential(
            nn.Flatten(),
            nn.Linear(seq_length, 128),
            nn.ReLU(),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Linear(64, 1),
        )

    def forward(self, x):
        return self.net(x)

```

Файл: app/services/crypto_market/point_of_entry_and_exit/other.py

(Допоміжні утиліти для ML моделей)

```

import numpy as np

def create_sequences(data, seq_length: int = 50):
    xs, ys = [], []
    for i in range(len(data) - seq_length):
        xs.append(data[i:i + seq_length])
        ys.append(data[i + seq_length])
    return np.array(xs), np.array(ys)

```

Файл: app/services/crypto_market/point_of_entry_and_exit/analysis.py

(ML-аналіз точок входу та виходу)

```

import numpy as np
import torch
import torch.nn as nn
from torch.utils.data import TensorDataset, DataLoader
from math import sqrt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sqlalchemy.ext.asyncio import AsyncSession

from app.config import ML_SEQ_LENGTH, ML_MAX CANDLES, ML_EXIT MULTIPLIERS
from app.services.crypto_market.point_of_entry_and_exit.models_build import LSTMModel,
GRUModel, MLPModel
from app.services.crypto_market.point_of_entry_and_exit.other import create_sequences
from app.db.queries.crypto_tickers import get_all_close_prices_by_symbol

def _mape(y_true: np.ndarray, y_pred: np.ndarray) -> float:
    mask = y_true != 0
    if not mask.any():
        return np.inf
    return float(np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])))

```

```

def _train_and_predict(model: nn.Module, x: np.ndarray, y: np.ndarray) -> np.ndarray:
    x_t = torch.tensor(x, dtype=torch.float32)
    y_t = torch.tensor(y, dtype=torch.float32)
    loader = DataLoader(TensorDataset(x_t, y_t), batch_size=32, shuffle=True)
    optimizer = torch.optim.Adam(model.parameters())
    loss_fn = nn.MSELoss()

    model.train()
    for xb, yb in loader:
        pred = model(xb).squeeze(-1)
        loss = loss_fn(pred, yb.squeeze(-1))
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    model.eval()
    with torch.no_grad():
        return model(x_t).squeeze(-1).numpy()

async def point_of_entry_and_exit(
    symbol: str, trading_style: str, session: AsyncSession
) -> str:
    result = await get_all_close_prices_by_symbol(session, symbol)
    closes = [r[0] for r in result.fetchall()]

    if len(closes) < 100:
        return "Недостатньо історичних даних для аналізу."

    closes_arr = np.array(closes[-ML_MAX_CANDLES:]).reshape(-1, 1)
    scaler = MinMaxScaler()
    scaled = scaler.fit_transform(closes_arr)

    x, y = create_sequences(scaled, seq_length=ML_SEQ_LENGTH)

    models: dict[str, nn.Module] = {
        "LSTM": LSTMModel(),
        "GRU": GRUModel(),
        "MLP": MLPModel(seq_length=ML_SEQ_LENGTH),
    }

    metrics = {}
    for name, model in models.items():
        print(f"Навчання моделі: {name}")
        preds = _train_and_predict(model, x, y)
        y_flat = y.flatten()
        metrics[name] = {
            "mae": mean_absolute_error(y_flat, preds),
            "rmse": sqrt(mean_squared_error(y_flat, preds)),
            "r2": r2_score(y_flat, preds),
            "mape": _mape(y_flat, preds),
        }

    best_name = min(metrics, key=lambda k: metrics[k]["mae"] + metrics[k]["rmse"])
    best_model = models[best_name]

    last_seq = torch.tensor(
        scaled[-ML_SEQ_LENGTH:].reshape(1, ML_SEQ_LENGTH, 1), dtype=torch.float32
    )
    best_model.eval()
    with torch.no_grad():
        predicted_scaled = best_model(last_seq).item()

    predicted_price = scaler.inverse_transform([[predicted_scaled]])[0][0]
    current_price = float(closes_arr[-1])
    direction = "LONG" if predicted_price > current_price else "SHORT"
    multiplier = ML_EXIT_MULTIPLIERS.get(trading_style.lower(), 0.02)

    exit_price = (
        predicted_price * (1 + multiplier)
        if direction == "LONG"
        else predicted_price * (1 - multiplier)
    )

```

```
m = metrics[best_name]
return (
    f"{symbol.upper()} {direction}\n\n"
    f"Entry point: {current_price:.2f}\n"
    f"Exit point: {exit_price:.2f}\n\n"
    f"{best_name} "
    f"(R2: {m['r2']:.6f}, MAE: {m['mae']:.6f}, "
    f"RMSE: {m['rmse']:.6f}, MAPE: {m['mape']:.6f})"
)
```