НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК 004.8+336.76

До захисту допущено
Завідувач кафедри ММСА
_____ Оксана ТИМОЩУК
«___» _____ 2025 р.

Магістерська дисертація
на здобуття ступеня магістра
за освітньо-професійною програмою «Системний аналіз фінансового ринку»
зі спеціальності 124 «Системний аналіз»
на тему: «Розробка та дослідження прогнозних моделей у системах алгоритмічної торгівлі»

Виконав:
Студент 2 курсу, групи КА-42мп
Ярінко Богдан Богданович _____

Науковий керівник:
Директор ННК ІПСА НТУУ КПІ ім. Ігоря Сікорського,
член-кореспондент НАН України,
доктор фізико-математичних наук,
професор Касьянов Павло Олегович _____

Консультант з нормоконтролю:
доцент кафедри ММСА, к.ф.-м.н.,
Статкевич Віталій Михайлович _____

Рецензент:
Завідувач кафедри інтегральних та диференціальних рівнянь
КНУ ім. Тараса Шевченка,
доктор фізико-математичних наук,
професор Капустян Олексій Володимирович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань
Студент: _____

Київ - 2025

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти — другий (магістерський)
Спеціальність — 124 «Системний аналіз»
Освітньо-професійною програмою «Системний аналіз фінансового ринку»

ЗАТВЕРДЖУЮ
Завідувач кафедри ММСА
_____ Оксана ТИМОЩУК
«___» _____ 2025 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Ярінку Богдану Богдановичу

1. Тема дисертації: «Розробка та дослідження прогнозних моделей у системах алгоритмічної торгівлі», науковий керівник дисертації професор, доктор фізико-математичних наук Касьянов Павло Олегович,
затверджені наказом по університету від 06 листопада 2025 року № 4837-с.

2. Строк подання студентом дисертації: _____

3. Об’єкт дослідження: процес побудови та експлуатації прогнозних моделей у системах алгоритмічної торгівлі.

4. Предмет дослідження: методи діагностики, валідації та вдосконалення алгоритмів машинного навчання на фінансових часових рядах.

5. Перелік завдань, які потрібно розробити:
1) аналіз сучасних методів алгоритмічної торгівлі та підходів машинного навчання;
2) обґрунтування методології попередньої обробки фінансових даних;
3) дослідження ефективності прогнозних моделей та їхніх недоліків;
4) розробка системи для автоматизованої діагностики та вдосконалення торгових стратегій;
5) верифікація результатів дослідження та розробка стартап-проекту.

6. Перелік графічного (ілюстративного матеріалу):
1) рисунки;
2) таблиці;

3) презентація.

7. Орієнтовний перелік публікацій.

Ярінко Б. Б. Розробка та дослідження прогнозних моделей у системах алгоритмічної торгівлі. Збірник доповідей IV науково-практичної конференції «Системні науки та інформатика», 25–29 листопада 2025 року, Київ. К., НН ІПСА КПІ ім. Ігоря Сікорського, 2025, 6 с.

8. Консультанти розділів дисертації

Розділ	Прізвище та ініціали	Підпис, дата	
		завдання видав	завдання прийняв
-	-	-	-

9. Дата видачі завдання: 02 вересня 2025 року

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації (МД)	Строк виконання етапів магістерської дисертації	Примітка
1	Затвердження теми МД. Ознайомлення зі структурою МД згідно з Положенням про державну атестацію студентів НТУУ «КПІ ім. І. Сікорського»	01.09.2025-10.09.2025	виконано
2	Ознайомлення з ДСТУ 3008:2015.	11.09.2025-17.09.2025	виконано
3	Перший розділ. Теоретичні основи побудови прогнозних моделей у системах алгоритмічної торгівлі	18.09.2025-01.10.2025	виконано
4	Другий розділ. Методи розробки, валідації та інтерпретації прогнозних моделей для алгоритмічної торгівлі	02.10.2025-15.10.2025	виконано
6	Третій розділ. Експериментальні дослідження та підходи до вдосконалення прогнозних моделей	16.10.2025-12.11.2025	виконано
7	Четвертий розділ. Стартап-проект.	13.11.2025-19.11.2025	виконано

Студент _____ Богдан ЯРІНКО

Науковий керівник дисертації _____ Павло КАСЬЯНОВ

РЕФЕРАТ

Магістерська дисертація: 108 с., 6 рис., 12 табл., 1 додаток, 40 джерел.

АЛГОРИТМІЧНА ТОРГІВЛЯ, МАШИННЕ НАВЧАННЯ, ПРОГНОЗНІ МОДЕЛІ, ДІАГНОСТИКА МОДЕЛЕЙ, ФІНАНСОВІ ЧАСОВІ РЯДИ, НЕЙРОННІ МЕРЕЖІ, БАЗА ЗНАНЬ.

У магістерській дисертації досліджено проблему підвищення ефективності та робастності торгових стратегій в умовах нестаціонарності фінансових ринків. Розроблено систему автоматизованої діагностики прогнозних моделей, яка, на відміну від класичних підходів, фокусується на виявленні та корекції систематичних помилок алгоритмів, таких як ігнорування ринкових режимів та впевненості прогнозу (Додаток А). Експериментальна верифікація підтвердила ефективність запропонованої методології: приріст метрики F1-Score склав від 10% до 64,1% . В роботі розглядались такі моделі, як LSTM, XGBoost, Random Forest, логістична регресія. Розроблено стартап-проект для комерціалізації створеної технології, проведено аналіз конкурентного середовища та сформовано маркетингову стратегію виведення продукту на ринок.

ABSTRACT

Master’s thesis: 108 pp., 6 figures, 12 tables, 1 appendix, 40 references.

ALGORITHMIC TRADING, MACHINE LEARNING, PREDICTIVE MODELS, MODEL DIAGNOSTICS, FINANCIAL TIME SERIES, NEURAL NETWORKS, KNOWLEDGE BASE.

The master's thesis investigates the problem of enhancing the efficiency and robustness of trading strategies under conditions of financial market non-stationarity. An automated diagnostic system for predictive models has been developed. Unlike classical approaches, it focuses on identifying and correcting systematic algorithm errors, such as the disregard for market regimes and forecast confidence (Appendix A). Experimental verification confirmed the effectiveness of the proposed methodology. The increase in the F1-Score metric ranged from 10% to 64.1%, depending on the models examined, such as LSTM, XGBoost, Random Forest, and logistic regression. A startup project has been developed to commercialize the created technology, including a competitive landscape analysis and a marketing strategy for product market entry.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БЗ – База Знань

ADX – індекс середньої спрямованості руху

ARCH – авторегресійна умовна гетероскедастичність

ARIMA – авторегресійна інтегрована модель ковзного середнього

ATR – середній істинний діапазон волатильності

AUC – площа під кривою

CV – перехресна перевірка

GARCH – узагальнена авторегресійна умовна гетероскедастичність

GRU – керований рекурентний блок

LSTM – довга короткочасна пам’ять

MACD – сходження-розходження ковзних середніх

MAE – середня абсолютна помилка

ML – машинне навчання

MLOps – практики для управління розробкою моделей машинного навчання.

MSE – середньоквадратична помилка

OFI – дисбаланс потоку ордерів

RMSE – корінь середньоквадратичної помилки

RNN – рекурентна нейронна мережа

SHAP – адитивні пояснення Шеплі

TFT – часовий трансформер

VAR – векторна авторегресія

XGBoost – екстремальний градієнтний бустинг

ВСТУП

Сучасні фінансові ринки функціонують як складні динамічні системи, що характеризуються високим рівнем шуму, нестаціонарністю та нелінійністю процесів ціноутворення. Еволюція обчислювальних потужностей та поява великих масивів даних зумовили домінування алгоритмічних стратегій, що поступово витісняють класичні методи ручної торгівлі. Економетричні підходи, базуються на жорстких припущеннях лінійності та нормальності розподілу, що обмежує їхню ефективність в умовах сучасної мікроструктури ринку. Натомість методи машинного навчання, зокрема ансамблеві алгоритми Random Forest і XGBoost та архітектури глибокого навчання типу LSTM, демонструють значно вищий потенціал у виявленні прихованих ринкових закономірностей. Проте практичне впровадження цих методів супроводжується критичними викликами, серед яких схильність до перенавчання, складність інтерпретації рішень та нездатність адаптуватися до зміни ринкових режимів без спеціалізованої діагностики.

Аналіз сучасного стану проблеми за даними літературних джерел показує, що можливості наявних засобів розробки можуть бути недостатніми для повного задоволення потреб дослідників. Наявні MLOps-платформи ефективно автоматизують процеси тренування та розгортання моделей, однак не надають інструментарію для глибокого семантичного аналізу причин їхньої неефективності. Більшість помилок торгових стратегій, таких як збитки на боковому ринку або асиметрія прогнозів, залишаються невиявленими на етапі стандартного тестування. Це зумовлює об’єктивну необхідність створення спеціалізованої системи автоматизованої діагностики, яка базується на формалізації експертних знань та дозволяє виявляти систематичні помилки прогнозних моделей, трансформуючи процес їх вдосконалення з інтуїтивного пошуку в структуровану інженерну процедуру.

Для досягнення мети дослідження, яка полягає у підвищенні ефективності алгоритмічних стратегій, в роботі реалізовано комплексний методологічний підхід. Центральним рішенням роботи є розробка архітектури системи на основі документо-орієнтованої бази знань, яка містить формалізовані правила детекторів проблем та коректорів. Це дозволяє автоматично ідентифікувати такі недоліки, як мультиколінеарність вхідних ознак, ігнорування впевненості прогнозу та вразливість до зміни волатильності.

Практична цінність отриманих результатів підтверджується експериментальною перевіркою, яка продемонструвала суттєве зростання метрики F1-Score для досліджуваних моделей. Розроблений програмний продукт та методологія можуть бути застосовані інвестиційними фондами, проп-трейдинговими компаніями та фінтех-стартапами для автоматизації аудиту торгових алгоритмів, підвищення їхньої робастності та скорочення часу розробки нових інвестиційних продуктів. Результати роботи створюють підґрунтя для побудови адаптивних торгових систем, здатних ефективно функціонувати в умовах високої невизначеності криптовалютних та фондових ринків.

Метою роботи є підвищення ефективності торгових стратегій шляхом розробки методології та програмного засобу для автоматизованого виявлення та усунення систематичних помилок прогнозних моделей. Об’єктом дослідження є процес побудови та експлуатації прогнозних моделей у системах алгоритмічної торгівлі. Предмет дослідження – методи діагностики, валідації та вдосконалення алгоритмів машинного навчання на фінансових часових рядах. Наукова новизна одержаних результатів полягає в розробці концептуального підходу до діагностики фінансових моделей на основі формалізованої бази знань, що дозволяє, на відміну від існуючих методів, автоматизувати виявлення прихованих патернів неефективності та забезпечити перехід від ручного аналізу до системного інжинірингу торгових стратегій.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ ПРОГНОЗНИХ МОДЕЛЕЙ У СИСТЕМАХ АЛГОРИТМІЧНОЇ ТОРГІВЛІ

1.1 Еволюція та сучасний стан алгоритмічної торгівлі

Сучасні фінансові ринки являють собою складні динамічні системи, що характеризуються високим рівнем шуму, нестаціонарністю та нелінійністю. Розвиток інформаційних технологій та збільшення обчислювальних потужностей призвели до фундаментальної трансформації біржової торгівлі, де домінуючу роль зайняли системи алгоритмічного трейдингу. Алгоритмічна торгівля визначається як процес використання комп'ютерних програм для автоматичного виконання торговельних ордерів згідно із заздалегідь визначеними правилами та стратегіями, що базуються на математичних моделях, статистичному аналізі та методах машинного навчання [1].

Фундаментальною теоретичною проблемою, що лежить в основі розробки будь-якої прогнозної моделі, є відношення до гіпотези ефективного ринку. У своїй формі вона стверджує, що ціни активів миттєво та повною мірою відображають усю наявну інформацію, що робить спроби прогнозування майбутніх цін на основі історичних даних або публічної інформації марними [2]. Однак емпіричні дослідження останніх десятиліть та практичний успіх хедж-фондів переконливо демонструють наявність ринкових неефективностей, аномалій та патернів, які можуть бути експлуатовані. Дослідники зазначають, що ринки можуть демонструвати передбачуваність певною мірою, особливо на коротких часових інтервалах або в періоди високої волатильності, використовуючи такі техніки, як пробій діапазону або ковзні середні.

У цьому контексті предиктивне моделювання виступає як хребет сучасного прийняття рішень у фінансовій індустрії. Воно дозволяє трейдерам та інвесторам аналізувати колосальні обсяги історичних даних, виявляти

приховані тренди та прогнозувати майбутні рухи ринку з точністю, що перевищує випадкове вгадування [3]. Сучасна парадигма торгівлі на основі штучного інтелекту не обмежується лише ідентифікацією трендів; вона включає глибокий аналіз факторів, що рухають ринком, інтеграцію аналізу настроїв, макроекономічних показників та оцінку думок експертів для отримання комплексної картини ринку [4].

Для коректної побудови прогнозних моделей критично важливим є розуміння специфічних властивостей фінансових даних, які відрізняють їх від інших типів часових рядів, наприклад, метеорологічних або інженерних.

1. Більшість фінансових рядів є нестаціонарними, тобто їхні статистичні моменти (середнє значення, дисперсія, коваріація) змінюються з часом. Це порушує припущення статистичних моделей, таких як лінійна регресія, і вимагає застосування процедур диференціювання або логарифмування для приведення ряду до стаціонарного вигляду. Тест Дікі-Фуллера є стандартним інструментом для перевірки наявності одиничного кореня та стаціонарності даних [5].

2. Автокореляція – це міра лінійної залежності між значеннями часового ряду в різні моменти часу. Наявність автокореляції в залишках моделі свідчить про те, що модель не врахувала всю доступну інформацію з минулого, і в даних залишається структура, яку можна використати для прогнозу.

3. Кластеризація волатильності – це явище, коли періоди високої волатильності мають тенденцію групуватися разом, так само як і періоди низької волатильності. Це означає, що волатильність є автокорельованою, що є передумовою для використання моделей типу GARCH [6].

4. Важкі хвости розподілу – емпіричні розподіли доходностей фінансових активів часто мають вищий ексцес, ніж нормальний розподіл. Це означає, що екстремальні події трапляються частіше, ніж передбачає теорія нормального розподілу, що є критичним фактором для управління ризиками [6].

Врахування цих властивостей є обов'язковим при виборі архітектури моделі. Ігнорування нестаціонарності або важких хвостів може призвести до хибних прогнозів та недооцінки ризиків, відомої як ризик моделі.

1.2. Економетричні підходи до прогнозування

Історичний розвиток методів прогнозування почався з лінійних стохастичних моделей, розроблених у середині XX століття. Ці методи, хоч і мають обмеження в сучасних умовах високочастотної торгівлі, залишаються фундаментом для розуміння часових рядів і часто використовуються як базові моделі для оцінки ефективності складніших алгоритмів.

Модель ARIMA, формалізована Боксом і Дженкінсом у 1970-х роках, стала революційним інструментом для моделювання умовного середнього часового ряду. Вона базується на припущенні, що майбутнє значення змінної є лінійною комбінацією її минулих значень та минулих помилок прогнозу, що ілюструється формулою (1.1) [7]:

$$\phi(B)(1 - B)^d y_t = \theta(B)\epsilon_t \quad (1.1)$$

де:

y_t – значення часового ряду в момент t,

B – оператор зсуву назад,

$\phi(B)$ – поліном авторегресії порядку p, що описує залежність від минулих значень,

$(1 - B)^d$ – оператор диференціювання порядку d, необхідний для усунення нестаціонарності тренду,

$\theta(B)$ – поліном ковзного середнього порядку q, що моделює залежність від минулих помилок,

ϵ_t – білий шум.

Процес побудови моделі ARIMA включає ідентифікацію параметрів за допомогою аналізу автокореляційної функції ACF та часткової автокореляційної функції PACF, оцінку коефіцієнтів зазвичай методом максимальної правдоподібності та діагностику залишків [8].

Попри свою популярність, ARIMA має суттєві недоліки при роботі з фінансовими даними.

1. Лінійність – ARIMA здатна моделювати лише лінійні залежності. Фінансові ринки, як складні адаптивні системи, насичені нелінійними взаємодіями, які вона ігнорує, що призводить до низької точності прогнозів у періоди турбулентності.

2. Вимога стаціонарності – модель вимагає приведення даних до стаціонарного вигляду. Диференціювання може призвести до втрати інформації про довгострокові тренди.

3. Неможливість моделювання волатильності – ARIMA фокусується на прогнозі середнього значення, ігноруючи змінну дисперсію, що є критичним для трейдингу [6].

Для подолання проблеми кластеризації волатильності Роберт Енгл (1982) та Тім Боллерслев (1986) розробили моделі ARCH (Autoregressive Conditional Heteroskedasticity) та GARCH (Generalized ARCH). Ці моделі фокусуються на прогнозуванні не самої ціни, а її дисперсії тобто волатильності σ_t^2, як показано у формулі (1.2). Стандартна модель GARCH(p,q) задається рівнянням для умовної дисперсії:

$$\sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \, \epsilon_{\{t-i\}}^2 + \sum_{j=1}^{p} \beta_j \, \sigma_{\{t-j\}}^2 \qquad (1.2)$$

де поточна волатильність залежить від минулих квадратів помилок (ϵ_{t-j}^2, ARCH-члени) та минулих значень самої волатильності (σ_{t-j}^2, GARCH-члени) [9].

GARCH є незамінним інструментом для управління ризиками та ціноутворення опціонів. Проте, базова модель GARCH припускає симетричну

реакцію волатильності на позитивні та негативні шоки. У реальності існує ефект важеля: негативні новини зазвичай викликають сильніше зростання волатильності, ніж позитивні новини аналогічної сили. Для врахування цього ефекту використовуються асиметричні модифікації, такі як Exponential GARCH або GJR-GARCH. Також моделі GARCH часто критикують за повільну адаптацію до різких змін ринкових режимів та обмежену здатність моделювати складну нелінійну динаміку, притаманну сучасним високочастотним даним.

У той час як ARIMA та GARCH є одновимірними моделями, Векторна авторегресія дозволяє аналізувати взаємозв'язки між декількома часовими рядами одночасно. Це критично для систем, що враховують вплив макроекономічних показників або кореляції між різними активами.

У моделі VAR кожна змінна системи розглядається як лінійна функція від своїх власних лагових значень та лагових значень усіх інших змінних у системі, що може бути формалізовано у формулі (1.3):

$$Y_t = c + A_1 Y_{t-1} + \cdots + A_p Y_{t-p} + e_t \quad (1.3)$$

VAR широко використовується центральними банками та інституційними інвесторами для структурного аналізу та тестування причинності за Грейнджером. Однак, VAR страждає від проблеми прокляття розмірності: кількість параметрів для оцінки зростає квадратично зі збільшенням кількості змінних, що призводить до перенавчання на обмежених вибірках. Крім того, лінійна природа моделі обмежує її здатність прогнозувати складні економічні взаємодії під час криз, що стимулює розвиток нелінійних розширень, таких як Deep VAR на базі нейронних мереж [12].

1.3 Машинне навчання для алгоритмічної торгівлі та нелінійний аналіз

Зростання доступності великих даних та обчислювальних потужностей зумовило перехід від параметричних економетричних моделей до непараметричних алгоритмів машинного навчання. ML-моделі не роблять жорстких припущень щодо розподілу даних і здатні апроксимувати складні нелінійні функції, що робить їх більш придатними для зашумлених та хаотичних фінансових ринків [13].

Серед алгоритмів машинного навчання найбільшу ефективність у завданнях прогнозування продемонстрували ансамблеві методи на основі дерев рішень.

Алгоритм Random Forest використовує техніку беггінгу, він будує велику кількість незалежних дерев рішень на випадкових підмножинах даних та ознак, а потім усереднює їхні прогнози.

1. Перевагою є висока стійкість до перенавчання, здатність обробляти велику кількість ознак, вбудована оцінка важливості факторів [14].
2. Він часто використовується як надійний базовий алгоритм для задач класифікації ринкових режимів або прогнозу напрямку ціни, де інтерпретованість має значення.

Градієнтний бустинг будує ансамбль дерев послідовно: кожне нове дерево навчається виправляти помилки, зроблені попередніми деревами. Алгоритм XGBoost став де-факто стандартом у змаганнях з аналізу даних та в індустрії завдяки своїй швидкодії та ефективності. Він використовує градієнтний спуск для мінімізації функції втрат, додаючи регуляризаційні члени для контролю складності моделі та запобігання перенавчанню. Він також ефективно обробляє пропущені дані та підтримує паралельні обчислення. Дослідження показують, що XGBoost здатний захоплювати складні нелінійні патерни краще за лінійні моделі. У порівняльних тестах на

прогнозування сонячної енергії та фінансових ризиків часто демонструє результати, близькі до нейронних мереж, іноді перевершуючи їх у задачах з табличними даними [15].

На відміну від RNN, дерева рішень не мають вбудованої пам’яті послідовностей. Тому для використання XGBoost у трейдингу необхідно проводити ретельний інжиніринг ознак, створюючи лагові змінні технічні індикатори та змінні часу, щоб перетворити задачу часового ряду на задачу навчання з учителем.

Ключова відмінність полягає в компромісі між зміщенням та дисперсією. Економетричні моделі мають високе зміщення через жорсткі лінійні припущення, але низьку дисперсію. Натомість, Random Forest, XGBoost мають низьке зміщення, тобто можуть вивчити складну форму, але високу дисперсію і тому схильні до перенавчання на шумі. У таблиці 1.1 наведено порівняння ключових характеристик.

Таблиця 1.1 Порівняння статистичних моделей та методів машинного навчання

Характеристика	Статистичні моделі (ARIMA, GARCH)	Машинне навчання (Random Forest, XGBoost)
Припущення про дані	Жорсткі (лінійність, стаціонарність, нормальність)	Гнучкі (непараметричні, не вимагають нормальності)
Обробка нелінійності	Обмежена (потребує специфікації)	Висока (автоматично виявляє нелінійні зв’язки)
Інтерпретованість	Висока (зрозумілі коефіцієнти)	Середня/Низька (важливість ознак, але «чорна скринька»)
Вимоги до даних	Працюють на малих вибірках	Потребують великих обсягів даних для навчання
Схильність до перенавчання	Низька (якщо правильно специфікована)	Висока (потребує крос-валідації та регуляризації)
Застосування в HFT	Обмежене (повільна адаптація)	Широке (швидкий інференс, висока точність)

1.4 Архітектури глибокого навчання для аналізу часових рядів та послідовностей

Глибоке навчання представляє собою клас методів машинного навчання, що використовують багатошарові штучні нейронні мережі для автоматичного вилучення ознак високого рівня з сирих даних. У контексті часових рядів ці моделі стали домінуючим інструментом завдяки своїй здатності моделювати часові залежності довільної довжини [16].

Стандартні нейронні мережі прямого поширення не мають пам’яті, тому вони неефективні для послідовних даних. Рекурентні нейронні мережі вирішують цю проблему, передаючи прихований стан від одного кроку до наступного. Однак звичайні RNN страждають від проблеми затухання градієнта, що унеможливлює навчання на довгих послідовностях [17].

Запропонована Хохрайтером та Шмідхубером [10] архітектура LSTM вирішує проблему градієнтів за допомогою спеціальної комірки пам’яті та механізму воріт, які регулюють потік інформації:

1) ворота забування вирішують, яку інформацію з попереднього стану комірки слід відкинути,

2) вхідні ворота визначають, яку нову інформацію записати в комірку,

3) вихідні ворота формують вихідний прихований стан h_t на основі стану комірки (див. також [17]).

Завдяки цьому LSTM здатна пам’ятати важливі події протягом довгих проміжків часу і ігнорувати незначний шум. Це робить її ідеальною для фінансових рядів, де події мають довгостроковий вплив. Дослідження на даних акцій Tesla показали, що модель LSTM досягає точності 94% у прогнозуванні тренду, перевершуючи як традиційні статистичні моделі, так і прості RNN. У порівнянні з ARIMA, LSTM демонструє значно меншу середньоквадратичну помилку та кращі показники AUC у задачах оцінки

ризиків (0.8522 проти 0.8105 у XGBoost) [18]. Особливо ефективною LSTM є у прогнозуванні волатильності, де вона перевершує моделі GARCH за умов наявності достатньої кількості даних для навчання

Gated Recurrent Unit є спрощеною модифікацією LSTM, яка об'єднує ворота забування та вхідні ворота в єдині ворота оновлення. GRU має менше параметрів, що прискорює навчання та зменшує ризик перенавчання на менших наборах даних. Хоча у згаданому дослідженні Tesla GRU показала дещо гірший результат (на 4.5% нижчу точність, ніж Transformer/LSTM), вона залишається популярною альтернативою завдяки обчислювальній ефективності.

Революцію у глибокому навчанні спричинила поява архітектури Transformer. На відміну від RNN, які обробляють дані послідовно крок за кроком, Трансформери обробляють всю послідовність одночасно, тобто паралельно, використовуючи механізм само-уваги [19].

Механізм уваги дозволяє моделі оцінювати важливість кожного елемента вхідної послідовності по відношенню до інших, незалежно від їх відстані у часі. Математично це реалізується через обчислення векторів запиту (Q), ключа (K) та значення (V), і представлено у формулі (1.4):

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1.4}$$

де множник $\frac{1}{\sqrt{(d_k)}}$ забезпечує стабільність градієнтів.

Для фінансових рядів це означає здатність моделі миттєво пов'язувати поточну ціну з подією, що сталася місяць тому, не проходячи через усі проміжні дні, як це робить LSTM. Це дозволяє краще захоплювати складні глобальні залежності [19].

У експериментах з прогнозування акцій Tesla базовий трансформер показав точність 94.4%, що порівнянно з LSTM, але модель може страждати від квадратичної складності обчислень при дуже довгих рядах.

Спеціально для задач прогнозування часових рядів була розроблена архітектура Temporal Fusion Transformer. Вона поєднує переваги LSTM для

локальної обробки послідовностей та механізмів уваги для довгострокових залежностей, додаючи критично важливий для фінансів компонент – інтерпретованість [20].

Наведемо ключові компоненти TFT.

1. Gating Mechanisms дозволяють моделі адаптивно пропускати непотрібні шари, спрощуючи архітектуру, якщо дані прості.

2. Variable Selection Networks автоматично визначають та зважують важливість кожної вхідної змінної в кожен момент часу. Це дає трейдеру розуміння, чому модель зробила прогноз, наприклад, сьогодні важливий обсяг, а вчора була важлива волатильність.

3. Static Covariate Encoders інтегрують статичні метадані, наприклад, сектор економіки, до якого належить акція, у процес прогнозування.

4. За допомогою Quantile Regression прогнозує не лише одне значення, а й інтервали , що дозволяє оцінювати невизначеність прогнозу.

Ця модель демонструє state-of-the-art результати на багатьох фінансових бенчмарках, перевершуючи як прості LSTM, так і ARIMA, особливо у мульти-горизонтному прогнозуванні [21].

1.5 Виклики адаптації моделей до нестаціонарності ринкових даних

Головним викликом при впровадженні моделей у реальну торгівлю є мінливість ринкового середовища. Модель, навчена на даних 2020 року, може бути абсолютно непридатною для ринку 2025 року. У такому випадку виникає безліч помилок, пов’язаних з неправильною оцінкою ризиків та недостовірністю прогнозів, деякі з них це дрейф концепції, перенавчання та ігнорування ринкових режимів.

Дрейф концепції виникає, коли статистичні властивості цільової змінної, яку модель намагається передбачити, змінюються з часом [22]. В алгоритмічній торгівлі це проявляється як деградація ефективності стратегії.

Вкажемо типи дрейфу:

1) раптовий – різка зміна умов, наприклад, початок пандемії COVID-19;

2) поступовий – повільна зміна інфляційних очікувань або ринкової ліквідності;

3) сезонний – повторювані зміни, пов’язані з бізнес-циклами.

Для боротьби з дрейфом використовуються методи Adaptive Learning: ковзні вікна для постійного перенавчання моделі, зважування прикладів та ансамблі моделей, що динамічно змінюються. Алгоритм Autoregressive Drift Detection Method дозволяє виявляти дрейф у режимі реального часу, порівнюючи помилку прогнозу з динамічним порогом [23].

Ринки періодично перемикаються між різними режимами бичачий, ведмежий, боковий, висока волатильність. Ідентифікація поточного режиму є критичною для вибору відповідної стратегії.

Приховані марковські моделі є популярним ймовірнісним підходом, який припускає, що спостережувані дані генеруються прихованими станами. Модель оцінює ймовірність перебування в кожному стані та матрицю ймовірностей переходу між ними. Вона ефективно виявляє переходи між спокійним і турбулентним ринком, що дозволяє покращити коефіцієнт Шарпа портфеля шляхом уникнення інвестицій у періоди високого ризику. Однак модель вимагає задання кількості станів апріорі та базується на припущенні про стаціонарність розподілів всередині режимів [24].

Методи кластеризації, такі як K-Means або Gaussian Mixture Models, використовуються для групування ринкових станів на основі подібності багатовимірних ознак без розмітки вчителем.

Кластеризація на основі метрики Вассерштейна не робить жорстких припущень про форму розподілу даних, що дозволяє виявляти режими з

різними статистичними властивостями більш гнучко. Вона дозволяє моделювати режими як суміш нормальних розподілів, надаючи ймовірнісну оцінку приналежності до режиму [25].

1.6. Підходи до валідації та оцінювання ефективності прогнозних моделей

Стандартні методи перевірки моделей, прийняті в загальному машинному навчанні, наприклад, випадкова крос-валідація, є категорично неприйнятними для фінансових часових рядів. Використання майбутньої інформації при навчанні призводить до оптимістичних результатів бек-тестів, які розвалюються в реальній торгівлі.

Маркос Лопес де Прадо у своїй фундаментальній праці «Advances in Financial Machine Learning» запропонував метод Purged Cross-Validation [26].

Суть методу полягає у двох модифікаціях стандартної процедури. Розглянемо їх послідовно.

1. Очищення – видалення з тренувального набору всіх спостережень, мітки яких перекриваються у часі з даними тестового набору. Це необхідно, оскільки фінансові мітки, наприклад, дохідність за наступні 5 днів, використовують інформацію з майбутнього по відношенню до моменту прийняття рішення.

2. Ембарго – додавання захисного буфера після кожного тестового періоду перед початком наступного тренувального періоду. Це усуває залишкову серійну кореляцію, яка може призвести до витоку інформації.

Для отримання більш надійної статистики ефективності використовується Combinatorial Purged. Замість одного розбиття на тренувальну та тестову вибірки, метод генерує всі можливі комбінації, що дозволяє створити сотні сценаріїв тестування на одних і тих самих історичних

даних. Це дозволяє оцінити не лише точкове значення прибутковості, а й розподіл результатів, розрахувати ймовірність перенавчання та отримати більш реалістичну оцінку коефіцієнта Шарпа [27].

Окрім стандартних метрик регресії MSE, MAE, RMSE та класифікації Accuracy, F1-score, AUC, для оцінки торгових моделей критично важливо використовувати фінансові метрики, перерахуємо їх.

1. Коефіцієнт Шарпа – відношення середньої надлишкової прибутковості до волатильності портфеля. Це стандарт індустрії для оцінки дохідності з поправкою на ризик [28].

2. Максимальна просадка – найбільше падіння капіталу від піку до дна. Це міра ризику, яка показує потенційні збитки в найгіршому сценарії [28].

3. Profit Factor – відношення валового прибутку до валового збитку.

Дослідження показують, що оптимізація моделі безпосередньо на фінансові метрики, наприклад, використання функції втрат Sharpe Loss у нейромережах дає кращі результати, ніж мінімізація середньоквадратичної помилки.

1.7. Висновки до першого розділу

Проведений у першому розділі аналіз теоретичних засад та літературних джерел дозволяє зробити ряд ключових висновків, які визначають напрямок подальшого дослідження.

По-перше, спостерігається парадигмальний зсув: від лінійних економетричних моделей, таких як ARIMA та GARCH, до нелінійних методів глибокого навчання. Хоча економетричні моделі забезпечують теоретичну прозорість, їхня ефективність у моделюванні складної динаміки сучасних ринків є обмеженою. У цьому контексті алгоритми на основі LSTM та Transformers демонструють значно вищу точність.

По-друге, перевага архітектур з пам’яттю є безсумнівною. Моделі LSTM та GRU значною мірою перевершують традиційні методи через здатність утримувати довгострокові залежності та уникати проблеми затухання градієнта. Особливо наглядною є архітектура Temporal Fusion Transformer, яка поєднує в собі точність механізмів уваги з необхідною для фінансових інституцій інтерпретованістю.

По-третє, критичність адаптації до середовища не може бути недооціненою. Статичні моделі стикаються з ризиком застарівання через дрейф концепції. Тому сучасні системи алгоритмічної торгівлі повинні інтегрувати модулі виявлення ринкових режимів для забезпечення динамічної адаптації стратегій до змінюваних ринкових умов.

Нарешті, валідація виступає як фундамент надійності, традиційні методи крос-валідації можуть призводити до помилкових висновків щодо прибутковості стратегій. Застосування нових методологій, таких як Purged Cross-Validation та Combinatorial CV, є необхідною умовою для отримання надійних результатів дослідження, що допоможе мінімізувати ризик перенавчання.

Таким чином, ці висновки формують основу для подальших досліджень і сприяють розвитку більш ефективних і адаптивних моделей у сфері алгоритмічної торгівлі.

РОЗДІЛ 2 МЕТОДИ РОЗРОБКИ, ВАЛІДАЦІЇ ТА ІНТЕРПРЕТАЦІЇ ПРОГНОЗНИХ МОДЕЛЕЙ ДЛЯ АЛГОРИТМІЧНОЇ ТОРГІВЛІ

2.1. Теоретичні засади формування прогнозної моделі

Сучасний ландшафт фінансових ринків характеризується безпрецедентним зростанням обсягів даних, збільшенням швидкості транзакцій та домінуванням алгоритмічних учасників. У цьому середовищі економетричні підходи, що базуються на лінійних припущеннях та жорстких статистичних розподілах, поступаються місцем адаптивним методам машинного навчання та глибокого навчання. Цей розділ присвячений детальній розробці методологічного апарату для побудови торгових систем, які здатні ефективно функціонувати в умовах високої невизначеності, нестаціонарності та низького відношення сигналу до шуму.

Мета цього розділу – не просто перелічити доступні алгоритми, а сформувати цілісний, науково обґрунтований конвеєр дослідження, який охоплює всі етапи: від специфічної попередньої обробки фінансових даних до суворої валідації результатів з урахуванням ефекту множинного тестування. Особлива увага приділяється вирішенню фундаментальних проблем фінансового моделювання, таких як втрата пам’яті часового ряду при диференціюванні, некоректність стандартних методів крос-валідації для часових рядів та проблема чорної скриньки у складних нейромережевих архітектурах.

Методологія базується на синтезі академічних досліджень у галузі мікроструктури ринку та прикладних розробок у сфері фінансового інжинірингу, зокрема працях Лопеса де Прадо, які заклали основу фінансового машинного навчання як окремої дисципліни. Розглянуто перехід від евристичних правил до статистично значущих прогнозів, що вимагає впровадження спеціалізованих метрик, таких як Deflated Sharpe Ratio, та

методів інтерпретації, адаптованих для рекурентних нейронних мереж, таких як TimeSHAP.

2.2. Постановка задачі прогнозування

Першим критичним рішенням у розробці торгової системи є математичне формулювання цільової змінної. Традиційно прогнозування цін активів розглядалося як задача регресії, де метою моделі є мінімізація відхилення прогнозованої ціни від фактичної. Проте, як свідчать останні дослідження, цей підхід має суттєві обмеження в контексті алгоритмічної торгівлі.

Регресійні моделі, що мінімізують середньоквадратичну помилку, намагаються апроксимувати безперервну функцію ціни. В умовах фінансових ринків, які характеризуються високою стохастичністю та наявністю важких хвостів у розподілах, такі моделі часто стають надмірно чутливими до шуму та викидів. Вони намагаються вгадати точне значення ціни, що є надзвичайно складним завданням, часто ігноруючи більш важливий для трейдера фактор – напрямок руху. Як зазначає дослідник Liang et al, висока випадковість і волатильність роблять регресію неефективною для захоплення різких ринкових змін [29]. Більше того, метрика MSE не розрізняє малу помилку у правильному напрямку та малу помилку у неправильному напрямку, хоча для торгової стратегії це різниця між прибутком та збитком.

Альтернативою є формулювання задачі як класифікації. У цьому випадку простір вихідних значень дискретизується. Замість прогнозування точної ціни модель навчається визначати клас майбутнього стану ринку.

Дослідження демонструють, що методи класифікації, такі як SVM, градієнтний бустинг та глибокі нейронні мережі з класифікаційними виходами, демонструють вищу робастність [30]. Вони фокусуються на

побудові розділяючих гіперплощин у багатовимірному просторі ознак, що дозволяє краще фільтрувати ринковий шум.

Вибір функції втрат визначає ландшафт, у якому модель шукає оптимальні параметри. Для задач класифікації стандартним вибором є категоріальна перехресна ентропія (формула (2.1)):

$$L_{CE} = -\sum_{i=1}^{C} y_i \, log(\widehat{y_i}) \tag{2.1}$$

де y_i – істинна мітка класу, а $\widehat{y_i}$ – передбачена ймовірність.

Проте, специфіка фінансових задач вимагає модифікації стандартних функцій втрат. Важливою інновацією є введення регуляризації на основі впевненості [31]. Це передбачає включення до функції втрат компонента, який штрафує модель за високу впевненість у помилкових прогнозах, або навпаки, заохочує вищу впевненість лише у випадках з високою ймовірністю успіху.

Такий підхід дозволяє створювати торгові політики, які утримуються від входу в ринок, якщо впевненість моделі не перевищує певний поріг, що безпосередньо покращує коефіцієнт Шарпа стратегії шляхом зменшення кількості збиткових угод.

2.3. Інженерія фінансових даних та мікроструктурний аналіз

Якість вхідних даних є фундаментом будь-якої прогнозної моделі. У фінансовій сфері дані є зашумленими, нестаціонарними та гетероскедастичними.

Традиційний підхід до аналізу часових рядів базується на формуванні барів за фіксованими проміжками часу – наприклад, 1 хвилина, 1 година, 1 день. Проте, цей метод має фундаментальний недолік: хронологічний час не є еквівалентом ринкового часу. Активність на ринку розподілена нерівномірно.

Період відкриття ринку може містити стільки ж інформації, скільки кілька годин обіднього затишшя.

Використання свічкових барів породжує дві основні статистичні проблеми.

1. Порушення нормальності розподілу – через значну різницю обсягів торгів у кожному барі, розподіл прибутковостей стає лептокуртичним та асиметричним та з наявністю важких хвостів. Це ускладнює застосування параметричних статистичних тестів, які припускають нормальність.

2. Гетероскедастичність – дисперсія прибутковостей змінюється з часом і корелює з внутрішньоденними циклами активності, що створює нестабільність для моделей машинного навчання.

Для вирішення цих проблем у методології використовуються бари, що базуються на внутрішній активності ринку, зокрема Dollar Bars, тобто бари грошового обсягу. Dollar Bar формується кожного разу, коли кумулятивна вартість угод досягає заздалегідь визначеного порогу. Порівняння часових та доларових барів продемонстровано в таблиці 2.1.

Умова формування бару може бути виражена через формулу (2.2):

$$\sum_{i=1}^{N} p_i \cdot v_i \geq T \tag{2.2}$$

де p_i – ціна i-ї угоди, v_i – розмір i-ї угоди.

Таблиця 2.1 Порівняння типів барів у фінансовій методології

Характеристика	Time Bars	Dollar Bars
Тригер формування	Фіксований час (напр., 5 хв)	Фіксована грошова сума
Чутливість до активності	Низька (багато шуму в спокійний час)	Висока (синхронізується з потоком інформації)
Статистичні властивості	Гетероскедастичність, ненормальність	Найближчий до нормального розподілу
Врахування зміни цін	Ні	Так (при зростанні ціни потрібно менше акцій для бару)

Використання Dollar Bars має теоретичне обґрунтування: згідно з гіпотезою про підпорядковані процеси, ціна змінюється з кожною одиницею інформації, що надходить на ринок. Оскільки обсяг грошей, що переходить з рук в руки, є проксі для обсягу інформації, семплювання за такими свічками вирівнює потік інформації. Це призводить до кращої статистичної поведінки: відновлення нормальності розподілу прибутковостей та зменшення автокореляції волатильності [33]. Динамічне налаштування порогу також є важливою частиною реалізації. Фіксований поріг може бути неефективним при довгострокових змінах вартості активу або загальної ліквідності ринку.

Більшість моделей по типу ARIMA та алгоритмів ML вимагають, щоб вхідні дані були стаціонарними, тобто мали незмінність математичного сподівання, дисперсії та автоковаріації в часі. Цінові ряди є нестаціонарними. Стандартна практика полягає у взятті перших різниць

Хоча це робить ряд стаціонарним, це також знищує всю довгострокову пам’ять про історію цін. Ряд перетворюється на процес без пам’яті, що унеможливлює виявлення довгострокових трендів нейронними мережами.

Дробове диференціювання, дозволяє знайти баланс між стаціонарністю та збереженням інформації. Оператор дробового диференціювання $(1-B)^d$ для дійсного числа d визначається через розклад у ряд Тейлора, згідно з формулою (2.3):

$$(1\ -\ B)^d\ = \sum_{k=0}^{\infty} \binom{d}{k} (-B)^k\ = \sum_{k=0}^{\infty} \omega_k B^k \tag{2.3}$$

ваги ω_k обчислюються ітеративно за формулою (2.4):

$$\omega_0\ =\ 1,\ \ \omega_k\ =\ -\omega_{k-1} \frac{(d\ -\ k\ +\ 1)}{k} \tag{2.4}$$

На відміну від цілочисельного диференціювання ($d = 1$), де $\omega_0 = 1$, $\omega_1 = -1$, а всі інші ваги дорівнюють 0 (пам’ять лише на 1 крок), при $0 < d < 1$ ваги спадають поступово, зберігаючи інформацію про далеке минуле [33].

Для підвищення точності короткострокових прогнозів модель збагачується даними мікроструктури ринку, отриманими з лімітної книги

ордерів. Центральним предиктором є Order Flow Imbalance, який вимірює дисбаланс між потоками ордерів на купівлю та продаж на найкращих рівнях цін.

OFI базується на спостереженні за змінами стану книги ордерів. Нехай q_t^b та q_t^a – обсяги на найкращих рівнях попиту та пропозиції у момент t, а p_t^b та p_t^a – відповідні ціни. Потік ордерів e_t для кожного боку визначаються наступним чином.

Для сторони попиту e_t^b:

1. якщо $p_t^b > p_{t-1}^b$, це означає покращення ціни покупцями, отже весь новий обсяг q_t^b вважається притоком ліквідності: $e_t^b = q_t^b$;

2. якщо $p_t^b = p_{t-1}^b$, зміна обсягу відображає додавання або скасування ордерів: $e_t^b = q_t^b - q_{t-1}^b$;

3. якщо $p_t^b < p_{t-1}^b$, це означає погіршення ціни: $e_t^b = -q_{t-1}^b$.

Аналогічна логіка застосовується до сторони пропозиції e_t^a, але зі зворотним знаком впливу на ціну.

Інтегральний показник OFI за період T розраховується за формулою (2.5):

$$OFI_T = \sum_{t=1}^{T} \left(e_t^b - e_t^a \right) \tag{2.5}$$

Високе позитивне значення OFI свідчить про агресивний тиск покупців, що з високою ймовірністю призведе до зростання ціни. Мультирівневий OFI, дозволяє розраховувати дисбаланс не лише для першого рівня, а й для глибини стакану 5-10 рівнів. Це дозволяє моделі бачити приховану ліквідність та виявляти спроби маніпуляцій або великі інституційні ордери, що розміщені подалі від спреду.

Додатковою нормалізацією є ділення OFI на загальний обсяг торгів, що робить показник порівнюваним у різні періоди активності [34].

Найбільш вразливим місцем прогнозних стратегій є метод розмітки даних. Стандартний метод фіксованого горизонту, де мітка y_i визначається

знаком прибутковості через фіксований час h ($r_{t,t+h}$), ігнорує траєкторію руху ціни. На практиці трейдер може бути вибитий з позиції по стоп-лосу задовго до часу $t + h$ через високу волатильність, навіть якщо кінцевий прогноз був правильним.

Для наближення моделі до реальних умов торгівлі застосовується метод потрійного бар'єру, запропонований Лопесом де Прадо [36]. Для кожного спостереження встановлюються три межі виходу з позиції:

1) верхній горизонтальний бар'єр встановлюється на рівні $P_t \cdot (1 + \tau \cdot \sigma_t)$, де σ_t – поточна волатильність, τ – коефіцієнт ширини;

2) нижній горизонтальний бар'єр встановлюється на рівні $P_t \cdot (1 - \tau \cdot \sigma_t)$;

3) вертикальний бар'єр встановлюється як максимальний час утримання позиції, наприклад, кількість барів.

Критичною особливістю є динамічна ширина бар'єрів. Поріг прибутку/збитку визначається як функція від щоденної волатильності σ_t. Це гарантує, що мітки є статистично еквівалентними у різні ринкові режими: у періоди високої волатильності бар'єри розширюються, щоб уникнути випадкового спрацювання від шуму, а в спокійні періоди вони звужуються [35].

Мітка y_i формується наступним чином:

1) y_i = 1 (Лонг) – ціна торкнулася верхнього бар'єру першою;

2) y_i = -1 (Шорт) – ціна торкнулася нижнього бар'єру першою;

3) y_i = 0 (Холд) – ціна досягла вертикального бар'єру без торкання горизонтальних або, альтернативно, клас визначається знаком прибутковості на момент виходу, якщо він недостатньо великий для впевненого сигналу.

Цей метод перетворює задачу прогнозування ціни на задачу прогнозування ймовірності досягнення прибутку раніше за стоп-лос, що є прямою оптимізацією стратегії співвідношення ризику до винагороди.

Простір гіперпараметрів сучасних торгових моделей таких як кількість шарів, крок навчання, параметри дробового диференціювання, пороги

потрійного бар’єру є багатовимірним і складним. Пошук по сітці є обчислювально неможливим. Виходом з цього може бути Байєсівська оптимізація. Метод будує сурогатну імовірнісну модель, яка апроксимує залежність цільової функції від гіперпараметрів. На кожному кроці алгоритм обирає наступний набір параметрів, максимізуючи функцію придбання. Це дозволяє ефективно досліджувати простір, балансуючи між розвідкою нових зон та уточненням параметрів у перспективних областях, знаходячи оптимум за мінімальну кількість дорогих циклів навчання [39].

2.4. Метрики оцінки ефективності та проблема перенавчання

Оцінка якості моделі в алгоритмічній торгівлі вимагає виходу за межі стандартних метрик класифікації таких як accuracy F1-score. Необхідно оцінювати економічну доцільність та статистичну значущість результату з урахуванням ризику.

Базовою метрикою для оцінки портфеля є коефіцієнт Шарпа, який визначає відношення надлишкової прибутковості до волатильності за формулою (2.6):

$$SR = \frac{E}{\sigma_p} \tag{2.6}$$

Для більш точної оцінки ризику також застосовують такі показники:

1. Sortino Ratio – враховує лише волатильність негативних результатів, що є більш релевантним для асиметричних стратегій;
2. Calmar Ratio – відношення річної прибутковості до максимальної просадки, що показує ефективність відновлення після втрат.

Однією з головних причин провалу алгоритмічних фондів є перенавчання на історії. Дослідники часто проводять тисячі симуляцій з

різними параметрами, обираючи найкращу. При великій кількості спроб високий коефіцієнт Шарпа може бути отриманий суто випадково.

Для боротьби з цим явищем у методологію впроваджено метрику Deflated Sharpe Ratio [36]. Він коригує оцінку базового коефіцієнта, враховуючи кількість незалежних випробувань, які були проведені до вибору фінальної моделі.

DSR розраховується як ймовірність того, що істинний коефіцієнт Шарпа стратегії є позитивним, при умові, що ми вже відкинули гіпотезу про випадковість найкращого результату серед N випробувань.

2.5. Процедура валідації прогнозних моделей

Стандартна крос валідація, що широко використовується в машинному навчанні, базується на припущенні, що спостереження є незалежними та однаково розподіленими. Для часових рядів це припущення хибне. Використання випадкового перемішування призводить до витоку інформації оскільки тренувальні дані можуть містити інформацію, корельовану з тестовими даними через серійну кореляцію або перекриття міток у методі потрійного бар’єру.

Для коректної оцінки використовується метод Purged K-Fold Cross-Validation. Цей метод вводить два механізми захисту від витоку даних.

1. Очищення – з тренувального набору видаляються всі спостереження, чиї мітки перекриваються в часі з будь-яким спостереженням у тестовому наборі. Це критично для методу потрійного бар’єру, де тривалість угоди є змінною.

2. Ембарго – оскільки фінансові кореляції згасають поступово, навіть після видалення перекриттів може залишатися залишкова залежність. Ембарго

додає захисний буфер, наприклад, 1% від обсягу даних після кожного тестового блоку, який також виключається з тренування.

Додатковим рівнем валідації є Walk-Forward Validation (ковзне вікно), що імітує реальний процес еволюції моделі. Тренувальне вікно зсувається в часі, і модель постійно перенавчається. Це дозволяє оцінити стійкість стратегії до зміни ринкових режимів.

Для автоматичного виявлення моменту, коли модель застаріла і потребує перенавчання, використовуються детектори дрейфу концепції, такі як Page-Hinkley Test або ADWIN. Ці алгоритми моніторять статистичні властивості помилки прогнозування в реальному часі. Якщо середня помилка суттєво зростає, спрацьовує тригер для перенавчання моделі або переходу в захисний режим. Page-Hinkley є особливо ефективним для виявлення різких змін у фінансових рядах [37].

2.6. Інтерпретація моделей

Використання глибоких нейронних мереж, зокрема, створює проблему чорної скриньки. Регулятори та портфельні керуючі вимагають розуміння логіки прийняття рішень.

Для інтерпретації використовується метод SHAP, який базується на теорії кооперативних ігор. Він розподіляє виграш між гравцями як значення Шеплі. Це забезпечує адитивність та локальну точність пояснень.

SHAP дозволяє будувати графіки залежності, де видно не лише важливість ознаки, але і напрямок впливу. Наприклад, можна візуалізувати, як високі значення волатильності впливають на ймовірність сигналу купівлі – позитивно чи негативно.

Стандартний KernelSHAP розглядає ознаки як незалежні, що не підходить для рекурентних мереж, де порядок надходження даних має

вирішальне значення. Однак TimeSHAP адаптує механізм збурень для послідовностей. Замість того, щоб маскувати окремі пікселі чи слова, він маскує цілі часові кроки або групи подій.

Вкажемо ключові компоненти TimeSHAP.

1. Обрізка – алгоритм автоматично визначає, наскільки далеко в минуле дивиться модель. Якщо події, що відбулися 50 барів тому, мають нульовий внесок у прогноз, TimeSHAP обрізає їх, фокусуючись на значущому вікні. Це значно зменшує обчислювальну складність.

2. Event-level vs Feature-level explanations – TimeSHAP дозволяє розрізнити, коли сталася важлива подія і що саме сталося [38].

Це дозволяє верифікувати, чи модель дійсно вивчила фундаментальні ринкові закономірності, чи просто запам’ятала шум.

2.7. Автоматизація експериментів та MLOps

Розвиток алгоритмічної торгівлі вимагає не лише високої точності прогнозних моделей, але й ефективної організації процесу їх створення, валідації та впровадження. Сучасний підхід передбачає автоматизацію експериментів, що включає формалізацію конвеєру підготовки даних, тренування моделей, оцінки їх продуктивності, управління версіями моделей і даних, а також безперервний моніторинг роботи у продуктивному середовищі. Основною метою такої автоматизації є підвищення відтворюваності результатів, мінімізація людських помилок та забезпечення стійкості торгових стратегій до змін ринкових умов.

Автоматизація включає кілька ключових компонентів. По-перше, організація експериментів передбачає стандартизовану процедуру підготовки даних: від очищення та нормалізації до побудови часових вікон, барів та інженерії ознак мікроструктури ринку. Кожен крок конвеєру документується, параметри зберігаються у версіях, що дозволяє відтворювати

експериментальні умови для майбутніх запусків. По-друге, тренування моделей інтегрується з системами відстеження метрик та логування, що дозволяє автоматично збирати інформацію про продуктивність, ризик-показники, розподіли прогнозів та ймовірності помилок. По-третє, у конвеєр вбудовані механізми моніторингу дрейфу даних та концепцій, які автоматично сигналізують про необхідність перенавчання або адаптації моделі до нових ринкових умов.

Важливою складовою автоматизації є управління життєвим циклом моделі, яке охоплює всі стадії від прототипування до деплою та обслуговування у продуктивному середовищі. Це включає стандартизовані процедури версіонування моделей, управління залежностями та середовищами виконання, автоматичне тестування та валідацію моделей перед релізом, а також інтеграцію з системами моніторингу ризиків і фінансових показників. Така інфраструктура дозволяє швидко відновлювати експерименти, повторно використовувати існуючі дані та моделі, а також забезпечує надійність та прозорість у процесі прийняття торгових рішень.

Додатково автоматизація експериментів сприяє реалізації концепції continuous training та continuous deployment у фінансових моделях. Механізми безперервного навчання дозволяють моделі адаптуватися до зміни ринкових режимів, оперативно враховувати дрейф концепцій та оновлювати параметри стратегії. Застосування MLOps підходів забезпечує повний аудит усіх змін, зберігає історію результатів та дозволяє проводити ретроспективний аналіз ефективності. Крім того, інтеграція з хмарними або локальними обчислювальними ресурсами дозволяє масштабувати тренування моделей та оптимізацію параметрів у великомасштабних експериментах.

Таким чином, автоматизація експериментів і впровадження MLOps формує фундамент для побудови стійких, адаптивних і керованих торгових стратегій, що здатні працювати в умовах високої волатильності та складної динаміки фінансових ринків. Вона забезпечує прозорість процесу розробки, стандартизовану оцінку ефективності моделей, контроль ризиків та

безперервне вдосконалення алгоритмічних систем на основі емпіричних даних.

2.8 Висновки до другого розділу

Розроблена методологія пропонує комплексний підхід до створення алгоритмічних торгових систем, який адресує специфічні виклики фінансових ринків. Перехід від Time Bars до Dollar Bars та використання дробового диференціювання забезпечують статистичну коректність вхідних даних. Впровадження методу потрійного бар'єру та мікроструктурних ознак наближає модель до реалій ринкової механіки. Застосування Purged K-Fold CV та метрики Deflated Sharpe Ratio створює надійний бар'єр проти перенавчання, відсіюючи помилкові відкриття. Нарешті, інтеграція TimeSHAP забезпечує прозорість, керованість та можливість масштабування процесу дослідження. Цей методологічний фундамент є основою для практичних експериментів, для дослідника.

РОЗДІЛ 3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА ПІДХОДИ ДО ВДОСКОНАЛЕННЯ ПРОГНОЗНИХ МОДЕЛЕЙ

3.1. Постановка експерименту та вибір моделей, даних і метрик оцінювання

Для забезпечення об’єктивності, відтворюваності та наукової обґрунтованості дослідження, постановка експерименту включає чітке визначення набору моделей, даних, методології валідації та критеріїв оцінки. В якості основного фінансового інструменту для аналізу було обрано історичні денні котирування криптовалютної пари ETH/USD. Цей актив є оптимальним для даного дослідження через його високу ліквідність, що мінімізує ринкові мікроструктурні шуми, та значну волатильність, яка забезпечує наявність тривалих періодів з різними ринковими режимами. На основі сирих даних було сформовано набір предикторів, що включають ключові класи технічних індикаторів з модифікованими параметрами для виявлення як короткострокових, так і середньострокових закономірностей. Експеримент сфокусовано на вирішенні задачі бінарної класифікації: прогнозування напрямку руху ціни закриття наступного торгового дня, де цільова змінна приймає значення 1 для зростання ціни та 0 в іншому випадку.

Для проведення всебічного порівняльного аналізу було обрано чотири алгоритми, що представляють різні парадигми машинного та глибокого навчання:

1) логістична регресія – лінійний метод, що слугує еталоном для оцінки ефективності більш складних підходів;

2) випадковий ліс – ансамблевий метод, що базується на усередненні прогнозів множини незалежних дерев рішень і відомий своєю стійкістю до перенавчання;

3) градієнтний бустинг – послідовний ансамблевий метод, який вважається одним з найефективніших алгоритмів для роботи зі структурованими даними.

4) LSTM – модель рекурентної нейронної мережі, спеціалізована для аналізу часових послідовностей та виявлення залежностей у часі.

Для уникнення витоку даних з майбутнього, весь набір даних було розділено хронологічно на тренувальну (70%), валідаційну (15%) та тестову (15%) вибірки. Валідаційна вибірка використовувалася для проміжної оцінки та глибокого аналізу помилок моделей, тоді як тестова залишалася недоторканою до фінального етапу для отримання неупередженої оцінки. Для оцінки якості класифікаторів було обрано Accuracy та F1-Score. Остання є особливо важливою, оскільки вона враховує баланс між точністю та повнотою і є більш чутливою до здатності моделі правильно ідентифікувати позитивний клас, що є критичним у контексті алгоритмічної торгівлі, де хибно-позитивні сигнали ведуть до збиткових угод.

3.2. Проведення експериментів з базовими моделями

На основі визначеної методології було проведено серію експериментів з навчання чотирьох обраних моделей на тренувальній вибірці. Для забезпечення рівних умов порівняння, моделі Logistic Regression, Random Forest та XGBoost використовували однаковий набір ознак у табличному форматі. Для моделі LSTM дані були перетворені у формат послідовностей з довжиною ретроспективного вікна у 30 періодів, що дозволяє мережі аналізувати динаміку індикаторів за останній місяць для формування прогнозу на наступний день. Процес навчання був стандартизований і включав масштабування ознак за допомогою MinMaxScaler для приведення їх до діапазону [0, 1], що є необхідним для коректної роботи логістичної регресії та

покращує збіжність нейронної мережі. Навчені об’єкти було збережено для подальшого застосування до валідаційної та тестової вибірок, що гарантує відсутність витоку інформації про розподіл даних з майбутніх періодів. Кожна модель була навчена з використанням стандартних реалізацій з бібліотек scikit-learn, xgboost та tensorflow.keras з базовими гіперпараметрами, щоб оцінити їхню вихідну продуктивність без глибокої оптимізації. Після завершення навчання кожна модель була застосована для генерації прогнозів на валідаційній вибірці. Отримані прогнози, разом з ймовірностями класі, були збережені для подальшого кількісного порівняння та глибокого якісного аналізу за допомогою розробленої системи.

3.3. Порівняльний аналіз продуктивності моделей на валідаційних даних

Кількісне порівняння ефективності навчених моделей на валідаційній вибірці дозволило зробити первинні висновки щодо їхньої прогнозної здатності на нових, небачених даних. Розраховані значення метрик Accuracy та F1-Score для кожної з чотирьох моделей зведено у таблиці 3.1.

Таблиця 3.1 Порівняльні результати продуктивності базових моделей на тестовій вибірці

Модель	Accuracy	F1-Score
Logistic Regression	0.507	0.412
Random Forest	0.523	0.454
XGBoost	0.505	0.479
LSTM	0.483	0.574

Аналіз отриманих емпіричних даних демонструє відмінності в ефективності досліджуваних алгоритмів залежно від обраної метрики

оцінювання. Найбільш відмінний результат показала нейронна мережа LSTM. Попри найнижчу загальну точність на рівні 0,483, ця модель досягла найвищого показника F1-Score 0,574. Це свідчить про те, що архітектура LSTM краще за інші методи справляється з ідентифікацією цільового класу, хоча і схильна до більшої кількості помилкових спрацьовувань у загальному потоці даних.

Ансамблеві методи на основі дерев рішень продемонстрували вищу стабільність прогнозування. Алгоритм Random Forest забезпечив найкращу загальну точність 0,523 при помірному значенні F1-Score 0,454. Модель XGBoost показала збалансовані результати з точністю 0,505 та другим за величиною показником F1-Score 0,479. Водночас лінійна модель Logistic Regression виявилася конкурентоспроможною за точністю 0,507, але продемонструвала найнижчу ефективність за метрикою F1-Score.

Отримані результати спростовують припущення про однозначну перевагу якогось одного підходу. Спостерігається чіткий компроміс між загальною точністю класифікації, де лідирує Random Forest, та здатністю вловлювати позитивні сигнали, де найкращою виявилася LSTM. Вибір оптимальної моделі для побудови торгової стратегії залежатиме від пріоритетності мінімізації ризиків або ж максимізації кількості потенційних угод. Однак, найважливішим висновком є те, що навіть найкраща модель не досягає рівня, достатнього для побудови надійної торгової стратегії. F1-Score нижче 0,5 вказує на те, що модель все ще робить значну кількість помилок. Це ключове спостереження підтверджує центральну гіпотезу дослідження: успіх полягає не у простому виборі найкращого алгоритму, а в необхідності глибокого аналізу причин, чому існуючі моделі працюють недостатньо ефективно.

3.4. Детальний аналіз помилок та ідентифікація систематичних недоліків моделей

Попередній порівняльний аналіз показав, що, незважаючи на різницю в архітектурах, усі протестовані моделі, від простої Логістичної регресії до складної LSTM, демонструють обмежену та недостатню для практичного застосування прогнозну здатність. Це спонукало до проведення більш глибокого дослідження, спрямованого не на порівняння загальних метрик, а на якісний аналіз природи самих помилок. Метою цього етапу було перевірити гіпотезу про те, що помилки моделей не є випадковим шумом, а мають систематичний характер і виникають за певних, ідентифікованих ринкових умов та внутрішніх властивостей моделі. Такий підхід дозволяє виявити спільні сліпі зони, притаманні різним алгоритмам, та сформулювати універсальні принципи для їх вдосконалення. Для аналізу використовувалися прогнози та ймовірності, згенеровані всіма чотирма моделями на валідаційній вибірці, що дозволило провести крос-модельний аналіз та виявити спільні патерни.

Першим виявленим патерном стала низька продуктивність на боковому ринку. Фінансові ринки існують у трьох основних станах: висхідний тренд, низхідний тренд та боковий рух, коли ціна коливається у вузькому діапазоні без чіткого напрямку. Для діагностики цієї проблеми було використано індикатор ADX, який вимірює силу тренду. Значення ADX нижче 20 традиційно вважаються ознакою слабкого тренду або флету. Аналіз продуктивності моделей у розрізі цього індикатора виявив чітку закономірність: коли ADX вказував на наявність сильного тренду, метрики F1-Score були значно вищими, ніж у періоди, коли ринок знаходився у флеті. Як показано на рисунку 3.1, під час бокового руху моделі, особливо ансамблеві, починали генерувати велику кількість хибних сигналів, реагуючи на незначні ринкові шуми, які вони помилково інтерпретували як початок нового руху. Це

свідчить про те, що моделі, навчені розпізнавати трендові патерни, стають неефективними, коли основна парадигма ринку змінюється, і не мають вбудованого механізму для ідентифікації стану невизначеності.

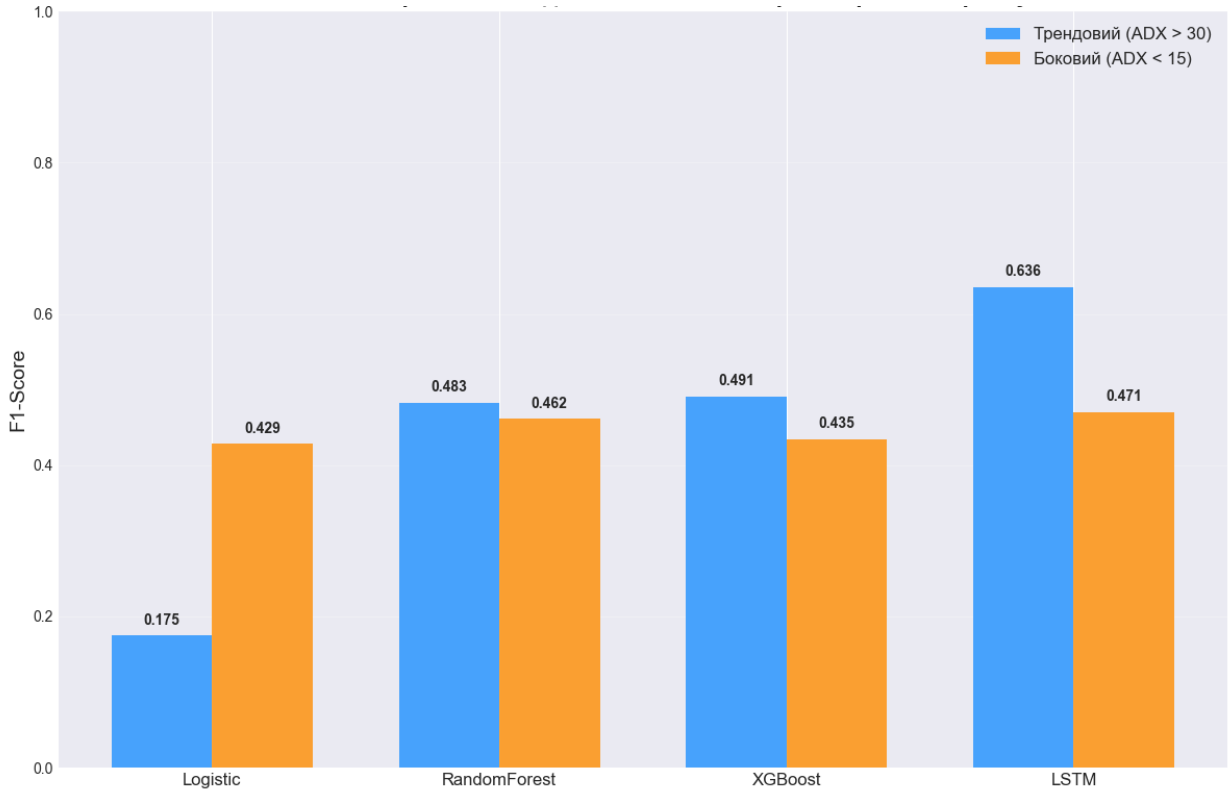

Рисунок 3.1 Порівняння F1-Score моделей в трендовому та боковому ринкових режимах.

Другим важливим спостереженням стало ігнорування моделями власної впевненості у прогнозах. Більшість класифікаторів (окрім Логістичної регресії, яка є природно ймовірнісною) генерують не тільки фінальний прогноз (0 або 1), а й імовірність належності до кожного класу. Ця ймовірність може розглядатися як міра впевненості моделі. Аналіз впливу фільтрації прогнозів за рівнем впевненості (понад 65%) на точність моделей, представлений на рис. 3.1, виявив неочікувані результати. Всупереч припущенню, що вища задекларована впевненість моделі корелює з вищою точністю її передбачень, емпіричні дані демонструють протилежну або нульову тенденцію. Для моделей Logistic Regression та LSTM застосування

цього фільтру призвело до падіння точності. Такий результат свідчить про те, що ці архітектури або взагалі не генерували прогнозів із впевненістю вище 65%, або ж абсолютна більшість таких високоімвірних прогнозів виявилася помилковою.

Ситуація з ансамблевими моделями також суперечить початковій гіпотезі. Застосування фільтру впевненості не покращило, а помітно погіршило їхню загальну результативність, що продемонстровано на рисунку 3.2. Точність моделі RandomForest знизилася з базового рівня 0,512 для всіх прогнозів до 0,442 для прогнозів із високою впевненістю, Аналогічна динаміка спостерігається і для XGBoost, де точність впала з 0,491 до 0,468. Це критичне спостереження вказує на те, що висока заявлена впевненість досліджуваних моделей не є надійним індикатором істинності прогнозу. На практиці відбір найбільш упевнених сигналів не тільки не підвищує якість торгових рішень, а й може призвести до систематичного відбору помилкових прогнозів, що робить стандартний підхід до використання ймовірнісних оцінок цих моделей неефективним.

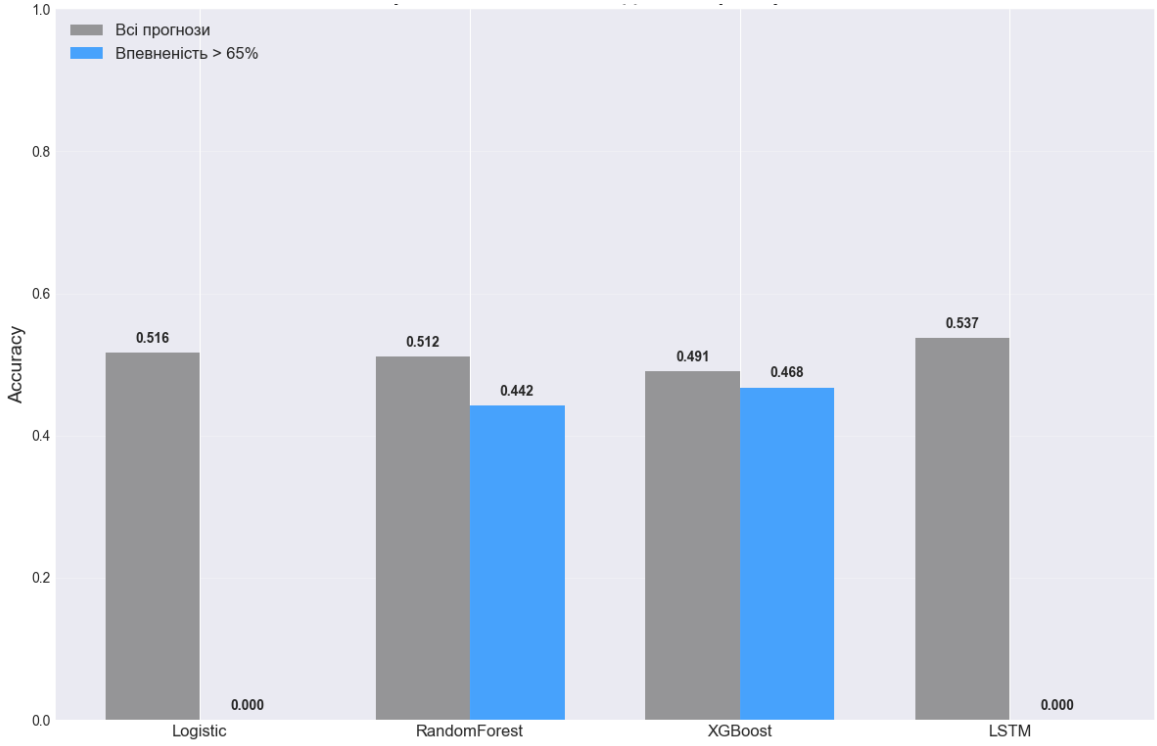

Рисунок 3.2 Залежність точності прогнозу від впевненості моделей.

Третім важливим спостереженням стало виявлення чіткої часової залежності в ефективності алгоритму XGBoost. Детальний розріз результативності за днями тижня спростував попередню гіпотезу про рівномірне погіршення якості прогнозів у вихідні дні через низьку ліквідність активів. Дані демонструють, що динаміка ефективності моделі має складнішу, нелінійну структуру.

Найвищі показники точності та F1-Score були зафіксовані у понеділок та неділю. Це свідчить про те, що модель успішно ідентифікує патерни, які виникають під час відкриття торгового тижня та підготовки ринку до нових сесій наприкінці вихідних. Натомість субота виявилася найскладнішим днем для прогнозування з критично низькими метриками, де F1-Score падає до рівня 0,305. Середина тижня, зокрема вівторок і середа, також характеризується зниженням прогностичної здатності моделі порівняно з піковими днями. Така статистика вказує на те, що алгоритм XGBoost найкраще працює на етапах зародження або відновлення трендів, але втрачає стабільність у періоди ринкової невизначеності або шуму, які найчастіше спостерігаються в суботу та середину робочого тижня (рисунок 3.3).

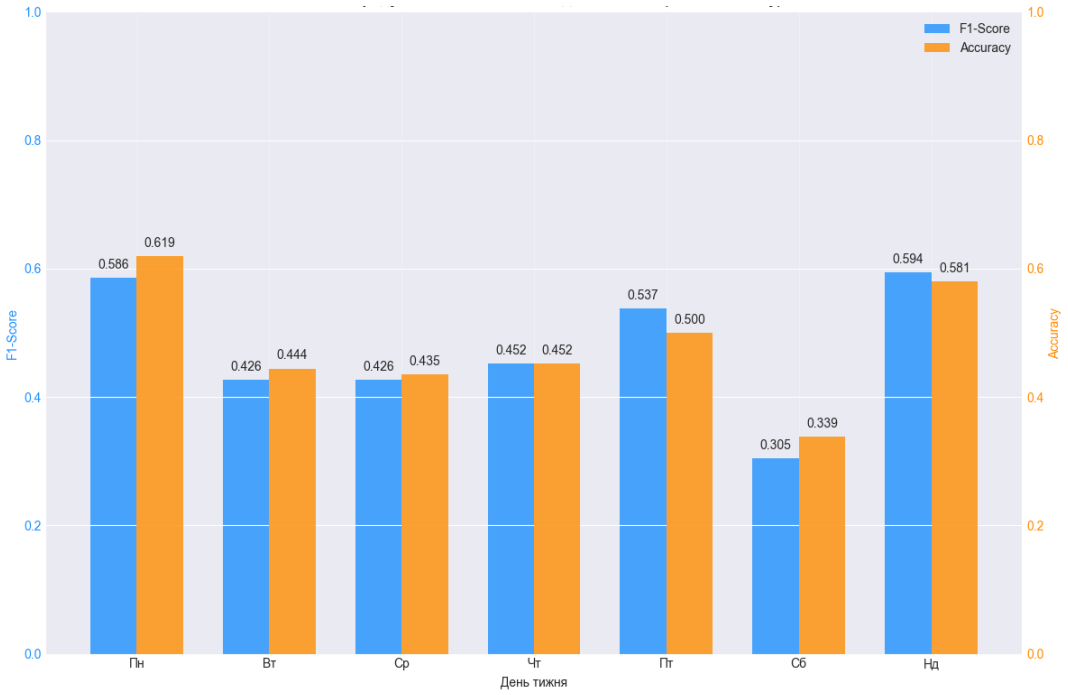

Рисунок 3.3 Продуктивність моделі XGBoost за днями тижня

Четвертим патерном стала виражена асиметрія прогностичної здатності моделей залежно від напрямку торгівлі, а саме для позицій Long та Short. Як демонструє графік порівняння точності моделей для різноспрямованих позицій на рисунку 3.4, більшість алгоритмів не є універсальними та демонструють суттєвий перекіс у бік одного з напрямків.

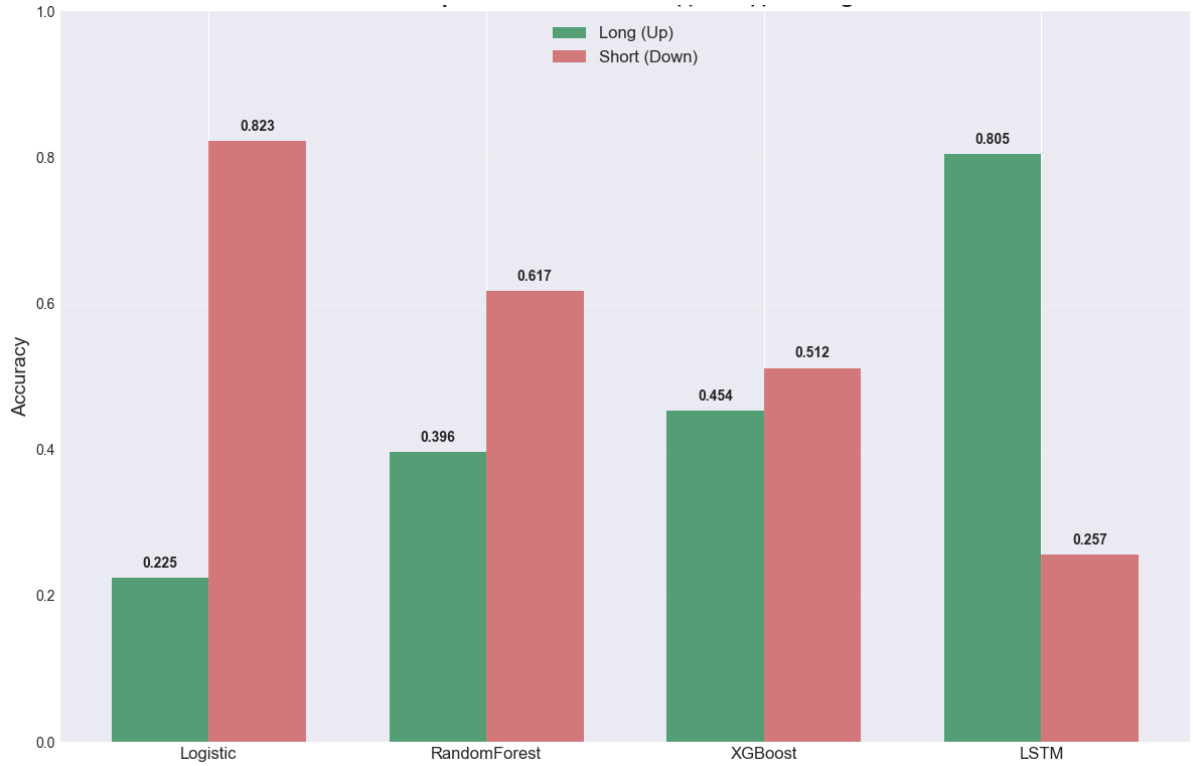

Рисунок 3.4 Порівняння точності моделей для Long і Short позицій

Найбільш показовим є приклад моделі Logistic Regression. Її точність для коротких позицій Short сягає вражаючого рівня 0.823, тоді як точність для довгих позицій Long становить лише 0.225. Це свідчить про те, що модель фактично навчилася передбачати лише падіння ринку, ігноруючи сигнали зростання. Діаметрально протилежну картину демонструє нейронна мережа LSTM. Вона показує високу ефективність у визначенні висхідних трендів з точністю 0.805, але виявляється абсолютно безпорадною при прогнозуванні падінь, де її точність знижується до 0.257.

Така полярність вказує на те, що ці моделі замість вивчення реальних ринкових механізмів просто підлаштувалися під домінуючі тренди в навчальній вибірці або запам’ятали специфічні умови розподілу класів. Винятком на цьому фоні виступає модель XGBoost. Вона продемонструвала найбільш збалансовану поведінку з точністю 0.454 для Long та 0.512 для Short позицій. Хоча ці показники не є високими, відсутність критичного дисбалансу робить XGBoost більш надійним кандидатом для побудови стабільної торгової системи, яка не залежатиме критично від глобального напрямку руху ринку. Виявлена асиметрія інших моделей створює неприйнятні ризики, оскільки зміна глобального тренду призведе до повної втрати їхньої ефективності.

3.5. Узагальнення результатів і формування типових патернів проблем прогнозних моделей

Проведений на попередніх етапах глибокий аналіз помилок дозволяє перейти від часткових спостережень до узагальнення та систематизації. Результати дослідження чітко показали, що недоліки протестованих моделей не є унікальними для конкретного алгоритму, а скоріше відображають фундаментальні виклики, пов’язані з природою фінансових часових рядів. На основі виявлених закономірностей можна сформулювати три основні класи типових патернів проблем, з якими стикаються дослідники при побудові прогнозних моделей для алгоритмічної торгівлі:

Домінуючою тенденцією, що спостерігається у всіх досліджуваних моделях, є контекстна сліпота. Цей патерн об’єднує групу проблем, пов’язаних з нездатністю моделей адаптуватися до зміни зовнішніх умов та ринкових режимів. Моделі, навчені на агрегованих історичних даних, застосовують одну й ту саму статичну логіку, ігноруючи динамічний контекст. В рамках дослідження було виявлено два яскравих прояви цього патерну.

1. Низька продуктивність на боковому ринку – моделі ефективно розпізнають трендові рухи, але стають джерелом хибних сигналів та збитків, коли ринок переходить у стан флету. Емпірично це виявляється у значному падінні метрик якості в періоди, коли індикатор сили тренду знаходиться на низьких рівнях.

2. Ігнорування періодичних патернів – моделі не враховують часові закономірності, такі як зміна волатильності та ліквідності залежно від торгової сесії або дня тижня. Це призводить до систематичного погіршення якості прогнозів у певні, прогнозовані проміжки часу.

Паралельно з цим, було зафіксовано часте неефективне використання інформації. Цей патерн стосується проблем, пов’язаних з тим, як моделі обробляють та інтерпретують інформацію, закладену як у вхідних ознаках, так і у власних вихідних даних. Під час дослідження було ідентифіковано два ключові недоліки.

1. Інформаційна надлишковість – моделі отримують на вхід набір ознак зі значною мультиколінеарністю, що ускладнює їх інтерпретацію, знижує стабільність і може призводити до перенавчання. Це було підтверджено аналізом кореляційної матриці та розподілу важливості ознак.

2. Ігнорування впевненості прогнозу – моделі генерують велику кількість прогнозів з низькою ймовірністю (близькою до 50%), які, як показав аналіз, мають якість, близьку до випадкового вгадування. Стандартний підхід, що однаково довіряє всім сигналам, призводить до накопичення збитків від великої кількості невпевнених та хибних прогнозів.

Аналіз також виявив такий патерн як процедурні та методологічні ризики.

Ця група проблем стосується не самої моделі, а процесу її розробки, валідації та управління ризиками. Звіти, згенеровані для моделей, виявили потенційні ризики, які є критично важливими для практичного застосування.

1. Ризик переоптимізації – процес розробки та валідації моделі без чіткого розділення даних на тренувальну, валідаційну та відкладену тестову

вибірки створює високий ризик підгонки моделі під історичні дані, що робить її нежиттєздатною в майбутньому.

2. Ігнорування управління ризиками – висока потенційна прибутковість стратегії може супроводжуватися неприйнятно великими просадками капіталу, що робить її занадто ризикованою для реального використання.

Формулювання цих патернів є ключовим кроком, оскільки воно переводить проблему з рівня модель погано працює на рівень модель погано працює, тому що вона сліпа до ринкового режиму, не використовує власну впевненість та розроблена з методологічними ризиками. Таке чітке визначення проблем відкриває шлях до їх цілеспрямованого та систематичного вирішення.

3.6. Обґрунтування потреби у створенні системи автоматизованої діагностики та вдосконалення моделей

Процес виявлення та аналізу описаних патернів проблем, проведений в рамках даного дослідження, був виконаний вручну і вимагав значних часових затрат, експертних знань та написання спеціалізованого коду для кожного окремого аналізу. У реальних умовах, коли дослідник працює з десятками моделей та гіпотез, такий ручний підхід стає головним вузьким місцем, що сповільнює ітеративний процес вдосконалення моделі.

Ця проблема вказує на нагальну потребу в систематизації та автоматизації процесу діагностики. Якщо типові проблеми та методи їх виявлення можна формалізувати, то можливо створити програмний інструмент, який би виконував цю роботу автоматично. Така система могла б приймати на вхід результати роботи будь-якої моделі і, спираючись на формалізовану базу знань, проводити комплексний аудит моделі. Вона б

автоматично перевіряла наявність типових симптомів тобто патернів проблем і надавала досліднику структурований звіт з діагнозом та, що найважливіше, з конкретними, пріоритезованими рекомендаціями щодо подальших кроків для вдосконалення.

Створення такої системи дозволило б вирішити кілька ключових завдань:

1) значне скорочення часу, що витрачається на рутинну діагностику від днів до хвилин, тим самим прискоривши дослідницький цикл;

2) підвищення об’єктивності та глибини аналізу застосовувши широкий спектр стандартизованих тестів, які дослідник може пропустити при ручному аналізі;

3) демократизація знань за рахунок зроблення передових практик аналізу моделей доступними навіть для дослідників-початківців, інкапсулюючи експертні знання у вигляді програмних правил та рекомендацій.

Таким чином, розробка системи є не просто технічним завданням, а логічним та необхідним наступним кроком, що випливає з результатів експериментального дослідження. Вона є інструментом, що дозволяє перейти від одноразових експериментів до створення стійкого, відтворюваного та ефективного процесу ітеративного покращення прогнозних моделей в алгоритмічній торгівлі.

3.7. Розробка системи автоматизованої діагностики та покращення моделей

3.7.1. Архітектура та принципи функціонування системи автоматизованої діагностики

У рамках даного дослідження було розроблено концепцію та програмну реалізацію спеціалізованої платформи призначеної для автоматизації процесів діагностики та вдосконалення прогнозних моделей у сфері алгоритмічної торгівлі. Необхідність створення такої системи зумовлена складністю виявлення причин низької ефективності алгоритмів на фінансових ринках, де стандартні метрики часто не відображають реальної придатності моделі до торгівлі. Запропоноване рішення дозволяє перейти від інтуїтивного пошуку помилок до стандартизованого, відтворюваного процесу аудиту, який базується на формалізованих експертних знаннях. Система виступає інтелектуальною надбудовою над класичним конвеєром машинного навчання та забезпечує трансформацію сирих результатів моделювання у структурований план оптимізації торгової стратегії.

Проектування архітектури системи здійснювалося за модульним принципом, що дозволяє забезпечити гнучкість налаштування та легку інтеграцію нових діагностичних правил без зміни базового коду. Як зображено на рисунку 3.5 структурна організація платформи складається з трьох логічних рівнів: вхідного шару даних, ядра системи діагностики та вихідного шару звітності.

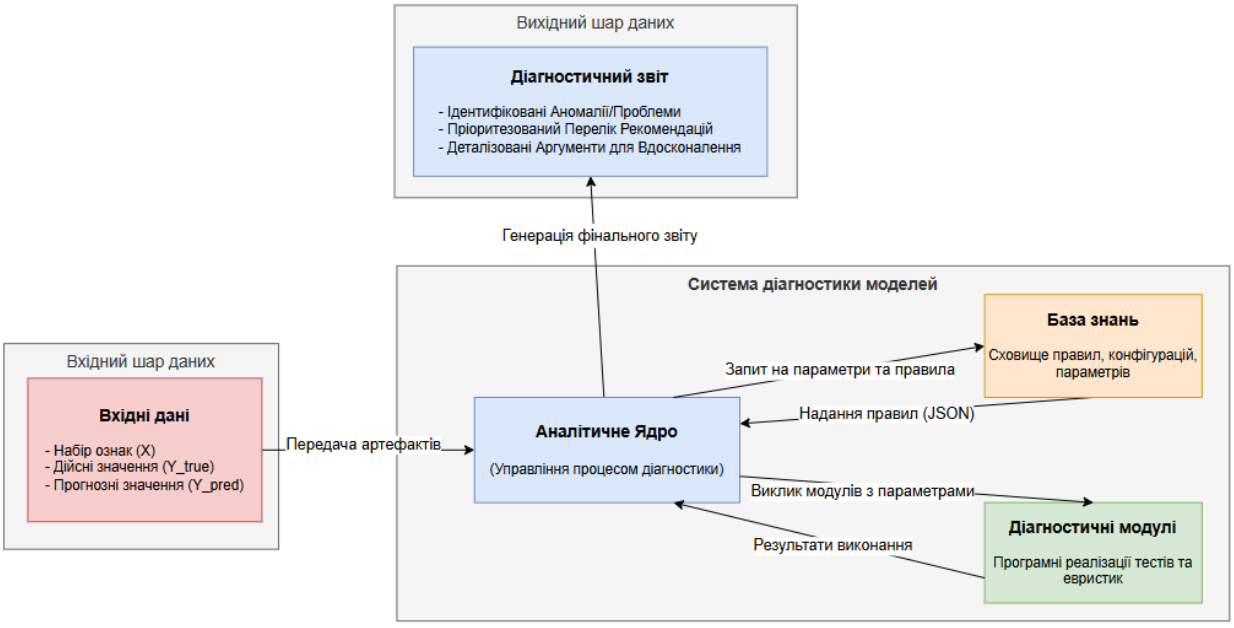

Рисунок 3.5 Концептуальна архітектура системи діагностики моделей

Вхідний шар відповідає за отримання та попередню обробку артефактів роботи моделі, які включають набір ознак, вектор дійсних значень цільової змінної та вектор прогнозних значень, згенерованих алгоритмом. Ці дані передаються до центрального компонента системи – аналітичного ядра.

Аналітичне ядро виконує роль оркестратора всього діагностичного процесу. Воно керує потоком виконання, ініціюючи запити до бази знань для отримання актуальних конфігурацій та правил перевірки. На основі отриманих інструкцій ядро викликає відповідні діагностичні модулі, які є ізольованими програмними реалізаціями специфічних тестів та евристик. Така організація взаємодії компонентів забезпечує послідовну верифікацію моделі на наявність широкого спектру відомих проблем, від статистичних аномалій у даних до специфічних помилок ринкової логіки.

Результатом роботи системи є генерація комплексного діагностичного звіту, який формується на вихідному шарі. Цей документ містить деталізований перелік ідентифікованих аномалій, аргументацію щодо причин їх виникнення та пріоритезований список рекомендацій для вдосконалення моделі.

3.7.2. Проектування бази знань на основі виявлених патернів проблем

Фундаментальною відмінністю розробленої системи від існуючих аналогів є використання гнучкої, документо-орієнтованої бази знань, яка виступає сховищем формалізованого експертного досвіду. Замість жорсткого кодування логіки перевірок у тілі програми, правила діагностики винесені у зовнішні конфігураційні файли, що дозволяє динамічно розширювати функціонал системи.

Структура кожного запису в БЗ уніфікована та складається з двох ключових блоків, що проілюстровано на рисунку 3.6. Першим компонентом є об’єкт-детектор, який містить логіку ідентифікації проблеми. Він описує необхідні вхідні дані, статистичні методи перевірки та порогові значення, при досягненні яких фіксується наявність симптому певної проблеми. Наприклад, правило може визначати критичний рівень кореляції між ознаками або допустиму межу розбіжності точності на тренувальній та тестовій вибірках.

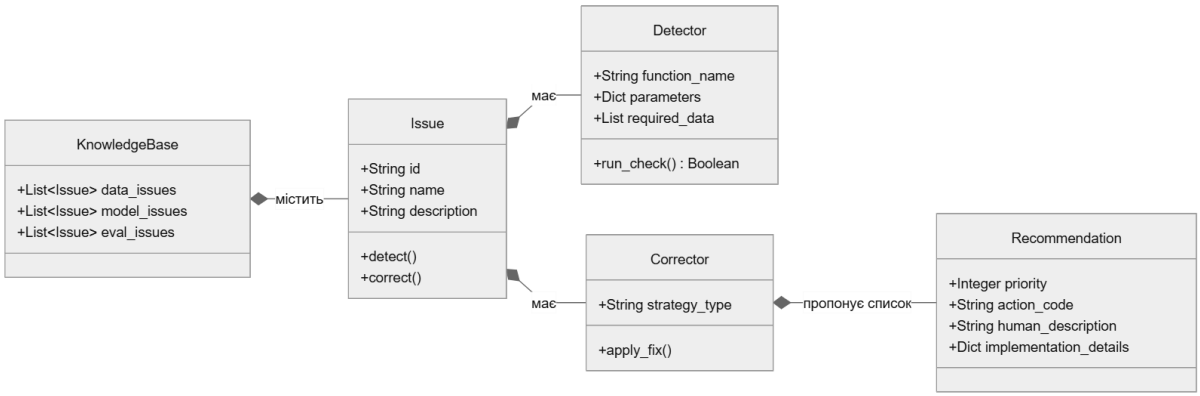

Рисунок 3.6. Структура документа «Правило» в Базі Знань

Другим компонентом правила є об’єкт-коректор, який містить інструкції для вирішення виявленої проблеми. Цей блок зберігає інформацію про рекомендовані дії, такі як методи трансформації даних, алгоритми відбору

ознак або стратегії налаштування гіперпараметрів. Завдяки такій двокомпонентній структурі система здатна не лише констатувати факт неефективності, а й автоматично пропонувати науково обґрунтовані методи її усунення. Це дозволяє реалізувати повний цикл вдосконалення моделі, де кожен етап базується на перевірених емпіричних закономірностях, виявлених у ході попередніх експериментальних досліджень.

3.8. Практичне застосування системи для автоматизованого аналізу та вдосконалення моделей

Теоретична концепція системи була перевірена на практиці шляхом її застосування для комплексного аналізу всіх чотирьох раніше навчених базових моделей: Logistic Regression, Random Forest, XGBoost та LSTM. Цей етап імітує реальний сценарій роботи дослідника, який, отримавши початкові результати, використовує автоматизовану систему для глибокої діагностики кожної моделі-кандидата, порівняння їхніх слабких місць та формування єдиної, обґрунтованої стратегії для подальшого вдосконалення.

На вхід системи було послідовно подано артефакти роботи кожної з чотирьох моделей на валідаційній вибірці (набори ознак, справжні значення, прогнози, ймовірності, метрики). Аналітичне ядро системи застосувало логіку детекторів з бази знань, згенерувавши індивідуальні діагностичні звіти для кожного алгоритму. Аналіз цих звітів дозволив виявити як спільні, так і унікальні для кожної моделі проблеми.

Результати проведеного крос-модельного діагностування дозволили виявити низку фундаментальних проблем, які мають як універсальний, так і специфічний характер для різних архітектур. Найбільш поширеним недоліком, ідентифікованим для всіх чотирьох алгоритмів, стала наявність надлишкових ознак у вхідних даних, що свідчить про високу

мультиколінеарність між стандартними технічними індикаторами та необхідність їх попередньої ортогоналізації або відбору. Іншим критичним зауваженням системи стало ігнорування рівня впевненості прогнозу, оскільки всі моделі генерували сигнали навіть за низької ймовірності правильності рішення, що в умовах зашумлених фінансових даних призводить до значної кількості хибних спрацювань. Також діагностика вказала на високий ризик переоптимізації стратегій через використання статичного розбиття даних, рекомендувавши перехід до більш робастних методів валідації типу Walk-Forward для запобігання підгляданню в майбутнє.

Поглиблений аналіз специфічних вразливостей виявив чітку диференціацію проблем залежно від типу алгоритму. Для моделей Logistic Regression та LSTM було зафіксовано статистично значущу автокореляцію помилок, що вказує на те, що навіть рекурентна нейронна мережа при поточному наборі ознак не змогла повністю декомпозувати часову структуру ряду та залишила частину детермінованої інформації у залишках. Водночас ансамблеві методи Random Forest та XGBoost продемонстрували класичні ознаки перенавчання та схильність до надмірних просадок капіталу, що пояснюється відсутністю жорстких обмежень складності дерев та механізмів регуляризації у базових налаштуваннях. Крім того, окремі моделі виявили критичну чутливість до зміни фаз ринку, систематично втрачаючи ефективність під час бокового руху цін, що підтвердило гіпотезу про необхідність явно ідентифікувати стан ринку як окрему ознаку для навчання.

На основі синтезу отриманих діагностичних висновків було розроблено єдиний комплексний план модернізації, який охоплює всі етапи побудови торгової системи від обробки даних до логіки виконання угод. Стратегія вдосконалення передбачає глибокий інжиніринг ознак, в рамках якого корельовані вхідні змінні замінюються на комбіновані показники, такі як гістограма MACD та ширина смуг Боллінджера, а також вводиться новий індикатор ринкового режиму на базі ADX для розрізнення трендових і флетових станів. Для усунення перенавчання та підвищення узагальнюючої

здатності моделей заплановано проведення процедури оптимізації гіперпараметрів через пошук по сітці, що дозволить знайти баланс між зміщенням та дисперсією. Фінальним етапом модернізації є інтеграція у торговий алгоритм фільтрів впевненості, які відсікають слабкі сигнали з ймовірністю нижче порогового значення, та впровадження динамічного управління ризиками через адаптивний Stop-Loss, що базується на поточній волатильності активу. Реалізація цього системного підходу дозволяє трансформувати набір базових алгоритмів у цілісну адаптивну торгову стратегію, готову до фінальної перевірки на незалежних тестових даних.

3.9. Фінальна верифікація та порівняння результатів базових та вдосконалених моделей

Фінальним та найбільш важливим етапом експериментального дослідження є оцінка ефективності запропонованого підходу на недоторканій тестовій вибірці. Цей крок дозволяє отримати неупереджену оцінку того, чи призвів комплексний план вдосконалення, сформований системою, до реального покращення узагальнюючої здатності моделей. Для цього всі чотири алгоритми були перенавчені з урахуванням застосованих корекцій та оцінені на тестових даних.

Результати порівняння продуктивності базових та вдосконалених версій кожної моделі зведено у фінальній порівняльній таблиці 3.2.

Аналіз результатів фінальної верифікації підтверджує ефективність запропонованої методології вдосконалення моделей системою для всіх типів досліджуваних алгоритмів. Отримані дані дозволяють зробити висновки щодо різної чутливості архітектур до проведеної оптимізації.

Таблиця 3.2 Фінальне порівняння продуктивності моделей на тестовій вибірці до та після застосування рекомендацій системи

Модель	F1-Score (Baseline)	F1-Score (Improved)	Зміна F1, %
Logistic Regression	0.412	0.676	+64.1%
Random Forest	0.454	0.550	+21.1%
XGBoost	0.479	0.527	+10.0%
LSTM	0.574	0.672	+17.1%

По-перше, абсолютним лідером за темпами приросту ефективності стала модель Logistic Regression. Показник F1-Score для цього алгоритму зріс на 64,1 відсотка і досяг найвищого значення серед усіх моделей на рівні 0,676. Такий результат свідчить про те, що усунення мультиколінеарності та введення нових контекстних ознак мають критичне значення саме для лінійних моделей. Очищення даних дозволило цьому простому алгоритму розкрити свій потенціал і перевершити складніші ансамблеві методи.

По-друге, нейронна мережа LSTM підтвердила свою високу здатність до прогнозування часових рядів. Зберігши високі стартові позиції, вона покращила свій результат на 17,1 відсотка та досягла показника 0,672, що є другим найкращим результатом у тесті. Це спростовує побоювання щодо складності навчання рекурентних мереж на цих даних і доводить, що при правильній підготовці вхідних векторів LSTM здатна формувати стійкі та точні торгові сигнали.

По-третє, ансамблеві методи Random Forest та XGBoost продемонстрували помірне, але стабільне зростання якості прогнозування. Random Forest покращив свій показник на 21,1 відсотка до рівня 0,550, тоді як приріст для XGBoost склав 10,0 відсотків із фінальним результатом 0,527. Менша динаміка покращення цих моделей порівняно з лінійною регресією може пояснюватися тим, що алгоритми на основі дерев рішень вже на базовому етапі здатні частково компенсувати шуми та нелінійні залежності в

даних, тому ефект від зовнішньої оптимізації для них виявився менш масштабним, ніж для лінійної моделі.

3.10 Висновки до третього розділу

Основний висновок даного експериментального розділу є багатогранним. Систематичний, керований даними підхід до аналізу та вдосконалення, реалізований в системі, довів свою ефективність для цілого класу моделей, дозволивши значно покращити їхню узагальнюючу здатність. У межах експериментів система була апробована на різнорідних алгоритмах – Random Forest, XGBoost, LSTM та Logistic Regression – що дало змогу оцінити її працездатність як для класичних моделей, так і для глибинних архітектур.Водночас, дослідження виявило, що не існує універсального рецепту вдосконалення, і ефективність рекомендацій може залежати від специфіки самого алгоритму. Розроблена система виступає як потужний інструмент, що формалізує та автоматизує процес діагностики, дозволяючи дослідникам швидко виявляти слабкі місця та генерувати обґрунтовані гіпотези для подальшого тестування. Це доводить, що ключ до створення надійних прогнозних систем лежить у синергії між потужними алгоритмами та інтелектуальними інструментами для їх глибокої, ітеративної та модельно-специфічної діагностики.

РОЗДІЛ 4 РОЗРОБКА ВЛАСНОГО СТАРТАП ПРОЕКТУ

4.1 План розробки стартапу та масштабування його на ринок

Впровадження розробленої системи автоматизованої діагностики прогнозних моделей у реальний сектор економіки вимагає чіткого планування етапів перетворення наукової розробки на комерційний продукт. План розробки стартапу базується на принципах ітеративної розробки та методології ощадливого виробництва, що передбачає поетапне створення цінності для кінцевого споживача з постійною перевіркою продуктових гіпотез. В основу дорожньої карти проекту покладено концепцію поступової еволюції від лабораторного прототипу системи діагностики до масштабованої хмарної платформи, здатної обслуговувати тисячі користувачів одночасно. Життєвий цикл реалізації проекту розділено на чотири послідовні фази, кожна з яких має чітко визначені завдання, необхідні ресурси та критерії успішного завершення.

Перша фаза, яка класифікується як етап науково-дослідних та дослідно-конструкторських робіт, сфокусована на фіналізації архітектури бази знань та перетворенні програмного коду, розробленого в рамках магістерської дисертації, на стабільне серверне ядро. Ключовим завданням цього періоду є наповнення бази знань новими правилами детектування помилок, що охоплюють ширший спектр фінансових інструментів, окрім криптовалют, а також розробка уніфікованого інтерфейсу прикладного програмування для взаємодії зовнішніх систем з аналітичним ядром. Важливим аспектом є проведення юридичної експертизи та захист прав інтелектуальної власності на розроблені алгоритми аналізу впевненості та стабільності моделей.

Друга фаза передбачає створення мінімально життєздатного продукту, орієнтованого на отримання зворотного зв’язку від перших користувачів. На цьому етапі розробляється веб-інтерфейс користувача, який забезпечує зручне

завантаження даних та візуалізацію діагностичних звітів без необхідності встановлення спеціалізованого програмного забезпечення. Особлива увага приділяється створенню підсистеми білінгу та управління підписками, а також інтеграції з хмарними провайдерами для забезпечення відмовостійкості системи. Завершенням цього етапу є проведення закритого бета-тестування серед обмеженої групи кількісних трейдерів та аналітиків, що дозволить виявити критичні помилки та оптимізувати користувацький досвід перед публічним релізом.

Третя фаза знаменує собою вихід на ринок та початок активної комерційної експлуатації системи. Основні зусилля на цьому етапі спрямовуються на реалізацію маркетингової стратегії, залучення трафіку та конвертацію безкоштовних користувачів у платних передплатників. Передбачається запуск партнерських програм з освітніми платформами для трейдерів та постачальниками фінансових даних, що дозволить швидко сформувати базу лояльних клієнтів. Технічна команда в цей час фокусується на оптимізації швидкодії алгоритмів та додаванні функціоналу для підтримки високочастотної торгівлі.

Четверта фаза присвячена масштабуванню бізнесу та виходу на корпоративний сегмент ринку. Стратегія масштабування передбачає розробку спеціалізованих рішень для інвестиційних фондів та проп-трейдингових компаній, які потребують глибокої інтеграції системи діагностики у власні внутрішні процеси розробки стратегій. Це вимагатиме створення відокремлених інсталяцій системи, що працюють у закритому контурі замовника для забезпечення максимальної безпеки даних. Детальний календарний план-графік реалізації цих етапів із зазначенням термінів та відповідальних осіб наведено в таблиці 4.1.

Таблиця 4.1 Календарний план-графік реалізації стартап-проекту

Етап та назва робіт	Тривалість реалізації, міс.	Очікувані результати етапу
1. Дослідження та розробка	1–3	Створено стабільне ядро системи
1.1. Оптимізація алгоритмів ядра	1	Зменшено час обробки даних на 40 відсотків
1.2. Розширення Бази Знань	1	Додано 20 нових правил діагностики ризиків
1.3. Розробка API-шлюзу	1	Реалізовано RESTful API для зовнішніх запитів
2. Створення MVP та інфраструктури	2–4	Запущено публічну бета-версію
2.1. Розробка Frontend-частини	2	Створено особистий кабінет користувача
2.2. Налаштування хмарної інфраструктури	1	Розгорнуто кластер Kubernetes для масштабування
2.3. Інтеграція платіжних шлюзів	1	Підключено Stripe та криптовалютні платежі
3. Комерційний запуск	5–7	Отримано першу виручку
3.1. Маркетингова кампанія	2	Залучено перші 1000 реєстрацій на платформі
3.2. Контент-маркетинг та SEO	Постійно	Опубліковано 10 кейс-стаді ефективності системи
3.3. Запуск реферальної програми	1	Сформовано мережу партнерів-інфлюенсерів

Для успішного закріплення на різних ринках передбачається адаптація продукту під вимоги місцевих фінансових регуляторів та локалізація інтерфейсу. Технологічне масштабування забезпечуватиметься переходом на мікросервісну архітектуру, що дозволить незалежно збільшувати потужність обчислювальних модулів при зростанні навантаження. Окремим вектором розвитку стане відкриття програмного інтерфейсу для сторонніх розробників, що дозволить створити екосистему додатків навколо платформи діагностики, перетворюючи стартап з монолітного продукту на інфраструктурну платформу для індустрії алгоритмічної торгівлі.

4.2 Опис ідеї стартап-проекту

В основу ідеї стартап-проекту покладено концепцію створення спеціалізованої інтелектуальної платформи для автоматизованого аудиту та вдосконалення алгоритмів у сфері алгоритмічної торгівлі. Головна ідея проекту полягає у вирішенні фундаментальної проблеми індустрії, яка пов'язана з відсутністю стандартизованих інструментів для глибокої діагностики причин неефективності торгових стратегій. Існуючі на ринку рішення здебільшого фокусуються на двох крайніх полюсах: це або класичні системи бек-тестування, які лише констатують історичну прибутковість без пояснення причин збитків, або загальні MLOps-платформи, які технічно обслуговують життєвий цикл моделей, але не враховують специфіку фінансових ринків. Запропонований продукт заповнює цю нішу, виступаючи у ролі віртуального головного аналітика, який здатний автоматично перевірити модель на наявність прихованих дефектів, таких як мультиколінеарність вхідних даних, ігнорування ринкових режимів або підглядання у майбутнє, та надати конкретні рекомендації щодо їх усунення.

Сутність інноваційної пропозиції полягає у використанні унікальної архітектури на базі документо-орієнтованої бази знань, що містить формалізований експертний досвід у вигляді правил-детекторів та правил-коректорів. Користувач системи завантажує результати роботи своєї моделі, після чого аналітичне ядро проводить серію перехресних перевірок, використовуючи статистичні тести та евристичні алгоритми. На відміну від конкурентів, система не просто виявляє проблему, наприклад, низьку точність на боковому ринку, а й пропонує алгоритмічне рішення, як-от додавання фільтру волатильності або зміну архітектури вхідних ознак. Цінність продукту для клієнта вимірюється у значному скороченні часу на розробку стратегій та мінімізації фінансових ризиків, пов'язаних із запуском перенавчених алгоритмів у реальну торгівлю, що підтверджено експериментально зростанням метрик ефективності моделей.

Для визначення ринкового позиціонування та унікальних переваг розробленого продукту було проведено порівняльний аналіз із основними класами існуючих рішень, результати якого наведено в таблиці 4.2.

Таблиця 4.2. Порівняльна характеристика ідеї проекту з існуючими аналогами

Характеристика порівняння	Традиційні бек-тестери	Універсальні AutoML платформи	Запропонована система діагностики
Основний об’єкт аналізу	Фінансовий результат (P&L)	Гіперпараметри моделі	Логіка та семантика помилок
Метод виявлення проблем	Ручний аналіз графіків	Перебір метрик якості	Автоматичні детектори та тести

Продовження таблиці 4.2

Наявність рекомендацій	Відсутні	Автоматичний вибір кращої моделі	Детальний план вдосконалення
Інтерпретація результатів	Низька (чорна скринька)	Середня (SHAP-значення)	Висока (причинно-наслідкові зв'язки)
Врахування специфіки ринку	Високе	Низьке	Високе (режими, мікроструктура)
Рівень автоматизації аудиту	Відсутній	Частковий	Повний

Як видно з наведеної таблиці, ключовою конкурентною перевагою стартап-проекту є фокус на інтерпретації та виправленні внутрішньої логіки торгових стратегій, чого не пропонують існуючі альтернативи. У той час як бек-тестери показують, скільки грошей втратила стратегія, а AutoML платформи намагаються оптимізувати математичні параметри, запропонована система пояснює, чому саме виникли втрати, та надає інженерні рішення для виправлення ситуації. Це дозволяє позиціонувати проект як необхідний інструмент третього покоління для індустрії алгоритмічної торгівлі, який доповнює існуючу інфраструктуру інтелектуальним шаром діагностики та прийняття рішень.

4.3 Технологічний аудит ідеї проекту

Успішна комерціалізація розробленої системи автоматизованої діагностики прогнозних моделей значною мірою залежить від правильності вибору технологічного стеку та архітектурних рішень, що лежать в основі продукту. Технологічний аудит ідеї проекту спрямований на оцінку реалізованості запропонованих алгоритмів, доступності необхідних ресурсів та здатності системи до масштабування в умовах реального ринку. В основу програмної реалізації покладено мову програмування Python, яка на сьогодні є де-факто промисловим стандартом у сфері фінансового аналізу та машинного навчання. Цей вибір обумовлений наявністю розвиненої екосистеми бібліотек з відкритим вихідним кодом, що дозволяє суттєво знизити собівартість розробки та прискорити виведення продукту на ринок. Архітектура системи спроектована за модульним принципом, де ключовим інноваційним елементом є відокремлення логіки діагностики від програмного коду виконання. Реалізація бази знань у форматі документо-орієнтованої структури дозволяє динамічно оновлювати правила перевірки моделей без необхідності зупинки сервісу чи перекомпіляції ядра. Це забезпечує системі високу гнучкість та адаптивність до появи нових типів ринкових аномалій.

Критично важливим аспектом технологічного аудиту є оцінка можливості обробки великих масивів фінансових даних у режимі реального часу. Для забезпечення необхідної швидкодії в системі використовуються векторизовані обчислення та методи розпаралелювання процесів, що дозволяє проводити комплексний аудит торгової стратегії за лічені хвилини. Інфраструктурне рішення базується на використанні хмарних технологій та контейнеризації, що гарантує відмовостійкість сервісу та можливість автоматичного масштабування обчислювальних потужностей при зростанні навантаження. Важливою технологічною перевагою є використання уніфікованих протоколів обміну даними, що спрощує інтеграцію системи з

існуючими торговими терміналами та платформами розробки стратегій. Результати аудиту ключових технологічних компонентів стартап-проекту, їх доступності та ступеня готовності наведено в таблиці 4.3.

Таблиця 4.3. Технологічний аудит компонентів стартап-проекту

Компонент системи	Обрана технологія або рішення	Обґрунтування вибору та статус реалізації
Аналітичне ядро	Python, NumPy, Pandas, Scikit-learn, tensorflow.keras	Забезпечує високу швидкість математичних обчислень та сумісність з ML-моделями користувачів. Реалізовано на рівні робочого прототипу.
База Знань	JSON-схема, NoSQL сховище	Дозволяє зберігати ієрархічні правила діагностики та легко їх редагувати. Структура розроблена, наповнення триває.
Обробка даних	Власні алгоритми	Унікальна технологія препроцесингу, що відновлює нормальність розподілу даних. Алгоритми протестовано та верифіковано.

Продовження таблиці 4.3

Інтерфейс	React.js, RESTful API	Забезпечує реактивність вебдодатку та зручність взаємодії користувача з системою. Знаходиться на стадії проектування MVP.
Інфраструктура	Docker, Kubernetes, AWS/GCP	Гарантує ізоляцію процесів користувачів та автоматичне масштабування. Технології доступні та широко використовуються.

Проведений технологічний аудит підтверджує, що ідея проекту є технічно реалізованою з використанням існуючих, перевірених часом технологій. Обраний стек дозволяє мінімізувати ризики розробки, оскільки базується на стабільних компонентах з широкою підтримкою спільноти. Ключові технологічні бар'єри, пов'язані зі складністю обробки нестаціонарних фінансових рядів, успішно подолані в рамках наукової частини магістерської дисертації шляхом розробки власних алгоритмів препроцесингу та валідації. Це створює міцний технологічний фундамент для переходу до наступних етапів комерціалізації та запуску продукту.

4.4 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкового потенціалу та аналіз конкурентного середовища є критично важливим етапом планування комерціалізації розробленої системи, оскільки це дозволяє оцінити обсяг цільового ринку та ідентифікувати ключові фактори успіху продукту. Сучасний глобальний ринок програмного забезпечення для алгоритмічної торгівлі демонструє стійку тенденцію до зростання, що зумовлено демократизацією фінансових інструментів та збільшенням доступності біржових даних для приватних інвесторів. Основним драйвером попиту на аналітичні інструменти є перехід значної частини роздрібних трейдерів від ручного управління капіталом до використання автоматизованих торгових систем та методів машинного навчання. Цей сегмент користувачів, який можна охарактеризувати як професійні роздрібні інвестори, стикається з проблемою відсутності інструментів інституційного рівня для перевірки якості своїх стратегій. Існуючі на ринку платформи пропонують або занадто спрощений функціонал бек-тестування, або надмірно складні та дорогі рішення, орієнтовані на великі інвестиційні банки. Стартап-проект націлений на заповнення цієї ринкової ніші, пропонуючи доступний інструмент глибокої діагностики, що базується на передових наукових методах.

Для формування ефективної стратегії виходу на ринок було проведено сегментацію потенційних споживачів продукту. Аналіз дозволив виділити три ключові групи клієнтів, які мають специфічні потреби та різний рівень платоспроможності. Перша група включає індивідуальних розробників торгових роботів та дата-саєнтистів, які потребують інструментів для валідації своїх моделей перед запуском у реальну торгівлю. Друга група складається з невеликих проп-трейдингових компаній та хедж-фондів, які зацікавлені у стандартизації процесів розробки та контролю ризиків. Третя група охоплює освітні платформи та курси з алгоритмічного трейдингу, які можуть

використовувати систему як навчальний інструмент для студентів. Детальна характеристика виділених ринкових сегментів та оцінка їхньої привабливості наведені в таблиці 4.4.

Таблиця 4.4. Характеристика потенційних сегментів ринку для стартап-проекту

Сегмент ринку	Потреби та больові точки клієнтів	Рівень платоспроможності	Потенціал охоплення
Індивідуальні алготрейдери	Потреба у перевірці гіпотез, уникненні збитків від перенавчання, економія часу на аналіз помилок	Середній	Високий
Проп-трейдингові фірми	Необхідність аудиту стратегій кандидатів, контроль ризиків, автоматизація відбору кращих алгоритмів	Високий	Середній
Освітні платформи	Потреба в наочній демонстрації помилок, автоматизація перевірки студентських робіт	Середній	Низький
Розробники торгових ботів на замовлення	Підтвердження якості коду перед замовником, прискорення здачі проектів	Високий	Середній

Аналіз конкурентного середовища за моделлю п’яти сил Портера свідчить про помірний рівень загрози з боку існуючих гравців та високий поріг входу для нових конкурентів. Основна конкуренція розгортається у сегменті

платформ для бек-тестування, проте ніша спеціалізованих систем діагностики та аудиту залишається малоосвоєною. Бар'єром для входу нових учасників є необхідність наявності глибокої експертизи у сфері фінансового машинного навчання та наявність сформованої бази знань про типи ринкових аномалій, що є ключовим інтелектуальним активом даного проекту. Загроза з боку товарів-замінників, таких як ручний аналіз або використання стандартних метрик бібліотек машинного навчання, знижується через зростаючу складність моделей, яку неможливо ефективно проаналізувати без спеціалізованих інструментів.

Для комплексної оцінки внутрішнього потенціалу проекту та зовнішніх ринкових факторів було проведено SWOT-аналіз. Цей метод дозволив ідентифікувати сильні сторони продукту, які необхідно розвивати, слабкі сторони, що потребують нівелювання, а також можливості для зростання та потенційні загрози. Результати аналізу, представлені в таблиці 4.5, демонструють, що проект має сильні технологічні переваги завдяки унікальній методології діагностики, але потребує значних маркетингових зусиль для побудови впізнаваності бренду.

Таблиця 4.5. SWOT-аналіз стартап-проекту

Сильні сторони (Strengths)	Слабкі сторони (Weaknesses)
Унікальна методологія виявлення семантичних помилок моделей. Висока швидкість обробки даних завдяки оптимізованому ядру. Можливість надання конкретних рекомендацій замість абстрактних метрик. Гнучка архітектура бази знань.	Відсутність сформованої клієнтської бази на старті. Обмежений маркетинговий бюджет. Залежність від сторонніх джерел історичних даних. Необхідність постійного оновлення правил під нові типи ринків.

Продовження таблиці 4.5

Можливості (Opportunities)	Загрози (Threats)
Зростання ринку криптовалют та інтересу до автоматичної торгівлі. Можливість ліцензування технології великим брокерам. Розширення функціоналу для підтримки фондового та валютного ринків. Створення екосистеми додатків.	Поява аналогічного функціоналу у великих платформ-конкурентів. Зміна регуляторної політики щодо алгоритмічної торгівлі. Ризик кібератак та витоку інтелектуальної власності користувачів.

За результатами проведеного аналізу ринкових можливостей можна зробити висновок про високу інвестиційну привабливість проекту. Поєднання зростаючого попиту на інструменти автоматизації, наявність чітко визначеної цільової аудиторії та унікальна ціннісна пропозиція створюють сприятливі умови для успішного виходу на ринок. Стратегія розвитку повинна фокусуватися на швидкому зайнятті ніші інструментів для професійної діагностики, використовуючи технологічну перевагу як основний драйвер конкурентоспроможності.

4.5 Розроблення ринкової стратегії стартап-проекту

Формування ринкової стратегії інноваційного продукту є комплексним процесом, який визначає довгостроковий вектор розвитку компанії та способи досягнення конкурентних переваг в обраному ринковому сегменті. Для розробленої системи автоматизованої діагностики торгових моделей обрано стратегію концентрованого зростання, яка передбачає фокусування ресурсів на вузькій спеціалізованій ніші інструментів для аудиту та валідації алгоритмів. Цей вибір обумовлений високою технологічною складністю

продукту та наявністю чітко окресленої цільової аудиторії, яка має специфічні потреби у підвищенні надійності торгових операцій. Стратегія ринкового позиціонування базується на диференціації продукту за критерієм інтелектуальної складової. На відміну від масових платформ для бек-тестування, які конкурують переважно ціною та швидкістю обробки даних, даний стартап позиціонується як експертна система преміум-класу, що надає унікальну аналітичну цінність через інтерпретацію причинно-наслідкових зв'язків помилок.

Ключовим елементом ринкової стратегії є вибір моделі монетизації, яка повинна забезпечити стабільний грошовий потік та водночас сприяти швидкому залученню користувачів. Для даного проекту обрано гібридну бізнес-модель надання програмного забезпечення як послуги, що поєднує безкоштовний базовий доступ для ознайомлення з функціоналом та багаторівневу систему платних підписок для професійного використання. Така стратегія дозволяє знизити бар'єр входу для нових клієнтів, формуючи лояльну спільноту користувачів, частина з яких згодом конвертується у платних підписників. Цінова політика розроблена з урахуванням платоспроможності різних сегментів цільової аудиторії та цінності, яку продукт створює для користувача, а саме економії часу розробників та запобігання фінансовим втратам. Детальну структуру тарифних планів та їх функціональне наповнення наведено в таблиці 4.6.

Таблиця 4.6. Стратегія ціноутворення та тарифні плани стартап-проекту

Назва тарифного плану	Цільова аудиторія	Вартість	Ключовий функціонал та обмеження
Базовий (Start)	Студенти, початківці, дослідники	Безкоштовно	Доступ до базових детекторів даних, ліміт 5 перевірок на місяць, стандартна підтримка

Продовження таблиці 4.6

Професійний (Pro)	Приватні алготрейдери, фрілансери	49 доларів на місяць	Повний доступ до Бази Знань, необмежена кількість перевірок, доступ до API, експорт звітів у PDF
Корпоративний (Enterprise)	Хедж-фонди, проп-трейдингові фірми	Договірна (від 500 доларів)	Розгортання на серверах клієнта, персональна адаптація правил, SLA підтримка 24/7, аудит безпеки

Стратегія розповсюдження продукту базується на використанні прямих каналів продажів через власний вебсайт та API-інтерфейс, що дозволяє зберегти повний контроль над взаємодією з клієнтом та маржинальністю бізнесу. Для сегмента індивідуальних користувачів основним каналом залучення виступає контент-маркетинг та пошукова оптимізація, спрямована на вирішення конкретних технічних проблем трейдерів. Для корпоративного сегмента передбачено використання прямих продажів та участь у профільних фінтех-конференціях. Важливим стратегічним кроком є створення партнерської екосистеми, в рамках якої система діагностики може пропонуватися як додатковий модуль до існуючих торгових терміналів або брокерських платформ. Це дозволить використовувати клієнтську базу партнерів для швидкого масштабування без значних витрат на маркетинг.

Для забезпечення стійкості бізнесу в умовах конкурентної боротьби розроблено стратегію захисту ринкової частки. Вона базується на створенні високих витрат перемикання для клієнтів шляхом глибокої інтеграції системи у їхні робочі процеси та накопичення історії аналізу моделей. Крім того, постійне оновлення та розширення Бази Знань створює ефект мережевої цінності, де кожна нова знайдена ринкова аномалія покращує якість

діагностики для всіх користувачів платформи. Для систематизації обраних стратегічних напрямків та інструментів їх реалізації було розроблено карту стратегічних ініціатив, представлену в таблиці 4.7.

Таблиця 4.7. Карта стратегічних ініціатив ринкової стратегії

Стратегічний напрямок	Інструменти реалізації	Очікуваний ефект
Проникнення на ринок	Агресивний інтернет-маркетинг, безкоштовний тріал-період, публікація наукових статей	Формування впізнаваності бренду та первинної бази користувачів
Розвиток продукту	Додавання підтримки нових класів активів, інтеграція з популярними бібліотеками Python	Підвищення цінності продукту та утримання існуючих клієнтів
Ринкова експансія	Локалізація інтерфейсу, адаптація під вимоги регуляторів США та ЄС	Вихід на нові географічні ринки з високою платоспроможністю
Диверсифікація	Створення освітніх курсів на базі платформи, консалтингові послуги	Отримання додаткових джерел доходу та експертний статус

Реалізація розробленої ринкової стратегії дозволить стартап-проекту зайняти стійку позицію в ніші аналітичних інструментів для алгоритмічної торгівлі. Поєднання гнучкої цінової політики, орієнтації на технологічну перевагу та багаторівневої системи дистрибуції створює передумови для

досягнення запланованих фінансових показників та забезпечення повернення інвестицій.

4.6 Розроблення маркетингової програми стартап-проекту

Розробка ефективної маркетингової програми є завершальним етапом планування виведення інноваційного продукту на ринок, який спрямований на практичну реалізацію обраної ринкової стратегії через систему конкретних оперативних заходів. Специфіка продукту, яким є складна аналітична система для діагностики алгоритмічних моделей, диктує необхідність відмови від традиційних методів агресивної реклами на користь концепції вхідного маркетингу та експертного позиціонування. Маркетингова програма стартап-проекту базується на принципі побудови довіри через демонстрацію компетентності, оскільки цільова аудиторія, що складається з квант-аналітиків та розробників торгових роботів, характеризується високим рівнем технічної грамотності та скептицизмом до неперевірених рішень. Головним завданням маркетингових комунікацій є донесення до споживача ключової цінності продукту, яка полягає не в обіцянці надприбутків, а в гарантуванні технічної надійності та математичної коректності торгових стратегій.

Основою маркетингової програми є контент-стратегія, спрямована на створення глибоких аналітичних матеріалів, що розкривають типові помилки в алгоритмічній торгівлі та способи їх вирішення за допомогою розробленої системи. Передбачається публікація серії технічних статей на профільних ресурсах та у спеціалізованих спільнотах розробників, де на реальних прикладах демонструється, як використання доларових барів або фільтрація за впевненістю покращують метрики стратегій. Такий підхід дозволяє органічно залучати цільовий трафік через пошукові системи за низькочастотними професійними запитами. Важливим елементом програми є вірусний

маркетинг, який реалізується через надання безкоштовного доступу до обмеженої версії діагностичного звіту. Користувачі, отримавши цінну інформацію про слабкі місця своєї моделі, стають природними промоутерами сервісу, ділячись результатами аудиту в соціальних мережах та на форумах.

Комунікаційна політика передбачає використання омніканального підходу для охоплення різних сегментів аудиторії. Для індивідуальних трейдерів основним каналом комунікації є соціальні мережі та відеохостинги, де розміщуються навчальні матеріали та відеоінструкції з використання платформи. Для корпоративного сегмента використовується прямий маркетинг через професійну соціальну мережу LinkedIn та участь у галузевих конференціях з фінансових технологій. Особлива увага приділяється побудові ком'юніті навколо продукту шляхом створення закритого клубу користувачів, де учасники можуть обмінюватися досвідом та отримувати консультації від розробників системи. Це дозволяє не лише утримувати існуючих клієнтів, а й отримувати швидкий зворотний зв'язок для вдосконалення продукту. Комплекс основних маркетингових заходів, канали їх реалізації та очікувані результати систематизовано в таблиці 4.8.

Таблиця 4.8. Програма маркетингових заходів для просування стартап-проекту

Маркетинговий інструмент	Канали комунікації	Зміст заходу	Очікуваний результат
Контент-маркетинг	Medium, Habr, GitHub, власний блог	Публікація кейс-стаді, оглядів наукових статей, прикладів коду з використанням API системи	Формування іміджу експерта, залучення органічного пошукового трафіку, навчання користувачів

Продовження таблиці 4.8

Пошукова оптимізація	Google, Bing, YouTube	Оптимізація сайту під технічні запити, створення відеоуроків з налаштування моделей	Зростання позицій у видачі, зниження вартості залучення клієнта
Соціальний маркетинг	Reddit, Twitter, Telegram, Discord	Участь у дискусіях, допомога новачкам, анонси оновлень, проведення AMA-сесій	Побудова лояльної спільноти, отримання зворотного зв'язку, вірусне розповсюдження
E-mail маркетинг	Розсилка по базі зареєстрованих користувачів	Тригерні листи про завершення аналізу, дайджести новин ринку, персональні пропозиції	Повернення користувачів на платформу, підвищення конверсії у платну підписку

Важливим компонентом маркетингової програми є система стимулювання збуту, яка спрямована на подолання бар'єру першої покупки. Для цього впроваджується багаторівнева система лояльності, що передбачає знижки при оплаті річної підписки та бонуси за залучення нових користувачів. Для корпоративних клієнтів розроблено програму пілотного тестування, яка дозволяє безкоштовно використовувати повний функціонал системи протягом обмеженого періоду для оцінки її ефективності в умовах реальної

інфраструктури замовника. Ефективність маркетингової програми оцінюється за системою ключових показників, що включають вартість залучення одного клієнта, довічну цінність клієнта та коефіцієнт відтоку. Постійний моніторинг цих метрик дозволяє оперативно коригувати маркетинговий бюджет та перерозподіляти ресурси на найбільш ефективні канали просування, забезпечуючи стабільне зростання користувацької бази та фінансових показників проекту.

4.7 Висновки до четвертого розділу

У ході виконання четвертого розділу магістерської дисертації було проведено комплексне обґрунтування доцільності комерціалізації розробленої системи автоматизованої діагностики прогнозних моделей у форматі інноваційного стартап-проекту. Детальний аналіз ринкового середовища дозволив ідентифікувати перспективну нішу інструментів для глибокого семантичного аудиту алгоритмічних стратегій, яка наразі залишається неохопленою традиційними платформами для бек-тестування та універсальними рішеннями машинного навчання. Проведений технологічний аудит підтвердив технічну реалістичність впровадження системи з використанням обраного стеку на базі мови програмування Python та хмарної мікросервісної архітектури, що гарантує високу продуктивність обробки фінансових даних та здатність платформи до автоматичного масштабування.

Розроблена бізнес-модель, що базується на принципах надання програмного забезпечення як послуги з гібридною системою монетизації, дозволяє ефективно збалансувати процеси залучення користувачів та генерації прибутку, охоплюючи як масовий сегмент індивідуальних дослідників, так і високомаржинальний корпоративний сектор. Запропонована маркетингова стратегія, орієнтована на експертне позиціонування, контент-маркетинг та

побудову професійного ком'юніті, повністю відповідає специфіці складного високотехнологічного продукту та дозволяє оптимізувати витрати на просування. Сформований план-графік реалізації проекту та карта стратегічних ініціатив демонструють чітку траєкторію розвитку від створення мінімально життєздатного продукту до експансії на міжнародні ринки, що у сукупності з унікальною ціннісною пропозицією підтверджує високу інвестиційну привабливість та комерційний потенціал розробленої системи діагностики.

ВИСНОВКИ

У магістерській дисертації вирішено актуальне науково-прикладне завдання підвищення ефективності та робастності систем алгоритмічної торгівлі в умовах високої невизначеності та нестаціонарності фінансових ринків. Проведене дослідження дозволило досягти поставленої мети шляхом розробки комплексної методології та програмного інструментарію для автоматизованої діагностики прогнозних моделей, що забезпечило перехід від інтуїтивного пошуку оптимальних параметрів до систематичного інженерного процесу вдосконалення торгових стратегій. У ході роботи було виконано теоретичний аналіз, який довів обмеженість класичних економетричних підходів та виявив фундаментальні проблеми застосування методів машинного навчання, зокрема їхню вразливість до змін ринкових режимів та схильність до перенавчання на зашумлених даних.

Найбільш важливим науковим результатом роботи є розробка концепції та програмна реалізація системи діагностики на основі формалізованої бази знань. Запропоноване рішення, на відміну від існуючих аналогів, фокусується не на максимізації метрик на історичних даних, а на виявленні семантичних причин помилок алгоритмів, таких як мультиколінеарність вхідних ознак, ігнорування впевненості прогнозу та асиметрія результатів у різних фазах ринку. Обґрунтовано та впроваджено удосконалені методи підготовки даних, зокрема використання барів грошового обсягу та дробового диференціювання, що дозволило відновити нормальність статистичних розподілів та зберегти інформаційну пам'ять часових рядів.

Достовірність отриманих результатів підтверджено серією експериментів на історичних даних криптовалютної пари ETH/USD із використанням методу очищеної перехресної перевірки, що виключає можливість підглядання у майбутнє. Кількісний аналіз продемонстрував високу ефективність розробленої методології вдосконалення моделей. Після

застосування автоматично згенерованих рекомендацій системи метрика F1-Score для моделі Логістичної регресії зросла на шістдесят чотири цілих і одну десяту відсотка, досягнувши рівня 0,676, що стало найкращим результатом серед усіх протестованих алгоритмів. Це експериментально доводить, що правильний інжиніринг ознак та усунення структурних помилок стратегії мають критично більший вплив на кінцевий фінансовий результат, ніж вибір складної архітектури моделі.

Практичне значення роботи полягає у створенні готового до використання програмного продукту, який дозволяє інвестиційним фондам та індивідуальним розробникам значно скоротити час на аудит торгових роботів та мінімізувати фінансові ризики. На основі отриманих результатів розроблено повноцінний стартап-проект з чіткою бізнес-моделлю та стратегією виходу на ринок, що підтверджує комерційний потенціал наукової розробки. За результатами дослідження рекомендовано використовувати запропоновану систему автоматизованої діагностики як обов'язковий етап валідації будь-яких алгоритмічних стратегій, а також впроваджувати розроблені методи фільтрації сигналів за рівнем впевненості для підвищення стабільності торгових операцій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Wang Z., Ventre C. A Financial Time Series Denoiser Based on Diffusion Model. Data Science in Finance and Economics. 2021. Vol. 1, Issue 4, P. 345-361.
URL: https://www.aimspress.com/article/doi/10.3934/DSFE.2021019?viewType=HTML (дата звернення: 10.11.2025).

2. Shuozhe Li, Zachery B. Shulwolf, Risto Miikkulainen. Deep Learning for Financial Time Series Forecasting. arXiv, 2025. 8 p. URL: https://arxiv.org/html/2502.09625v1 (дата звернення: 10.11.2025).

3. Kaur T. Machine Learning Algorithms for Trading: Predictive Modeling and Portfolio Optimization (Part 2). Medium. 2024. URL: https://medium.com/@tkaur77t/machine-learning-algorithms-for-trading-predictive-modeling-and-portfolio-optimization-part-2-7edb60da83e8 (дата звернення: 15.11.2025).

4. Al-Nefaie A. H., Aldhyani T. H. Using Machine Learning on Macroeconomic, Technical, and Sentiment Indicators for Stock Market Forecasting. Information. 2025. Vol. 16, No. 7. 584, 41 p. URL: https://www.mdpi.com/2078-2489/16/7/584 (дата звернення: 15.11.2025).

5. Gwokkwan Sun, Shuhan Deng. Financial Time Series Forecasting: A Comparison Between Traditional Methods and AI-Driven Techniques. Journal of Computer Signal and System Research, 2024. Vol. 2, No. 2, P. 86-93. URL: https://www.researchgate.net/publication/390275409_Financial_Time_Series_Forecasting_A_Comparison_Between_Traditional_Methods_and_AI-Driven_Techniques (дата звернення: 15.11.2025).

6. Grachev O. Y., Application of Time Series Models (ARIMA, GARCH, and ARMA-GARCH) for Stock Market Forecasting. Honors Capstones. 2017, 177. 51 p. URL:

https://huskiecommons.lib.niu.edu/cgi/viewcontent.cgi?article=1176&context=studentengagement-honorscapstones (дата звернення: 15.11.2025).

7. Orozco-Castaneda J. M., Alzate-Vargas S., Bedoya-Valencia D. Evaluating Volatility Using an ANFIS Model for Financial Time Series Prediction. Risks, 2024. Vol. 12, No. 10. 156. 15p. URL: https://www.mdpi.com/2227-9091/12/10/156 (дата звернення: 15.11.2025).

8. Ruiheng Chen. ARIMA vs. Machine Learning in Portfolio Return Forecasting: A Comparative Study Integrating GARCH-Based Volatility Estimation and Value-at-Risk Applications. 2nd International Conference on Innovations in Applied Mathematics, Physics, and Astronomy. 2025. P. 419-428. URL: https://www.scitepress.org/Papers/2025/138270/138270.pdf (дата звернення: 15.11.2025).

9. Asokan Mowniesh. A study of forecasts in Financial Time Series using Machine Learning methods. Master's thesis, Linkoping University, Department of Computer and Information Science, 2022. 49 p. URL: https://www.diva-portal.org/smash/get/diva2:1671481/FULLTEXT01.pdf (дата звернення: 15.11.2025).

10. Hochreiter S., Schmidhuber J. Long short-term memory. Neural computation, 1997, Vol. 9(8), P. 1735-1780. URL: https://doi.org/10.1162/neco.1997.9.8.1735

11. Ab A. D. Mastering Forecasting: Unveiling the Power of VAR Modeling for Dynamic Time Series Prediction. Medium. 2023. URL: https://medium.com/@Alidotab/mastering-forecasting-unveiling-the-power-of-var-modeling-for-dynamic-time-series-prediction-1b87a7d63b4b (дата звернення: 15.11.2025).

12. Agusti M., Vidal-Quadras Costa I., Altmeyer P. Deep vector autoregression for macroeconomic data. BIS IFC Bulletin, 2022. No. 59. 45 p. URL: https://www.bis.org/ifc/publ/ifcb59_39.pdf (дата звернення: 15.11.2025).

13. Sanjay Agal, Niyati Dhirubhai Odedra. Impact of Predictive Analytics on Algorithmic Trading Enhancing Strategy Performance and Profitability. Degres,

2024. Vol. 10. No 2. P. 184-198. URL: https://www.researchgate.net/publication/389166973_Impact_of_Predictive_Analytics_on_Algorithmic_Trading_Enhancing_Strategy_Performance_and_Profitability (дата звернення: 15.11.2025).

14. Wohlwend B. Decision Tree, Random Forest, and XGBoost: An Exploration into the Heart of Machine Learning. Medium. 2020. URL: https://medium.com/@brandon93.w/decision-tree-random-forest-and-xgboost-an-exploration-into-the-heart-of-machine-learning-90dc212f4948 (дата звернення: 15.11.2025).

15. Afsaneh Mollasalehi, Armin Farhadi. Advanced Machine Learning Techniques. arXiv, 2025. 9 p. URL: https://arxiv.org/abs/2509.24059 (дата звернення: 15.11.2025).

16. Kabir R., Bhadra D., Ridoy M., Milanova M. Machine Learning and Knowledge Extraction. MDPI. 2025. Vol. 7, No. 1. 22 p. URL: https://www.mdpi.com/2413-4155/7/1/7 (дата звернення: 15.11.2025).

17. Cheng Zhang, Nilam Nur Amir Sjarif, Roslina Ibrahim, Deep learning models for price forecasting of financial time series: A review of recent advancements: 2020-2022. arXiv, 2023. 37 p. URL: https://arxiv.org/pdf/2305.04811 (дата звернення: 15.11.2025).

18. Jue Xiao, Tingting Dengj, Shuochen Bi. Comparative Analysis of LSTM, GRU, and Transformer Models for Stock Price Prediction. arXiv, 2024. 6 p. URL: https://arxiv.org/abs/2411.05790 (дата звернення: 15.11.2025).

19. Bergmann D., Stryker C. What is the attention mechanism? IBM. URL: https://www.ibm.com/think/topics/attention-mechanism (дата звернення: 20.11.2025).

20. Pei Y., Cartlidge J., Mandal A., Gold D., Marcilio E., Mazzon R.Cross-Modal Temporal Fusion for Financial Market Forecasting. arXiv, 2025. 8 p. URL: https://arxiv.org/abs/2504.13522 (дата звернення: 20.11.2025).

21. Arash Peik, Mohammad Ali Zare Chahooki, Amin Milani Fard, Mehdi A. Sarram. Adaptive Temporal Fusion Transformers for Cryptocurrency Price

Prediction. arXiv, 2025. 17 p. URL: https://arxiv.org/html/2509.10542v1 (дата звернення: 20.11.2025).

22. Hovakimyan G., Bravo J. M. Evolving Strategies in Machine Learning: A Systematic Review of Concept Drift Detection. Information, 2024. Vol. 15, No. 12. 24 p. URL: https://www.mdpi.com/2078-2489/15/12/786 (дата звернення: 20.11.2025).

23. Qiuyan Xiang, Lingling Zi, Xin Cong, Yan Wang. Concept Drift Adaptation Methods under the Deep Learning Framework: A Literature Review. Applied Sciences, 2023. Vol. 13, No. 11. 24 p. URL: https://www.mdpi.com/2076-3417/13/11/6515 (дата звернення: 20.11.2025).

24. Wang M., Lin Y., Mikhelson I. Regime-Switching Factor Investing with Hidden Markov Models. Risk and Financial Management, 2020. Vol. 13, No. 12. 15 p. URL: https://www.mdpi.com/1911-8074/13/12/311 (дата звернення: 20.11.2025).

25. Market Regime Detection: From Hidden Markov Models to Wasserstein Clustering. Hikmah Technologies. 2025 URL: https://publication.hikmahtechnologies.com/market-regime-detection-from-hidden-markov-models-to-wasserstein-clustering-6ba0a09559dc?gi=71577be4db50 (дата звернення: 20.11.2025).

26. Purged cross-validation. Wikipedia. URL: https://en.wikipedia.org/wiki/Purged_cross-validation (дата звернення: 20.11.2025).

27. Gort B., Yang B. The Combinatorial Purged Cross-Validation Method. Towards AI. 2023. URL: https://towardsai.net/p/l/the-combinatorial-purged-cross-validation-method (дата звернення: 20.11.2025).

28. Young B. Time Series Analysis in Algo Trading. LuxAlgo. 2025 URL: https://www.luxalgo.com/blog/time-series-analysis-in-algo-trading/ (дата звернення: 20.11.2025).

29. Jiang J., Yang C., Wang X., Li B. Why Regression? Binary Encoding Classification Brings Confidence to Stock Market Index Price Prediction. arXiv,

2025. 14 p. URL: https://arxiv.org/html/2506.03153v1 (дата звернення: 20.11.2025).

30. Chakravorty A., Elsayed N. A Comparative Study of Machine Learning Algorithms for Stock Price Prediction Using Insider Trading Data. arXiv, 2025. 5 p. URL: https://arxiv.org/html/2502.08728v1 (дата звернення: 20.11.2025).

31. Junzhe Jiang, Chang Yang, Xinrun Wang, Bo Li. Why Regression? Binary Encoding Classification Brings Confidence to Stock Market Index Price Prediction. arXiv, 2025. 14 p. URL: https://arxiv.org/abs/2506.03153 (дата звернення: 20.11.2025).

32. Meyer M., Gonah M. Machine Learning Trading Essentials: Financial Data Structures. Hudson & Thames. URL: https://hudsonthames.org/machine-learning-trading-essentials-part-1-financial-data-structures/ (дата звернення: 20.11.2025).

33. Tails Azimuth. Fractional Differentiation and Memory. RiskLab, 2024. URL: https://www.risklab.ai/research/financial-data-science/fractional_differentiation (дата звернення: 20.11.2025).

34. Rama Cont, Mihai Cucuringu, Chao Zhang. Cross-impact of order flow imbalance in equity markets. Quantitative Finance, 2023. Vol. 23, No. 10. P 1373-1393. URL: https://www.tandfonline.com/doi/full/10.1080/14697688.2023.2236159#d1e199 (дата звернення: 20.11.2025).

35. Data Labelling. MlFinPy Documentation. URL: https://mlfinpy.readthedocs.io/en/latest/Labelling.html (дата звернення: 20.11.2025).

36. Bailey D. H., Lopez de Prado M. The Deflated Sharpe Ratio: Correcting for selection bias, backtest overfitting and non-normality. Journal of Portfolio Management. 2014. 22 p. URL: https://pdfs.semanticscholar.org/c215/d0a2064ce1a3565d276475abc84305418f0f.pdf (дата звернення: 20.11.2025).

37. Or Jacobi. Concept Drift: 8 Detection Methods. Coralogix, 2021. URL: https://coralogix.com/ai-blog/concept-drift-8-detection-methods/ (дата звернення: 20.11.2025).

38. Joao Bento. TimeSHAP: Explaining Recurrent Models through Sequence Perturbations. Feedzai Tech, 2023. URL: https://medium.com/feedzaitech/timeshap-explaining-recurrent-models-through-sequence-perturbations-41f2324bfe5f (дата звернення: 20.11.2025).

39. Peng Liu. Seeking Better Sharpe Ratios via Bayesian optimization. Journal of Portfolio Management, 2023, 49(7). 14 p. URL: https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?params=/context/lkcsb_research/article/8471/&path_info=seeking_better_sharpe.pdf (дата звернення: 20.11.2025).

40. Ярінко Б. Б. Розробка та дослідження прогнозних моделей у системах алгоритмічної торгівлі. Збірник доповідей IV науково-практичної конференції «Системні науки та інформатика», 25–29 листопада 2025 року, Київ. К., НН ІПСА КПІ ім. Ігоря Сікорського, 2025, 6 с.

ДОДАТОК А

Лістинг системи

system/analyzer.py:

```
import json
import pandas as pd
from . import detectors


class Analyzer:
    def __init__(self, knowledge_base_path: str):
        """
        Ініціалізує аналізатор, завантажуючи базу знань.
        :param knowledge_base_path: Шлях до JSON-файлу з базою знань.
        """
        try:
            with open(knowledge_base_path, "r", encoding="utf-8") as f:
                self.kb = json.load(f)
            print("Базу знань успішно завантажено.")
        except FileNotFoundError:
            print(
                f"Помилка: файл бази знань не знайдено за шляхом
'{knowledge_base_path}'"
            )
            self.kb = {}

        self.results = []
        # Словник для динамічного виклику функцій
        self.detector_functions = {
            "detect_multicollinearity": detectors.detect_multicollinearity,
            "detect_class_imbalance": detectors.detect_class_imbalance,
            "detect_overfitting": detectors.detect_overfitting,
            "detect_error_autocorrelation":
detectors.detect_error_autocorrelation,
            "detect_volatility_instability":
detectors.detect_volatility_instability,
            "detect_concept_drift": detectors.detect_concept_drift,
            # Нові детектори
            "detect_lookahead_bias": detectors.detect_lookahead_bias,
            "detect_indicator_dependency":
detectors.detect_indicator_dependency,
            "detect_excessive_trading_frequency":
detectors.detect_excessive_trading_frequency,
            "detect_flat_market_instability":
detectors.detect_flat_market_instability,
            # Просунуті детектори
            "analyze_prediction_confidence":
detectors.analyze_prediction_confidence,
            "detect_data_snooping_risk": detectors.detect_data_snooping_risk,
            "detect_excessive_drawdown": detectors.detect_excessive_drawdown,
            "find_systematic_error_patterns":
detectors.find_systematic_error_patterns,
        }
```

```
    def run_analysis(
        self,
        X_df,
        y_true,
        y_pred,
        model=None,
        y_prob=None,
        train_metrics=None,
        validation_metrics=None,
    ):
        """
        Запускає повний цикл аналізу моделі на основі наданих даних.
        :param X_df: DataFrame з ознаками (валідаційна/тестова вибірка).
        :param y_true: Series зі справжніми значеннями.
        :param y_pred: Series з прогнозами моделі.
        :param model: Навчена модель (напр., XGBoost) для аналізу feature
importances.
        :param y_prob: Series з імовірностями прогнозів.
        :param train_metrics: Словник з метриками на тренувальній вибірці
(для overfitting).
        :param validation_metrics: Словник з метриками на валідаційній
вибірці (для overfitting).
        """
        self.results = []  # Очищуємо результати попереднього аналізу
        print("\n--- Запуск аналізу системи ---")

        if not self.kb:
            print("Аналіз неможливий: база знань порожня.")
            return

        for category, rules in self.kb.items():
            print(f"\n[Клас] Перевірка проблем типу: '{category}'")
            for rule in rules:
                detector_func_name = rule["detector"].get("function")

                if detector_func_name in self.detector_functions:
                    detector_func =
self.detector_functions[detector_func_name]
                    detector_params = rule["detector"].get("parameters", {})

                    # Формуємо аргументи для функції-детектора
                    kwargs = {
                        "X_df": X_df,
                        "y_true": y_true,
                        "y_pred": y_pred,
                        "model": model,
                        "y_prob": y_prob,
                        "train_metrics": train_metrics,
                        "validation_metrics": validation_metrics,
                        detector_params,
                    }

                    detection_result = detector_func(kwargs)

                    if detection_result and detection_result.get("detected"):
                        print(f"  [!] Виявлено проблему: '{rule['name']}'")
```

```
                        print(f"      Деталі: {detection_result['details']}")
                        self.results.append(
                            {
                                "rule_id": rule["id"],
                                "name": rule["name"],
                                "corrector": rule["corrector"],
                            }
                        )
                    # else:
                    #     print(
                    #         f"  [✓] Проблем за патерном '{rule['name']}'
не виявлено."
                    #     )
                else:
                    print(
                        f"  [!] Попередження: детектор '{detector_func_name}'
не реалізовано або відсутній у system/detectors.py."
                    )

        return self.generate_report()

    def generate_report(self):
        """
        Форматує та повертає фінальний звіт з рекомендаціями.
        """
        if not self.results:
            report = "--- Звіт системи ---\n\nПроблем не виявлено. Модель
пройшла базову діагностику."
            print(report)
            return report

        report = "--- Звіт та рекомендації системи ---\n"
        for result in self.results:
            report += f"\n----------------------------------------\n"
            report += f"ПРОБЛЕМА: {result['name']} (ID:
{result['rule_id']})\n"
            report += "----------------------------------------\n"
            report += "РЕКОМЕНДОВАНІ ДІЇ:\n"

            recommendations = sorted(
                result["corrector"]["recommendations"], key=lambda x:
x["priority"]
            )
            for rec in recommendations:
                report += f"  [{rec['priority']}] {rec['action']}:\n"
                report += f"      {rec['description']}\n"
                if "implementation" in rec:
                    impl = rec["implementation"]
                    if isinstance(impl, dict):
                        impl_str = ", ".join([f"{k}: {v}" for k, v in
impl.items()])
                        report += f"      -> Деталі реалізації: {impl_str}\n"
                    else:
                        report += f"      -> Деталі реалізації: {impl}\n"

        print(report)
```

```
        return report
```

system/detectors.py:

```
import numpy as np
import pandas as pd
from sklearn.metrics import f1_score
from statsmodels.stats.stattools import durbin_watson  # type: ignore

# --- Detectors for Data Issues ---


def detect_multicollinearity(
    X_df: pd.DataFrame, correlation_threshold: float = 0.95, kwargs
) -> dict:
    """
    Виявляє висококорельовані ознаки (мультиколінеарність).
    KB-DATA-FEAT
    """
    corr_matrix = X_df.corr().abs()
    upper_tri = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
k=1).astype(bool))

    highly_correlated_pairs = []
    for column in upper_tri.columns:
        correlated_features = upper_tri.index[
            upper_tri[column] > correlation_threshold
        ].tolist()
        if correlated_features:
            for feature in correlated_features:
                highly_correlated_pairs.append(tuple(sorted((column,
feature))))

    unique_pairs = sorted(list(set(highly_correlated_pairs)))
    if unique_pairs:
        return {
            "detected": True,
            "details": f"Знайдено {len(unique_pairs)} пар(и) ознак з
кореляцією > {correlation_threshold}",
            "payload": {"correlated_pairs": unique_pairs},
        }
    return {"detected": False}


def detect_class_imbalance(
    y_true: pd.Series, imbalance_threshold: float = 0.65, kwargs
) -> dict:
    """
    Виявляє дисбаланс класів у цільовій змінній.
    KB-DATA-IMB
    """
    class_ratio = y_true.value_counts(normalize=True).max()
    if class_ratio > imbalance_threshold:
        return {
            "detected": True,
```

```
            "details": f"Домінуючий клас складає {class_ratio:.2%}, що
перевищує поріг {imbalance_threshold:.2%}",
            "payload": {"class_ratio": class_ratio},
        }
    return {"detected": False}


# --- Detectors for Model Issues ---


def detect_overfitting(
    train_metrics: dict,
    validation_metrics: dict,
    performance_gap_threshold: float = 0.15,
    kwargs,
) -> dict:
    """
    Виявляє перенавчання, порівнюючи метрики на тренувальній та валідаційній
вибірках.
    KB-MODEL-OVR
    """
    if not train_metrics or not validation_metrics:
        return {
            "detected": False,
            "details": "Метрики для тренувальної або валідаційної вибірки не
надано.",
        }

    metric = "f1_score"  # Пріоритетна метрика для перевірки
    train_score = train_metrics.get(metric)
    val_score = validation_metrics.get(metric)

    if train_score is None or val_score is None:
        return {"detected": False, "details": f"Метрика '{metric}'
відсутня."}

    gap = train_score - val_score
    if gap > performance_gap_threshold:
        return {
            "detected": True,
            "details": f"Розрив у {metric.replace('_', ' ')} між train та
validation ({gap:.2%}) перевищує поріг ({performance_gap_threshold:.2%})",
            "payload": {
                "performance_gap": gap,
                "train_score": train_score,
                "validation_score": val_score,
            },
        }
    return {"detected": False}


def detect_error_autocorrelation(
    y_true: pd.Series,
    y_pred: pd.Series,
    durbin_watson_thresholds: list = [1.5, 2.5],
    kwargs,
```

```
) -> dict:
    """
    Виявляє автокореляцію помилок за допомогою тесту Дарбіна-Вотсона.
    KB-MODEL-AUTOCORR
    """
    # Працює для регресії, для класифікації (0/1) це менш інформативно, але
можливо
    residuals = y_true - y_pred
    if residuals.var() < 1e-9:  # Якщо помилок немає, автокореляції теж
        return {"detected": False}

    dw_statistic = durbin_watson(residuals)

    if not (durbin_watson_thresholds[0] < dw_statistic <
durbin_watson_thresholds[1]):
        autocorr_type = "позитивна" if dw_statistic < 2 else "негативна"
        return {
            "detected": True,
            "details": f"Статистика Дарбіна-Вотсона ({dw_statistic:.2f}) поза
межами безпечного діапазону ({durbin_watson_thresholds}). Виявлено
{autocorr_type} автокореляцію помилок.",
            "payload": {"durbin_watson_statistic": dw_statistic},
        }
    return {"detected": False}

# --- Detectors for Evaluation Issues ---

def detect_volatility_instability(
    X_df: pd.DataFrame,
    y_true: pd.Series,
    y_pred: pd.Series,
    volatility_feature_pattern: str = "ATR",
    performance_drop_threshold: float = 0.05,
    kwargs,
) -> dict:
    """
    Перевіряє, чи падає якість моделі на волатильному ринку.
    KB-EVAL-VOL
    """
    vol_col = next(
        (
            col
            for col in X_df.columns
            if volatility_feature_pattern.upper() in col.upper()
        ),
        None,
    )
    if not vol_col:
        return {
            "detected": False,
            "details": f"Ознака волатильності з патерном
'{volatility_feature_pattern}' не знайдена.",
        }

    analysis_df = pd.DataFrame(
        {"y_true": y_true, "y_pred": y_pred, "volatility": X_df[vol_col]}
```

```
    )

    # Визначаємо періоди низької та високої волатильності
    vol_threshold = analysis_df["volatility"].quantile(0.75)  # Топ 25%
найволатильніших
    low_vol_df = analysis_df[analysis_df["volatility"] < vol_threshold]
    high_vol_df = analysis_df[analysis_df["volatility"] >= vol_threshold]

    if len(low_vol_df) == 0 or len(high_vol_df) == 0:
        return {
            "detected": False,
            "details": "Недостатньо даних для порівняння волатильних
періодів.",
        }

    f1_low_vol = f1_score(low_vol_df["y_true"], low_vol_df["y_pred"])
    f1_high_vol = f1_score(high_vol_df["y_true"], high_vol_df["y_pred"])

    performance_drop = f1_low_vol - f1_high_vol

    if performance_drop > performance_drop_threshold:
        return {
            "detected": True,
            "details": f"F1-Score падає на {performance_drop:.2%} у періоди
високої волатильності (з {f1_low_vol:.2f} до {f1_high_vol:.2f}), що перевищує
поріг {performance_drop_threshold:.2%}.",
            "payload": {
                "f1_low_vol": f1_low_vol,
                "f1_high_vol": f1_high_vol,
                "performance_drop": performance_drop,
            },
        }
    return {"detected": False}

def detect_concept_drift(
    y_true: pd.Series, y_pred: pd.Series, time_window: str = "90D", kwargs
) -> dict:
    """
    Виявляє дрейф концепту, аналізуючи падіння якості моделі з часом.
    KB-EVAL-DRIFT
    """
    if not isinstance(y_true.index, pd.DatetimeIndex):
        return {
            "detected": False,
            "details": "Індекс даних не є DatetimeIndex, аналіз неможливий.",
        }

    results_df = pd.DataFrame({"y_true": y_true, "y_pred": y_pred})
    # Розраховуємо F1-Score по ковзному вікну
    f1_scores = (
        results_df["y_true"]
        .rolling(window=time_window)
        .apply(
            lambda x: f1_score(
                results_df.loc[x.index, "y_true"], results_df.loc[x.index,
"y_pred"]
```

```
            ),
            raw=False,
        )
    )
    f1_scores = f1_scores.dropna()

    if len(f1_scores) < 2:
        return {"detected": False, "details": "Недостатньо даних для аналізу
тренду."}

    # Проста перевірка тренду: чи є останні 30% періоду значно гіршими за
перші 30%
    split_point1 = int(len(f1_scores) * 0.3)
    split_point2 = int(len(f1_scores) * 0.7)

    mean_first_part = f1_scores.iloc[:split_point1].mean()
    mean_last_part = f1_scores.iloc[split_point2:].mean()

    if mean_first_part - mean_last_part > 0.07:  # Якщо F1 впав більше ніж на
7%
        return {
            "detected": True,
            "details": f"Спостерігається деградація F1-Score з часом: середня
якість впала з {mean_first_part:.2f} до {mean_last_part:.2f}.",
            "payload": {
                "mean_f1_start": mean_first_part,
                "mean_f1_end": mean_last_part,
            },
        }
    return {"detected": False}


def detect_lookahead_bias(
    validation_metrics: dict, abnormal_score_threshold: float = 0.9, kwargs
) -> dict:
    """
    Виявляє потенційне 'підглядання у майбутнє', аналізуючи аномально високі
метрики.
    KB-DATA-BIAS
    """
    metric = "f1_score"
    val_score = validation_metrics.get(metric)
    if val_score and val_score > abnormal_score_threshold:
        return {
            "detected": True,
            "details": f"F1-Score на валідації ({val_score:.2f}) є аномально
високим, що може свідчити про 'підглядання у майбутнє'.",
            "payload": {"validation_f1_score": val_score},
        }
    return {"detected": False}

def detect_indicator_dependency(
    X_df: pd.DataFrame, model, importance_threshold: float = 0.4, kwargs
) -> dict:
    """
```

```
    Виявляє небезпечну залежність від одного індикатора на основі feature
importances.
    KB-MODEL-DEPD
    """
    if not hasattr(model, "feature_importances_"):
        return {
            "detected": False,
            "details": "Модель не має атрибуту 'feature_importances_', аналіз
неможливий.",
        }

    try:
        importances = model.feature_importances_
        feature_names = X_df.columns
        feature_importance_df = pd.DataFrame(
            {"feature": feature_names, "importance": importances}
        ).sort_values(by="importance", ascending=False)

        most_important_feature = feature_importance_df.iloc[0]

        if most_important_feature["importance"] > importance_threshold:
            return {
                "detected": True,
                "details": f"Ознака '{most_important_feature['feature']}' має
надзвичайно високу важливість ({most_important_feature['importance']:.2%}),
що створює ризик залежності.",
                "payload": {
                    "feature": most_important_feature["feature"],
                    "importance": most_important_feature["importance"],
                },
            }
    except Exception as e:
        return {
            "detected": False,
            "details": f"Під час аналізу важливості ознак сталася помилка:
{e}",
        }

    return {"detected": False}

def detect_excessive_trading_frequency(
    y_pred: pd.Series, max_trades_per_day: int = 10, kwargs
) -> dict:
    """
    Виявляє надмірну частоту торгівлі.
    KB-EVAL-COSTS
    """
    if not isinstance(y_pred.index, pd.DatetimeIndex):
        return {"detected": False, "details": "Індекс не є часовим."}

    trades = (y_pred.diff() != 0).astype(int)
    trades_per_day = trades.resample("D").sum()

    avg_trades = trades_per_day.mean()
    if avg_trades > max_trades_per_day:
        return {
```

```
            "detected": True,
            "details": f"Середня кількість угод на день ({avg_trades:.1f})
перевищує поріг ({max_trades_per_day}).",
            "payload": {"average_trades_per_day": avg_trades},
        }
    return {"detected": False}

def detect_flat_market_instability(
    X_df: pd.DataFrame,
    y_true: pd.Series,
    y_pred: pd.Series,
    adx_threshold: int = 20,
    performance_drop_threshold: float = 0.05,
    kwargs,
) -> dict:
    """
    Перевіряє, чи падає якість моделі на боковому ринку.
    KB-EVAL-FLAT
    """
    # Потрібна ознака ADX, поки що заглушка, якщо її немає
    adx_col = next((col for col in X_df.columns if "ADX" in col.upper()),
None)
    if not adx_col:
        return {"detected": False, "details": "Ознака ADX не знайдена."}

    analysis_df = pd.DataFrame(
        {"y_true": y_true, "y_pred": y_pred, "adx": X_df[adx_col]}
    )

    trending_market = analysis_df[analysis_df["adx"] > adx_threshold]
    flat_market = analysis_df[analysis_df["adx"] <= adx_threshold]

    if len(trending_market) == 0 or len(flat_market) == 0:
        return {
            "detected": False,
            "details": "Недостатньо даних для порівняння ринкових режимів.",
        }

    f1_trending = f1_score(trending_market["y_true"],
trending_market["y_pred"])
    f1_flat = f1_score(flat_market["y_true"], flat_market["y_pred"])

    drop = f1_trending - f1_flat
    if drop > performance_drop_threshold:
        return {
            "detected": True,
            "details": f"F1-Score падає на {drop:.2%} під час бокового ринку
(з {f1_trending:.2f} до {f1_flat:.2f}).",
            "payload": {
                "f1_trending": f1_trending,
                "f1_flat": f1_flat,
                "performance_drop": drop,
            },
        }
    return {"detected": False}
```

```
def analyze_prediction_confidence(
    y_prob: pd.Series, confidence_threshold: float = 0.6, kwargs
) -> dict:
    """
    Аналізує впевненість прогнозів моделі.
    KB-MODEL-CONF
    """
    # y_prob - це ймовірності класу 1
    if y_prob is None:
        return {
            "detected": False,
            "details": "Ймовірності прогнозів (y_prob) не надано.",
        }

    # Зона невпевненості: наприклад, між 40% і 60%
    low_confidence_zone = 1 - confidence_threshold
    uncertain_preds = y_prob[
        (y_prob > low_confidence_zone) & (y_prob < confidence_threshold)
    ]

    uncertain_ratio = len(uncertain_preds) / len(y_prob) if len(y_prob) > 0
else 0

    if uncertain_ratio > 0.5:  # Якщо більше 50% прогнозів - невпевнені
        return {
            "detected": True,
            "details": f"{uncertain_ratio:.1%} прогнозів знаходяться у 'зоні
невпевненості' (між {low_confidence_zone:.0%} та
{confidence_threshold:.0%}).",
            "payload": {"uncertain_predictions_ratio": uncertain_ratio},
        }
    return {"detected": False}


def detect_data_snooping_risk(kwargs) -> dict:
    """
    Надає рекомендацію щодо перевірки на переоптимізацію.
    KB-BT-SNOOP
    """
    # Цю проблему складно виявити автоматично, тому детектор носить
інформаційний характер
    return {
        "detected": True,  # Завжди спрацьовує, щоб нагадати користувачу
        "details": "Рекомендовано провести Walk-Forward аналіз або тестування
на повністю нових Out-of-Sample даних для валідації стійкості моделі.",
        "payload": {"manual_check_required": True},
    }


def detect_excessive_drawdown(
    y_true: pd.Series, y_pred: pd.Series, max_drawdown_threshold: float =
0.20, kwargs
) -> dict:
    """
    Розраховує максимальну просадку капіталу.
```

```
    KB-RISK-DRAWDOWN
    """
    returns = (y_true.shift(-1) - y_true) * y_pred  # Проста симуляція
прибутку
    if returns.isnull().all():
        return {"detected": False, "details": "Неможливо розрахувати
прибутковість."}

    cumulative_returns = (1 + returns).cumprod()
    peak = cumulative_returns.cummax()
    drawdown = (cumulative_returns - peak) / peak
    max_drawdown = drawdown.min()

    if abs(max_drawdown) > max_drawdown_threshold:
        return {
            "detected": True,
            "details": f"Максимальна просадка капіталу
({abs(max_drawdown):.2%}) перевищує поріг ({max_drawdown_threshold:.2%}).",
            "payload": {"max_drawdown": abs(max_drawdown)},
        }
    return {"detected": False}


def find_systematic_error_patterns(
    y_true: pd.Series,
    y_pred: pd.Series,
    error_rate_multiplier: float = 2.0,
    min_count: int = 10,
    periods: list = None,
    kwargs,
) -> dict:
    """
    Шукає системні помилки за різними часовими періодами:
    години, день тижня, день місяця, місяць (за замовчуванням).
    KB-EVAL-REGIME
    Параметри:
      - error_rate_multiplier: або float для всіх періодів, або dict
{'hour':2.0, ...}
      - min_count: мінімальна кількість спостережень в групі, щоб її
враховувати
      - periods: список періодів для перевірки, наприклад ['hour','weekday']
    """
    if not isinstance(y_true.index, pd.DatetimeIndex):
        return {"detected": False, "details": "Індекс не є часовим."}

    if periods is None:
        periods = ["hour", "weekday", "day", "month"]

    # дозволяємо задавати multiplier як число або як dict для різних періодів
    def get_multiplier(period):
        if isinstance(error_rate_multiplier, dict):
            return error_rate_multiplier.get(period, 2.0)
        return float(error_rate_multiplier)

    errors = (y_true != y_pred).astype(int)
    overall_error_rate = errors.mean()
```

```
    period_groupers = {
        "hour": errors.index.hour,
        "weekday": errors.index.dayofweek,  # Monday=0..Sunday=6
        "day": errors.index.day,  # day of month 1..31
        "month": errors.index.month,  # 1..12
    }
    problematic = {}
    for period in periods:
        if period not in period_groupers:
            continue
        grouper = period_groupers[period]
        grouped = errors.groupby(grouper)
        # compute mean error rate and counts per bucket
        stats = grouped.agg(["mean", "count"])  # DataFrame with columns
mean,count
        stats.columns = ["error_rate", "count"]
        # filter by min_count to avoid noisy buckets
        stats = stats[stats["count"] >= min_count]
        if stats.empty:
            continue
        multiplier = get_multiplier(period)
        mask = stats["error_rate"] > overall_error_rate * multiplier
        if mask.any():
            detected_buckets = stats[mask]
            # convert to simple dict: bucket_value -> {error_rate, count}
            buckets_info = {
                int(idx): {
                    "error_rate": float(row["error_rate"]),
                    "count": int(row["count"]),
                }
                for idx, row in detected_buckets.iterrows()
            }
            problematic[period] = {
                "overall_error_rate": float(overall_error_rate),
                "multiplier": multiplier,
                "buckets": buckets_info,
            }

    if problematic:
        # короткий текст для деталі
        summary_parts = []
        for p, info in problematic.items():
            buckets = list(info["buckets"].keys())
            summary_parts.append(f"{p}: {buckets}")
        details = (
            "Знайдено аномально високі рівні помилок за періодами -> "
            + "; ".join(summary_parts)
        )
        return {
            "detected": True,
            "details": details,
            "payload": {
                "overall_error_rate": float(overall_error_rate),
                "min_count": min_count,
                "problematic_periods": problematic,
            },
```

```
        }

    return {"detected": False}
```

base.json:

```
{
  "Data Issues": [
    {
      "id": "KB-DATA-FEAT",
      "name": "Надлишкові або зашумлені ознаки",
      "description": "Висока кореляція між ознаками (мультиколінеарність)
ускладнює модель та підвищує ризик перенавчання.",
      "detector": {
        "function": "detect_multicollinearity",
        "parameters": {
          "correlation_threshold": 0.95
        }
      },
      "corrector": {
        "type": "feature_engineering",
        "recommendations": [
          {
            "priority": 1,
            "action": "create_interaction_features",
            "description": "Для груп скорельованих індикаторів (напр.,
компоненти MACD, Bollinger Bands) створіть комбіновані ознаки (MACD_hist,
BB_width, BB_percent), які зберігають інформацію, але зменшують
надлишковість.",
            "implementation": {
              "MACD": "df['MACD_hist'] = df['MACD...'] - df['MACDs...']",
              "BollingerBands": "df['BB_width'] = df['BBB...']"
            }
          },
          {
            "priority": 2,
            "action": "apply_regularization",
            "description": "Використовуйте L1-регуляризацію (reg_alpha в
XGBoost), яка автоматично 'обнуляє' ваги найменш корисних ознак під час
навчання.",
            "implementation": { "hyperparameter": "reg_alpha",
"suggested_range": [0.01, 1.0] }
          }
        ]
      }
    },
    {
      "id": "KB-DATA-IMB",
      "name": "Дисбаланс класів",
      "description": "Нерівномірний розподіл класів у цільовій змінній, що
змушує модель ігнорувати рідкісний клас.",
      "detector": {
        "function": "detect_class_imbalance",
        "parameters": {
          "imbalance_threshold": 0.65
        }
      },
```

```
      "corrector": {
        "type": "model_parameter",
        "recommendations": [
          {
            "priority": 1,
            "action": "use_class_weights",
            "description": "Використовуйте ваги класів у моделі (параметр
`scale_pos_weight` в XGBoost), щоб сильніше штрафувати за помилки на
рідкісному класі.",
            "implementation": { "hyperparameter": "scale_pos_weight",
"formula": "count(negative_class) / count(positive_class)" }
          }
        ]
      }
    },
    {
      "id": "KB-DATA-BIAS",
      "name": "Підглядання у майбутнє (Look-ahead Bias)",
      "description": "Модель випадково використовує інформацію, яка не була б
доступна в реальний час, що призводить до нереалістично високих
результатів.",
      "detector": {
        "function": "detect_lookahead_bias",
        "parameters": {
          "abnormal_score_threshold": 0.9
        }
      },
      "corrector": {
        "type": "training_strategy",
        "recommendations": [
          {
            "priority": 1,
            "action": "ensure_causality_in_features",
            "description": "Переконайтеся, що всі розрахунки ознак (ковзні
середні, нормалізація) використовують тільки минулі дані. Scaler має
навчатися на тренувальних даних і лише потім застосовуватися до нових.",
            "implementation": {}
          },
          {
            "priority": 2,
            "action": "use_timeseriessplit",
            "description": "Для крос-валідації використовуйте
TimeSeriesSplit, щоб тренувальна вибірка завжди передувала тестовій у часі.",
            "implementation": { "tool":
"sklearn.model_selection.TimeSeriesSplit" }
          }
        ]
      }
    }
  ],
  "Model Issues": [
    {
      "id": "KB-MODEL-OVR",
      "name": "Перенавчання (Overfitting)",
      "description": "Модель показує відмінні результати на тренувальній
вибірці, але значно гірші на нових (валідаційних) даних.",
```

```
      "detector": {
        "function": "detect_overfitting",
        "parameters": {
          "performance_gap_threshold": 0.15
        },
        "required_data": ["train_metrics", "validation_metrics"]
      },
      "corrector": {
        "type": "hyperparameter_tuning",
        "recommendations": [
          {
            "priority": 1,
            "action": "reduce_model_complexity",
            "description": "Зменшіть складність моделі. Для деревних моделей
це означає зменшення `max_depth` (глибини дерев) та збільшення
`min_child_weight`.",
            "implementation": { "hyperparameters": ["max_depth",
"min_child_weight"] }
          },
          {
            "priority": 2,
            "action": "increase_regularization",
            "description": "Посильте L1/L2 регуляризацію (`reg_alpha`,
`reg_lambda`), щоб 'штрафувати' модель за надто складні рішення.",
            "implementation": { "hyperparameters": ["reg_alpha",
"reg_lambda"] }
          }
        ]
      }
    },
    {
        "id": "KB-MODEL-AUTOCORR",
        "name": "Автокореляція помилок",
        "description": "Помилки моделі не є випадковими, а залежать одна від
одної, що вказує на пропущений часовий патерн.",
        "detector": {
          "function": "detect_error_autocorrelation",
          "parameters": {
            "durbin_watson_thresholds": [1.5, 2.5]
          }
        },
        "corrector": {
          "type": "feature_engineering",
          "recommendations": [
            {
              "priority": 1,
              "action": "add_lag_features",
              "description": "Додайте лаги цільової змінної та/або самих
помилок як нові ознаки, щоб модель могла врахувати попередні стани системи.",
              "implementation": { "features": ["target_lag_1", "error_lag_1"]
}
            }
          ]
        }
    },
    {
```

```
        "id": "KB-MODEL-DEPD",
        "name": "Залежність від одного індикатора",
        "description": "Продуктивність моделі критично залежить від однієї
ознаки або групи однотипних індикаторів, що робить її нестійкою.",
        "detector": {
          "function": "detect_indicator_dependency",
          "parameters": {
            "importance_threshold": 0.4
          },
          "required_data": ["feature_importances"]
        },
        "corrector": {
          "type": "feature_engineering",
          "recommendations": [
            {
              "priority": 1,
              "action": "diversify_features",
              "description": "Додайте індикатори з різних категорій: трендові
(Moving Averages), осцилятори (Stochastic), об'єму (On-Balance Volume),
волатильності (Bollinger Bands).",
              "implementation": {}
            },
            {
              "priority": 2,
              "action": "use_ensemble_methods",
              "description": "Створіть кілька моделей, кожна з яких
фокусується на своїй групі індикаторів, а потім усередніть їхні прогнози.",
              "implementation": {}
            }
          ]
        }
    },
    {
        "id": "KB-MODEL-CONF",
        "name": "Ігнорування 'впевненості' моделі",
        "description": "Модель генерує багато прогнозів з низькою ймовірністю
(близько 50%), що робить сигнали ненадійними.",
        "detector": {
          "function": "analyze_prediction_confidence",
          "parameters": {
            "confidence_threshold": 0.6
          },
          "required_data": ["y_prob"]
        },
        "corrector": {
          "type": "trading_strategy",
          "recommendations": [
            {
              "priority": 1,
              "action": "add_confidence_threshold",
              "description": "Входьте в угоду, тільки якщо ймовірність
прогнозу перевищує поріг (напр., 60% або 65%).",
              "implementation": {}
            },
            {
              "priority": 2,
```

```
              "action": "use_confidence_for_position_sizing",
              "description": "Використовуйте ймовірність для визначення
розміру позиції: вища впевненість — більша позиція.",
              "implementation": {}
            }
          ]
        }
    }
  ],
  "Evaluation Issues": [
    {
      "id": "KB-EVAL-VOL",
      "name": "Нестабільність на волатильному ринку",
      "description": "Якість прогнозів моделі значно погіршується в періоди
високої ринкової волатильності.",
      "detector": {
        "function": "detect_volatility_instability",
        "parameters": {
          "volatility_feature_pattern": "ATR",
          "performance_drop_threshold": 0.1
        }
      },
      "corrector": {
        "type": "feature_engineering",
        "recommendations": [
          {
            "priority": 1,
            "action": "add_regime_feature",
            "description": "Створіть бінарну ознаку 'is_high_volatility', щоб
модель могла явно розрізняти ринкові режими.",
            "implementation": { "expression": "(df['ATR...'] >
df['ATR...'].rolling(30).mean()).astype(int)" }
          }
        ]
      }
    },
    {
      "id": "KB-EVAL-DRIFT",
      "name": "Дрейф концепту (Concept Drift)",
      "description": "Продуктивність моделі деградує з часом, оскільки
ринкові закономірності змінюються.",
      "detector": {
        "function": "detect_concept_drift",
        "parameters": {
            "time_window": "90D"
        }
      },
      "corrector": {
        "type": "training_strategy",
        "recommendations": [
            {
                "priority": 1,
                "action": "retrain_on_rolling_window",
                "description": "Регулярно перенавчайте модель на найновіших
даних, використовуючи 'ковзне вікно' для тренування.",
                "implementation": {}
```

```
            }
          ]
        }
    },
    {
        "id": "KB-EVAL-COSTS",
        "name": "Надмірна частота торгівлі",
        "description": "Модель генерує занадто багато сигналів, що може
призвести до збитків через комісії та прослизання.",
        "detector": {
          "function": "detect_excessive_trading_frequency",
          "parameters": {
            "max_trades_per_day": 10
          }
        },
        "corrector": {
          "type": "feature_engineering",
          "recommendations": [
            {
              "priority": 1,
              "action": "add_cooldown_filter",
              "description": "Введіть 'період охолодження' після кожної
угоди, щоб уникнути миттєвих контр-сигналів. Додайте ознаку 'час з останньої
угоди'.",
              "implementation": { "feature": "time_since_last_trade" }
            },
            {
              "priority": 2,
              "action": "increase_timeframe",
              "description": "Агрегуйте дані до більших таймфреймів (напр., з
5-хвилинних до 15-хвилинних), щоб зменшити ринковий шум.",
              "implementation": {}
            }
          ]
        }
    },
    {
        "id": "KB-EVAL-FLAT",
        "name": "Низька продуктивність на боковому ринку",
        "description": "Модель втрачає ефективність, коли на ринку відсутній
чіткий тренд (флет).",
        "detector": {
          "function": "detect_flat_market_instability",
          "parameters": {
            "adx_threshold": 20,
            "performance_drop_threshold": 0.03
          }
        },
        "corrector": {
          "type": "feature_engineering",
          "recommendations": [
            {
              "priority": 1,
              "action": "add_market_regime_filter",
              "description": "Створіть ознаку 'ринкового режиму' (напр., на
основі індикатора ADX), щоб модель могла розрізняти тренд та флет.",
```

```
              "implementation": { "indicator": "ADX", "expression":
"(df['ADX_14'] < 20).astype(int)" }
            },
            {
              "priority": 2,
              "action": "use_dynamic_targets",
              "description": "Впровадьте динамічні цілі для take-profit/stop-
loss, що залежать від поточної волатильності (ATR).",
              "implementation": { "indicator": "ATR" }
            }
          ]
        }
    },
    {
        "id": "KB-BT-SNOOP",
        "name": "Ризик переоптимізації (Data Snooping)",
        "description": "Продуктивність моделі значно падає на нових даних, що
не використовувались при оптимізації, вказуючи на 'підгонку' під історію.",
        "detector": {
          "function": "detect_data_snooping_risk",
          "parameters": {
            "performance_drop_threshold": 0.1
          }
        },
        "corrector": {
          "type": "training_strategy",
          "recommendations": [
            {
              "priority": 1,
              "action": "use_walk_forward_validation",
              "description": "Використовуйте послідовну валідацію (Walk-
Forward), щоб імітувати реальну торгівлю, тренуючи на минулих даних і
тестуючи на майбутніх.",
              "implementation": {}
            },
            {
              "priority": 2,
              "action": "reserve_true_out_of_sample_data",
              "description": "Зарезервуйте фінальний тестовий набір, який
ніколи не використовувався для налаштування моделі, для остаточної, чесної
оцінки.",
              "implementation": {}
            }
          ]
        }
    },
    {
        "id": "KB-RISK-DRAWDOWN",
        "name": "Надмірна просадка капіталу",
        "description": "Стратегія показує періоди катастрофічних втрат, що
робить її занадто ризикованою для реального використання.",
        "detector": {
          "function": "detect_excessive_drawdown",
          "parameters": {
            "max_drawdown_threshold": 0.20
          }
```

```
        },
        "corrector": {
          "type": "risk_management",
          "recommendations": [
            {
              "priority": 1,
              "action": "implement_dynamic_stop_loss",
              "description": "Використовуйте динамічний Stop-Loss, що
базується на поточній волатильності ринку (наприклад, 2 * ATR).",
              "implementation": { "indicator": "ATR" }
            },
            {
              "priority": 2,
              "action": "optimize_position_sizing",
              "description": "Зменшуйте розмір позиції після серії збитків,
щоб обмежити подальші втрати.",
              "implementation": {}
            }
          ]
        }
    },
    {
        "id": "KB-EVAL-REGIME",
        "name": "Системні помилки у певних режимах",
        "description": "Модель систематично помиляється за певних умов
(напр., час доби, день тижня), що вказує на невивчений патерн.",
        "detector": {
          "function": "find_systematic_error_patterns",
          "parameters": {
            "error_rate_multiplier": 1.5
          }
        },
        "corrector": {
          "type": "trading_strategy",
          "recommendations": [
            {
              "priority": 1,
              "action": "add_time_based_features",
              "description": "Створіть ознаки, що відображають час доби, день
тижня або близькість до відкриття/закриття ринку.",
              "implementation": { "features": ["hour_of_day", "day_of_week"]
}
            },
            {
              "priority": 2,
              "action": "implement_time_filters",
              "description": "Забороніть торгівлю в періоди, коли модель
систематично показує погані результати (напр., перші 30 хвилин сесії).",
              "implementation": {}
            }
          ]
        }
    }
  ]
}
```