

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КІЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

**НН Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу**

До захисту допущено:
Завідувач кафедри
_____ Оксана ТИМОЩУК
«____» 2023 р.

**Дипломна робота
на здобуття ступеня бакалавра
за освітньо–професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»
на тему: «Дослідження ефективності алгоритмів машинного навчання
для задачі управління активами на ринку криптовалют з урахуванням
оцінки довіри користувачів до криптовалюти»**

Виконав: студент IV курсу, групи КА–93
Кротенко Нікіта Сергійович

Керівник: асистент кафедри ММСА
Канцедал Георгій Олегович

Консультант з нормоконтролю: к.ф.–м.н.
Статкевич Віталій Михайлович

Консультант з економічного розділу: к.е.н., доцент
Рошина Надія Василівна

Рецензент: к.ф.–м.н., доцент, доцент кафедри загальної
фізики і математики Полтавського національного
педагогічного університету імені В. Г. Короленка
Хорольський Олексій Вікторович

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____

Київ – 2023 року

**Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
НН Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу**

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 124 «Системний аналіз»
Освітня програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Оксана ТИМОЩУК
«__» травня 2023 р.

**ЗАВДАННЯ
на дипломну роботу студенту
Кротенко Нікіті Сергійовичу**

1. Тема роботи «Дослідження ефективності алгоритмів машинного навчання для задачі управління активами на ринку криптовалют з урахуванням оцінки довіри користувачів до криптовалюти», керівник роботи Канцедал Георгій Олегович, асистент кафедри ММСА, затверджені наказом по університету від «30» травня 2023 р. № 2065–с.
2. Термін подання студентом роботи: 12.06.2023.
3. Вихідні дані до роботи: програмний продукт, призначений для збору і обробки даних, розробки, налаштування і порівняння різних моделей машинного навчання.
4. Зміст роботи: 1. Огляд предметної області. 2. Огляд методів машинного навчання для задачі передбачення часових рядів; 3. Створення програмного продукту; 4. Функціонально–вартісний аналіз програмного продукту.

5. Перелік ілюстративного матеріалу: презентація.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рошина Н.В., к.е.н., доцент		

7. Дата видачі завдання: 17.04.2023

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Налаштування хмарного серверу, бази даних і похвилинного завантаження даних	01.08.2022 — 20.09.2022	виконано
2.	Огляд літературних джерел	17.04.2023 — 23.04.2023	виконано
3.	Перший розділ. Огляд предметної області.	24.04.2023- 30.04.2023	виконано
4.	Другий розділ. Огляд методів машинного навчання.	01.05.2023 — 07.05.2023	виконано
5.	Третій розділ. Створення програмного продукту.	15.05.2023 — 21.05.2023	виконано
6.	Четвертий розділ. Функціонально-вартісний аналіз програмного продукту.	22.05.2023 — 28.05.2023	виконано
7.	Оформлення роботи.	29.05.2023- 05.06.2023	виконано

Студент _____

Нікіта КРОТЕНКО

Керівник _____

Георгій КАНЦЕДАЛ

РЕФЕРАТ

Дипломна робота: 177с., 37 рис., 16 табл., 2 додатки, 23 джерела.

МАШИННЕ НАВЧАННЯ, ГЛИБОКЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, LSTM, 1D-CNN, АНСАМБЛЬ, ПРОГНОЗУВАННЯ ЧАСОВОГО РЯДУ, ПОПЕРЕДНЯ ОБРОБКА ДАНИХ, ПЕРЕХРЕСНА ПЕРЕВІРКА, ПІДБІР ГІПЕРПАРАМЕТРІВ, КРИПТОВАЛЮТИ, BINANCE, УПРАВЛІННЯ РИЗИКАМИ, УПРАВЛІННЯ АКТИВАМИ.

Для проведення цього дослідження було розроблено базу даних і Python додаток для завантаження щовицьливих свічкових даних з криптовалютної біржі, а потім розміщено на віддаленому сервері.

Потім дані були оброблені та на них було визначено ключові точки, де рекомендувалося закривати короткі і відкривати довгі позиції (локальні мініуми) або закривати довгі і відкривати короткі позиції (локальні максимуми).

Для прогнозування та прийняття рішень було розроблено три моделі нейронних мереж: довготрива короткочасна пам'ять (LSTM), одновимірна згорткова нейронна мережа (1D-CNN) і модель ансамблю, що поєднує LSTM і 1D-CNN. Щоб забезпечити надійну оцінку продуктивності, було реалізовано методологію перехресної перевірки з розбиттям часового ряду на тренувальні і тестові вибірки. Крім того, було проведено автоматичний пошук гіперпараметрів для оптимізації конфігурації моделей.

Дослідження також передбачало розробку цільового стовпця та користувальських показників, адаптованих до конкретного завдання управління активами на ринку криптовалют. Ці показники були розроблені для оцінки ефективності моделей з точки зору прибутковості та управління ризиками.

Результати цього дослідження показують ефективність алгоритмів машинного навчання для управління активами на ринку криптовалют, враховуючи рейтинги довіри користувачів до криптовалют.

ABSTRACT

Bachelor thesis: 177 p., 37 fig., 16 tabl., 2 append., 23 sources.

MACHINE LEARNING, DEEP LEARNING, NEURAL NETWORKS, LSTM, 1D-CNN, ENSEMBLE, TIME-SERIES PREDICTION, DATA PREPROCESSING, CROSS-VALIDATION, HYPERPARAMETER TUNING, CRYPTOCURRENCIES, BINANCE, RISK MANAGEMENT, ASSET MANAGEMENT

To conduct this research, a database and Python application to load per-minute candlestick data were developed and then hosted on a remote server, to collect data from cryptocurrency exchange.

The data was then processed, and key points were identified where it was recommended to close short and open long positions (local minima) or close long and open short positions (local maxima).

Three neural network models were developed for prediction and decision-making: Long Short-Term Memory (LSTM), a one-dimensional Convolutional Neural Network (1D-CNN), and an ensemble model combining both LSTM and 1D-CNN. A time-series train-test split and cross-validation methodology were implemented to ensure reliable performance assessment. An automatic hyperparameter search was also conducted to optimize the models' configuration.

The research also involved the development of a target column and custom metrics tailored to the specific task of asset management in the cryptocurrency market. These metrics were designed to evaluate the performance of the models in terms of profitability and risk management.

This research provides insights into the effectiveness of machine learning algorithms for asset management in the cryptocurrency market, considering users' trust ratings towards cryptocurrencies.

ЗМІСТ

ЗМІСТ	6
ВСТУП.....	11
РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ.....	13
1.1 Біржа	13
1.1.1 Визначення біржі.....	13
1.1.2 Основні функції біржі	13
1.1.3 Типи бірж.....	15
1.1.4 Регуляція та нагляд за біржовою діяльністю	16
1.1.5 Огляд криптовалютної біржі Binance.....	18
1.2 Криптовалюта	19
1.2.1 Визначення криптовалюти.....	19
1.2.2 Опис технологій блокчейн.....	19
1.2.3 Основні принципи криптовалютних систем	20
1.2.4 Особливості криптовалюти ВТС	20
1.3 Ф'ючерси	21
1.3.1 Визначення ф'ючерсів	21
1.3.2 Роль ф'ючерсів у фінансових ринках	22
1.3.3 Типи ф'ючерсів	23
1.3.4 Механізм фінансування для perpetуальних ф'ючерсів.....	24
1.3.5 Торгове плече в ф'ючерсних контрактах	25
1.3.6 Ліквідація ф'ючерсних позицій	26
1.4 Висновки до розділу 1	27

РОЗДІЛ 2 ОГЛЯД МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ЗАДАЧІ	
ПЕРЕДБАЧЕННЯ ЧАСОВИХ РЯДІВ	29
2.1 Авторегресійні моделі.....	29
2.1.1 AR (Авторегресія)	29
2.1.2 MA (Середня ковзна)	29
2.1.3 ARMA (Авторегресія середньої ковзної).....	30
2.1.4 ARIMA (Авторегресія інтегрована середня ковзна)	30
2.1.5 Висновок про авторегресійні моделі.....	30
2.2 Експоненційно зважені моделі	31
2.2.1 SES (Експоненційне згладжування просте)	31
2.2.2 Holt–Winters (Тренд і сезонність)	31
2.2.3 Висновок про експоненційно зважені моделі.....	32
2.3 Рекурентні нейронні мережі (RNN)	33
2.3.1 LSTM (Довготривала краткосрочна пам'ять).....	33
2.3.2 GRU (Однічна управляюча рекурентна одиниця)	33
2.3.3 Bidirectional LSTM (Двостороння LSTM)	33
2.3.4 Висновок про прогнозуючі моделі на основі дерев рішень.....	34
2.4 Прогнозуючі моделі на основі дерев рішень	34
2.4.1 Random Forest (Випадковий ліс).....	34
2.4.2 Gradient Boosting (Градієнтний бустинг)	35
2.4.3 Висновок про прогнозуючі моделі на основі дерев рішень.....	35
2.5 Згорткові нейронні мережі (CNN).....	36
2.5.1 1D CNN (Одновимірна згорткова нейронна мережа).....	36
2.5.2 Висновок про згорткові нейронні мережі	36
2.6 Метрики для задачі регресії.....	37

2.7. Градієнтний метод.....	38
2.8. Складові частини нейронних мереж	40
2.8.1. Dense (або повнозв'язний) шар	40
2.8.2. Активаційні функції	41
2.8.3. Dropout	43
2.8.4. BatchNormalization.....	45
2.8.5. Регуляризація.....	46
2.8.6. Оптимізатори	47
2.9 Висновки до розділу 2.....	51
РОЗДІЛ 3 СТВОРЕННЯ ПРОГРАМНОГО ПРОДУКТУ	52
3.1. Налаштування збору даних і бази даних на хмарному сервері	52
3.1.1 Загальний огляд.....	52
3.1.2 Огляд Python–додатку	53
3.2. Створення розмітки для машинного навчання	55
3.2.1 Загальний огляд.....	55
3.2.2. Визначення допоміжних функцій.....	56
3.2.3 Пошук локальних екстремумів.....	58
3.2.4. Пошук початкових екстремумів	60
3.2.5. Пошук глобальних екстремумів на відрізках	62
3.2.6. Пошук неправильно визначених екстремумів	66
3.2.7. Пошук пропущених екстремумів	67
3.2.8. Повторення кроків 3.2.6 і 3.2.7 доки кількість екстремумів не стабілізується	69
3.2.9. Огляд результатів	70
3.3. Генерація нових ознак.....	71

3.3.1. Генерація по–хвилинних ознак.....	71
3.3.2. Генерація по–контрактних ознак.....	77
3.4. Створення цільового показника і метрики	78
3.4.1. Загальний огляд.....	78
3.4.2. Перший цільовий показник	79
3.4.3. Другий цільовий показник	80
3.4.4. Фінальна обробка цільових показників	81
3.4.5. Метрика.....	82
3.5. Попередня обробка даних.....	86
3.6. Розбиття даних на тренувальну, тестову і валідаційну вибірки	87
3.7. Створення моделі на основі LSTM шару і підбір гіперпараметрів	87
3.7.1. Підготовка даних для LSTM шару	87
3.7.2. Підбір гіпер–параметрів моделі.....	88
3.7.3. Структура оптимальної моделі (рис 3.26).....	89
3.7.4 Тренування моделі та огляд результатів (рис. 3.27).....	90
3.8. Створення моделі на основі 1D-CNN шару і підбір гіперпараметрів ...	90
3.8.1. Підготовка даних для 1D-CNN шару	91
3.8.2. Структура оптимальної моделі (рис. 3.28).....	91
3.8.3. Тренування моделі і огляд результатів (рис 3.29)	92
3.9. Створення ансамблевої моделі	93
3.9.1. Структура ансамблевої моделі (рис 3.30)	93
3.9.2. Тренування моделі і огляд результатів (рис 3.31)	94
3.10 Висновки до розділу 3	94
РОЗДІЛ 4 ФУНКЦІОНАЛЬНО–ВАРТИСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ	97

4.1 Постановка задачі проектування	97
4.2 Обґрунтування функцій програмного продукту	98
4.3 Обґрунтування системи параметрів програмного продукту	101
4.4 Аналіз експертного оцінювання параметрів	104
4.5 Аналіз рівня якості варіантів реалізації функцій.....	109
4.6 Економічний аналіз варіантів розробки ПП	110
4.7 Вибір кращого варіанту програмного продукту техніко–економічного рівня	116
4.8 Висновки до розділу 4.....	117
ВИСНОВКИ	118
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	119
ДОДАТОК А ЛІСТИНГ ПРОГРАМИ	122
ДОДАТОК Б ПРЕЗЕНТАЦІЯ.....	163

ВСТУП

Криптовалютний ринок є одним з найдинамічніших та найменш передбачуваних ринків у сучасному фінансовому світі. Значна волатильність та швидкі зміни цін роблять його привабливим для інвесторів, проте одночасно створюють складності із прийняттям ефективних управлінських рішень. У цьому контексті виникає потреба в застосуванні передових методів аналізу та прогнозування, таких як алгоритми машинного навчання, для покращення результативності управління активами на ринку криптовалют.

Це дослідження присвячене оцінці ефективності алгоритмів машинного навчання для управління активами на ринку криптовалют з урахуванням рейтингів довіри користувачів до криптовалют. Використання даних про довіру користувачів може бути важливим чинником у прийнятті управлінських рішень, оскільки ці рейтинги відображають ставлення споживачів до певних криптовалют та їх готовність ризикувати вкладеними коштами.

Для досягнення цієї мети, було використано дані про хвилинні свічки та інші показники, які були зібрані з однієї з найпопулярніших криптовалютних бірж - Binance. Для цього на хмарному сервері було створено базу даних і Python-додаток, який записував туди дані щохвилинно.

Після обробки даних, на основі аналізу локальних мінімумів та максимумів було виявлено ключові точки, де рекомендовано закривати короткі позиції та відкривати довгі, або закривати довгі позиції та відкривати короткі.

Окрім того, створено додаткові ознаки, які допомогли врахувати різні аспекти ринку криптовалют. Застосування передобробки та інженерії ознак дало змогу поліпшити якість даних та підготувати їх для подальшого аналізу.

Для прогнозування та прийняття управлінських рішень розроблено три моделі нейронних мереж: Long Short-Term Memory (LSTM), одновимірна

згорткова нейронна мережа (1D-CNN) та ансамбль моделей, що поєднує LSTM та 1D-CNN. Було використано методи поділу даних на тренувальну та тестову вибірки для часових рядів та перехресну перевірку для достовірної оцінки продуктивності моделей. Щоб досягти оптимальних результатів, було реалізовано автоматичний пошук гіперпараметрів для кожної моделі.

Дослідження також передбачало розробку цільового показника та користувацької метрики, яка дозволяє оцінити ефективність моделі з точки зору прибутковості та управління ризиками. Аналіз результатів надав нам важливі висновки щодо ефективності алгоритмів машинного навчання для управління активами на ринку криптовалют, зокрема з урахуванням довіри користувачів до криптовалют.

Це дослідження має значимість для інвесторів, фінансових установ та дослідників, які зацікавлені в застосуванні методів машинного навчання для покращення управління активами в галузі криптовалют. Результати дослідження сприятимуть підвищенню рівня прийняття рішень та забезпеченню більш вдалих інвестиційних стратегій.

РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Біржа

1.1.1 Визначення біржі

Біржа є спеціалізованою організацією, яка забезпечує механізм та інфраструктуру для проведення торгівлі різними фінансовими активами. Вона виступає як централізована платформа, де купуються та продаються цінні папери, товари, валюти, криптовалюти тощо.

Біржі грають важливу роль у фінансовій системі, дозволяючи компаніям, інвесторам та трейдерам здійснювати операції з активами. Вони забезпечують прозорість, легкість та безпеку укладання угод, а також формують цінові механізми для визначення вартості активів на основі попиту та пропозицій.

1.1.2 Основні функції біржі

Можна виділити три основні функції біржі.

1. Забезпечення ліквідності ринку. Одна з основних функцій біржі полягає у забезпеченні ліквідності ринку. Ліквідність визначається як здатність активу швидко та ефективно бути купленим або проданим без значних змін у ціні. Біржа виконує роль посередника, збираючи покупців та продавців разом та створюючи умови для виконання операцій з активами. Це забезпечує трейдерам можливість легко знайти контрагентів для торгівлі та швидко виконати свої замовлення. Біржа забезпечує ліквідність ринку шляхом створення механізмів, таких як

централізована платформа торгівлі, система збирання та відображення замовлень, технологічні рішення для швидкої передачі даних та інші інструменти, які сприяють ефективному проведенню торгівлі.

2. Зведення покупців та продавців. Біржа виконує роль посередника, що зводить покупців та продавців разом. Вона надає платформу, де трейдери можуть розміщувати свої замовлення на купівлю або продаж активів. Завдяки централізованій системі збирання та відображення замовлень, покупці та продавці мають можливість бачити наявні замовлення та вибирати ті, які відповідають їхнім потребам. Згодом, коли замовлення з покупцем і продавцем збігаються, біржа виконує операцію та реєструє її. Цей процес зведення покупців та продавців відбувається за допомогою механізмів, таких як ринкові ордери, лімітні ордери, стоп–ордери та інші типи замовлень. Кожен трейдер має можливість вибрати певний тип замовлення та параметри, що відповідають його потребам та торгівельній стратегії.
3. Прозорість та регулювання ринку. Біржа забезпечує прозорість та регулювання ринку, забезпечуючи правила та норми поведінки для учасників. Це забезпечує довіру та стабільність на ринку. Біржові правила визначають порядок подання та виконання замовлень, правила ведення торгівлі, вимоги до лістингу активів, механізми розгляду скарг та регулювання конфліктів інтересів. Регулювання ринку може здійснюватися самою біржею або за допомогою наглядових органів та регуляторів фінансового ринку. Такі органи можуть встановлювати правила для запобігання маніпуляціям ринком, незаконній діяльності та іншим ризикам. Прозорість ринку забезпечується шляхом публікації інформації про торговельні операції, обсяги, ціни та інші дані, які допомагають учасникам ринку аналізувати та приймати рішення.

Ці функції є важливими складовими, які забезпечують ефективну торгівлю та функціонування фінансового ринку.

1.1.3 Типи бірж

До основних типів бірж можна віднести наступні:

- 1) фондова біржа: це біржа, де торгуються акції та інші цінні папери компаній. Фондові біржі можуть бути національними (наприклад, New York Stock Exchange у США) або регіональними (наприклад, London Stock Exchange у Великобританії);
- 2) товарна біржа: на таких біржах торгуються товари, такі як нафта, золото, сіль, зерно і інші сировинні товари. Прикладами товарних бірж є Chicago Mercantile Exchange (CME) і London Metal Exchange (LME);
- 3) валютна біржа (Forex): це біржа, де проводиться торгівля валютами. Трейдери купують одну валюту, продавці продають іншу. Найбільш відомою валютною біржею є Foreign Exchange Market (FOREX), яка є децентралізованою та працює цілодобово;
- 4) ф'ючерсна біржа: на ф'ючерсних біржах торгуються ф'ючерсні контракти, що дозволяють купувати або продавати актив в майбутньому за фіксованою ціною. Найвідоміші ф'ючерсні біржі включають Chicago Mercantile Exchange (CME), Chicago Board Options Exchange (CBOE) та Eurex Exchange;
- 5) опціонна біржа: на опціонних біржах торгуються опціони, що надають право на купівлю або продаж активу за певною ціною протягом визначеного періоду. Наприклад, Chicago Board Options Exchange (CBOE) та International Securities Exchange (ISE);
- 6) біржа товарних ф'ючерсів (Commodity Futures Exchange): це спеціалізована біржа, де торгуються ф'ючерси на різноманітні сировинні товари, такі як нафта, газ, метали, сіль, зерно та інші. Прикладом є New York Mercantile Exchange (NYMEX);

7) криптовалютна біржа: це спеціалізована біржа, де трейдери можуть купувати та продавати криптовалюти, такі як Bitcoin, Ethereum, Tether та інші.

Детальніше варто розглянути криптовалютні біржі, що забезпечують платформу для обміну цифрових активів між різними учасниками ринку. Вони пропонують різні торгові пари, засновані на криптовалютах, дозволяючи трейдерам купувати і продавати ці активи на основі попиту та пропозиції. Криптовалютні біржі також можуть надавати інші послуги, такі як зберігання криптовалютних кошельків, маржеву торгівлю, пристрой для аналізу ринку та інші інструменти, що допомагають трейдерам в їхніх операціях з криптовалютами.

Криптовалютні біржі можуть мати свої особливості та функціональні можливості, включаючи різні типи ордерів (наприклад, ринкові ордери, лімітні ордери), механізми підтримки безпеки (наприклад, двофакторна аутентифікація, холодне зберігання криптовалют), відомості про ліквідність та обсяги торгів, а також інтерфейс користувача для виконання операцій на біржі.

1.1.4 Регуляція та нагляд за біржовою діяльністю

Біржева діяльність підлягає регулюванню та нагляду з боку спеціальних органів, які встановлюють правила та норми для функціонування бірж і забезпечують дотримання законодавства. Органи регулювання можуть бути національними фінансовими регуляторами, такими як комісії з цінних паперів і бірж (наприклад, Securities and Exchange Commission в США), центральними банками або спеціально уповноваженими органами.

Основна роль органів регулювання полягає у забезпечені стабільності та надійності фінансових ринків. Вони встановлюють правила для допуску активів до торгівлі, контролюють діяльність брокерських фірм та інших

учасників ринку, проводять моніторинг ризиків та здійснюють заходи для запобігання маніпуляціям ринком, шахрайству та іншим недобросовісним практикам.

Одним із основних завдань органів регулювання є забезпечення безпеки та стабільності біржевих ринків. Вони приймають рішення та встановлюють правила, які мають на меті захист інвесторів, зниження системних ризиків та запобігання нестабільності ринку.

До заходів безпеки та стабільності ринку можуть відноситися наступні.

1. Регулярні перевірки та аудити. Органи регулювання здійснюють перевірки фінансової діяльності бірж, брокерських фірм та інших учасників ринку. Це допомагає виявити порушення та недобросовісну діяльність, а також забезпечує відповідність фінансовим стандартам та правилам.
2. Капіталовкладення та резервування. Органи регулювання можуть встановлювати вимоги до мінімального капіталу, який повинні мати біржі та брокерські фірми для забезпечення фінансової стійкості. Вони також можуть вимагати створення резервних фондів, які можуть бути використані в разі необхідності для покриття непередбачуваних збитків або нестабільності ринку.
3. Моніторинг та контроль ризиків. Органи регулювання здійснюють моніторинг ризиків на фінансових ринках та встановлюють механізми контролю за ними. Це може включати встановлення обмежень на використання плеча, встановлення правил для захисту від великих коливань цін, визначення обов'язкових маржінних вимог та інші заходи для зниження фінансових ризиків та забезпечення стійкості ринку. Встановлення обов'язкових маржінних вимог означає, що учасники ринку повинні мати достатню суму коштів або активів як гарантію своїх позицій. Це допомагає зменшити можливість дефолту та збитків, які можуть виникнути внаслідок неблагоприятних рухів на ринку. Крім того, органи регулювання встановлюють правила для захисту від

великих коливань цін. Це може включати обмеження на використання плеча, тобто обмеження масштабу позицій, які можуть бути відкриті відносно наявних коштів. Це допомагає уникнути значних втрат у випадку різких змін цін.

Загальною метою заходів безпеки та стабільності ринку є забезпечення довіри учасників ринку до бірж та фінансової системи в цілому. Це важливо для притягнення інвесторів та забезпечення ефективної та безпечної торгівлі активами на біржі.

1.1.5 Огляд криптовалютної біржі Binance

Біржа Binance є однією з найбільш відомих та популярних криптовалютних бірж у світі. Заснована в 2017 році, вона пропонує широкий вибір торгових пар криптовалют і надає можливість обміну та торгівлі різними цифровими активами.

Однією з головних переваг біржі Binance для вашої задачі є наявність квартальних і перпетуальних ф'ючерсів. Квартальні ф'ючерси – це контракти, що закінчуються після трьох місяців, тоді як перпетуальні ф'ючерси – це контракти без обмеження терміну. Це дозволяє нам використовувати ці інструменти для розробки та використання алгоритмів машинного навчання для управління активами з урахуванням різних термінових вимог і стратегій.

Крім того, Binance надає доступ до історичних і актуальних даних через своє API (Application Programming Interface). Це означає, що ми можемо отримати доступ до різних типів даних, таких як ціни, обсяги, свічки та інші, які можуть бути використані для аналізу та розробки моделей машинного навчання. Цей доступ до даних дозволяє нам будувати моделі на основі достовірної та актуальної інформації, що покращує ефективність нашої роботи та прийняття рішень.

Отже, Binance з його квартальними і перпетуальними ф'ючерсами, а також доступом до історичних і актуальних даних через своє API, є відмінним вибором для мого дослідження ефективності алгоритмів машинного навчання для управління активами на ринку криптовалют.

1.2 Криптовалюта

1.2.1 Визначення криптовалюти

Криптовалюта – це цифровий або віртуальний актив, що використовує криптографію для забезпечення безпеки транзакцій, контролю створення нових одиниць та підтримки функціонування системи без потреби у центральному органі керування. Основна особливість криптовалюти полягає в тому, що вона не залежить від жодної центральної влади, такої як банк або уряд, що дає більшу свободу і контроль користувачам.

1.2.2 Опис технології блокчейн

Блокчейн є технологічною основою багатьох криптовалютних систем. Він представляє собою розподілену базу даних, яка зберігає всі транзакції, здійснені з криптовалютою. Блокчейн дозволяє створювати послідовні ланцюжки блоків, кожен з яких містить криптографічно підписані дані про транзакції. Це забезпечує надійність, цілісність і недоступність для змін блоків, що робить систему відкритою та надійною.

1.2.3 Основні принципи криптовалютних систем

До основних принципів криптовалютних систем можна віднести наступні:

- 1) децентралізація та відсутність центрального владу: криптовалюти працюють на принципі децентралізації, що означає, що вони не мають центрального органу керування або влади, рішення приймаються спільнотою учасників за допомогою консенсусу, що забезпечує демократичний контроль над системою;
- 2) анонімність та приватність транзакцій: криптовалюти дозволяють користувачам здійснювати транзакції з певним рівнем анонімності та приватності, вони використовують криптографію для захисту особистої інформації та забезпечення конфіденційності користувачів;
- 3) безпека та надійність криптовалют: криптовалюти мають високий рівень безпеки, завдяки застосуванню сучасних криптографічних протоколів та алгоритмів, вони використовують публічний ключовий криптографічний метод для підпису транзакцій, що гарантує цілісність та автентичність даних.

1.2.4 Особливості криптовалюти BTC

Криптовалюта BTC, що відома як Bitcoin, є першою та найбільш відомою криптовалютою у світі. Ось детальніше про особливості цієї криптовалюти:

- 1) перша криптовалюта: Bitcoin був першою криптовалютою, яка була запущена в 2009 році, він відіграв важливу роль у розвитку криптовалют та блокчейн-технології, ставши піонером у цій галузі;

- 2) децентралізованість: Bitcoin є децентралізованою криптовалютою, що означає, що він не підпорядковується жодному центральному органу або уряду, він працює на основі технології блокчейн, де транзакції підтверджуються мережею користувачів (майнерів);
- 3) обмежена кількість монет: відмінністю Bitcoin є обмежена кількість монет, що можуть бути створені (за дизайном, загальна кількість Bitcoin обмежена до 21 мільйона монет, що робить його рідкісним та цінним).
- 4) анонімність і приватність: хоча Bitcoin забезпечує певний рівень анонімності, він не є повністю анонімним, усі транзакції в мережі Bitcoin є публічно доступними, інформація про них записується в блокчейн (проте використання псевдонімів та додаткових заходів може допомогти зберегти конфіденційність користувача);
- 5) висока безпека: безпека Bitcoin забезпечується шифруванням та криптографічними принципами, кожна транзакція підписується цифровим підписом, що дозволяє перевірити її автентичність; також використовується система "доказу роботи" (Proof-of-Work), яка вимагає великої обчислювальної потужності для майнінгу нових блоків;
- 6) міжнародні перекази: Bitcoin дозволяє швидкі та недорогі міжнародні перекази коштів (оскільки він не залежить від банків та проміжних установ, операції можуть бути здійснені без обмежень та затримок).

1.3 Ф'ючерси

1.3.1 Визначення ф'ючерсів

Ф'ючерси – це тип похідного фінансового інструменту, який встановлює зобов'язання купити або продати актив (товар, валюту, фінансовий індекс, цінні папери тощо) за певною ціною (відомою як ціна ф'ючерсного

контракту) в певний майбутній момент часу. Ф'ючерси визначаються та узгоджуються на спеціалізованих біржах, де вони можуть бути торговані.

Ф'ючерсний контракт складається з декількох основних елементів:

- 1) актив, який буде куплений або проданий в майбутньому, це може бути будь-який актив, який може бути визначений та оцінений;
- 2) ціна ф'ючерсного контракту, за яку буде куплено або продано актив у майбутньому, вона встановлюється на момент укладення контракту та є обов'язковою для виконання;
- 3) майбутній момент часу: дата, коли фактично відбудеться купівля або продаж активу (цей термін може бути будь-яким визначенім часом в майбутньому);
- 4) обсяг контракту: це кількість активу, яка має бути куплена або продана в рамках ф'ючерсного контракту, обсяг може бути визначений умовами контракту або стандартом на відповідній біржі.

1.3.2 Роль ф'ючерсів у фінансових ринках

Ф'ючерси відіграють наступні ролі у фінансових ринках:

- 1) цінова відкритість та прозорість: ф'ючерси дозволяють встановлювати ціни активів на майбутній час, що створює цінову прозорість і відкритість, оскільки ціни ф'ючерсів можуть відображати прогнози та очікування ринку;
- 2) захист від цінових ризиків: ф'ючерси надають можливість захистити себе від змін цін на активи (якщо ви володієте активом і боїтесь зростання або падіння його ціни, ви можете укласти ф'ючерсний контракт, щоб зафіксувати ціну активу на майбутній час);
- 3) спекуляція та прибуткові можливості: ф'ючерси також надають можливість спекулювати на ціновій різниці активів, торгування

ф'ючерсами дозволяє інвесторам заробляти прибуток на зміні цін активів без необхідності володіння самим активом;

- 4) ліквідність та доступність: ф'ючерсні контракти торгуються на офіційних біржах, що забезпечує велику ліквідність та доступність для трейдерів (це означає, що учасники ринку можуть швидко купувати і продавати ф'ючерси без проблем).

Ф'ючерсні контракти використовуються в різних фінансових ринках, включаючи товарні ринки (наприклад, нафту, золото), фондові ринки, валютні ринки та індексні ринки. Вони грають важливу роль у забезпечені ліквідності, зниженні цінових ризиків та створенні можливостей для трейдерів та інвесторів.

1.3.3 Типи ф'ючерсів

Ф'ючерси можуть бути квартальними і перпетуальними

1. Квартальні ф'ючерси – це тип ф'ючерсних контрактів, у яких дата виконання контракту припадає на певний квартал року. Такі контракти мають фіксовану дату закінчення, яка зазвичай є останнім днем в місяці, що відповідає певному кварталу (наприклад, березень, червень, вересень, грудень). Квартальні ф'ючерси дозволяють трейдерам спрогнозувати ціну активу на певний квартал у майбутньому. Основна відмінність квартальних ф'ючерсів полягає в тому, що вони мають обмежений термін дії і закінчуються після встановленої дати. Після закінчення квартального ф'ючерсу можуть бути введені нові контракти на наступний квартал.
2. Перпетуальні ф'ючерси – це тип ф'ючерсних контрактів, які не мають встановленої дати закінчення. Вони тривають нескінченно довго (протягом "безстрокового" періоду). У таких ф'ючерсах дата виконання

контракту не настає, але вони все ж використовуються для торгівлі. Особливість перпетуальних ф'ючерсів полягає в тому, що вони використовують механізм фінансування для збереження ціни контракту близькою до ціни референтного активу. Цей механізм включає в себе обмін процентних ставок між трейдерами, які відкривають позиції на ринку ф'ючерсів.

1.3.4 Механізм фінансування для перпетуальних ф'ючерсів

Механізм фінансування у перпетуальних ф'ючерсах використовується для забезпечення цінової конвергенції між ф'ючерсним контрактом та реальною ціною базового активу (наприклад, криптовалюти). Він забезпечує вирівнювання ціни ф'ючерсу з реальною ціною активу на ринку.

Механізм фінансування в перпетуальних ф'ючерсах використовує додатковий елемент, відомий як "фінансовий розрахунок", який відбувається між короткою і довгою позиціями трейдерів на основі різниці між ціною ф'ючерсу та референтною ціною (наприклад, ціною на ринку spot).

Якщо ціна ф'ючерсу вища за референтну ціну, тоді коротка позиція (трейдер, який продав ф'ючерс) зобов'язаний сплатити фінансовий розрахунок довгої позиції (трейдер, який купив ф'ючерс). Цей розрахунок підвищує вартість короткої позиції і компенсує різницю між ціною ф'ючерсу та референтною ціною.

Зворотно, якщо ціна ф'ючерсу нижча за референтну ціну, тоді коротка позиція отримує фінансовий розрахунок від довгої позиції, що знижує вартість короткої позиції і компенсує різницю між ціною ф'ючерсу та референтною ціною.

Механізм фінансування регулярно виконується на основі фіксованого інтервалу часу, наприклад, кожні 8 годин. Це забезпечує постійну цінову конвергенцію між ф'ючерсним контрактом та реальною ціною активу.

Отже, механізм фінансування в перпетуальних ф'ючерсах допомагає зберігати цінову стабільність та забезпечує трейдерам можливість отримувати прибуток з цінових різниць між ф'ючерсом та референтною ціною. Він є важливою складовою для трейдерів, які використовують перпетуальні ф'ючерси для управління активами на ринку криптовалют.

1.3.5 Торгове плече в ф'ючерсних контрактах

Торгове плече є важливим аспектом ф'ючерсних контрактів і використовується для збільшення потенційних прибутків або ризиків трейдера. Воно дозволяє трейдерам використовувати позики для здійснення угоди на більшу кількість активів, ніж їх доступний капітал.

Поняття торгового плеча визначається як співвідношення між розміром позики, наданої брокером або біржею, і власним капіталом трейдера. Зазвичай воно представляється у вигляді співвідношення або множника, наприклад, 1:10 або 10x, що означає, що трейдер може укласти угоду на 10 разів більшу кількість активів, ніж його власний капітал.

Використання торгового плеча має два основних аспекти – збільшення потенційних прибутків і збільшення ризиків. За рахунок використання позикових коштів, трейдер може отримати більше прибутку від навіть незначних рухів цін активів. Наприклад, при використанні плеча 1:10, рух ціни активу на 1% може привести до збільшення прибутку на 10%.

Однак, важливо пам'ятати, що разом зі збільшеними потенційними прибутками, торгове плече також збільшує ризики. Якщо ціна активу рухається протилежно до прогнозу трейдера, втрати також збільшуються

відповідно до множника торгового плеча. Тому трейдерам потрібно бути уважними і враховувати ризики, пов'язані з використанням великого торгового плеча.

Торгове плече є інструментом, який дозволяє трейдерам збільшити свій потенційний прибуток, але вони також повинні бути готовими до можливих збитків. Розумне управління ризиками та обережне використання торгового плеча є важливими аспектами успішної торгівлі ф'ючерсами.

1.3.6 Ліквідація ф'ючерсних позицій

При торгівлі з використанням плеча (також відомого як маржин–трейдинг), трейдер має можливість використовувати позичені кошти від біржі або брокера для збільшення своєї торговельної позиції. Це дозволяє трейдеру отримувати більший потенційний прибуток при невеликому власному капіталі, але також приносить певні ризики.

Одним з ризиків, пов'язаних з торгівлею з плечем, є можливість швидкої ліквідації позиції при зміні ціни активу в протилежну сторону. Коли ціна активу рухається проти трейдера, його маржевий рахунок може стати недостатнім для покриття збитків, що викликає автоматичну ліквідацію його позиції.

Механізм ліквідації полягає у продажу або купівлі активу трейдером за поточного ринковою ціною з метою погашення заборгованості перед біржею або брокером. Якщо ціна рухається швидко і трейдер не має достатнього часу або можливості поповнити свій маржевий рахунок, його позиція буде автоматично ліквідована для запобігання подальшим збиткам.

Це означає, що трейдер може втратити всю свою власну і позичену суму на маржевому рахунку при несприятливих рухах цін. Ліквідація стає

захисним механізмом для біржі або брокера, але при цьому трейдер понесе втрати.

Важливо ретельно керувати ризиками та використовувати ефективні стратегії управління грошима при торгівлі з плечем, щоб уникнути непередбачуваних ліквідацій та збитків. Трейдерам рекомендується вивчати та розуміти умови торгівлі з плечем, маржін–вимоги та правила біржі або брокера, щоб уникнути негативних наслідків.

1.4 Висновки до розділу 1

У цьому розділі було проведено детальний огляд предметної області, пов'язаної з управлінням активами на ринку криптовалют з урахуванням оцінки довіри користувачів.

Починаючи з розгляду загальних аспектів біржі, були визначені її сутність, роль у фінансовій системі та основні функції. Також були розглянуті різні типи бірж, зокрема криптовалютні, з підкресленням особливостей криптовалютної біржі Binance та її переваг для розглянутої задачі.

Далі, у пункті 2, була проаналізована криптовалюта як основний актив на ринку криптовалют. Було детально розглянуто її сутність, особливості та роль блокчейну у криптовалютних системах. Основні принципи криптовалютних систем, такі як децентралізація, анонімність та безпека, були також висвітлені.

У пункті 3 було досліджено ф'ючерси як фінансові інструменти. Визначено їх сутність та роль у фінансових ринках. Були розглянуті два типи ф'ючерсів – квартальні та перпетуальні, і виявлено їх відмінності та особливості. Були наведені приклади криптовалютних ф'ючерсів. Нарешті, були досліжені механізми фінансування в перпетуальних ф'ючерсах та роль

торгового плеча в ф'ючерсних контрактах. Також була розглянута ліквідація ф'ючерсних позицій та її ризики для трейдерів.

Загальною метою розділу було розглянути основні аспекти біржової та криптовалютної діяльності, ф'ючерсів та їх особливостей. Ці особливості є важливими для більш детального розуміння предметної галузі. Розуміння цих особливостей є необхідним при обґрунтуванні деяких прийнятих в подальшій роботі рішень.

РОЗДІЛ 2 ОГЛЯД МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ЗАДАЧІ ПЕРЕДБАЧЕННЯ ЧАСОВИХ РЯДІВ

2.1 Авторегресійні моделі

2.1.1 AR (Авторегресія)

Авторегресійна модель (AR) базується на ідеї, що майбутні значення часового ряду залежать від попередніх значень ряду. У моделі AR використовується попередній часовий шаг або кілька попередніх шагів для передбачення майбутніх значень. Параметр моделі AR вказує, скільки попередніх шагів враховується. Зазвичай для побудови моделі AR використовують методи оцінки, такі як метод найменших квадратів.

2.1.2 MA (Середня ковзна)

Модель середньої ковзної (MA) використовує середнє значення попередніх спостережень для передбачення майбутніх значень часового ряду. У моделі MA враховуються лише попередні значення, а не їх послідовність, як у моделі AR. Параметр моделі MA вказує, скільки попередніх значень використовується для обчислення середнього значення.

2.1.3 ARMA (Авторегресія середньої ковзної)

Модель авторегресії середньої ковзної (ARMA) поєднує в собі як авторегресійну компоненту (AR), так і компоненту середньої ковзної (MA). Ця модель використовує попередні значення часового ряду, а також середнє значення попередніх значень для передбачення майбутніх значень.

2.1.4 ARIMA (Авторегресія інтегрована середня ковзна)

Модель авторегресії інтегрованої середньої ковзної (ARIMA) включає в себе компоненту інтегрованості, що дозволяє моделі управляти трендом або стаціонарністю часового ряду. ARIMA модель використовує попередні значення, середнє значення та різницю між спостереженнями для передбачення майбутніх значень.

2.1.5 Висновок про авторегресійні моделі

Ці авторегресійні моделі є популярними методами для передбачення часових рядів. Кожна модель має свої властивості та переваги в залежності від характеристик ряду. Наприклад, модель AR ефективна для передбачення рядів з помірним трендом, тоді як ARIMA добре працює з рядами, що мають тренд та сезонність. Вибір певної моделі залежить від аналізу даних та їх властивостей.

2.2 Експоненційно зважені моделі

2.2.1 SES (Експоненційне згладжування просте)

Експоненційне згладжування просте (SES) є простою моделлю, яка використовує експоненційно зважені середні для передбачення майбутніх значень часового ряду. У моделі SES кожному спостереженню призначається вага, яка зменшується експоненційно залежно від віддаленості часових шагів. Це дозволяє моделі більш чутливо реагувати на останні значення ряду.

Формула для SES:

$$\hat{y}_{t+1} = \alpha \cdot y_t + (1 - \alpha) \cdot \hat{y}_t$$

Де \hat{y}_{t+1} – прогнозоване значення на часовому кроці $(t + 1)$,
 y_t – спостережене значення на часовому кроці (t) ,
 \hat{y}_t – прогнозоване значення на часовому кроці (t) ,
 α – параметр згладжування, де $(0 \leq \alpha \leq 1)$.

2.2.2 Holt–Winters (Тренд і сезонність)

Модель Хольта–Вінтерса (Holt–Winters) використовується для передбачення часових рядів, які мають явний тренд і сезонність. Ця модель включає в себе компоненту тренду, яка враховує зміну значень ряду з часом, а також компоненту сезонності, яка моделює регулярні коливання ряду відповідно до сезонних патернів. Модель Хольта–Вінтерса використовує експоненційно зважені середні для оцінки тренду та сезонності та передбачення майбутніх значень.

$$\hat{y}_{t+m} = (l_t + m \cdot b_t) \cdot s_{t-m+1+(k \bmod m)}$$

$$l_t = \alpha \cdot (y_t / s_{t-m}) + (1 - \alpha) \cdot (l_{t-1} + b_{t-1})$$

$$b_t = \beta \cdot (l_t - l_{t-1}) + (1 - \beta) \cdot b_{t-1}$$

$$s_t = \gamma \cdot (y_t / (l_{t-1} + b_{t-1})) + (1 - \gamma) \cdot s_{t-m}$$

де \hat{y}_{t+m} – прогнозоване значення на часовому кроці $(t + m)$,
 l_t – рівень на часовому кроці (t) ,
– тренд на часовому кроці (t) ,
 s_t – сезонна складова на часовому кроці (t) ,
 y_t – спостережене значення на часовому кроці (t) ,
 m – довжина сезонного циклу,
 k – цілочисельна змінна, яка вказує на поточний сезон.

2.2.3 Висновок про експоненційно зважені моделі

Експоненційно зважені моделі є ефективними інструментами для передбачення часових рядів, особливо коли враховується тренд або сезонність. Модель SES є простою та легко інтерпретованою, а Holt–Winters дозволяє додатково моделювати тренд і сезонність. Вибір конкретної моделі залежить від характеристик ряду та його особливостей.

2.3 Рекурентні нейронні мережі (RNN)

2.3.1 LSTM (Довготривала краткосрочна пам'ять)

LSTM (Довготривала краткосрочна пам'ять) є одним з найпопулярніших типів RNN. Вона вирішує проблему зникнення градієнту, що часто виникає при тренуванні глибоких нейронних мереж. LSTM має спеціальні блоки пам'яті, які можуть запам'ятувати і забувати інформацію залежно від входних даних та поточного стану мережі. Це дозволяє LSTM зберігати довгострокові залежності в часових рядах.

2.3.2 GRU (Одинична управлююча рекурентна одиниця)

GRU (Одинична управлююча рекурентна одиниця) є іншим типом RNN, який також вирішує проблему зникнення градієнту. Вона використовує меншу кількість параметрів порівняно з LSTM, що робить її більш ефективною для навчання на великих наборах даних. GRU має гейти, які вирішують, яку інформацію треба передавати і яку треба забувати в кожному кроці часу.

2.3.3 Bidirectional LSTM (Двостороння LSTM)

Bidirectional LSTM (Двостороння LSTM) є комбінацією LSTM та двосторонньої моделі, яка дозволяє враховувати як минулі, так і майбутні контексти при прогнозуванні значень. Це дозволяє моделі отримувати

інформацію з обох напрямків і покращує її здатність до передбачення часових рядів.

2.3.4 Висновок про прогнозуючі моделі на основі дерев рішень

Рекурентні нейронні мережі, такі як LSTM, GRU та Bidirectional LSTM, є потужними моделями для аналізу та передбачення часових рядів. Вони добре підходять для моделювання послідовних залежностей та врахування довгострокових залежностей в даних. Застосування рекурентних нейронних мереж дозволяє отримати точні прогнози та враховувати контекст з обох напрямків.

2.4 Прогнозуючі моделі на основі дерев рішень

2.4.1 Random Forest (Випадковий ліс)

Random Forest (Випадковий ліс) – це ансамблева модель дерев рішень, де кожне дерево рішень тренується на випадковому під–наборі даних та випадковому під–наборі ознак. Коли потрібно зробити прогноз, кожне дерево вносить свій внесок, а потім результати агрегуються для отримання остаточного прогнозу. Random Forest добре справляється з уникненням перенавчання та здатна враховувати важливість ознак для передбачення.

2.4.2 Gradient Boosting (Градієнтний бустинг)

Gradient Boosting (Градієнтний бустинг) є іншим ансамблевим методом, де дерева рішень будуються послідовно, кожне на основі попереднього. Він використовує градієнтний спуск для пошуку оптимальних параметрів дерева. Кожне нове дерево спрямовується на виправлення помилок, зроблених попередніми деревами. Градієнтний бустинг зазвичай має високу точність прогнозу та може враховувати складні залежності в даних.

2.4.3 Висновок про прогнозуючі моделі на основі дерев рішень

Прогнозуючі моделі на основі дерев рішень є популярними методами для передбачення часових рядів. Вони використовують алгоритми дерев рішень, які розбивають дані на багато різних гілок на основі різних ознак і приймають рішення на основі цих гілок.

Такі моделі як Random Forest і Gradient Boosting, є ефективними методами для передбачення часових рядів. Вони враховують важливість ознак та здатні розпізнавати складні залежності в даних. Використання ансамблевих моделей, які комбінують декілька дерев рішень, може покращити точність прогнозу та зменшити ризик перенавчання.

2.5 Згорткові нейронні мережі (CNN)

2.5.1 1D CNN (Одновимірна згорткова нейронна мережа)

1D CNN (Одновимірна згорткова нейронна мережа) – це тип згорткових нейронних мереж, які використовуються для аналізу одновимірних даних, таких як часові ряди. Вони використовують фільтри, які здійснюють згортку через вхідні дані, щоб виділити важливі особливості та шаблони. Після згортки виконується пулінг (зведення), що дозволяє знизити розмірність даних. Потім отримані ознаки подаються на повнозв'язний шар для зроблення прогнозу.

2.5.2 Висновок про згорткові нейронні мережі

Згорткові нейронні мережі (Convolutional Neural Networks, CNN) є потужними моделями машинного навчання, які здатні до виявлення просторових та локальних залежностей у вхідних даних. Хоча вони в основному використовуються для обробки зображень, вони також можуть бути застосовані до аналізу часових рядів.

Згорткові нейронні мережі (CNN), зокрема 1D CNN, можуть бути ефективними моделями для передбачення часових рядів. Вони здатні виявляти шаблони, тренди та залежності в часових даних і забезпечувати точні прогнози. Згорткові нейронні мережі мають гнуочку архітектуру, що дозволяє їм адаптуватися до різних видів часових рядів та задач прогнозування. Використання згорткових нейронних мереж дозволяє вирішувати складні задачі передбачення в області часових рядів.

2.6 Метрики для задачі регресії

Вкажемо основні метрики для задачі регресії.

1. Середня абсолютна помилка (Mean Absolute Error, MAE): Ця метрика обчислює середнє значення абсолютнох відхилень між прогнозованими та справжніми значеннями. Вона вимірює абсолютну величину помилки і не враховує їхній напрямок.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

2. Середня квадратична помилка (Mean Squared Error, MSE): Ця метрика обчислює середнє значення квадратів відхилень між прогнозованими та справжніми значеннями. Вона вимірює середню величину квадратичної помилки.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

3. Корінь середньоквадратичної помилки (Root Mean Squared Error, RMSE): Це квадратний корінь з MSE. Використовується для вимірювання середнього значення квадратичної помилки в оригінальних одиницях вимірювання.

$$\text{RMSE} = \sqrt{\text{MSE}}$$

4. Коефіцієнт детермінації (Coefficient of Determination, R-squared): Ця метрика вимірює відповідність між прогнозованими та справжніми значеннями в межах 0 і 1. Вона вказує на частку дисперсії цільової

змінної, яка може бути пояснена моделлю. Значення близькі до 1 вказують на кращу точність прогнозу, тоді як значення близькі до 0 означають незначну або відсутню прогностичну здатність моделі.

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

Ці метрики допомагають оцінити якість прогнозування моделі для задачі передбачення часових рядів. Вибір конкретної метрики залежить від конкретних вимог та особливостей досліджуваної проблеми.

2.7. Градієнтний метод

Градієнт функції $f(\mathbf{x})$ позначається як $\nabla f(\mathbf{x})$ або $\text{grad}(f(\mathbf{x}))$. Він представляє собою вектор, в якому кожна компонента є похідною функції $f(\mathbf{x})$ по відповідному параметру. Наприклад, якщо у нас є функція $f(x_1, x_2)$, градієнт матиме вигляд:

$$\nabla f(x_1, x_2) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right)$$

Розглянемо кроки градієнтного методу.

1. Ініціалізація: Спочатку ми вибираємо початкове значення параметрів \mathbf{x}_0
2. Обчислення градієнта: Ми обчислюємо градієнт функції $f(\mathbf{x})$ в поточній точці \mathbf{x} . Це виконується, обчислюючи похідні функції по кожному параметру. Градієнт можна записати у векторній формі:

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right)$$

3. Оновлення параметрів: Ми оновлюємо значення параметрів, рухаючись в напрямку, протилежному до градієнта. Це робиться шляхом віднімання деякого кроку (кроку навчання або learning rate), помноженого на градієнт, від поточного значення параметрів. Оновлення можна записати наступним чином:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k),$$

де \mathbf{x}_k – поточне значення параметрів,
 α – крок навчання.

4. Перевірка умови зупинки: Ми перевіряємо певну умову зупинки, яка може включати, наприклад, обмеження на кількість ітерацій, дотичність або норму градієнта. Якщо умова зупинки не виконується, ми повертаємося до кроку 2 і продовжуємо ітерації.
5. Завершення: Коли умова зупинки виконується, ми вважаємо, що ми знайшли оптимальне значення параметрів \mathbf{x} .

Градієнтний метод є ітеративним процесом, який може бути повільним, особливо для функцій з вузькими або кривими мінімумами, або якщо крок навчання погано підібраний. Існує також багато модифікацій градієнтного методу, таких як стохастичний градієнтний метод, метод з моментом та інші, які дозволяють прискорити його збіжність та покращити продуктивність.

2.8. Складові частини нейронних мереж

2.8.1. Dense (або повнозв'язний) шар

Dense шар є одним з основних типів шарів у нейронних мережах. Він складається з набору нейронів, кожен з яких пов'язаний з усіма нейронами з попереднього шару. Кожен нейрон у Dense шарі приймає вхідні сигнали від усіх нейронів попереднього шару і виробляє вихідний сигнал, який передається наступному шару.

Dense шар працює на градієнтному методі, зокрема на алгоритмі зворотнього поширення помилки (backpropagation). У процесі тренування нейронної мережі, використовуючи зворотне поширення помилки, градієнти обчислюються з останнього шару назад до першого шару, що дозволяє змінювати ваги шарів, щоб зменшити помилку мережі.

Початкові ваги у Dense шарі зазвичай ініціалізуються випадковим чином. Існує кілька поширених методів ініціалізації ваг, таких як:

- випадкова ініціалізація: ваги ініціалізуються випадковими числами з певним розподілом, наприклад, з нормальним розподілом або рівномірним розподілом;
- ініціалізація Глорота (Xavier): ваги ініціалізуються випадковими числами з нормальним розподілом, в якому середнє значення залежить від кількості входів та виходів з нейрона;
- ініціалізація Хе (He): Ваги ініціалізуються випадковими числами з нормальним розподілом, в якому середнє значення залежить від кількості входів нейрому.

Ініціалізація початкових ваг є важливим етапом в тренуванні нейронної мережі, оскільки вона може вплинути на швидкість збіжності та якість моделі. Розумний вибір методу ініціалізації може поліпшити результати навчання мережі.

Нехай у нас є Dense шар з n нейронів у попередньому шарі та m нейронів у поточному шарі. Вага w_{ij} з'єднує нейрон i попереднього шару з нейроном j поточного шару. Позначимо вхідні сигнали до нейронів поточного шару як x_j , а активації нейронів як a_j . Тоді активація нейрону j поточного шару обчислюється як зважена сума вхідних сигналів з попереднього шару, залежно від ваг та зсуву:

$$a_j = \sum_{i=1}^n w_{ij}x_i + b_j,$$

де b_j – це зсув (bias) нейрону j .

2.8.2. Активаційні функції

Активаційні функції є важливою складовою нейронних мереж і використовуються для нелінійного перетворення вихідних сигналів нейронів. Вони дозволяють нейронній мережі моделювати складні нестационарні залежності між вхідними та вихідними даними. Різні активаційні функції мають різні властивості та використовуються залежно від конкретної задачі.

Ось деякі найпопулярніші активаційні функції.

1. Сигмоїда (Sigmoid) (рис. 2.1):

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

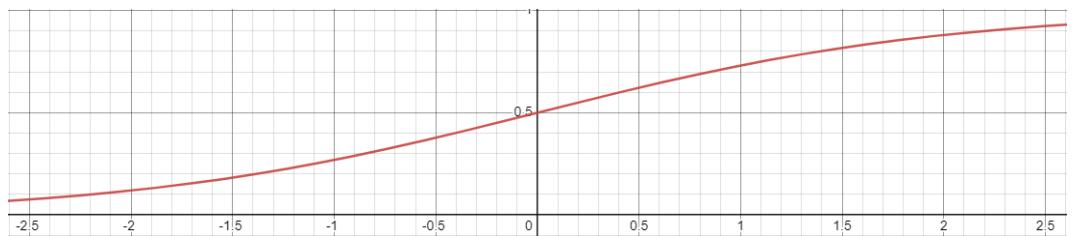


Рисунок 2.1 – Графік сигмоїди

Сигмоїда приймає будь-яке число і зображує його в діапазоні від 0 до 1. Вона має форму "S"-подібної кривої і використовується в задачах бінарної класифікації або як активаційна функція у внутрішніх шарах нейронних мереж. Сигмоїда може приводити до проблеми вивантаження градієнту (vanishing gradient problem), оскільки при великих абсолютних значеннях вхідного сигналу її похідна стає дуже мала.

2. ReLU (Rectified Linear Unit) (рис. 2.2):

$$ReLU(x) = \max(0, x)$$

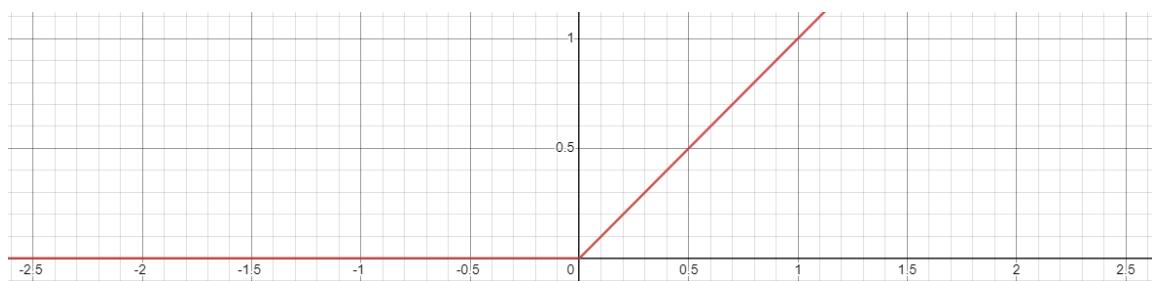


Рисунок 2.2 – Графік ReLU

ReLU функція повертає вхідний сигнал, якщо він більше за нуль, а в іншому випадку повертає нуль. Вона має просту форму та відсутність насиченості, що робить її більш ефективною для навчання нейронних мереж. ReLU широко використовується у багатошарових мережах та допомагає уникнути проблеми вивантаження градієнту. Проте, вона може виключати частину нейронів, які стають неактивними (dead neurons), коли вхідний сигнал менше або дорівнює нулю.

3. Гіперболічний тангенс (Tanh) (рис.2.3):

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

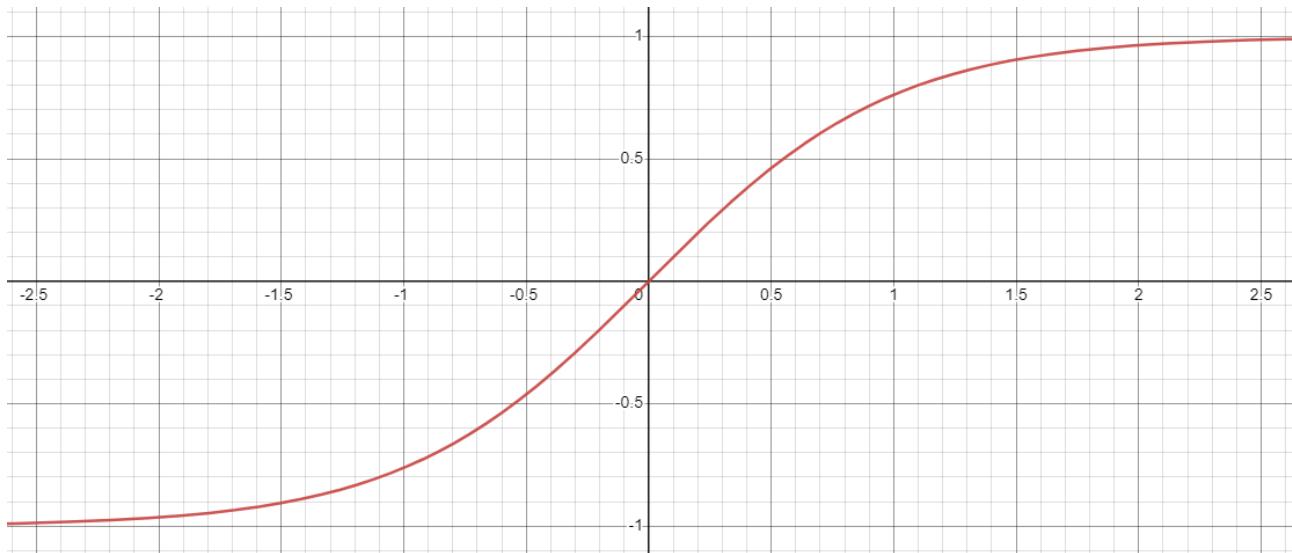


Рисунок 2.3 – Графік гіперболічного тангенса

Гіперболічний тангенс приймає будь-яке число і зображує його в діапазоні від -1 до 1. Вона подібна до сигмоїди, але має нульове середнє значення, що допомагає зберігати більшу інформацію в градієнті. Гіперболічний тангенс використовується у шарах нейронних мереж для отримання нелінійності та зменшення масштабу сигналу.

Активаційні функції дозволяють нейронній мережі набувати нелінійність та складні залежності між вхідними та вихідними даними. Вони грають важливу роль у проміжних шарах мережі, де вони допомагають виробляти відповідні вихідні сигнали на основі вхідних сигналів та ваг.

2.8.3. Dropout

Dropout є одним із методів регуляризації, що застосовується у нейронних мережах з метою запобігання перенавчанню (overfitting). Він випадковим чином "вимикає" (викидає) деякі нейрони в мережі на етапі навчання. Ідея полягає в тому, що мережа навчається з різними підмножинами нейронів на кожному кроці навчання, що дозволяє усереднювати прогнози з

різних комбінацій нейронів та знижує кореляцію між нейронами. Це забезпечує більш загальну і надійну роботу мережі та покращує її узагальнючу здатність.

Введемо індикатор випадкової змінної для кожного нейрона. Нехай d_i буде індикатором, який приймає значення 0 або 1, де 1 означає, що нейрон i активний (не викинутий), а 0 означає, що нейрон i вимкнутий (викинутий). Ймовірність викидання нейрона визначається параметром p , який вказує, який відсоток нейронів буде вимкнутий. Кожен нейрон має незалежну ймовірність p бути викинутим або залишитися активним.

Модифікована активація нейрона з використанням Dropout може бути описана наступною формулою:

$$a_i = \begin{cases} \frac{x_i}{p}, & d_i = 1 \\ 0, & d_i = 0 \end{cases},$$

де x_i – вхідний сигнал до нейрона i ,

a_i – модифікована активація нейрона i .

Під час навчання, на кожній ітерації, випадковим чином обираються значення d_i для кожного нейрона. На етапі передачі (тестування) Dropout не застосовується, але вихід активованих нейронів масштабується шляхом множення на ймовірність p .

Dropout допомагає уникнути перенавчання шляхом запобігання залежності між нейронами, сприяє роботі нейронної мережі з ширшим набором можливих конфігурацій та поліпшує її загальну узагальнючу здатність.

2.8.4. BatchNormalization

Batch Normalization є методом, який використовується у нейронних мережах з метою стабілізації та прискорення процесу навчання. Він нормалізує активації шляхом центрування та масштабування значень в батчі навчання. Це допомагає знизити варіацію активацій та полегшує передачу градієнту через мережу.

Для кожного нейрона у батчі обчислюються середнє значення та стандартне відхилення активацій. Далі, за допомогою цих статистичних значень, активації нормалізуються шляхом центрування та масштабування.

1. Обчислення середнього значення:

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i$$

де μ_B – середнє значення в батчі,

m – розмір батчу,

x_i – активація нейрона в батчі.

2. Обчислення стандартного відхилення:

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 ,$$

де σ_B^2 – дисперсія в батчі.

3. Центрування та масштабування активацій:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} ,$$

де \hat{x}_i – нормалізована активація нейрона i ,
 ϵ – малий додаток для захисту від ділення на нуль.

4. Масштабування та зсув активацій:

$$y_i = \gamma \hat{x}_i + \beta,$$

де y_i – остаточна активація нейрона i ,
 γ – параметр масштабування,
 β – параметр зсуву.

Batch Normalization дозволяє нейронним мережам більш стабільно та ефективно навчатися, поліпшує здатність мережі до узагальнення та допомагає уникнути проблем, пов'язаних з вибухом/зникненням градієнту під час навчання.

2.8.5. Регуляризація

L1–регуляризація, також відома як регуляризація з використанням L1–норми, включає до функції втрат штрафний член, який пропорційний сумі абсолютних значень параметрів моделі. Цей штраф сприяє зменшенню значень параметрів та розріджує модель, тобто деякі параметри можуть стати точно нульовими.

Формула для L1–регуляризації:

$$loss_{reg} = loss + \lambda \sum_{i=1}^N |\theta_i|,$$

де λ – параметр регуляризації,
 θ_i – параметр моделі,
 N – загальна кількість параметрів моделі.

L2–регуляризація, також відома як регуляризація з використанням L2–норми або регуляризація з використанням ваги фронту. Вона полягає в додаванні штрафного члена до функції втрат, який пропорційний сумі квадратів значень параметрів моделі. Цей штраф сприяє зменшенню величини параметрів і полегшує навчання моделі.

Формула для L2–регуляризації:

$$loss_{reg} = loss + \lambda \sum_{i=1}^N \theta_i^2,$$

де λ – параметр регуляризації,
 θ_i – параметр моделі,
 N – загальна кількість параметрів моделі.

Обидва методи регуляризації (L1 та L2) допомагають уникнути перенавчання та покращити загальну узагальнюючу здатність моделі. Вибір конкретного методу залежить від властивостей даних та вимог до моделі.

2.8.6. Оптимізатори

Оптимізатори в машинному навчанні використовуються для оновлення параметрів моделі з метою зменшення функції втрат та покращення роботи моделі. Одним з найефективніших і популярних оптимізаторів є Adam (Adaptive Moment Estimation).

Adam поєднує метод моментів та адаптивну швидкість навчання, що робить його ефективним для швидкого збіжності та обробки різноманітних

типів даних. Він використовує експоненційно зважену середню квадратів градієнтів для оновлення параметрів.

Формули для Adam оптимізатора:

1. Оновлення моменту першого порядку (середнього градієнту):

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t,$$

де m_t – момент першого порядку на кроці t ,
 g_t – градієнт на кроці t ,
 β_1 – експоненційні затухання для моменту першого порядку (зазвичай встановлюється на значення 0.9).

2. Оновлення моменту другого порядку (середнього квадрату градієнту):

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2,$$

де v_t – момент другого порядку на кроці t ,
 g_t – градієнт на кроці t ,
 β_2 – експоненційний затухання для моменту другого порядку (зазвичай встановлюється на значення 0.999).

3. Коригування моменту першого порядку:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t},$$

4. Коригування моменту другого порядку:

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t},$$

5. Оновлення параметрів:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t,$$

де θ_t – параметри на кроці t ,

η – швидкість навчання,

ϵ – невелике число для недопущення ділення на нуль.

Ці формули дозволяють ефективно оновлювати параметри моделі з використанням методу моментів та адаптивної швидкості навчання, що допомагає моделі збігатися до оптимальних значень та покращує її здатність до узагальнення.

Існує багато різних оптимізаторів, кожен з яких має свої особливості та переваги. Ось декілька інших популярних оптимізаторів.

1. Stochastic Gradient Descent (SGD): Це базовий оптимізатор, який оновлює параметри моделі, використовуючи градієнт функції втрат на кожному кроці навчання. Формула оновлення для SGD має вигляд:

$$\theta_t = \theta_{t-1} - \eta \cdot \nabla L(\theta_{t-1}),$$

де θ_t – параметри на кроці t ,

η – швидкість навчання,

$\nabla L(\theta_{t-1})$ – градієнт функції втрат.

2. RMSprop: Цей оптимізатор використовує експоненційно зважену середню квадратів градієнтів для оновлення параметрів моделі. Він дозволяє адаптивно регулювати швидкість навчання для кожного параметра. Формула оновлення для RMSprop має вигляд:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{v_t + \epsilon}} \cdot \nabla L(\theta_{t-1}),$$

де θ_t – параметри на кроці t ,

η – швидкість навчання,

v_t – експоненційно зважена середня квадратів градієнтів.

3. Adagrad: Цей оптимізатор адаптивно регулює швидкість навчання для кожного параметра, враховуючи раніше використані градієнти. Це дозволяє зменшити швидкість навчання для параметрів, які мають більш часті оновлення. Формула оновлення для Adagrad має вигляд:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot \nabla L(\theta_{t-1}),$$

де θ_t – параметри на кроці t ,

η – швидкість навчання,

G_t – сума квадратів градієнтів до кроку t .

4. Adadelta: Цей оптимізатор є розширенням методу Adagrad, який намагається зменшити проблему зменшення швидкості навчання з часом. Він використовує експоненційно зважену середню квадратів змін параметрів та градієнтів. Формули оновлення для Adadelta мають багато кроків та включають в себе поняття аккумулюваних середніх квадратів градієнтів та змін параметрів.

Це лише кілька прикладів оптимізаторів, які використовуються в машинному навчанні. Кожен оптимізатор має свої переваги та використовується в різних ситуаціях залежно від проблеми та типу даних.

2.9 Висновки до розділу 2

У цьому розділі були розглянуті різні методи машинного навчання, які можна використовувати для передбачення часових рядів. Починаючи з авторегресійних моделей, таких як AR, MA, ARMA і ARIMA, ми розглянули їх здатність моделювати залежності в часових рядах на основі попередніх значень.

Далі ми дослідили експоненційно зважені моделі, такі як SES і Holt–Winters, які крім попередніх значень враховують також тренд і сезонність.

Рекурентні нейронні мережі, такі як LSTM, GRU і Bidirectional LSTM, виявилися потужними моделями для передбачення часових рядів, оскільки вони можуть моделювати складні залежності в часі та використовувати попередні значення з великим кількістю кроків назад.

Прогнозуючі моделі на основі дерев рішень, такі як Random Forest і Gradient Boosting, представляють інший підхід до передбачення часових рядів, використовуючи ансамблі дерев для розроблення прогнозів.

Згорткові нейронні мережі, зокрема 1D CNN, є ефективними для виявлення локальних шаблонів в часових рядах, здатними автоматично витягати корисні ознаки зі вхідних даних.

Крім того, ми розглянули важливі метри для задачі регресії, такі як Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) і Coefficient of Determination (R-squared), які дозволяють оцінювати точність прогнозів моделей.

В цьому розділі ми представили широкий спектр методів машинного навчання для передбачення часових рядів, кожен з яких має свої переваги і може бути застосований залежно від конкретних вимог і характеристик даних. Вибір методу залежить від контексту завдання і доступних ресурсів.

РОЗДІЛ 3 СТВОРЕННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1. Налаштування збору даних і бази даних на хмарному сервері

3.1.1 Загальний огляд

За допомогою сервісу OVHcloud у серпні 2022 було арендовано хмарний сервер (рис. 3.1). Доступ до цього серверу був через SSH–тунель. На сервері було розгорнуто базу даних на основі СУБД Mysql. На цьому–ж сервері за допомогою GNU Screen було запущено Python–додаток, який щохвилино відправляє API–запит до Binance API, отримує дані, обробляє їх і заносить оброблені дані до БД. Під час налаштування БД і Python–додатку використовувалось розширення Remote – SSH для Visual Studio Code. Також, було налаштовано щоденні повідомлення в Telegram з короткою інформацією про ціни за останню добу і повідомлення про помилки в роботі додатка.

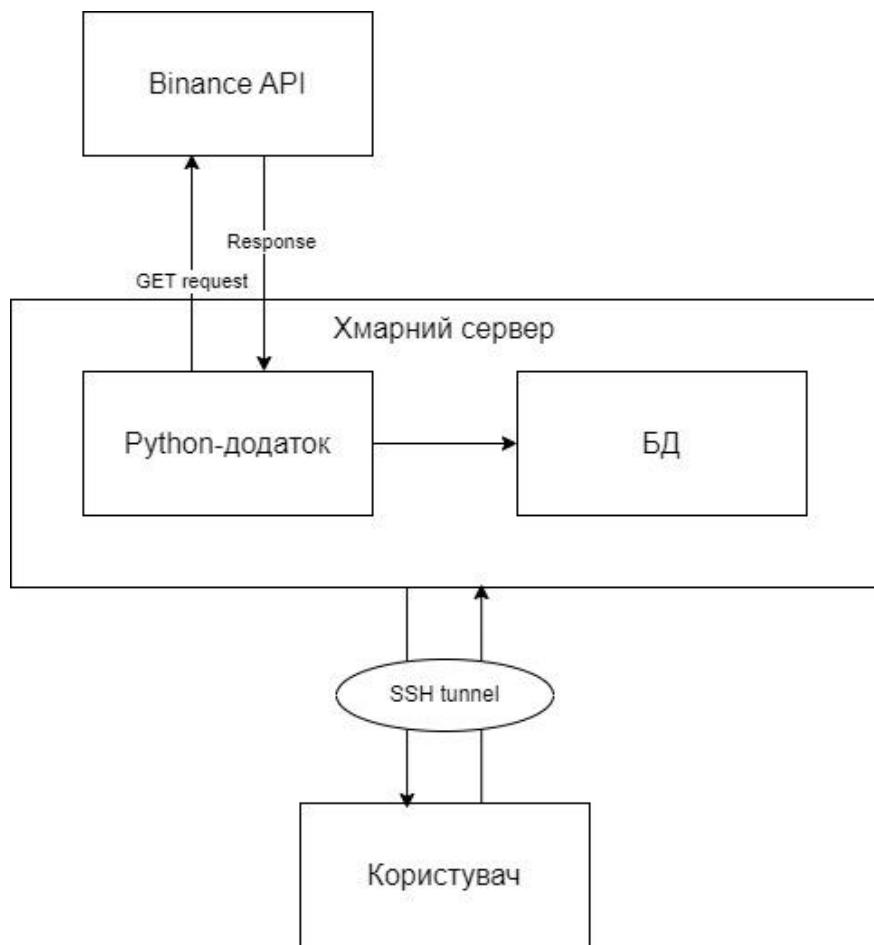


Рисунок 3.1 – Блок–схема організації хмарного сервера

3.1.2 Огляд Python–додатку

Python–додаток містить наступні функції:

- 1) `run_sql_script(script)` – на вхід приймає SQL запит у форматі `string`, повертає список кортежів;
- 2) `connect_ssh_tunnel(ssh_address, ssh_username, ssh_password)` – на вхід приймає дані для SSH тунелю, створює SSH тунель;
- 3) `mysql_connect()` – створює підключення до БД Mysql;
- 4) `create_sql_engine()` – створює SQL Engine; SQL Engine відповідає за обробку та виконання SQL–запитів, які надсилаються до бази даних; він взаємодіє з іншими компонентами СУБД, такими як оптимізатор

- запитів, планувальник та виконавець запитів, для ефективного оброблення та повернення результатів запиту;
- 5) `load_historical_data()` – циклічно надсилає запити до Binance API і завантажує усі доступні історичні дані по свічкам перпетуального і квартального контрактів, а також фандингу, потім обробляє їх і надсилає до БД;
 - 6) `backup_from_last_record()` – запитує timestamp останнього запису в БД і через Binance API запитує дані по всім свічкам перпетуального і квартального контрактів, а також фандингу що відбулися в інтервалі між цим timestamp і актуальним моментом;
 - 7) `stream()` – з Binance API отримує дані про останні свічки квартального і перпетуального контрактів, обробляє їх і заносить в БД;
 - 8) `get_bid_ask_data()` – з Binance API отримує моментальні дані про об'єми і ціни мінімального контракту продажу та максимального контракту купівлі (`bidPrice`, `bidQuantity`, `askPrice`, `askQuantity`), обробляє їх і заносить в БД;
 - 9) `send_telegram_message(message, chat_id, api_key)` – приймає на вхід текстове повідомлення у форматі `string`, `chat_id` – id користувача, `api_key` – API ключ від Telegram Bot API;
 - 10) `get_delivery(timestamp, output_type='timestamp')` – приймає на вхід момент часу у форматі `timestamp`, `output_type` – формат виводу, доступні значення: `["timestamp", "date"]`; повертає час закінчення кварталу, до якого належить цей момент;
 - 11) `get_quarter_time_left(timestamp)` – приймає на вхід момент часу у форматі `timestamp`, повертає кількість секунд, що залишилась до кінця кварталу;
 - 12) `get_quarter_symbol(pair, timestamp)` – приймає на вхід символ перпетуального контракту у форматі `string`, і час у форматі `timestamp`; повертає символ квартального контракту на момент часу `timestamp`;

- 13) daily_notification() – через Телеграм–бота надсилає повідомлення про виплати по 3 останніх фандингах а також значення максимальної розбіжності ціни bidPrice і askPrice і момент часу коли ця різниця була зафіксована;
- 14) run_backup_and_stream() – викликає функцію backup_from_last_record(), а потім за допомогою бібліотеки schedule щохвилини: на 0 секунді викликає функцію get_bid_ask_data, на 10 секунді викликає функцію stream; щодня о 8:10:30 UTC викликає функцію daily_notification()
- 15) main() – послідовно викликає функції connect_ssh_tunnel, create_sql_engine, run_backup_and_stream (для того щоб при виникненні помилок не доводилось перезапускати скрипт вручну, було реалізовано обробку помилок); при виникненні помилки надсилається повідомлення про помилку у Telegram–бота і через 300 секунд рекурсивно викликається функція main.

3.2. Створення розмітки для машинного навчання

3.2.1 Загальний огляд

Якщо точка вважається екстремумом, в колонці extrema вона матиме значення True, якщо ні – False відповідно. Мінімальний прибуток за один контракт повинен складати щонайменше 0.75% від суми відкриття контракту. Комісія за відкриття/закриття контракту становить 0.08% від суми контракту. Вважатимемо що у кожній з відмічених точок відбудуватиметься закриття попереднього контракту і відкриття протилежного йому. Торгове плече 5.

3.2.2. Визначення допоміжних функцій

Маємо наступні допоміжні функції.

1. `plot_selected_extremas` – зображує графік ціни контракту з плином часу і на цьому графіку виділяє: світло–зеленим кольором – усі екстремуми, червоним кольором – екстремуми визначені колонкою індикатором.

Вхідні параметри:

- `df` – таблиця формату `pandas.DataFrame`;
- `col` – колонка індикатор екстремуму;
- `window=(0,400)` – бажаний інтервал виводу;
- `color='red'` – бажаний колір виділення екстремумів;
- `figsize=(10,4)` – розмір графіку (в сотнях пікселів).

Прикладом є `plot_selected_extremas(df=df_razmetka, col='extr7', window=(400,800))` на рисунку 3.2:

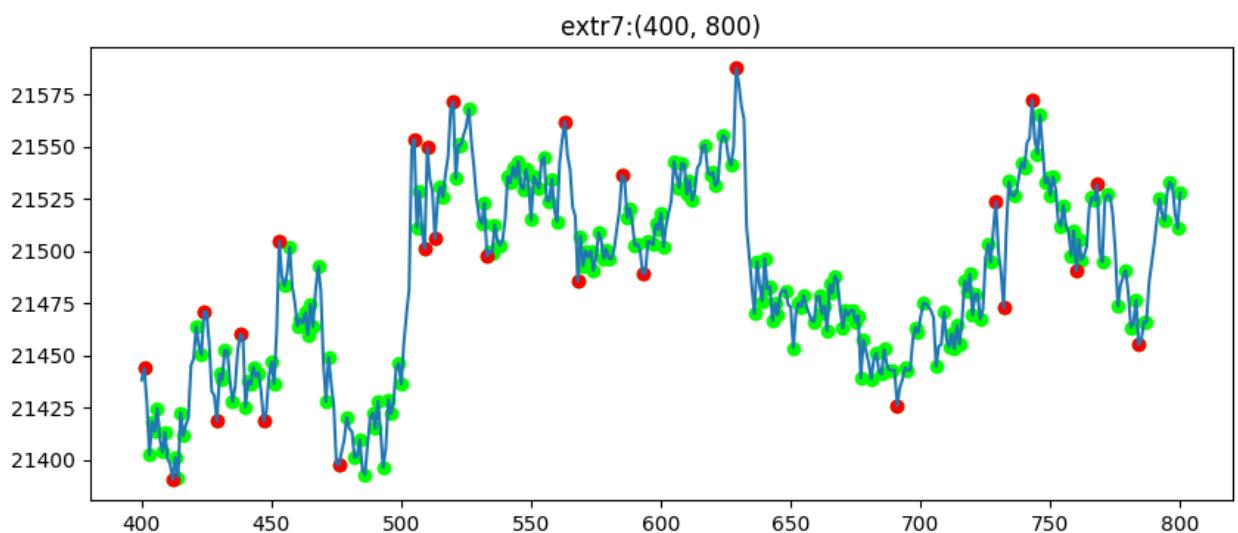


Рисунок 3.2 – Виділення максимумів з колонки “extr7” на відрізку [400:800]

2. `get_balance` – обчислення відносного прибутку від відкриття і закриття контракту, з урахуванням комісій і торгового плеча. Так як при створенні розмітки не було необхідності робити окремі функції для

шорт і лонг контрактів, прибутковість розраховується лише з урахуванням відносної зміни ціни.

Формула розрахунок прибутковості:

$$\text{profit} = (1 - \text{commision})^2 * (1 + \text{leverage} * \left| 1 - \frac{\text{Close}}{\text{Open}} \right|)$$

Вхідні параметри:

- open_price – ціна на момент відкриття контракту;
- close_price – ціна на момент закриття контракту;
- commission – комісія за відкриття/закриття контракту;
- leverage – розмір торгового плеча.

Приклад (рис. 3.3):

```
get_balance(10000, 11000)
✓ 0.0s
1.4976009600000004
```

Рисунок 3.3 – Приклад виконання функції `get_balance`

3. `get_balance_3contracts`:

Опис: Обчислення відносного прибутку від послідовності:

- відкриття контракту0;
- закриття контракту0;
- відкриття контракту1;
- закриття контракту1;
- відкриття контракту2;
- закриття контракту2;
- вхідні параметри;
- price0 – ціна активу в момент відкриття першого контракту;

- price1 – ціна активу в момент закриття першого і відкриття другого контракту,
- price2 – ціна активу в момент закриття другого відкриття третього контракту,
- price3 – ціна активу в момент закриття третього контракту.

Приклад (рис. 3.4):

```
get_balance_3contracts(10000, 11000, 9900, 10890), \
get_balance_3contracts(10000, 9000, 9900, 8910)
✓ 0.0s
(3.3588323654607315, 3.3588323654607306)
```

Рисунок 3.4 – Приклад get_balance_3contracts

3.2.3 Пошук локальних екстремумів

Нехай ми маємо 3 послідовні ціни: A, B, C.

Якщо $(B - A) * (C - B) < 0$ – точка B є екстремумом.

Можливі ситуації, коли ціна в точках A і B однаакова. В таких випадках ми йдемо по історії цін назад до моменту поки не дійдемо до моменту зміни ціни.

Приклад наведений на рис. 3.5:

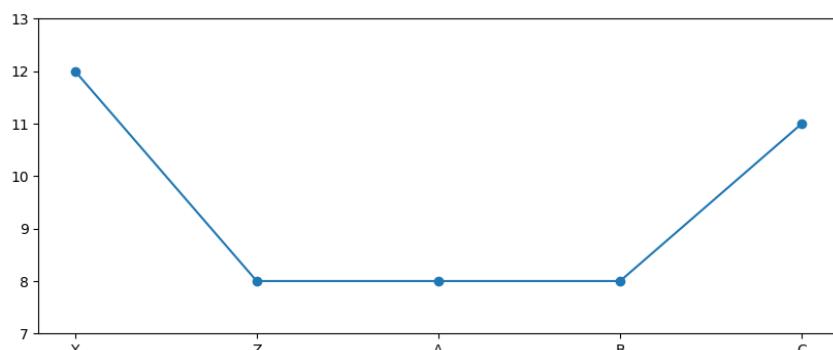


Рисунок 3.5 – Ілюстрація до прикладу знайдення екстремумів

Маємо наступні кроки:

- 1) $(B - A) = 0$ – пересуваємось на крок назад
- 2) $(B - A) = 0$ – пересуваємось на крок назад
- 3) $(Z - Y) \neq 0$ – записуємо це значення в колонку “diff_shift1”
- 4) Обчислюємо $(B - A) * (C - B)$, підставляючи $(Z - Y)$ на місце $(B - A)$
- 5) $(B - A) * (C - B) < 0$, отже точка В – екстремум.
- 6) якщо в точці В – “diff_shift1” < 0 , точка В – максимум, якщо в точці В – “diff_shift1” > 0 , точка В – мінімум.

Результат (рис. 3.6):

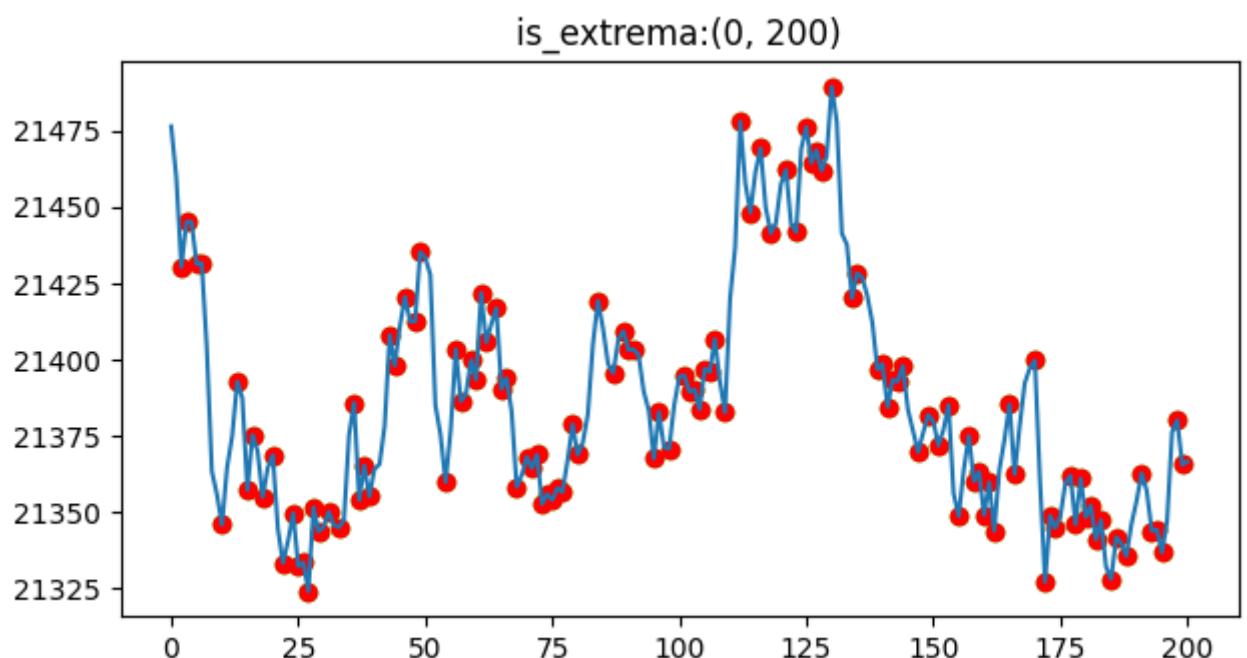


Рисунок 3.6 – Виділення локальних екстремумів

3.2.4. Пошук початкових екстремумів

Йдучи з початкової (нульової) точки і розглядаючи 3 наступні екстремуми, принципово можливі 4 сценарії. Розглядатимемо випадок, коли починаємо з шорт контракту, для лонг – теж саме, але інвертовано. Маємо наступні сценарії (рис. 3.7).

1. Додатковий прибуток від відкриття проміжних контрактів 1, 2 не перевищує 1.0752%. Найприбутковішою комбінацією виявилось закриття контракту в точці 3. В такому випадку ми виключаємо точки 1, 2.
2. Нульова точка залишається незмінною. Переходимо до розгляду точок: 0, 3, 4, 5.
3. Прибуток від відкриття контрактів 2, 3 не перевищує 1.0752%. В такому випадку ми виключаємо точки 2, 3. Точку 1 записуємо до екстремумів. Нульовою стає точка 1. Переходимо до розгляду точок: 1, 4, 5, 6.
4. Прибуток від послідовного відкриття контрактів 1,2,3 перевищує 1.0752% і є максимальним: Точку 1 записуємо до екстремумів. Нульовою стає точка 1. Переходимо до розгляду точок: 1, 2, 3, 4.
5. Жодна з комбінацій не перейшла поріг у 0.75%. В такому випадку ми виключаємо точки 1, 2, 3. Нульова точка залишається незмінною. Переходимо до розгляду точок 0, 4, 5, 6.

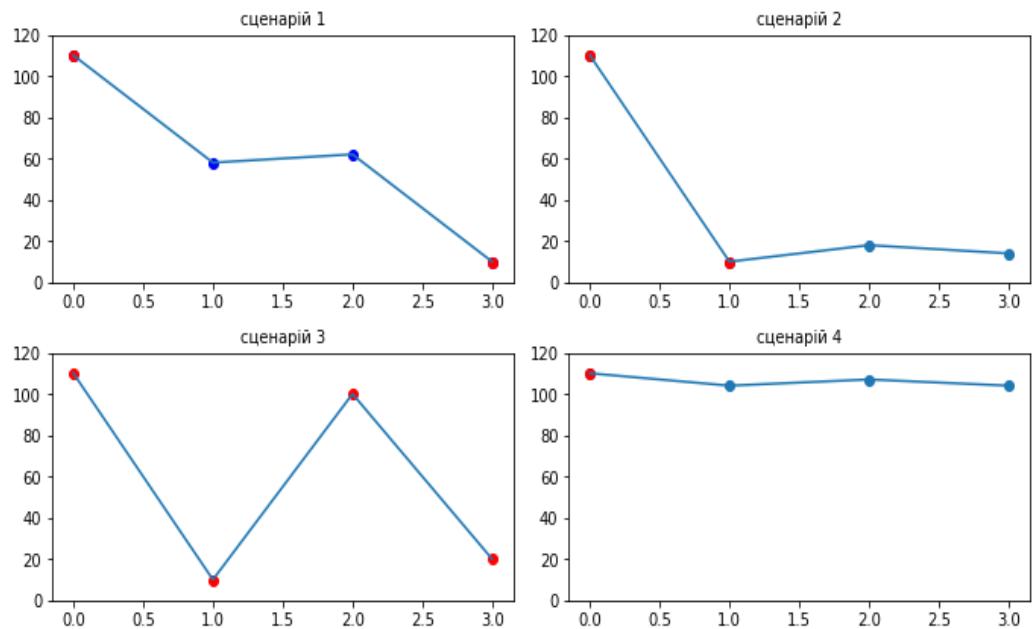


Рисунок 3.7 – Графіки можливих сценаріїв

Результат (рис. 3.8, табл. 3.1):

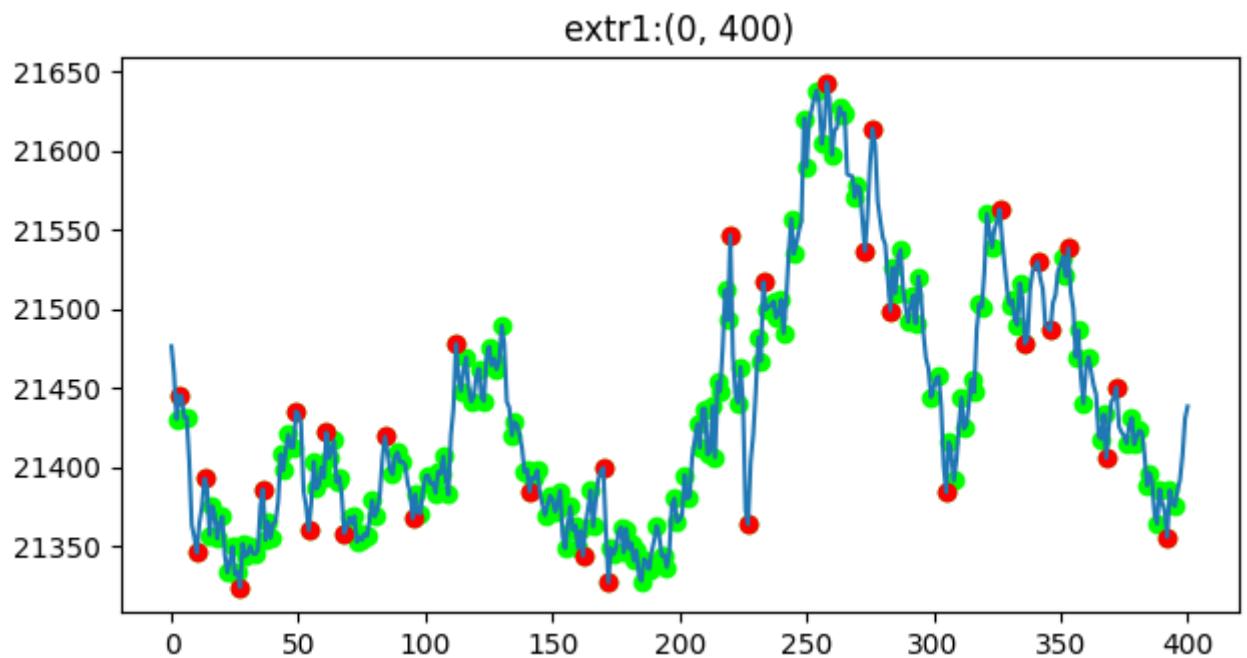


Рисунок 3.8 – Початкові екстремуми

Таблиця 3.1 – Проміжні результати

Екстремумів всього	188275
Екстремумів після кроку 3.2.4.	18620

Як бачимо по графіку (рис. 3.8) ці екстремуми не зовсім коректні. Без цього кроку пошук екстремумів можливий, але обчислення займатимуть набагато більше часу.

3.2.5. Пошук глобальних екстремумів на відрізках

Розглядаючи 3 послідовних екстремуми (назвемо їх left, mid, right) визначені після пошуку початкових екстремумів. Точку після right назовемо next_right. Якщо ціна в точці mid менше (більше) за ціну left – на відрізку [mid:right] шукаємо точку з мінімальною (максимальною) ціною – назовемо її index_extrema. Розглядатимемо випадок, коли починаємо з шорт контракту, для лонг – теж саме, але інвертовано. Принципово можливі 3 сценарії

1. index_extrema == mid: На цьому відрізку глобальний екстремум визначений правильно – переходимо до розгляду наступної трійки. (mid→left, right→mid, next_right→right) (рис. 3.9).

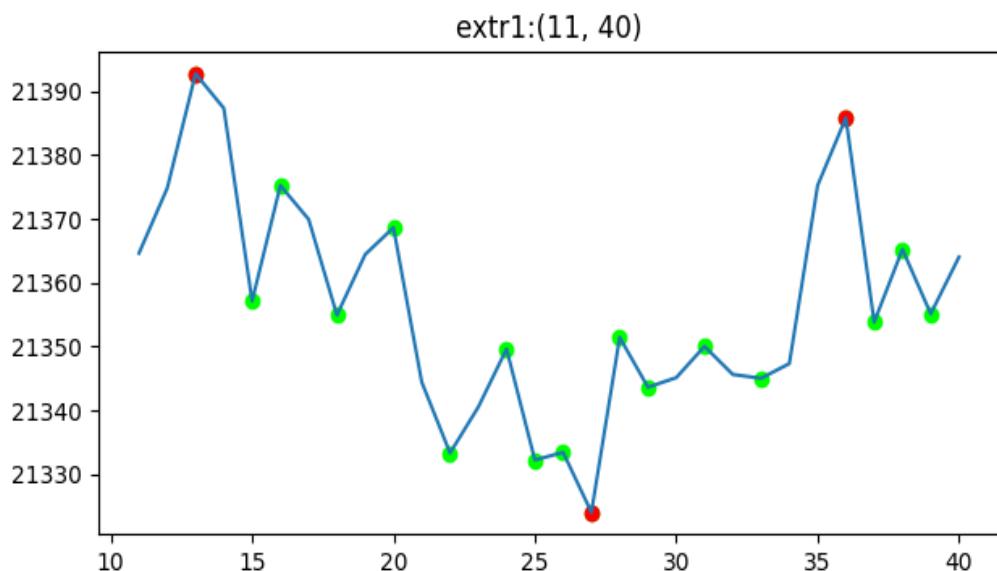


Рисунок 3.9 – Візуалізація сценарію 1

2. `index_extrema == right`: Екстремум `mid` визначений неправильно – видаляємо його і переходимо до розгляду наступної трійки. (`left→left, right→mid, next_right→right`) (рис. 3.10).

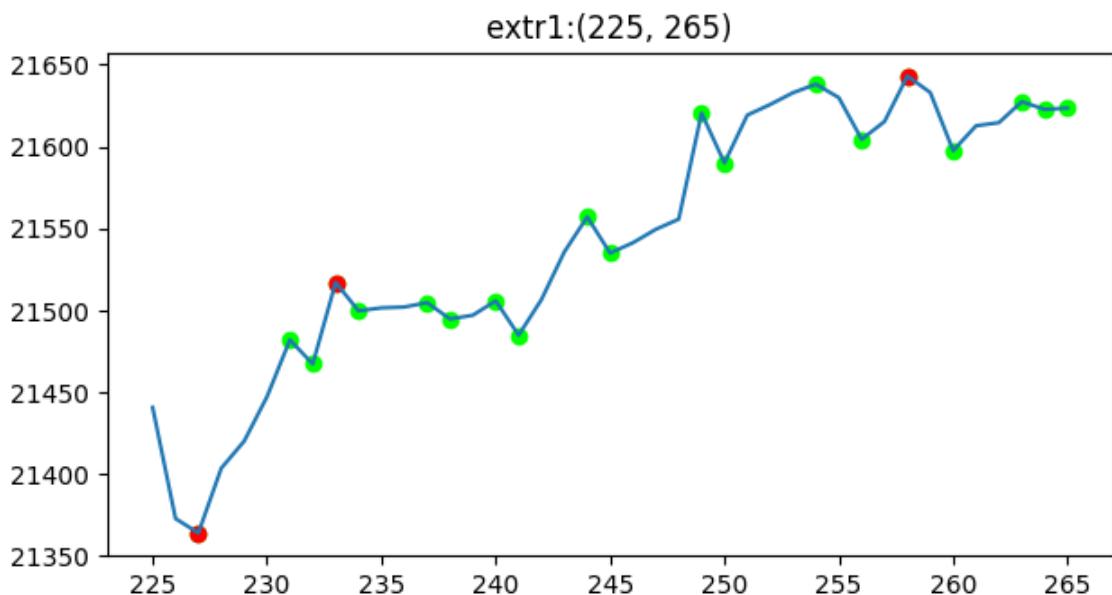


Рисунок 3.10 – Візуалізація сценарію 2

3. `(index_extrema != mid) & (index_extrema != right)`: `mid` не був глобальним екстремумом на цьому відрізку, тому замінюємо `mid` на `index_extrema` і

переходимо до розгляду наступної трійки (index_extrema→mid→left, right→mid, next_right→right)) (рис. 3.11).

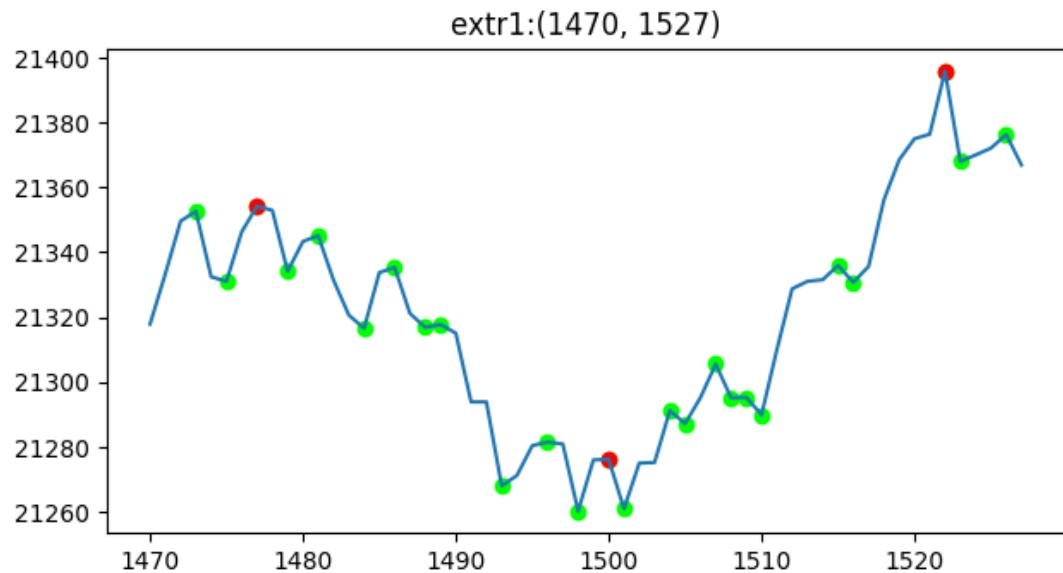


Рисунок 3.11 – Візуалізація сценарію 3

Результати (рис. 3.12 – 3.14, табл. 3.2):

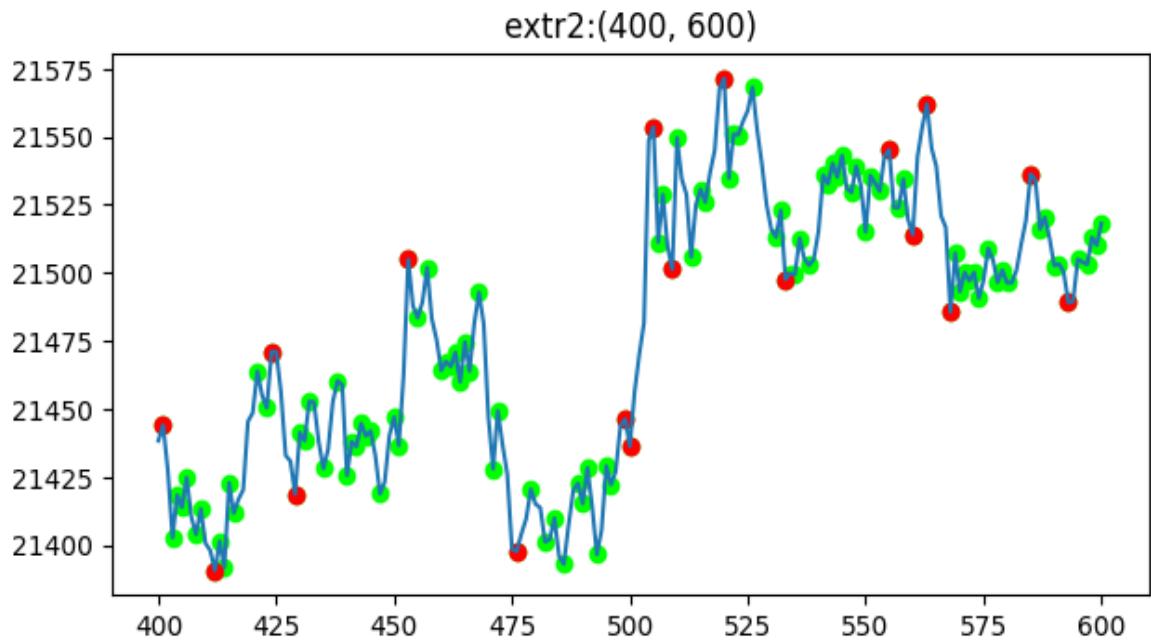


Рисунок 3.12 – Після знаходження глобальних екстремумів на відрізках

deleted:(400, 600)

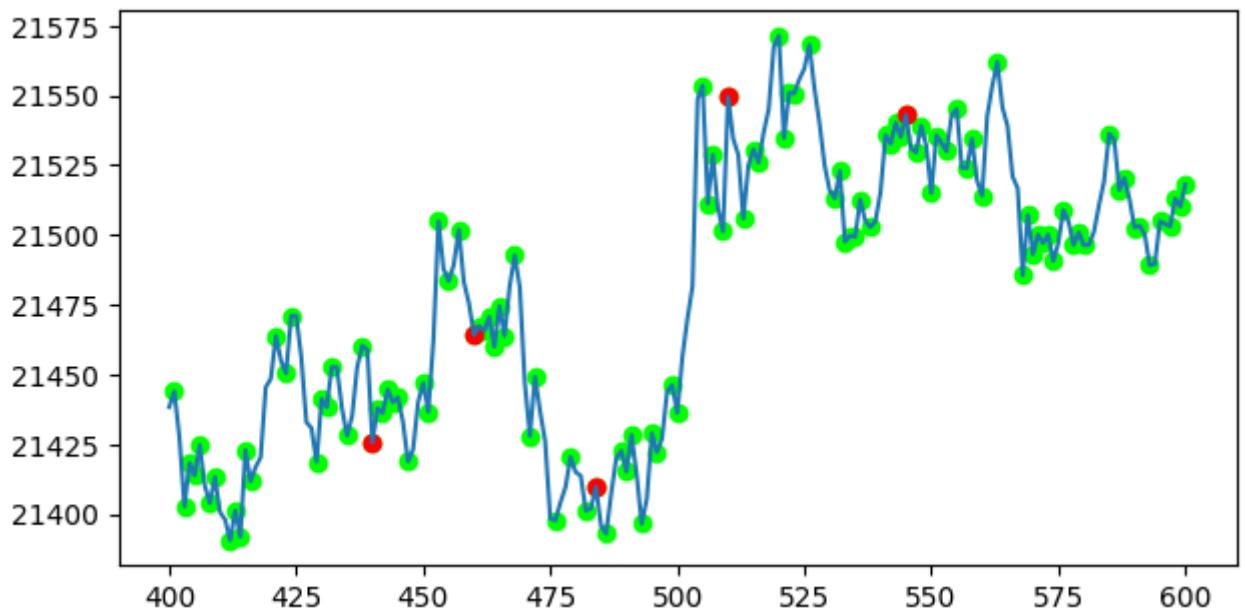


Рисунок 3.13 – Видалені екстремуми

added:(400, 600)

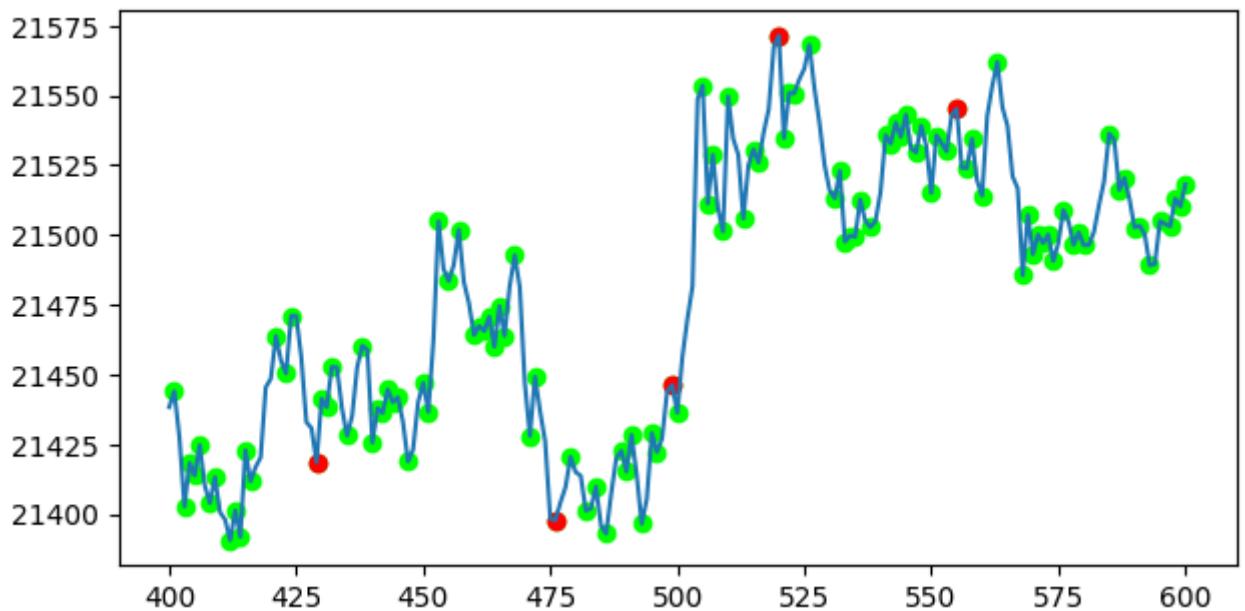


Рисунок 3.14 – Додані екстремуми

Таблиця 3.2 – Проміжні результати

Екстремумів всього	188275
Екстремумів після кроку 3.2.4.	18620
Екстремумів після кроку 3.2.5.	16024

3.2.6. Пошук неправильно визначених екстремумів

Нехай у точці 0 ми маємо відкритий контракт. Рахуємо прибуток від послідовного перевідкриття контрактів у точках 1, 2 і закриття у точці 3 – позначимо його $balance3$. І порівнюємо з прибутком від закриття контракту у точці 3 – позначимо його $balance1$. Якщо $balance3 * 1.0752 < balance1$: Видаляємо контракти 1, 2.

Результат (рис. 3.15 – 3.16, табл. 3.3):

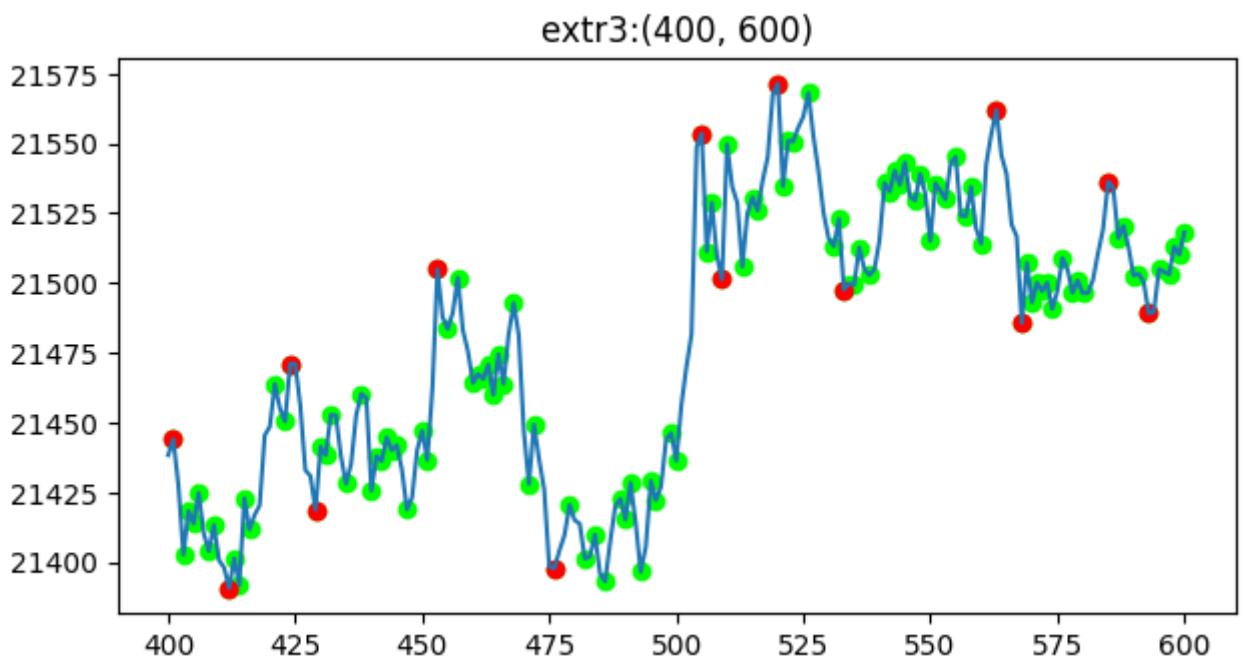


Рисунок 3.15 – Екстремуми після видалення зайвих

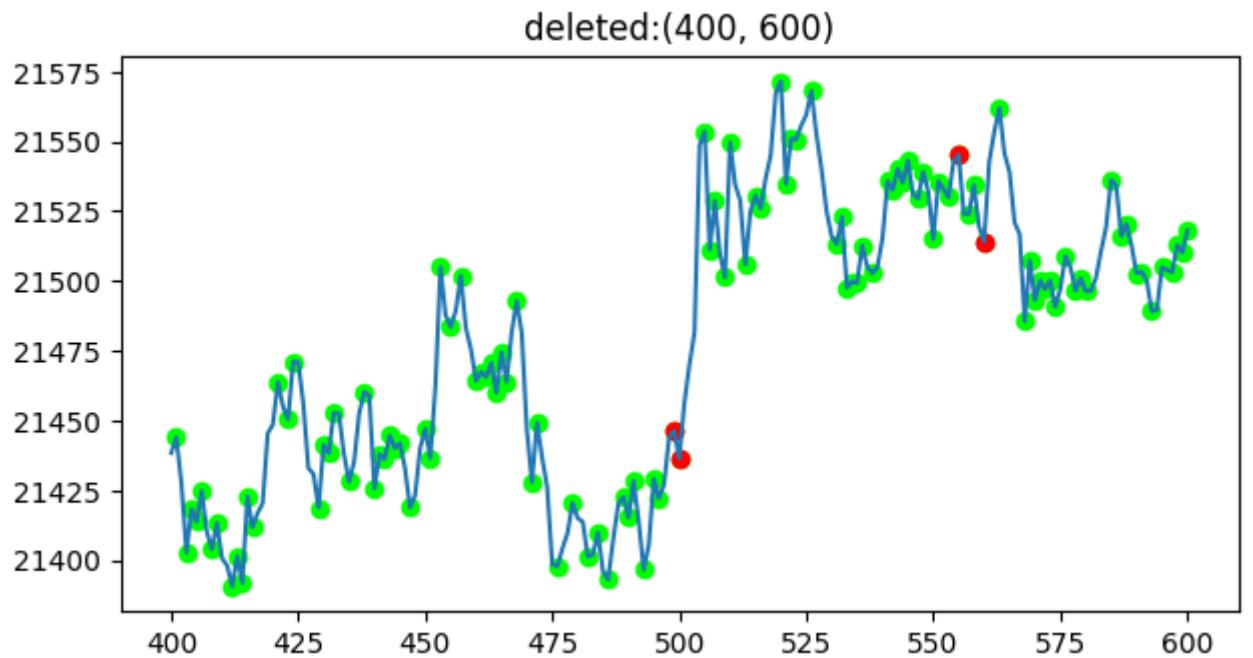


Рисунок 3.16 – Видалені екстремуми

Таблиця 3.3 – Проміжні результати

Екстремумів всього	188275
Екстремумів після кроку 3.2.4	18620
Екстремумів після кроку 3.2.5	16024
Екстремумів після кроку 3.2.6	15078

3.2.7. Пошук пропущених екстремумів

Беремо 2 послідовних контракти left , right . Нехай контракт відкрито в момент left . Прибуток від закриття контракту в момент right позначимо як balance1 . Дивимось чи можливо в інтервалі $(\text{left}:\text{right})$ перевідкрити 2 додаткових контракти (позначимо прибуток від такої послідовності як

`balance3`) так, щоб $\text{balance3} * 1.0752 > \text{balance1}$. Якщо можливо – додаємо ці контракти (рис. 3.17–3.18, табл. 3.4).

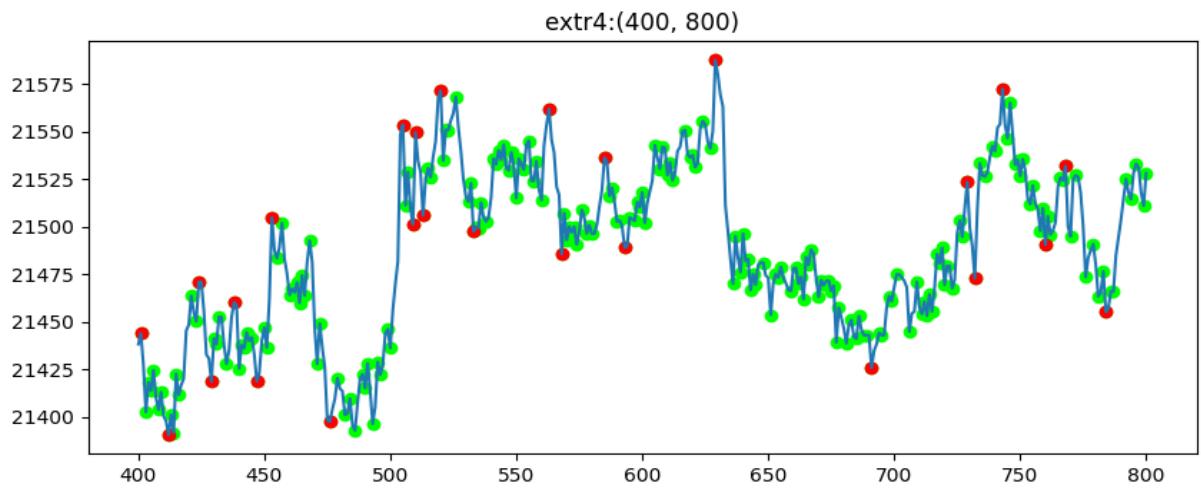


Рисунок 3.17 – Екстремуми після додавання пропущених

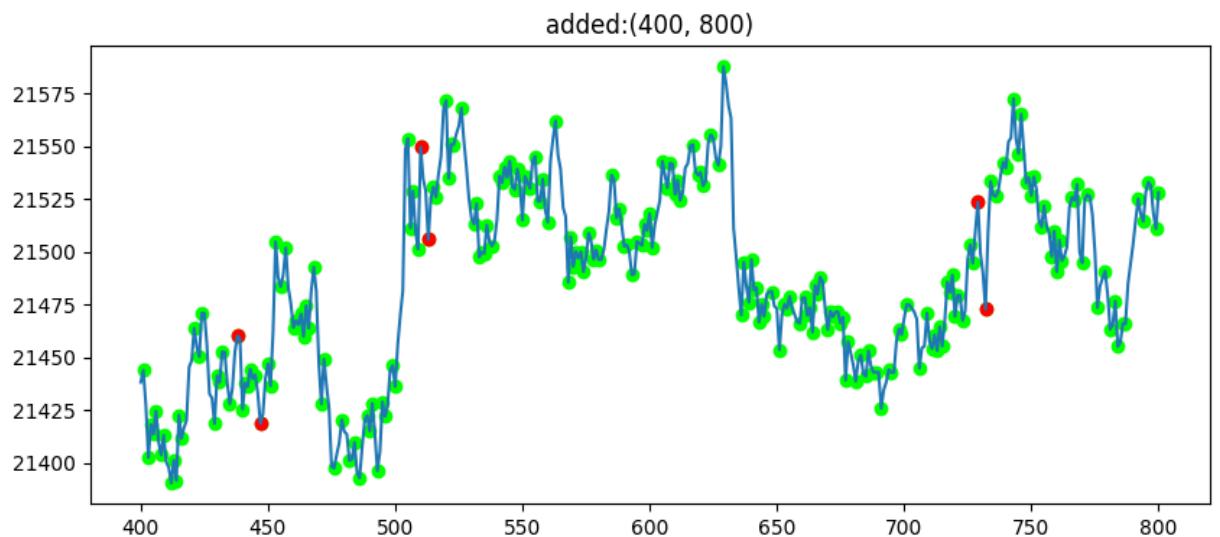


Рисунок 3.18 – Додані екстремуми

Таблиця 3.4 – Проміжні результати

Екстремумів всього	188275
Екстремумів після кроку 3.2.4	18620
Екстремумів після кроку 3.2.5	16024

Продовження таблиці 3.4

Екстремумів після кроку 3.2.6	15078
Екстремумів після кроку 3.2.7	18406

3.2.8. Повторення кроків 3.2.6 і 3.2.7 доки кількість екстремумів не стабілізується

Таблиця 3.5 – Остаточні результати

Екстремумів всього	188275
Екстремумів після кроку 3.2.4	18620
Екстремумів після кроку 3.2.5	16024
Екстремумів після кроку 3.2.6	15078
Екстремумів після кроку 3.2.7	18406
Екстремумів після кроку 3.2.8	18324

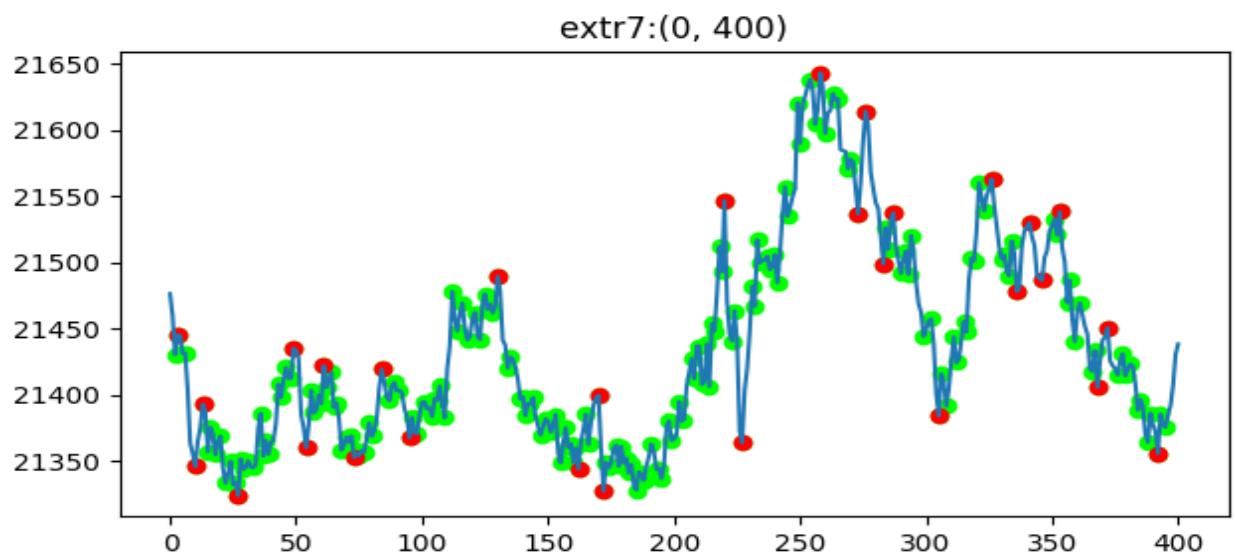


Рисунок 3.19 – Фінальний результат

3.2.9. Огляд результатів

Подивимось на розподіл тривалостей контрактів (рис. 3.20):

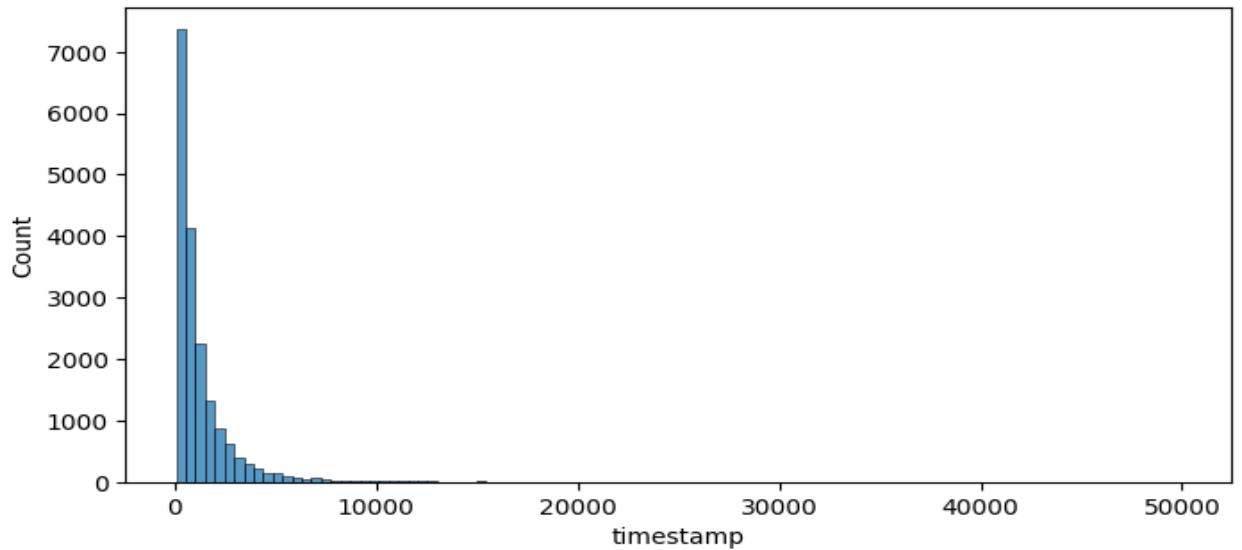


Рисунок 3.20 – Розподілення часу тривалості контрактів

Як бачимо – присутні викиди, тому подивимось на розподіл на відрізку [0:10000] (рис. 3.21).

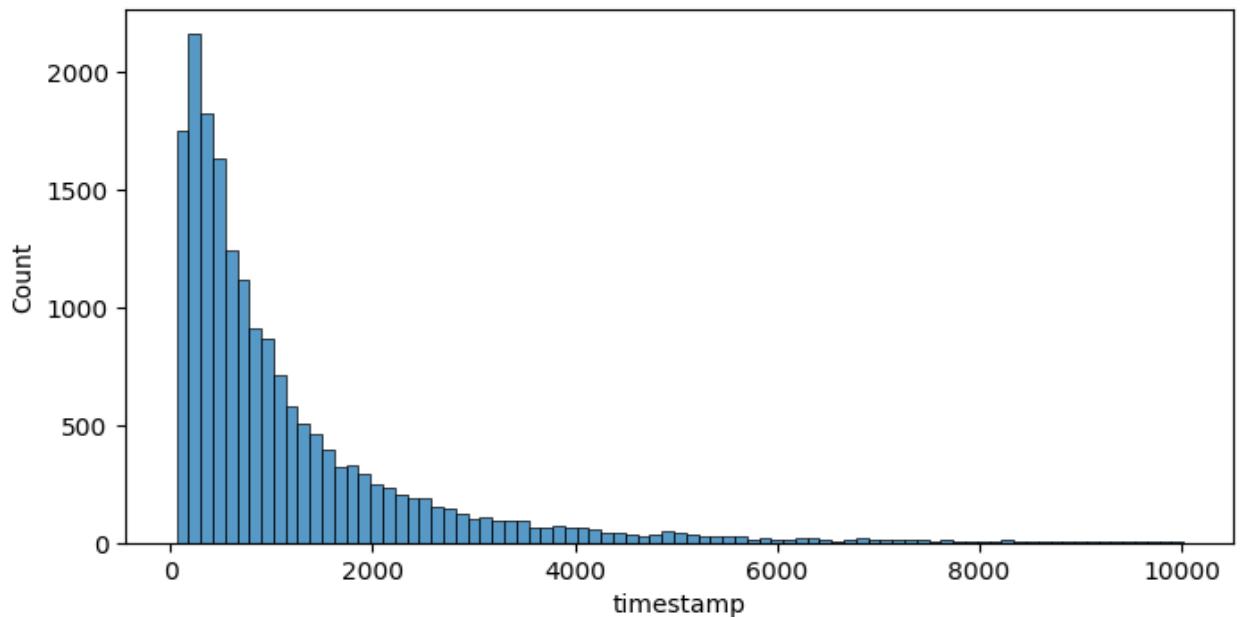


Рисунок 3.21 – Розподілення часу тривалості контрактів на відрізку [0:10000]

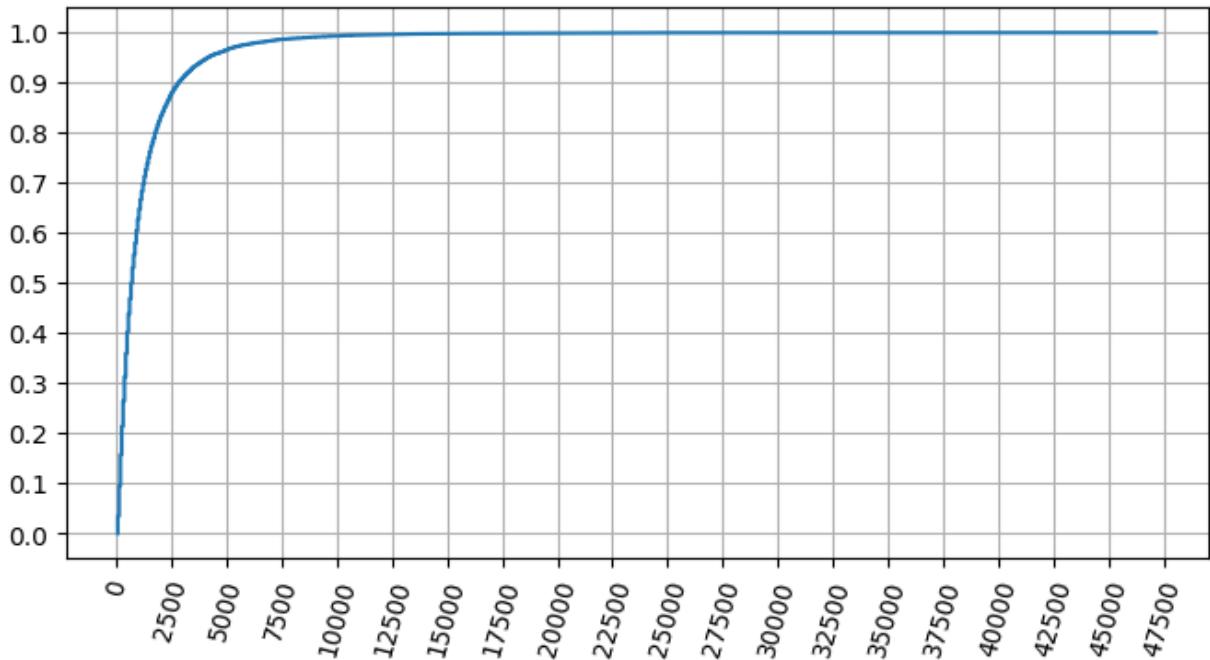


Рисунок 3.22 – Кумулятивний розподіл часу тривалостей контрактів

З рисунку 3.22 видно, що:

- 80.77% контрактів тривають до 30 хвилин;
- 93.48% контрактів тривають до 60.

3.3. Генерація нових ознак

3.3.1. Генерація по-хвилинних ознак

Маємо наступні ознаки.

1. `day_of_week` – день тижня – категоріальна змінна, тому перетворимо її до вигляду бінарного вектора за допомогою техніки OneHotEncoding, $E = [0, 1, 2, 3, 4, 5, 6]$.
2. `second_of_day` – секунда дня, $E = [0:86400]$.
3. `open_close_diff_perp` – різниця між ціною відкриття і закриття свічки перпетуального контракту:

$$open_close_diff_{perp} = open_{perp} - close_{perp},$$

E=ℝ.

4. high_low_diff_perp – різниця між максимальною і мінімальною ціною свічки перпетуального контракту:

$$high_low_diff_{perp} = high_{perp} - low_{perp},$$

E=ℝ+.

5. avg_trade_volume_perp – середній розмір перпетуального контракту, розраховується як сума усіх контрактів відкритих протягом хвилини, поділена на їх кількість:

$$avg_trade_volume_{perp} = \frac{volume_{perp}}{trades_{perp}},$$

E=ℝ+.

6. bid_ask_spread_abs_perp – різниця між ціною купівлі і ціною продажу на момент відкриття свічки для перпетуального контракту:

$$bid_ask_spread_abs_{perp} = askPrice_{perp} - bidPrice_{perp},$$

E=ℝ+.

7. bid_ask_spread_perp – відносна різниця між мінімальною ціною продажу і максимальною ціною купівлі на момент відкриття свічки для перпетуального контракту:

$$bid_ask_spread_{perp} = \frac{askPrice_{perp} - bidPrice_{perp}}{askPrice_{perp}},$$

E=[0:1].

8. `bid_ask_qty_spread_abs_perp` – різниця між об'ємами перпетуальних контрактів по мінімальній ціні продажу і максимальній ціні купівлі:

$$\text{bid_ask_qty_spread_abs}_{perp} = \text{askQty}_{perp} - \text{bidQty}_{perp},$$

E= \mathbb{R} .

9. `bid_ask_qty_spread_perg` – відносна різниця між об'ємами перпетуальних контрактів по мінімальній ціні продажу і максимальній ціні купівлі:

$$\text{bid_ask_qty_spread}_{perp} = \frac{\text{askQty}_{perp} - \text{bidQty}_{perp}}{\text{askQty}_{perp} + \text{bidQty}_{perp}},$$

E=[-1:1].

10. `bid_ask_qty_sum_perp` – сума об'ємів перпетуальних контрактів по мінімальній ціні продажу і максимальній ціні купівлі:

$$\text{bid_ask_qty_sum}_{perp} = \text{askQty}_{perp} + \text{bidQty}_{perp},$$

E= \mathbb{R}^+ .

11. `open_close_diff_cq` – різниця між ціною відкриття і закриття свічки квартального контракту:

$$\text{open_close_diff}_{cq} = \text{open}_{cq} - \text{close}_{cq},$$

E= \mathbb{R} .

12. `high_low_diff_cq` – різниця між максимальною і мінімальною ціною свічки квартального контракту:

$$high_low_diff_{cq} = high_{cq} - low_{cq},$$

E=ℝ+.

13.avg_trade_volume_cq – середній розмір квартального контракту, розраховується як сума усіх контрактів відкритих протягом хвилини, поділена на їх кількість:

$$avg_trade_volume_{cq} = \frac{volume_{cq}}{trades_{cq}},$$

E=ℝ+.

14.bid_ask_spread_abs_cq – різниця між ціною купівлі і ціною продажу на момент відкриття свічки для квартального контракту:

$$bid_ask_spread_abs_{cq} = askPrice_{cq} - bidPrice_{cq},$$

E=ℝ+.

15.bid_ask_spread_cq – відносна різниця між мінімальною ціною продажу і максимальною ціною купівлі на момент відкриття свічки для квартального контракту:

$$bid_ask_spread_{cq} = \frac{askPrice_{cq} - bidPrice_{cq}}{askPrice_{cq}},$$

E=[0:1].

16.bid_ask_qty_spread_abs_perp – різниця між об'ємами квартальних контрактів по мінімальній ціні продажу і максимальній ціні купівлі:

$$bid_ask_qty_spread_abs_{cq} = askQty_{cq} - bidQty_{cq},$$

E=ℝ.

17.bid_ask_qty_spread_cq – відносна різниця між об’ємами квартальних контрактів по мінімальній ціні продажу і максимальній ціні купівлі:

$$bid_ask_qty_spread_{cq} = \frac{askQty_{cq} - bidQty_{cq}}{askQty_{cq} + bidQty_{cq}},$$

E=[-1:1].

18.bid_ask_qty_sum_cq – сума об’ємів квартальних контрактів по мінімальній ціні продажу і максимальній ціні купівлі:

$$bid_ask_qty_sum_{cq} = askQty_{cq} + bidQty_{cq},$$

E=ℝ+

19.perp_cq_open_spread_abs – різниця між ціною відкриття свічок квартального і перпетуального контрактів:

$$perp_cq_open_spread_abs = open_{perp} - open_{cq},$$

E=ℝ.

20.perp_cq_open_spread – відносна різниця між ціною відкриття свічок квартального і перпетуального контрактів:

$$perp_cq_open_spread_abs = \frac{open_{perp} - open_{cq}}{open_{perp}},$$

E=ℝ.

short_term_expectations_X – різні варіанти розрахунку індикатора “короткострокові очікування”.

$$21. \quad short_term_expectations_1 = \frac{open_{perp} - open_{cq}}{time_left + 86400},$$

E=ℝ.

$$22. \quad short_term_expectations_2 = \frac{open_{perp} - open_{cq}}{time_left + 86400 * 9},$$

E=ℝ.

$$23. \quad short_term_expectations_3 = \frac{open_{perp} - open_{cq}}{time_left + 86400 * 90},$$

E=ℝ.

$$24. \quad short_term_expectations_4 = \frac{(open_{perp} - open_{cq})^2}{time_left + 86400},$$

E=ℝ+.

$$25. \quad short_term_expectations_5 = \frac{(open_{perp} - open_{cq})^2}{time_left + 86400 * 9},$$

E=ℝ+.

$$26. \quad short_term_expectations_6 = \frac{(open_{perp} - open_{cq})^2}{time_left + 86400 * 90},$$

E=ℝ+.

$$27. \quad short_term_expectations_7 = \frac{(open_{perp} - open_{cq})^2}{\sqrt{time_left + 86400}},$$

E=ℝ+.

$$28. \quad short_term_expectations_8 = \frac{(open_{perp} - open_{cq})^2}{\sqrt{time_left + 86400 * 9}},$$

E=ℝ+.

$$29. \quad short_term_expectations_9 = \frac{(open_{perp} - open_{cq})^2}{\sqrt{time_left + 86400 * 90}},$$

E=ℝ+.

$$30. \quad short_term_expectations_{10} = \frac{\sqrt{|open_{perp} - open_{cq}|}}{time_left + 86400},$$

E=ℝ+.

$$31. \quad short_term_expectations_{11} = \frac{\sqrt{|open_{perp} - open_{cq}|}}{time_left + 86400 * 9},$$

E=ℝ+.

$$32. \quad short_term_expectations_{12} = \frac{\sqrt{|open_{perp} - open_{cq}|}}{time_left + 86400 * 90},$$

E=ℝ+.

$$33. \quad short_term_expectations_{13} = \frac{\sqrt{|open_{perp} - open_{cq}|}}{\sqrt{time_left + 86400}},$$

E=ℝ+.

$$34. \quad short_term_expectations_{14} = \frac{\sqrt{|open_{perp} - open_{cq}|}}{\sqrt{time_left + 86400 * 9}},$$

E=ℝ+.

$$35. \quad short_term_expectations_{15} = \frac{\sqrt{|open_{perp} - open_{cq}|}}{\sqrt{time_left + 86400 * 90}},$$

E=ℝ+.

3.3.2. Генерація по—контрактних ознак

Маючи розмітку даних, ми можемо розділити наші дані на 18324 контракти, для цього кожному моменту часу призначаємо номер контракту, протягом який відповідає номеру контракту, що був відкритий у цей момент і створюємо нові фічі, що описують проміжок протягом якого було відкрито контракт, групуючи дані по номеру контракту.

Отримуємо:

1. З колонок ['open_perp', 'open_cq', 'bidPrice_perp', 'askPrice_perp', 'bidPrice_cq', 'askPrice_cq'] беремо показники:
 - a) min – мінімум;

- б) max – максимум;
- в) total_delta – різниця в ціні відкриття і ціні закриття контракту:

$$total_delta = price_{-1} - price_0;$$

- г) minmax_rel_diff – відносна різниця в максимальній і мінімальній ціні що були зафіксовані на проміжку доки було відкрито контракт:

$$minmax_rel_diff = \frac{\max - \min}{\max};$$

2. З колонок ['volume_perp', 'trades_perp', 'volume_cq', 'trades_cq', 'bidQty_perp', 'askQty_perp', 'bidQty_cq', 'askQty_cq'] беремо показники:
 - а) sum – сума;
 - б) mean – середнє.
3. З колонки 'time_left' беремо показник contract_duration – тривалість контракту в секундах.

3.4. Створення цільового показника і метрики

3.4.1. Загальний огляд

При виборі цільового показника було на вибір 2 варіанти:

- 1) розглядати задачу як задачу класифікації – в такому випадку, цільовий показник приймає значення – True, в рядках які є екстремумами, в інших рядках – False.

2) розглядати задачу як задачу регресії – в такому випадку цільовим показником буде ціна, при якій ми закриватимемо контракт.

Було прийнято рішення розглядати задачу як задачу регресії з наступних причин.

Недоліки задачі класифікації:

- 1) протягом хвилини ціна контракту коливається, а дані дискретні – при класифікації модель не змогла б отримати переваги від цих коливань і мала б закривати і відкривати контракт лише у фіксовану секунду хвилини і лише за ринковими ордерами;
- 2) більш того, беручи до уваги затримку з якою отримуються дані від Binance API а також час необхідний для обробки даних – протягом цієї затримки ситуація може змінитись непередбачувано;
- 3) цільовий показник був би незбалансований, що ускладнило б вибір метрики і потребувало додаткових маніпуляцій над даними.

Переваги задачі регресії:

- 1) при задачі регресії ми можемо відкривати лімітні ордери і таким чином ми не втрачаємо можливість отримати додаткову перевагу від короткострокових коливань контракту;
- 2) так як зі збільшенням часу протягом якого контракт відкритий – збільшуються ризики, за допомогою введення другої цільової змінної ми зможемо не лише перевідкривати контракти, але і закривати актуальний контракт не відкриваючи протилежний йому.

3.4.2. Перший цільовий показник

Під час створення розмітки ми орієнтувалися на ціну відкриття свічки. Однак, протягом хвилини ціна контракту коливається. Для того щоб врахувати ці коливання було прийнято наступні кроки.

1. Додатково визначено 2 колонки:

$$- \text{ up_border : } up_border = \frac{high + max(open, close)}{2} ;$$

$$- \text{ low_border : } low_border = \frac{low + min(open, close)}{2} .$$

2. Для локальних мінімумів (максимумів) першої цільовою функцією буде показник з колонки up_border (low_border). Цей цільовий показник буде працювати стабільніше ніж тільки high/low чи тільки open, тому, що:

- 1) на відміну від open – враховує потенційну перевагу від коливань контрактів;
- 2) при виборі high/low, навіть якщо уявити, що ми маємо ідеальну модель – не гарантовано, що нам вистачить об'єму, який готові покрити покупці/продажці;
- 3) при цілі high (low), якщо модель помилиться так що prediction більший (менший) за target – лімітний контракт не реалізується і ми втратимо вигідний момент для перевідкриття;
- 4) з вищезазначеного цільовою функцією ми зможемо розробити метрику, що краще підходить для задачі торгівлі, ніж звичайні MAE, MSE, RMSE чи R2.

3.4.3. Другий цільовий показник

Зі збільшенням часу протягом якого контракт відкритий – збільшуються ризики. Маючи тільки один цільовий показник ми передбачаємо тільки ціну, за якою ми закриємо актуальний контракт і відкриємо наступний протилежний йому. Ми хотіли б мінімізувати кількість контрактів, що тривають довше 30 хвилин.

Для цього створимо колонки:

- `min_over_upcoming_30min` – мінімальний показник `low_border`, що буде зафіксовано протягом наступних 30 хвилин;
- `max_over_upcoming_30min` – максимальний показник `up_border`, що буде зафіксовано протягом наступних 30 хвилин.

Якщо точка є максимумом – у другу цільову колонку буде внесено значення `min_over_upcoming_30min`. Мінімумом – значення `max_over_upcoming_30min`.

3.4.4. Фінальна обробка цільових показників

Тепер, маючи цільові показники для всіх рядків, що є екстремумами ми пересуваємо усі значення в цих колонках на клітинку назад і заповнюємо пропуски методом “`backfill`” – кожен пропуск заповнюється значенням наступної непорожньої клітинки.

Таким чином, після закриттяожної хвилини свічки відкриватиметься новий лімітний ордер по ціні передбаченій в цільовому показнику 1.

У випадках, коли прибуток від контракту відкритого по ЦП1 і закритого по ЦП2 не перевищує 1.075% – актуальний контракт буде закрито по ціні ЦП1, але нового контракту відкрито не буде.

3.4.5. Метрика

Так як цільовий показник 1 – ціна лімітного ордеру, який буде відкрито протягом наступної хвилини – змістимо усі передбачення на одну клітинку вперед. Наша мета: в точках закриття контрактів – простимулювати їх закриття. В усіх інших точках – простимулювати, щоб контракт не закрився раніше часу.

Розглянемо 4 можливих випадки.

1. Відкрито лонг контракт і у цю хвилину відбудеться закриття контракту (рис. 3.23).

- 1) якщо передбачувана ціна менше за ціну відкриття (Range 1) – ордер розраховується по ринковій ціні:

$$score = \sqrt{target - open};$$

- 2) якщо передбачувана ціна лежить в межах ціни відкриття і хвилинного максимуму (Range 2):

$$score = \sqrt{target - prediction};$$

- 3) якщо передбачувана ціна більша за хвилинний максимум (Range 3) – це означатиме, що ми пропустили момент для закриття, а тому такі випадки штрафуються сильніше:

$$score = 10 * \sqrt{|target - prediction|}.$$

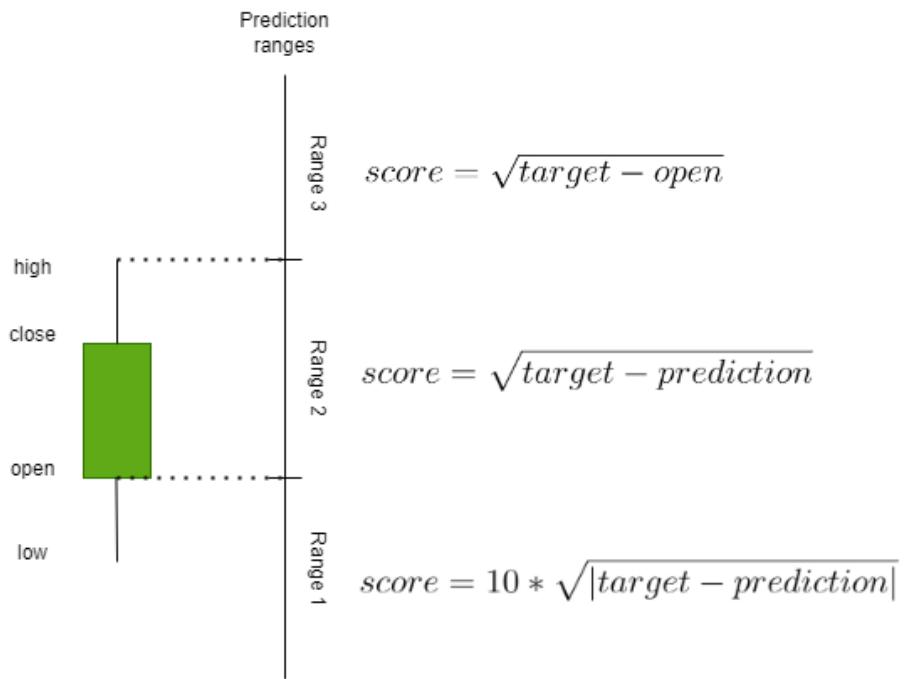


Рисунок 3.23 – Ілюстрація першого випадку

2. Відкрито шорт контракт і у цю хвилину відбудеться закриття контракту (рис 3.24).

1) якщо передбачувана ціна більша за ціну відкриття (Range 1) – ордер розраховується по ринковій ціні:

$$score = \sqrt{open - target};$$

2) якщо передбачувана ціна лежить в межах ціни відкриття і хвилинного мінімуму (Range 2):

$$score = \sqrt{prediction - target};$$

3) якщо передбачувана ціна менша за хвилинний мінімум (Range 3) – це означатиме, що ми пропустили момент для закриття, а тому такі випадки штрафуються сильніше:

$$score = 10 * \sqrt{|target - prediction|}.$$

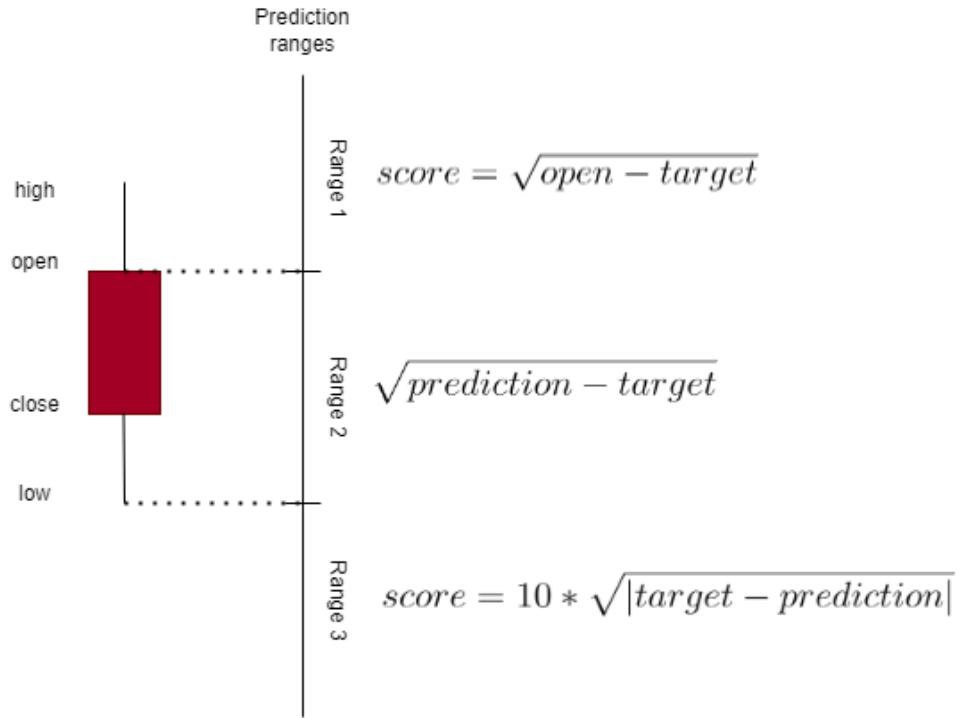


Рисунок 3.24 – Ілюстрація другого випадку

3. Відкрито лонг контракт і у цю хвилину не відбудеться закриття контракту.

1) якщо передбачувана ціна більша за максимальну хвилинну ціну:

$$score = \sqrt{prediction - target};$$

2) якщо передбачувана ціна менша за максимальну хвилинну ціну:

$$score = 10 * \sqrt{|prediction - target|}.$$

4. Відкрито шорт контракт і у цю хвилину не відбудеться закриття контракту.

1) якщо передбачувана ціна менша за мінімальну хвилинну ціну:

$$score = \sqrt{prediction - target};$$

2) якщо передбачувана ціна більша за мінімальну хвилинну ціну:

$$score = 10 * \sqrt{prediction - target}.$$

Для цільового показника 2 візьмемо звичайну MSE:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2.$$

3.5. Попередня обробка даних

Категоріальну ознаку `day_of_week` (день тижня) подамо у вигляді бінарного вектора за допомогою перетворення One-Hot Encoding.

One-Hot Encoding (однозначне кодування) – це метод представлення категоріальних змінних у вигляді бінарних векторів. При використанні One-Hot Encoding для змінної створюється новий набір бінарних змінних, де кожна змінна представляє одну унікальну категорію з початкової змінної.

Результат (табл. 3.6 – 3.7):

Таблиця 3.6 – Показник `day_of_week` до перетворення

<code>day_of_week</code>	0	1	2	3	4	5	6
--------------------------	---	---	---	---	---	---	---

Таблиця 3.7 – Показник `day_of_week` після перетворення.

<code>dow_0</code>	<code>dow_1</code>	<code>dow_2</code>	<code>dow_3</code>	<code>dow_4</code>	<code>dow_5</code>	<code>dow_6</code>
True	False	False	False	False	False	False
False	True	False	False	False	False	False
False	False	True	False	False	False	False
False	False	False	True	False	False	False
False	False	False	False	True	False	False
False	False	False	False	False	True	False
False	False	False	False	False	False	True

Після цього усі колонки стандартизуємо за формулою:

$$z_i = \frac{x_i - \bar{X}}{\sigma} .$$

3.6. Розбиття даних на тренувальну, тестову і валідаційну вибірки

Уся вибірка складає 381,497 рядків.

В якості валідаційної вибірки візьмемо останні 31,497 рядків.

В роботі з часовими рядами важливо, щоб тестова вибірка знаходилась після тренувальної. Для цього розділимо вибірку на 10 відрізків по 34,000 строк за допомогою техніки Nested Cross-Validation. Для цього скористаємось функцією `sklearn.model_selection.TimeSeriesSplit` з бібліотеки `scikit-learn` (рис. 3.25).

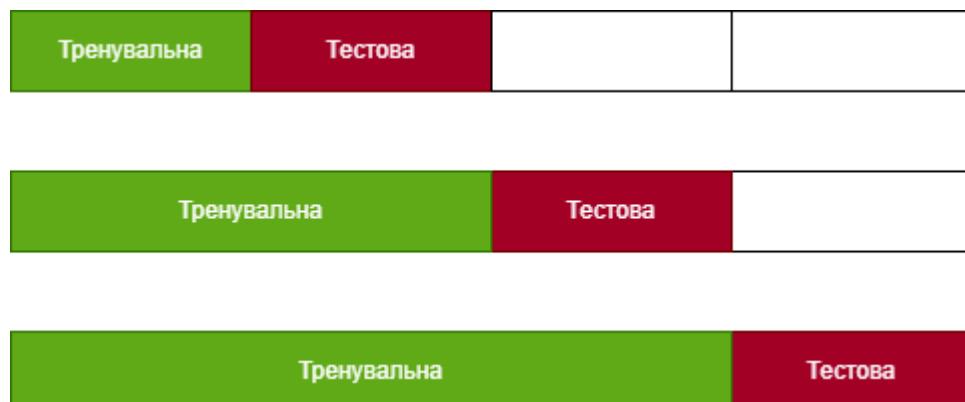


Рисунок 3.25 – Приклад Nested Cross-Validation з розбиттям вибірки на 4 відрізки.

3.7. Створення моделі на основі LSTM шару і підбір гіперпараметрів

3.7.1. Підготовка даних для LSTM шару

LSTM шар приймає дані у форматі тривимірного тензора. Розміри цього тензора визначаються таким чином: (кількість прикладів, кількість кроків часу, кількість ознак).

Візьмемо 60 кроків часу.

Для цього беремо по 60 строк від останнього (-1) запису, потім від передостаннього (-2) і т.д.

Отримуємо:

З початкової таблиці розмірністю 381403*110,
отримали тензор розмірністю 381344*60*110.

3.7.2. Підбір гіперпараметрів моделі

Для автоматизації процесу підбору гіперпараметрів було використано функцію `BayesianOptimizationOracle` з бібліотеки `keras_tuner`. Модель складатиметься з вхідного шару LSTM, Dense шарів [3,4,5]. Кількість нейронів в кожному шарі однакова і складає [32, 64, 128]. Після усіх Dense шарів можуть бути або не бути шари Dropout. Параметр `learning_rate` [1e-4, 1e-5, 1e-6].

Отже маємо $3 * 3 * 2 * 3 = 18$ – унікальних комбінацій гіперпараметрів.

Кожен набір гіперпараметрів тренуватимемо по 5 разів (Nested Cross-Validation) по 10 епох. Пізніше, знайшовши оптимальну комбінацію параметрів тренування запустимо на довший час.

Якби ми не мали обмеження часу або більші обчислювальні можливості, можна було б додати до перебору додаткові параметри, наприклад такі як:

- вибір активаційної функції. – На даний момент було взято тільки одну – Rectified Linear Unit (ReLU);
- додавання шарів BatchNormalization();
- додавання шарів Регуляризації();
- форма нейронної мережі – на даний момент кількість нейронів у всіх шарах однакова, тобто – усі мережі “прямокутні”; якщо відокремити кількість нейронів в кожному шарі – можна буде спробувати різні інші форми, проте це значно збільшить кількість можливих комбінацій;

- відокремлення шарів регуляризації та випадів (dropout) один від одного
- на даний момент регуляризація і випади dropout відбуваються або на всіх шарах одночасно, або не додаються на жодному.

3.7.3. Структура оптимальної моделі (рис 3.26)

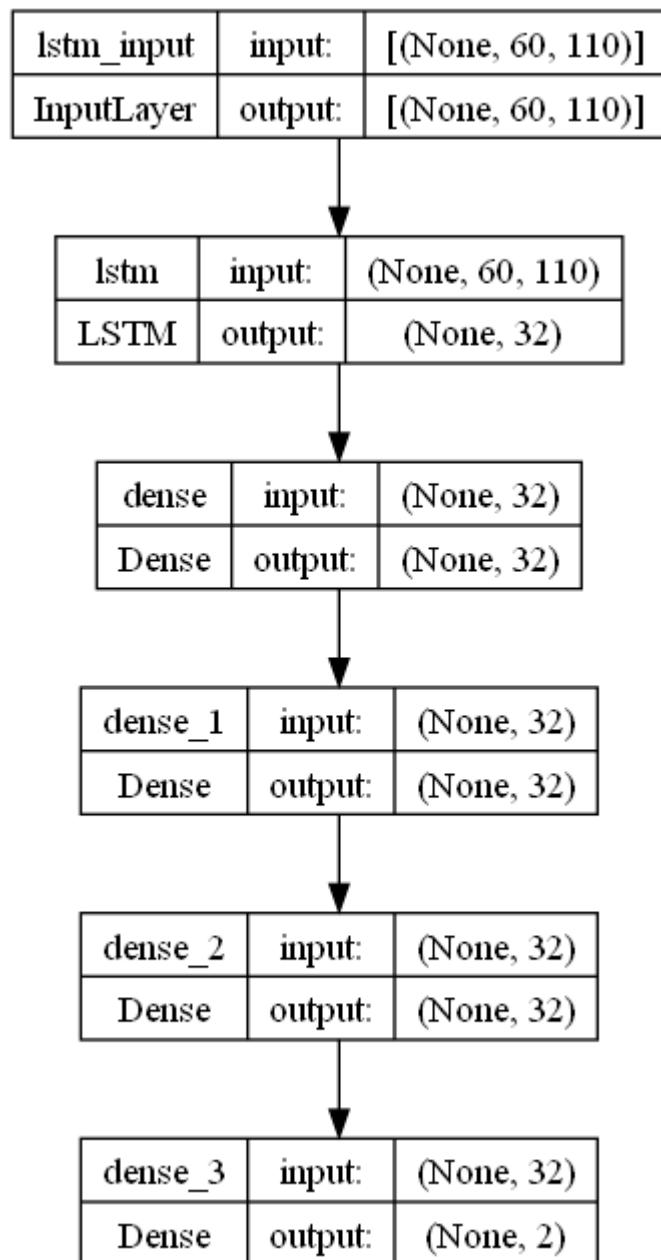


Рисунок 3.26 – Структура оптимальної моделі

3.7.4 Тренування моделі та огляд результатів (рис. 3.27)

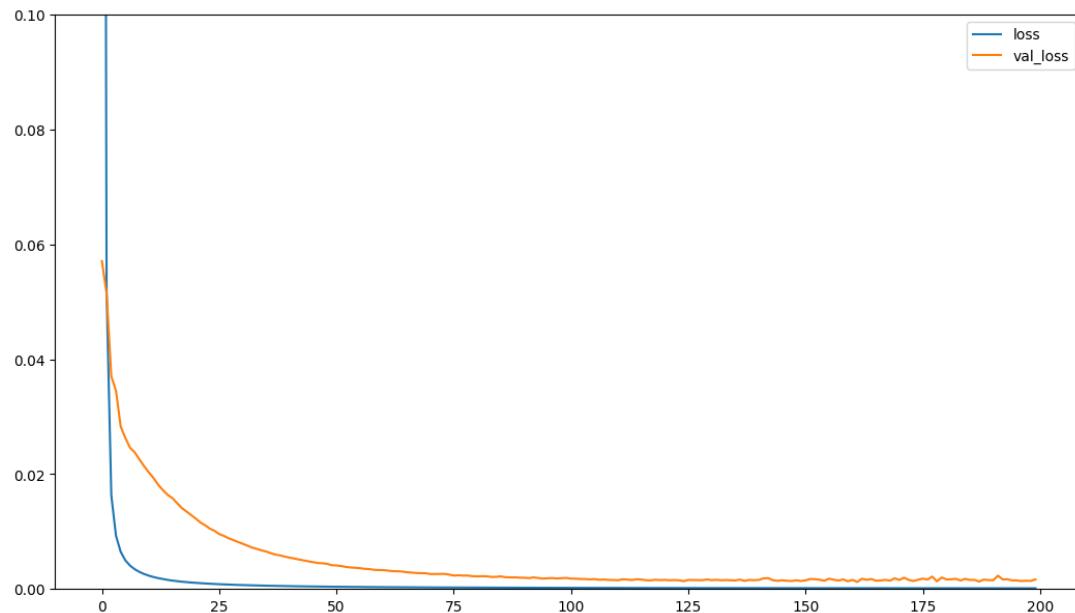


Рисунок 3.27 – Похибки навчальної та валідаційної вибірок

Таблиця 3.8 – Результати

Модель	Час навчання	Кількість епох	Похибка валідаційної вибірки	Похибка тренувальної вибірки
LSTM	5:33	140	6.184e-4	5.903e-5

3.8. Створення моделі на основі 1D-CNN шару і підбір гіперпараметрів

3.8.1. Підготовка даних для 1D-CNN шару

1D-CNN шар приймає дані у форматі тривимірного тензора, так само як і LSTM. Розміри цього тензора визначаються таким чином: (кількість прикладів, кількість кроків часу, кількість ознак).

Візьмемо 60 кроків часу.

Для цього беремо по 60 строк від останнього (-1) запису, потім від передостаннього (-2) і т.д.

Отримуємо:

З початкової таблиці розмірністю 381403*110, отримали тензор розмірністю 381344*60*110.

3.8.2. Структура оптимальної моделі (рис. 3.28)

Через більш високу кількість обчислень - час тренування моделі дуже високий, тому модель розраховувалась лише на одній комбінації гіперпараметрів. Так як для шару 1D-CNN кожна ознака потребує окремий шар - модель виявилася дуже “широкою”, тому на рисунку - вигляд моделі, який вона б мала, при наявності лише 3 ознак (фічей).

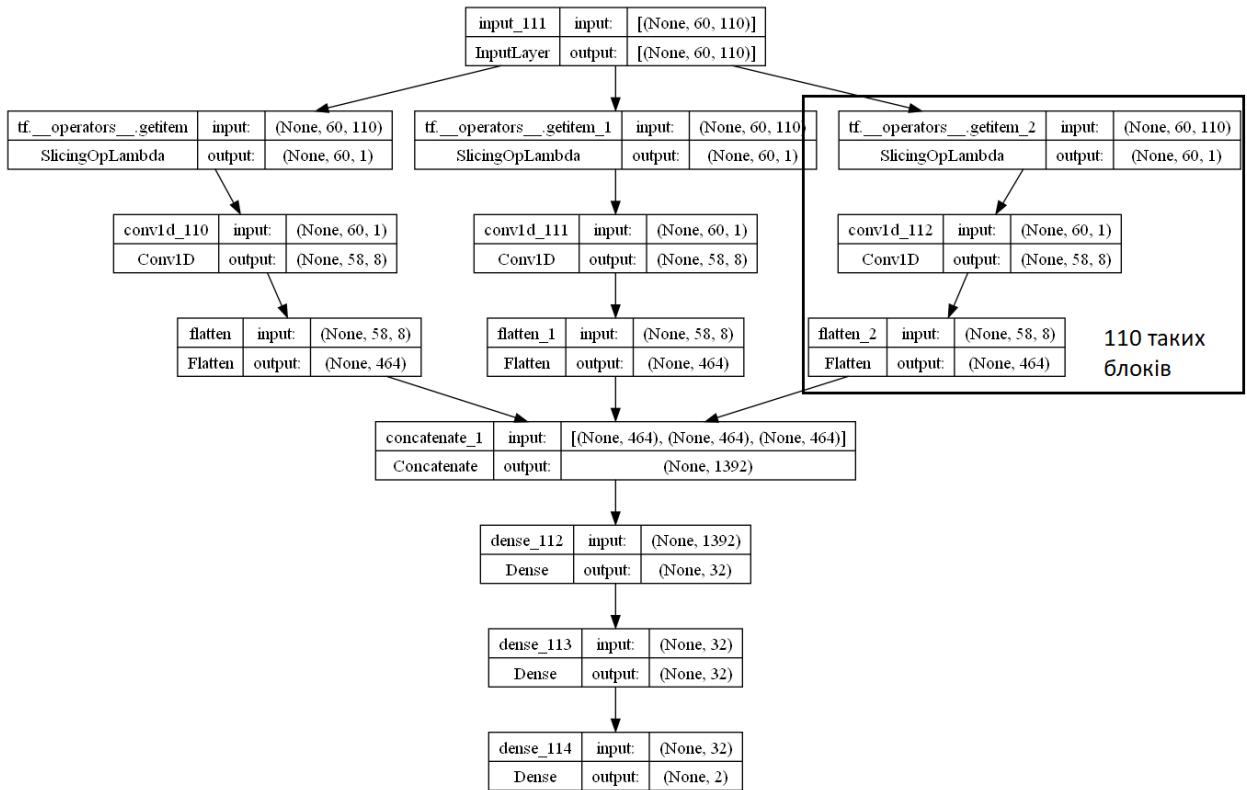


Рисунок 3.28 – Структура моделі

3.8.3. Тренування моделі і огляд результатів (рис 3.29)

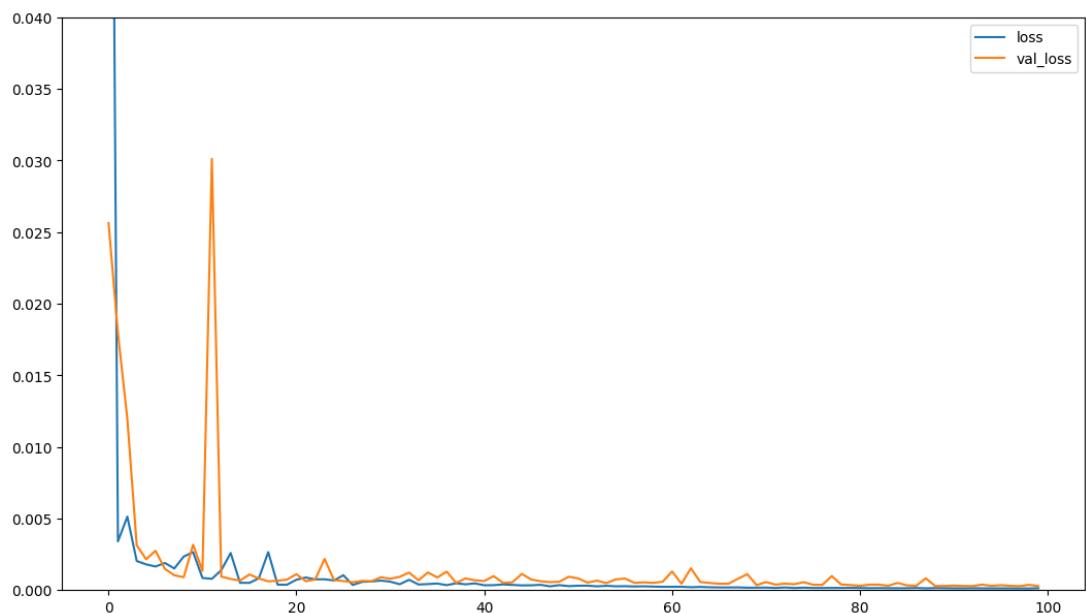


Рисунок 3.29 – Похибки навчальної та валідаційної вибірок

Таблиця 3.9 – Результати

Модель	Час навчання	Кількість епох	Похибка валідаційної вибірки	Похибка тренувальної вибірки
LSTM	5:33	140	6.184e-4	5.903e-5
1D-CNN	43:08	100	2.885e-4	1.358e-4

3.9. Створення ансамблевої моделі

3.9.1. Структура ансамблевої моделі (рис 3.30)

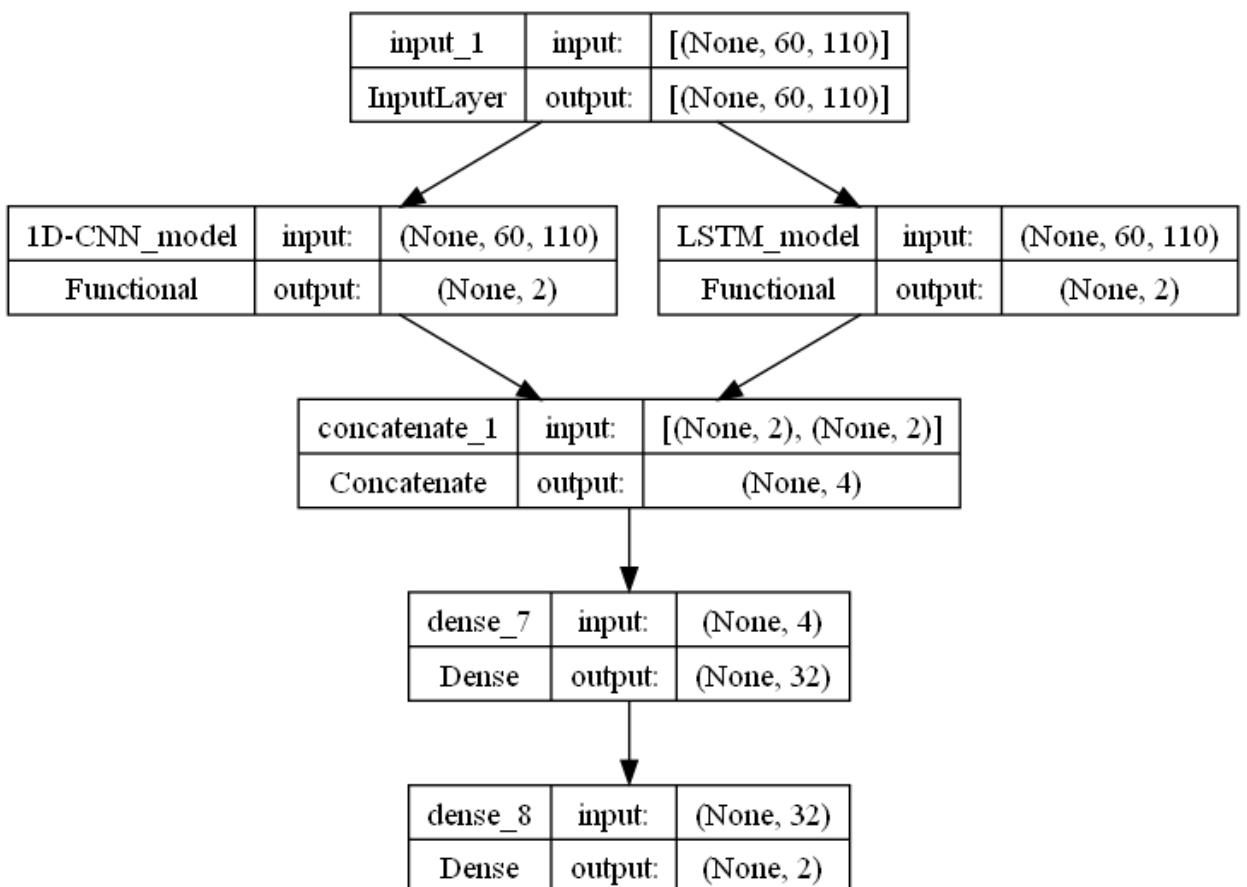


Рисунок 3.30 – Структура ансамблевої моделі

3.9.2. Тренування моделі і огляд результатів (рис 3.31)

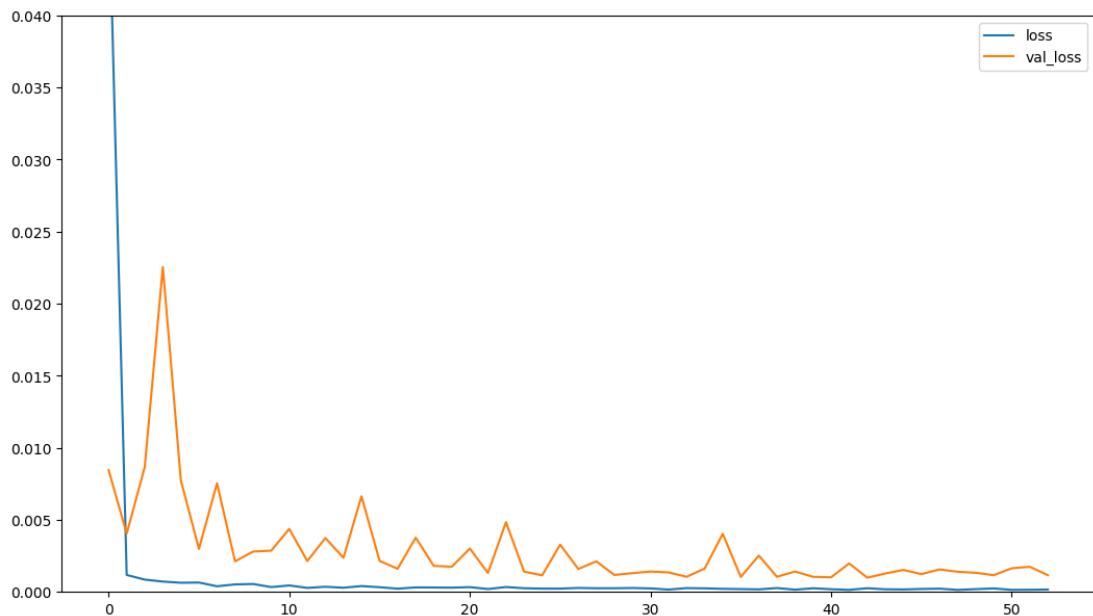


Рисунок 3.31 – Похибки навчальної та валідаційної вибірок

Таблиця 3.10 – Остаточні результати

Модель	Час навчання	Кількість епох	Похібка валідаційної вибірки	Похібка тренувальної вибірки
LSTM	5:33	140	6.184e-4	5.903e-5
1D-CNN	43:08	100	2.885e-4	1.358e-4
Ensemble	22:45	53	9.904e-4	1.841e-4

3.10 Висновки до розділу 3

У даному розділі було повністю описано весь процес дослідницької роботи, починаючи зі створення БД і автоматизації збору даних.

Було розроблено алгоритм виділення оптимальних точок закриття і відкриття контрактів, для подальшого створення цільового показника для алгоритмів машинного навчання.

На основі необроблених даних було згенеровано нові ознаки: по-хвилинні і по-контрактні, розроблено користувачьку метрику, більш оптимальну для задачі управління активами ніж звичайні MSE, R2 тощо.

Категоріальні показники було перетворено на бінарні вектори, а всі числові показники було стандартизовано.

Двовимірну матрицю значень розмірністю (кількість спостережень * кількість ознак) для обробки шарами LSTM і 1D-CNN було перетворено на тривимірний тензор розмірністю (кількість спостережень * кількість часових кроків * кількість ознак).

За принципом Nested Cross-Validation генеральну популяцію було розбито на тренувальні і тестові відрізки.

Було налаштовано автоматичний пошук гіперпараметрів. Зі знайденими гіпер-параметрами було натреновано три мережі. На основі LSTM шару, на основі 1D-CNN шару та на основі ансамблю цих двох моделей. Через брак ресурсів і часу для тренування моделей - не можна напевно стверджувати, що знайдені архітектури мереж є найоптимальнішими, оскільки:

- 1) множина можливих комбінацій гіперпараметрів неповна;
- 2) початкові ваги нейронної мережі випадково ініціалізуються перед тренуванням; ці ваги впливають на те, які важливі функції та особливості модель може навчитися розпізнавати; так як ми тренували кожну мережу лише по одному разу - результати можуть значно змінитись при перетренуванні навіть вже існуючих мереж.

Як можна покращити результати:

- 1) хоча й було налаштовано систему сповіщень, в даних все одно наявні пропуски в моментах, коли виникали помилки з боку Binance API або

Python додатку. Так як пропуски даних виникали не часто і зазвичай не перевищували 20 хвилин - дані було заповнено методом інтерполяції;

- 2) реалізувати фільтрацію ознак;
- 3) пошук моментів відкриття/закриття був реалізований спираючись на ціну відкриття свічки. Оптимальніше було б це зробити спираючись на ознаки “low_border”, “high_border”, створені пізніше;
- 4) реалізувати автоматичну генерацію ознак за допомогою бібліотек tsfresh/tsfel;
- 5) реалізувати моделі на основі дерев рішень (RandomForest) і моделі на основі градієнтного бустингу (XGBoost, LightGBM, CatBoost) та інших моделей, додати їх до ансамблевої моделі.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО–ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

Метою даного розділу є огляд альтернативних варіантів реалізації програмного продукту (ПП), що досліджує ефективність алгоритмів машинного навчання для задачі управління активами на ринку криптовалют з урахуванням оцінки довіри користувачів, з подальшим вибором оптимального на основі визначених характеристик, вимог та економічних факторів.

Дане дослідження здійснюється методами функціонально–вартісного аналізу (ФВА) – технології, що дозволяє оцінити реальну (не залежну від внутрішніх чинників) вартість продукту та виявити резерви щодо зменшення витрат за рахунок оптимізації виробництва, пошуку кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення.

Алгоритм ФВА складається з визначення послідовності етапів розробки продукту, повних витрат (річних) та їхніх джерел та кількості робочих часів, кінцевого розрахунку вартості ПП.

4.1 Постановка задачі проектування

У даній роботі метод ФВА застосовується для проведення техніко–економічного аналізу розробки ПП для дослідження ефективності машинного навчання для задачі управління активами на ринку криптовалют. Рішення стосовно проектування та реалізації компонентів цієї системи мають задовольнятися кожною її підсистемою, тому аналіз функцій ПП і буде фактичним аналізом.

Цільовий ПП має наступні технічні вимоги:

- 1) функціонування на персональних комп'ютерах зі стандартною конфігурацією;
- 2) доступ до даних у реальному часі;
- 3) швидкість обробки даних та отримання прогнозів;
- 4) зручний доступ до результатів дослідження;
- 5) мінімальні витрати на розробку та впровадження.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка ПП для дослідження ефективності алгоритмів машинного навчання у задачі управління активами на ринку криптовалют з урахуванням оцінки довіри користувачів. На її основі можна визначити наступні основні функції:

- 1) F_1 – вибір мови програмування;
- 2) F_2 – вибір бібліотеки для побудови нейронних мереж;
- 3) F_3 – вибір середовища розробки.

Кожна з вищезазначених функцій має декілька варіантів реалізації:

- 1) функція F_1 :
 - a) Python;
 - б) C++;
- 2) функція F_2 :
 - a) PyTorch;
 - б) TensorFlow;
- 3) функція F_3 :
 - a) Jupyter Notebook;
 - б) Visual Studio.

Для спрощення аналізу доцільно будувати морфологічну карту системи, що відображає множину всіх можливих варіантів їхньої реалізації (рис. 4.1).

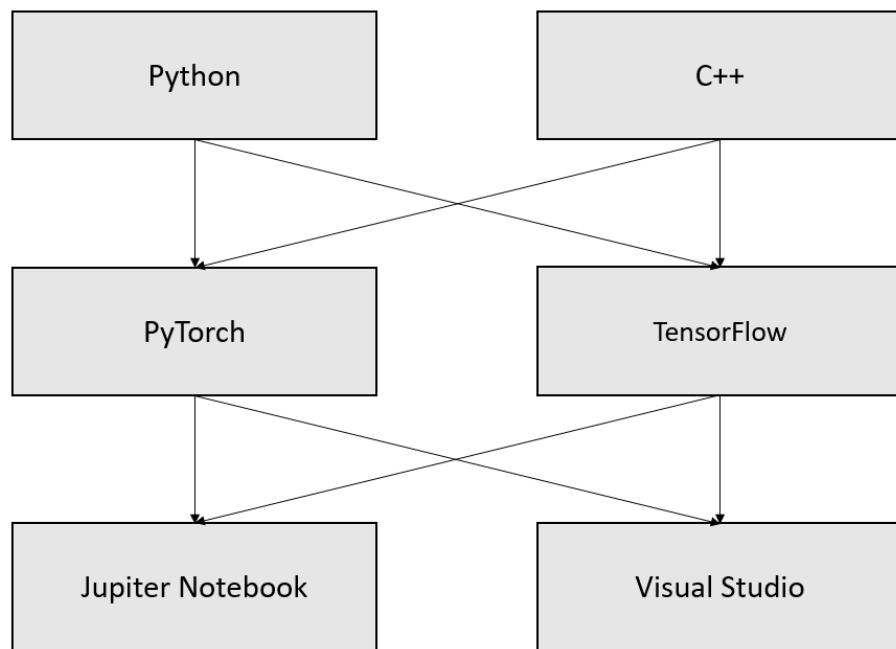


Рисунок 4.1 – Морфологічна карта

Позитивно–негативна матриця системи представлена у таблиці 4.1.

Таблиця 4.1 – Позитивно–негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	A	Простота синтаксису, висока продуктивність, широкий вибір бібліотек та фреймворків, розширюваність	Низька швидкість виконання, використання значного об'єму пам'яті
	B	Кросплатформеність, об'єктно–орієнтований підхід, розширюваність, ефективність	Складність синтаксису, велика кількість правил, необхідність керувати життєвим циклом об'єктів

Продовження таблиці 4.1

Функції	Варіанти реалізації	Переваги	Недоліки
F_2	A	Простий інтерфейс, динамічний граф обчислень, висока продуктивність, жива екосистема	Обмежена підтримка, невеликий обсяг вбудованих функцій
	B	Широкий функціонал, сумісність з іншими бібліотеками, розширюваність, зручна та повна документація	Складність у роботі та налаштуванні, використання великого об'єму пам'яті
F_3	A	Інтерактивність, розширюваність, підтримка різних мов програмування, легка візуалізація та відтворюваність даних, ефективність	Обмеження продуктивності, складність при роботі з великими проектами
	B	Потужність та функціональність, кросплатформеність, розширюваність та інтегрованість	Високі вимоги до обчислювальної потужності, великий об'єм встановлюваного пакету, складність інтерфейсу, ліцензування

З позитивно–негативної матриці видно, що певні переваги достатньо вагомими і перекриваються недоліками. На основі зіставлення цих даних між альтернативами можна відкинути варіанти, що не відповідають вимогам ПП.

Оновлені варіанти реалізації для основних функцій будуть наступними:

1. F_1 : доцільно надати перевагу простоті та багатофункціональноті у вигляді варіанту А.
2. F_2 : обидва варіанти (А та Б) є прийнятними, оскільки є повноцінними альтернативами одне одного.
3. F_3 : доцільно надати перевагу інтерактивності та широкій підтримці у вигляді варіанту Б.

Таким чином, будемо розглядати такий варіант реалізації ПП:

1. $F_1A - F_2A - F_3B$;
2. $F_1A - F_2B - F_3B$.

4.3 Обґрунтування системи параметрів програмного продукту

На основі вищерозглянутих даних визначаються основні параметри вибору, що фігуруватимуть при розрахунку коефіцієнта технічного рівня.

Для характеристики програмного продукту використовуються наступні параметри:

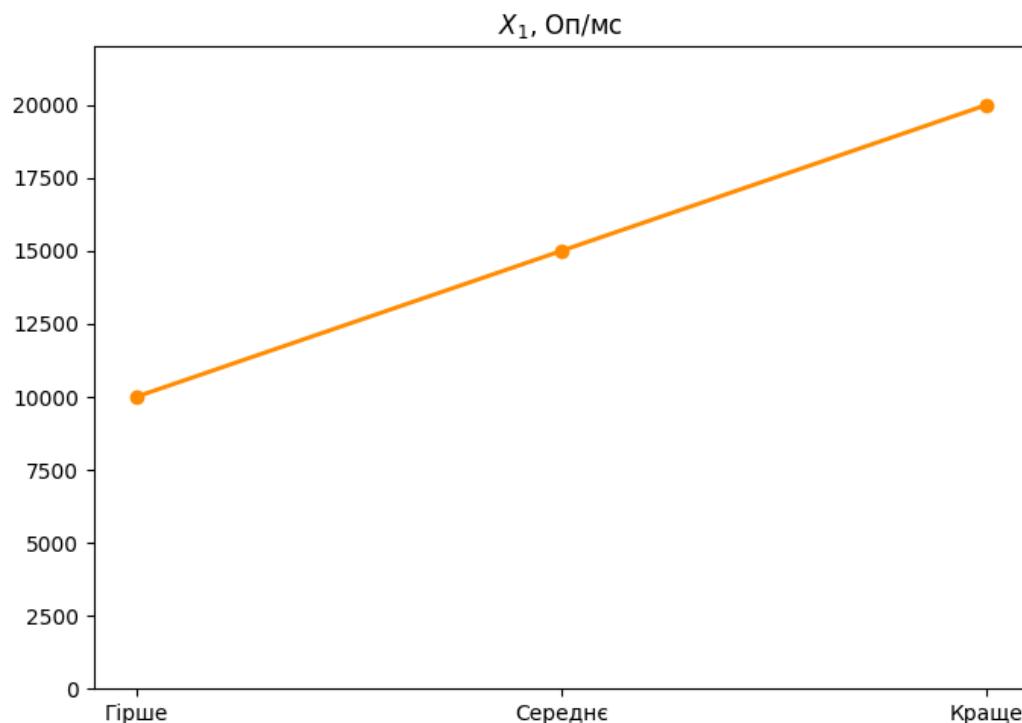
- 1) X_1 – швидкодія мови програмування (вимірюється у кількості операцій на мілісекунду);
- 2) X_2 – об'єм оперативної пам'яті (вимірюється у гігабайтах);
- 3) X_3 – час повної попередньої обробки даних (за даного рівня складності найдоцільніше вимірювати у мілісекундах);
- 4) X_4 – час навчання моделі (за даного рівня складності найдоцільніше вимірювати у хвилинах).

Гірші, середні та кращі значення параметрів визначаються експертно, спираючись на вимоги до розробки та експлуатації програмного продукту (табл. 4.2).

Таблиця 4.2 – Основні параметри програмного продукту

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X_1	Оп/мс	10000	15000	20000
Об'єм оперативної пам'яті	X_2	ГБ	4	16	32
Час попередньої обробки даних	X_3	хв	30	20	10
Час навчання моделі	X_4	хв	20	10	5

За даними таблиці 4.3 будуються графічні характеристики параметрів (рис. 4.2 – рис. 4.5).

Рисунок 4.2 – X_1 , швидкодія мови програмування

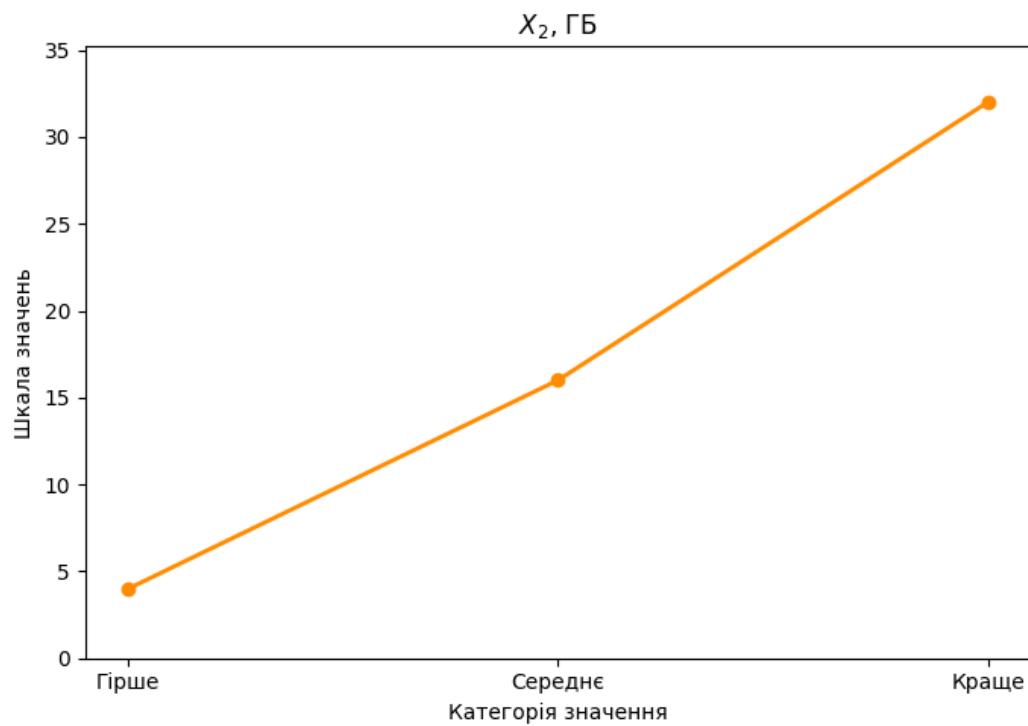


Рисунок 4.3 – X_2 , об'єм оперативної пам'яті

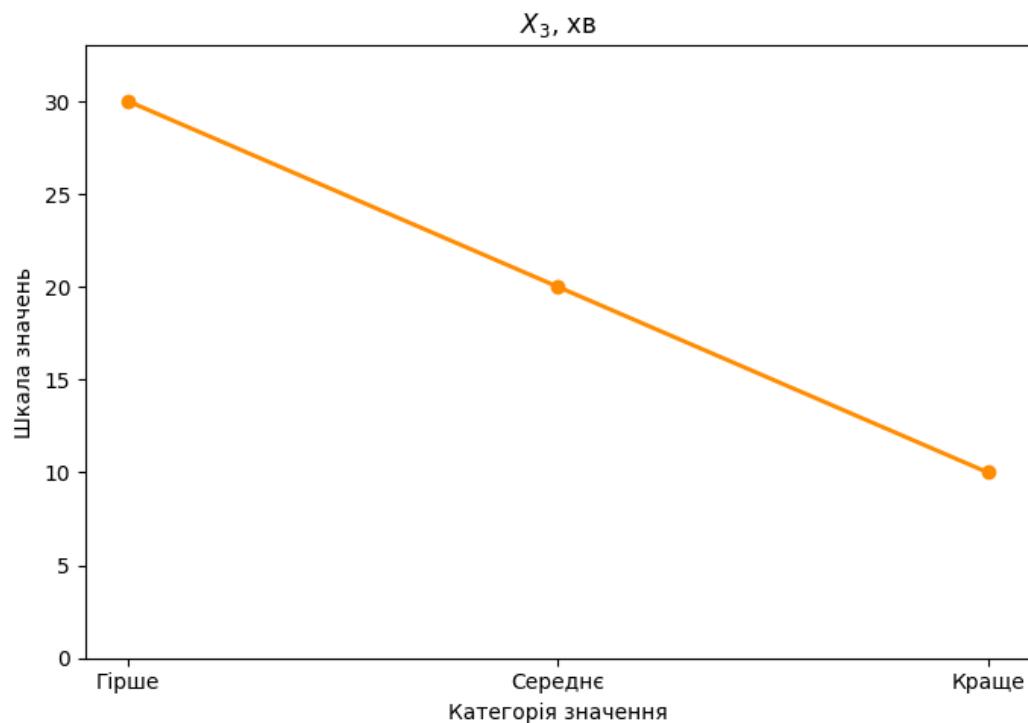


Рисунок 4.4 – X_3 , час попередньої обробки даних

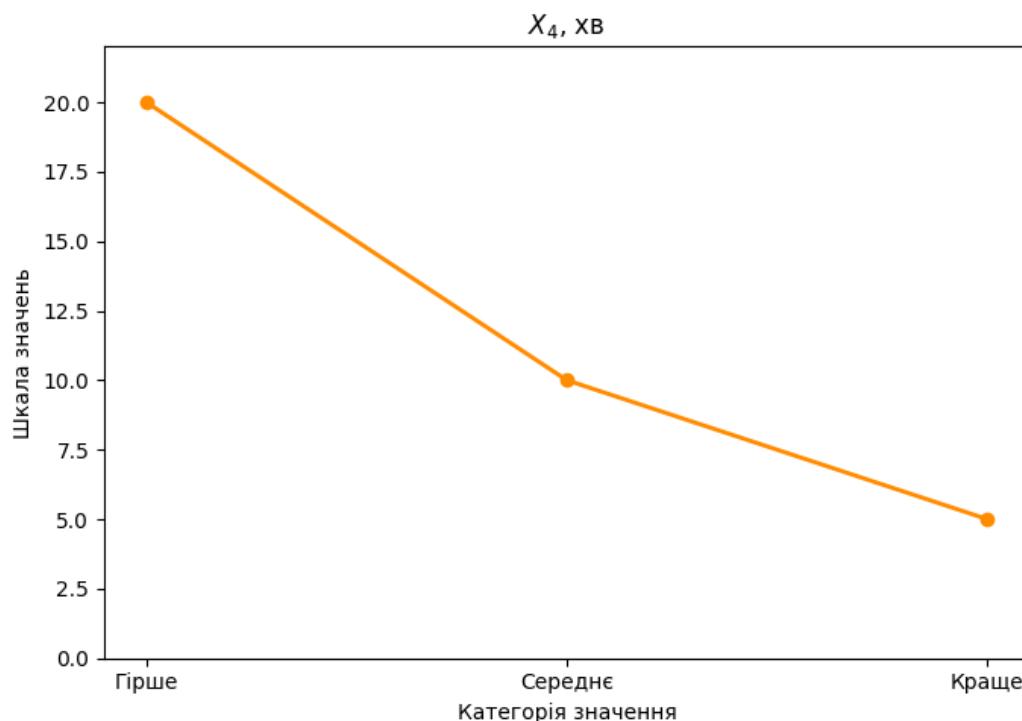


Рисунок 4.5 – X_4 , потенційний об'єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Значимість кожного параметра визначається методом попарного порівняння на основі оцінювання експертною комісією з 7 осіб, зважаючи на завдання розробити ПП, що дає найбільш точні результати.

Визначення коефіцієнтів значимості передбачає:

- 1) визначення рівня значимості параметра шляхом присвоєння різних рангів;
- 2) перевірку придатності експертних оцінок для подальшого використання;
- 3) визначення оцінки попарного пріоритету параметрів;
- 4) обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів Ri	Відхилення Δi	Δi2
			1	2	3	4	5	6	7			
X_1	Швидкодія мови програмування	Оп/мс	1	1	2	1	1	1	2	9	-8,5	72,25
X_2	Об'єм оперативної пам'яті	ГБ	2	3	1	2	2	2	1	13	-4,5	20,25
X_3	Час попередньої обробки даних	с	4	4	4	3	4	4	4	27	9,5	90,25
X_4	Час навчання моделі	мс	3	2	3	4	3	3	3	21	3,5	12,25
	Разом		10	10	10	10	10	10	10	70	0	195

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

- a) суму рангів кожного з параметрів і загальна сума рангів (табл. 4.3):

$$R_i = \sum_{j=1}^N r_{ij}, \quad (4.1)$$

$$R_{ij} = \frac{Nn(n+1)}{2}, \quad (4.2)$$

де N – число експертів;

n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij}, \quad (4.3)$$

$$T = \frac{1}{7} * 70 = 17,5$$

в) відхилення суми рангів кожного параметра від середньої суми рангів, сума яких повинна дорівнювати 0 (табл. 4.3):

$$\Delta_i = R_i - T. \quad (4.4)$$

г) загальна сума квадратів відхилення (табл. 4.3):

$$S = \sum_{i=1}^N \Delta_i^2, \quad (4.5)$$

Обчислимо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)}, \quad (4.6)$$

$$W = \frac{12 \cdot 195}{7^2(4^3 - 4)} = 0,796 > W_k = 0,67.$$

Експертне ранжування можна вважати достовірним, оскільки коефіцієнт узгодженості перевищує нормативний, що дорівнює 0,67.

На основі результатів ранжування, проводиться попарне порівняння всіх параметрів, результати якого подані у таблиці 4.4.

Таблиця 4.4 – Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X_1 і X_2	<	<	>	<	<	<	>	<	0,5
X_1 і X_3	<	<	<	<	<	<	<	<	0,5
X_1 і X_4	<	<	<	<	<	<	<	<	0,5
X_2 і X_3	<	<	<	<	<	<	<	<	0,5
X_2 і X_4	<	>	<	<	<	<	<	<	0,5
X_3 і X_4	>	>	>	<	>	>	>	>	1,5

Числове значення, що визначає ступінь переваги i -го параметра над j -тим, a_{ij} визначається за формулою (таблиця 4.4):

$$a_{ij} = \begin{cases} 1.5 & \text{при } X_i > X_j \\ 1.0 & \text{при } X_i = X_j \\ 0.5 & \text{при } X_i < X_j \end{cases} \quad (4.7)$$

Отримані числові оцінки є компонентами матриці $A = \|a_{ij}\|$ (табл. 4.5).

Для кожного параметра проводиться розрахунок вагомості K_{Bi} за наступними формулами:

$$K_{Bi} = \frac{b_i}{\sum_{i=1}^n b_i} \quad (4.8)$$

$$b_i = \sum_{j=1}^N a_{ij} \quad (4.9)$$

Відносні оцінки розраховуються, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b'_i}{\sum_{i=1}^n b'_i}, \quad (4.10)$$

$$b'_i = \sum_{j=1}^N a_{ij} b_j \quad (4.11)$$

Відповідно до таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому достатньо двох ітерацій.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітерація		Друга ітерація	
	X_1	X_2	X_3	X_4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$
X_1	1	0,5	0,5	0,5	2,5	0,156	9,25	0,157
X_2	1,5	1	0,5	0,5	3,5	0,219	12,25	0,208
X_3	1,5	1,5	1	1,5	5,5	0,344	21,25	0,36
X_4	1,5	1,5	0,5	1	4,5	0,281	16,25	0,275
Всього:					16	1	59	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів X_2 (об'єм оперативної пам'яті), X_3 (час попередньої обробки даних) та X_4 (час навчання моделі) відповідають технічним вимогам умов функціонування даного ПП. Абсолютне значення параметра X_1 (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП (таблиця 4.6) розраховується за наступною формулою:

$$K_K(j) = \sum_{i=1}^n K_{bi,j} B_{i,j}, \quad (4.12)$$

де n – кількість параметрів;

K_{bi} – коефіцієнт вагомості i -го параметра;

B_i – оцінка i -го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F_1	A	X_1	15000	6	0,157	0,942
F_2	A	X_2	16	8	0,208	1,664
	B	X_3	12	7	0,36	2,52
F_3	A	X_4	8	8	0,275	2,2

За даними з таблиці 4.6 за формулою:

$$K_K = K_{TY}[F_{1k}] + K_{TY}[F_{2k}] + \dots + K_{TY}[F_{zk}], \quad (4.13)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 0,942 + 1,664 + 2,2 = 4,804,$$

$$K_{K2} = 0,942 + 2,52 + 2,2 = 5,662.$$

З розрахунків випливає, що кращим є другий варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП необхідно спочатку провести розрахунок трудомісткості.

Всі варіанти складаються з двох окремих завдань:

1. Проектування ПП;
2. Розробка ПП.

Завдання 1 за ступенем новизни відноситься до групи А (задачі, які передбачають використання принципово нових методів розробки, проведення науково–дослідних робіт), завдання 2 – до групи Б (типові проектні рішення, оригінальні задачі і системи, які не мають аналогів). За складністю алгоритмів, що використовуються у завданні 1 належать до групи 1 (алгоритми оптимізації та моделювання систем і об'єктів), а в завданні 2 – до групи 3 (алгоритми, які реалізують стандартні методи рішень, а також не передбачають використання складних чисельних і логічних методів).

Для реалізації завдання 1 використовується довідкова інформація та експертний досвід, завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$T_0 = T_P \cdot K_{\Pi} \cdot K_{CK} \cdot K_M \cdot K_{CT} \cdot K_{CT.M}, \quad (4.14)$$

де T_P – трудомісткість розробки ПП;

K_{Π} – поправочний коефіцієнт;

K_{CK} – коефіцієнт на складність вихідної інформації;

K_M – коефіцієнт рівня мови програмування;

K_{CT} – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{CT.M}$ – коефіцієнт стандартного математичного забезпечення.

У всіх завданнях використовується мова програмування високого рівня, тому коефіцієнт $K_M = K_{CT.P} = 1$, також вважаємо, що за рівня складності контролю вихідної інформації (змінного типу) 5,2, а вихідної – 5,4 $K_{CK} = 1$.

Відповідно до норм часу для завдань розрахункового характеру, для першого маємо $T_P = 90$ людино–днів, $K_{\Pi} = 1,7$, $K_{CT} = 0,9$; для другого – $T_P = T_P = 27$ людино–днів, $K_{\Pi} = 0,9$, $K_{CT} = 0,8$. Якщо не враховувати коефіцієнти, що дорівнюють 1, то загальна трудомісткість програмування для кожного з завдань дорівнює:

$$T_1 = 90 \cdot 1,7 \cdot 0,9 = 137,7 \text{людино – днів},$$

$$T_2 = 27 \cdot 0,9 \cdot 0,8 = 19,44 \text{людино – днів}.$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (137,7 + 17,01 + 4,8 + 19,44) \cdot 8 = 1431,6 \text{людино} - \text{годин},$$

$$T_{II} = (137,7 + 17,01 + 6,91 + 19,44) \cdot 8 = 1448,48 \text{людино} - \text{годин}.$$

Отже, найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 28000 грн. та один дата-аналітик з окладом 22000. Визначимо середню зарплату за годину за формулою:

$$CЧ = \frac{M}{T_m \cdot t} \text{ грн.}, \quad (4.15)$$

де M – місячний оклад працівників;

T_m – кількість робочих днів на місяць;

t – кількість робочих годин на день.

$$CЧ = \frac{2 \cdot 28000 + 22000}{3 \cdot 21 \cdot 8} = 154,76 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою:

$$C_{зп} = CЧ \cdot T_i \cdot KД, \quad (4.16)$$

де $CЧ$ – величина погодинної оплати праці працівника;

T_i – трудомісткість відповідного завдання;

$KД$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$I. C_{3\Pi} = 154,76 \cdot 1431,6 \cdot 1,2 = 265865,3 \text{ грн.}$$

$$II. C_{3\Pi} = 154,76 \cdot 1448,48 \cdot 1,2 = 269000,12 \text{ грн.}$$

Відрахування на єдиний соціальний внесок (ЕСВ), що становить 22%, обчислюється наступним чином:

$$I. C_{\text{ВІД}} = C_{3\Pi} \cdot 0,22 = 265865,3 \cdot 0,22 = 58490,37 \text{ грн.}$$

$$II. C_{\text{ВІД}} = C_{3\Pi} \cdot 0,22 = 269000,12 \cdot 0,22 = 59180,03 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино–години. (СМ)

Оскільки одна ЕОМ обслуговує одного програміста з окладом 28000 грн., з коефіцієнтом зайнятості 0,2, отримаємо:

$$C_{\Gamma} = 12 \cdot M \cdot K_3 = 12 \cdot 28000 \cdot 0,2 = 67200 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{3\Pi} = C_{\Gamma} \cdot (1 + K_3) = 67200 \cdot (1 + 0,2) = 80640 \text{ грн.}$$

Відрахування на ЕСВ:

$$C_{\text{ВІД}} = C_{3\Pi} \cdot 0,22 = 80640 \cdot 0,22 = 17740,8 \text{ грн.}$$

Амортизаційні відрахування розраховуємо для амортизації 30% та вартості ЕОМ 30000 грн.

$$C_A = K_{TM} \cdot K_A \cdot I_{PP} = 1,15 \cdot 0,3 \cdot 30000 = 10350 \text{ грн.}$$

де K_{TM} – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

K_A – річна норма амортизації;

\mathcal{C}_{PR} – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$C_P = K_{TM} \cdot \mathcal{C}_{PR} \cdot K_P = 1,15 \cdot 30000 \cdot 0,05 = 1725 \text{ грн.}$$

де K_P – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$T_{EF} = (\Delta_K - \Delta_B - \Delta_C - \Delta_P) \cdot t \cdot K_B = (365 - 104 - 12 - 16) \cdot 8 \cdot 0,9 = 1677,6 \text{ години},$$

де Δ_K – календарна кількість днів у році;

Δ_B , Δ_C – відповідно кількість вихідних та свяtkovих днів;

Δ_P – кількість днів планових ремонтів устаткування;

t – кількість робочих годин в день, K_B – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$C_{EL} = T_{EF} \cdot N_C \cdot K_3 \cdot \mathcal{C}_{EH} = 1677,6 \cdot 0,3 \cdot 0,8 \cdot 4,79 = 1928,57 \text{ грн.}$$

де N_C – середньо-споживча потужність приладу;

K_3 – коефіцієнтом зайнятості приладу;

\mathcal{C}_{EH} – тариф за 1 КВт–годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$C_H = \mathcal{C}_{PR} \cdot 0,67 = 30000 \cdot 0,67 = 20100 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$C_{EKC} = C_{3\Pi} + C_{BID} + C_A + C_P + C_{EL} + C_H, \quad (4.17)$$

$$C_{EKC} = 80640 + 17740,8 + 10350 + 1725 + 1928,57 + 20100 = 132484,37 \text{ грн.}$$

Собівартість однієї машино–години ЕОМ дорівнюватиме:

$$C_{M-G} = \frac{C_{EKC}}{T_{EF}} = \frac{132484,37}{1677,6} = 78,97 \text{ грн/год.}$$

Оскільки всі роботи, пов’язані з розробкою ПП, проводяться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складатимуть:

$$C_M = C_{M-G} \cdot T, \quad (4.18)$$

$$\text{I. } C_M = 78,97 \cdot 1431,6 = 113053,45 \text{ грн.}$$

$$\text{II. } C_M = 78,97 \cdot 1448,48 = 114386,47 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$C_H = C_{3\Pi} \cdot 0,67, \quad (4.19)$$

$$\text{I. } C_H = 265865,3 \cdot 0,67 = 178129,75 \text{ грн.}$$

$$\text{II. } C_H = 269000,12 \cdot 0,67 = 180230,08 \text{ грн.}$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{PP} = C_{3\Pi} + C_{BID} + C_M + C_H, \quad (4.20)$$

$$\text{I. } C_{\text{ПП}} = 265865,3 + 58490,37 + 113053,45 + 178129,75 = 615538,87 \text{ грн.}$$

$$\text{II. } C_{\text{ПП}} = 269000,12 + 59180,03 + 114386,47 + 180230,08 = 622796,7 \text{ грн.}$$

4.7 Вибір кращого варіанту програмного продукту техніко–економічного рівня

Розрахуємо коефіцієнт техніко–економічного рівня для обох варіантів за формулою:

$$K_{\text{TEP}j} = \frac{K_{kj}}{C_{\Phi j}}, \quad (4.21)$$

$$K_{\text{TEP}1} = \frac{4,804}{615538,87} = 7,81 \cdot 10^{-6},$$

$$K_{\text{TEP}2} = \frac{5,662}{622796,7} = 9,09 \cdot 10^{-6}.$$

Отже, найбільш ефективним є варіант реалізації з більшим коефіцієнтом техніко–економічного – тобто другий варіант з $K_{\text{TEP}2} = 9,09 \cdot 10^{-6}$.

Найкращий варіант реалізації програмного продукту має наступні параметри за основними функціями.

1. Вибір мови програмування: Python.
2. Вибір для побудови нейронних мереж: TensorFlow.
3. Вибір середовища розробки: Jupyter Notebook.

Обраний варіант виконання програмного комплексу дозволяє розробникам зменшити вимоги до обчислювальної потужності та

використаного об'єму пам'яті, знизити витрати на виробництво та розробити більш гнучкий, зручний у плані роботи, отримання та візуалізації даних інтерфейс.

4.8 Висновки до розділу 4

У даному розділі було проведено повний функціонально–вартісний аналіз ПП та визначено оцінку його основних функцій.

Спочатку було визначено вимоги до ПП та його основні функціональні характеристики. Наступним кроком було обрано варіанти реалізації процесу розробки за кожною з визначених основних функцій, з яких розглядали лише доцільні з точки зору результатів позитивно–негативної матриці. Далі було визначено параметри, що характеризують програмний продукт, за результатами експертного оцінювання було проведено аналіз їхньої якості та застосовано розрахункові методи економічного аналізу для вибору найкращого програмного продукту.

ВИСНОВКИ

В процесі виконання даної роботи було пройдено усі основні етапи дослідження. Спочатку, заздалегідь було знайдено існуючі дані і налаштовано збір нових унікальних даних. Для цього було арендовано хмарний сервер, розгорнуто на ньому базу даних, розроблено і запущено на цьому сервері Python додаток, який щохвилини записував в цю базу даних різноманітні дані про ф'ючерсні контракти.

Було проведено дослідження предметної області і огляд існуючих рішень.

Після цього було завантажено зібраний дані і на їх основі було виділено точки для оптимального відкриття і закриття контрактів.

Потім було сгенеровано нові ознаки, перетворено дані і підготовано їх для обробки моделлю.

Було створено цільовий показник і користувальську метрику.

Далі, перш ніж тренувати моделі і порівнювати їх продуктивність, було підготовано розбиття часових даних на тренувальні і тестові вибірки, крос-валидація та пошук гіперпараметрів моделей.

Після цього було проведено тренування і порівняння результатів моделей.

Так як подібні дослідження потребують велику кількість обчислювальних ресурсів а також часу для перебору гіперпараметрів і тренування моделей — наведені моделі є лише суб-оптимальними і при подальшому налаштуванні гіперпараметрів і перетренуванні моделей їх продуктивність може суттєво покращитись.

Також, було зазначено деякі інші точки зросту що можуть покращити точність моделі, які через брак ресурсів не було реалізовано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Sokol D. D., Wetzer T. Exchanges and Alternative Trading Systems: Law and Regulation. 2017. 256 p.
2. Vigna P., Casey M. J. Cryptocurrency: How Bitcoin and Digital Money are Challenging the Global Economic Order Hardcover. Bodley Head, 2015. 384 p.
3. Hull J. C. Options, Futures, and Other Derivatives. Pearson, 2014. 896 p.
4. What are perpetual futures, quarterly futures and funding. *Binance*. URL: <https://www.binance.com/en/support/faq/what-are-perpetual-futures-and-quarterly-futures-d2a1afd5f829455c9ded23f0ca561a40>
5. Binance API documentation. *Binance*. URL: <https://binance-docs.github.io/apidocs/futures/en/>.
6. How to Create an ARIMA Model for Time Series Forecasting in Python. *Machine Learning Mastery*. URL: <https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/#:~:text=Autoregressive%20Integrated%20Moving%20Average%20Model,making%20skillful%20time%20series%20forecasts.>
7. Simple exponential smoothing. *OTexts*. URL: <https://otexts.com/fpp2/ses.html>
8. Holt–Winters' seasonal method. *OTexts*. URL: <https://otexts.com/fpp2/holt-winters.html>
9. Illustrated Guide to LSTM's and GRU's: A step by step explanation. *Towards Data Science*. URL: <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>
10. What does it mean by Bidirectional LSTM? *Analytics Vidhya*. URL: <https://medium.com/analytics-vidhya/what-does-it-mean-by-bidirectional-lstm-63d6838e34d9>.

- 11.Understanding Random Forest. *Towards Data Science.* URL:
<https://towardsdatascience.com/understanding-random-forest-58381e0602d2>
- 12.A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning. *Machine Learning Mastery.* URL:
<https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/>
- 13.1D Convolutional Neural Network Models for Human Activity Recognition. *Machine Learning Mastery.* URL: <https://machinelearningmastery.com/cnn-models-for-human-activity-recognition-time-series-classification/>
- 14.MAE, MSE, RMSE, Coefficient of Determination, Adjusted R Squared — Which Metric is Better? *Analytics Vidhya.* URL:
<https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e>
- 15.What is OVHcloud? *OVHcloud.* URL: <https://us.ovhcloud.com/about/>
- 16.Remote Development using SSH. *VisualStudio.* URL:
<https://code.visualstudio.com/docs/remote/ssh>
- 17.Python wrapper for the Binance Exchange REST API. *Binance.* URL:
<https://python-binance.readthedocs.io/en/latest/>
- 18.Schedule 1.2.0 documentation. *Schedule.* URL:
<https://schedule.readthedocs.io/en/stable/>
- 19.Telegram Bot API. *Telegram.* URL: <https://core.telegram.org/bots/api>
- 20.1D Convolutional Neural Network Models for Human Activity Recognition. *Machine Learning Mastery.* URL: <https://machinelearningmastery.com/cnn-models-for-human-activity-recognition-time-series-classification/>
- 21.How to do Time Series Split using Sklearn. *Medium.* URL:
https://medium.com/@Stan_DS/timeseries-split-with-sklearn-tips-8162c83612b9
- 22.Creating Custom Loss Functions in TensorFlow: Understanding the Theory and Practicalities. *Towards Data Science.* URL:

<https://towardsdatascience.com/creating-custom-loss-functions-in-tensorflow-understanding-the-theory-and-practicalities-383a19e387d6>

23. Introduction to the Keras Tuner. *Tensorflow.* URL:

https://www.tensorflow.org/tutorials/keras/keras_tuner

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

send_telegram_message.py

```

import requests
import json

requests.packages.urllib3.disable_warnings()

def send_telegram_message(message, chat_id, api_key):
    responses = {}
    headers = {'Content-Type': 'application/json',
               'Proxy-Authorization': 'Basic base64'}
    data_dict = {'chat_id': chat_id,
                 'text': message,
                 'parse_mode': 'HTML',
                 'disable_notification': True}
    data = json.dumps(data_dict)
    url = f"https://api.telegram.org/bot{api_key}/sendMessage"
    response = requests.post(url,
                             data=data,
                             headers=headers,
                             verify=False)
    return response

```

quarter_functions.py

```

from datetime import datetime, timezone
from dateutil.relativedelta import weekday, FR, relativedelta

def ceil_3(x):
    return (x + 2) // 3 * 3

def get_delivery(timestamp, type='timestamp'):
    types = ['timestamp', 'date']
    if type not in types:
        raise ValueError("Invalid type. Expected 'timestamp' or 'date'")

    time_from_timestamp = datetime.fromtimestamp(timestamp, tz=timezone.utc)

    delivery_date = time_from_timestamp +

```

```

relativedelta(month = ceil_3(time_from_timestamp.month)) + \
relativedelta(day=31, weekday=FR(-1), hour=8, minute=0, second=0)

if(delivery_date < time_from_timestamp):
    time_from_timestamp += relativedelta(months=2)
    delivery_date = time_from_timestamp + \
        relativedelta(month = ceil_3(time_from_timestamp.month)) + \
        relativedelta(day=31, weekday=FR(-1), hour=8, minute=0, second=0)

if(delivery_date < time_from_timestamp):
    raise Exception

if type=='timestamp':
    return int(delivery_date.timestamp())
if type=='date':
    return delivery_date

def get_quarter_time_left(timestamp):
    return get_delivery(timestamp) - timestamp

def get_quarter_symbol(pair, timestamp):
    delivery = get_delivery(timestamp, type='date')
    return f'{pair}_{delivery.year%100}{str(delivery.month).zfill(2)}{delivery.day}'"

```

script.py

```

from sshtunnel import SSHTunnelForwarder
import pymysql

import schedule
import pytz
from datetime import datetime, timedelta
from dateutil.relativedelta import FR, relativedelta
from time import sleep

import pandas as pd
import numpy as np

from sqlalchemy import create_engine
import textwrap
import logging
import configparser
from binance.client import Client

```

```
from quarter_functions import get_quarter_time_left, get_quarter_symbol

from send_telegram_message import send_telegram_message

logging.basicConfig(filename='logs.log', level=logging.INFO,
                    format='%(asctime)s:%(levelname)s:%(message)s')

config = configparser.ConfigParser()
config.read('api_data.ini')
client = Client(config['binance']['api_key'], config['binance']['api_secret'])

def run_sql_script(script):
    mysql_connect()
    cursor = connection.cursor()
    cursor.execute(script)
    connection.close()
    return cursor.fetchall()

def connect_ssh_tunnel(ssh_address, ssh_username, ssh_password):
    try:
        global tunnel
        tunnel = SSHTunnelForwarder((ssh_address),
                                     ssh_username=ssh_username,
                                     ssh_password=ssh_password,
                                     ssh_proxy_enabled=True,
                                     remote_bind_address=('localhost', 3306))
        tunnel.start()
    except BaseException as e:
        logging.info('ERROR: {}'.format(e))
    finally:
        if tunnel:
            tunnel.close

def mysql_connect():
    global connection
    connection = pymysql.connect(
        host = config['mysql']['host'],
        user = config['mysql']['user'],
        passwd = config['mysql']['passwd'],
        port = tunnel.local_bind_port
    )

def create_sql_engine():
    global engine
    engine = create_engine("mysql+pymysql://{}:{}@{}:{}{}".format(
        user = config['mysql']['user'],
        passwd = config['mysql']['passwd'],
        host = config['mysql']['host'],
        port = config['mysql']['port'],
        exchange_data
    ))
```

```

host = config['mysql']['host'],
port = tunnell.local_bind_port))

def load_historical_data():
    df_perpetual = pd.DataFrame()
    df_cq = pd.DataFrame()
    futures_history = pd.DataFrame()

    endTime = run_sql_script('SELECT MIN(timestamp) FROM `exchange data`.BTC_perpetual')
    endTime = endTime[0][0] * 1000 - 30000

    endTime_iterator = endTime
    shape_before, shape_after = -1, 0
    while(shape_before != shape_after):
        for i in range(20):
            shape_before = df_perpetual.shape[0]
            df_perpetual = df_perpetual.append(pd.DataFrame(
                client.futures_continous_klines(
                    pair='BTCUSDT',
                    contractType='PERPETUAL',
                    interval='1m',
                    endTime = endTime_iterator,
                    limit = 1500
                )
            ).iloc[:-1])
            shape_after = df_perpetual.shape[0]
            endTime_iterator = df_perpetual.iloc[-1,0] - 30000
            print(shape_after, end = '>')
        print('\n')
        endTime_iterator = endTime
        shape_before, shape_after = 0, 1
        while(shape_before != shape_after):
            for i in range(20):
                shape_before = df_cq.shape[0]
                df_cq = df_cq.append(pd.DataFrame(
                    client.futures_continous_klines(
                        pair='BTCUSDT',
                        contractType='CURRENT_QUARTER',
                        interval='1m',
                        endTime = endTime_iterator,
                        limit = 1500
                    )
                ).iloc[:-1])
                shape_after = df_cq.shape[0]
                endTime_iterator = df_cq.iloc[-1,0] - 30000
                print(shape_after, end = '>')
            print('\n')
    # параметр endTime не работает, пришлось немного подругому

```

```

startTime = 1568102400000
shape_before, shape_after = 1, 0
while(shape_before != shape_after):
    shape_before = futures_history.shape[0]
    futures_history = futures_history.append(
        pd.DataFrame.from_dict(client.futures_funding_rate(
            symbol = 'btcusdt',
            startTime = startTime,
            limit = 1000
        )).iloc[:,1:])
    )
    shape_after = futures_history.shape[0]
    startTime = futures_history.iloc[-1,0] + 30000
    print(shape_after, end = '>')
print('\n')
df_perpetual.drop(columns = [6,7,9,10,11], inplace = True)
df_perpetual.reset_index(drop = True, inplace= True)
df_perpetual.columns = ['timestamp', 'open', 'max', 'min', 'close', 'volume', 'number_of_trades']
df_perpetual['timestamp']//=1000

df_cq.drop(columns = [6,7,9,10,11], inplace = True)
df_cq.reset_index(drop = True, inplace= True)
df_cq.columns = ['timestamp', 'open', 'max', 'min', 'close', 'volume', 'number_of_trades']
df_cq['timestamp']//=1000

futures_history['fundingTime']//=1000

futures_history = pd.DataFrame(np.repeat(futures_history.values, 480, axis=0), columns=['timestamp','funding_rate'])
futures_history['timestamp'] = futures_history.apply(lambda row: row['timestamp'] - 60 * (480 - row.name%480-1), axis= 1)

df_perpetual = df_perpetual.merge(futures_history, how='left', on='timestamp')
df_cq['time_left'] = df_cq['timestamp'].apply(lambda x: get_quarter_time_left(x))

try:
    df_cq.to_sql('BTC_current_quarter', engine, if_exists = 'append', index = False)
    df_perpetual.to_sql('BTC_perpetual', engine, if_exists = 'append', index = False)
except BaseException as e:
    send_telegram_message('ERROR: {}'.format(e), config['telegram']['heorhii'],
config['telegram']['token_error'])
    send_telegram_message('ERROR: {}'.format(e), config['telegram']['nikita'],
config['telegram']['token_error'])
    logging.info('ERROR: {}'.format(e))

def backup_from_last_record():
    df_perpetual = pd.DataFrame()
    df_cq = pd.DataFrame()

```

```

startTime_perp = run_sql_script('SELECT MAX(timestamp) FROM `exchange
data` .BTC_perpetual')[0][0]
startTime_cq = run_sql_script('SELECT MAX(timestamp) FROM `exchange
data` .BTC_current_quarter')[0][0]

if ((datetime.now().timestamp() - min(startTime_perp, startTime_cq)) < 60):
    logging.info('No data to backup')
    return

startTime_perp = (startTime_perp + 30) * 1000
startTime_cq = (startTime_cq + 30) * 1000

logging.info('Starting restoring data from: {} / {}'.format(startTime_perp, startTime_cq))

startTime_iterator = startTime_perp
while startTime_iterator/1000 < datetime.today().timestamp():
    df_perpetual = df_perpetual.append(
        pd.DataFrame(client.futures_continuos_klines(
            pair='BTCUSDT',
            contractType='PERPETUAL',
            interval='1m',
            startTime = startTime_iterator,
            limit = 1500)
    )
)
startTime_iterator = df_perpetual.iloc[-1,0] + 30000

startTime_iterator = startTime_cq
while startTime_iterator/1000 < datetime.today().timestamp():
    df_cq = df_cq.append(
        pd.DataFrame(client.futures_continuos_klines(
            pair='BTCUSDT',
            contractType='CURRENT_QUARTER',
            interval='1m',
            startTime = startTime_iterator,
            limit = 1500)
    )
)
startTime_iterator = df_perpetual.iloc[-1,0] + 30000

logging.info('Data retrieved, {} / {} records successfully loaded'.format(df_perpetual.shape[0],
df_cq.shape[0]))

df_perpetual.drop(columns = [6,7,9,10,11], inplace = True)
df_perpetual.reset_index(drop = True, inplace = True)
df_perpetual.columns = ['timestamp', 'open', 'max', 'min', 'close', 'volume', 'number_of_trades']

```

```

df_perpetual['timestamp']//=1000

df_cq.drop(columns = [6,7,9,10,11], inplace = True)
df_perpetual.reset_index(drop = True, inplace = True)
df_cq.columns = ['timestamp', 'open', 'max', 'min', 'close', 'volume', 'number_of_trades']
df_cq['timestamp']//=1000

try:
    futures_history = pd.DataFrame.from_dict(
        client.futures_funding_rate(
            symbol = 'btcusdt',
            startTime = startTime_perp,
            limit = 1000
        ).iloc[:,1:]
    )
    futures_history['fundingTime']//=1000
    futures_history = pd.DataFrame(np.repeat(futures_history.values, 480, axis=0), columns=[
        'timestamp', 'funding_rate'])
    futures_history['timestamp'] = futures_history.apply(lambda row: row['timestamp'] - 60 * (480 - row.name%480-1), axis= 1)

    df_perpetual = pd.merge(df_perpetual, futures_history, how = 'left', on="timestamp")
    df_perpetual.fillna(client.futures_mark_price(symbol = 'BTCUSDT')['lastFundingRate'],
    inplace= True)
except BaseException:
    df_perpetual['funding_rate'] = client.futures_mark_price(symbol =
    'BTCUSDT')['lastFundingRate']
finally:
    pass

df_cq['time_left'] = df_cq['timestamp'].apply(lambda x: get_quarter_time_left(x))

try:
    df_cq.to_sql('BTC_current_quarter', engine, if_exists = 'append', index = False)
except BaseException as e:
    send_telegram_message('ERROR: {}'.format(e), config['telegram']['nikita'],
    config['telegram']['token_error'])
    send_telegram_message('ERROR: {}'.format(e), config['telegram']['heorhii'],
    config['telegram']['token_error'])
    logging.info('ERROR: {}'.format(e))

try:
    df_perpetual.to_sql('BTC_perpetual', engine, if_exists = 'append', index = False)
except BaseException as e:
    send_telegram_message('ERROR: {}'.format(e), config['telegram']['nikita'],
    config['telegram']['token_error'])
    send_telegram_message('ERROR: {}'.format(e), config['telegram']['heorhii'],
    config['telegram']['token_error'])
    logging.info('ERROR: {}'.format(e))

```

```

logging.info('data transferred successfully { }/{ } records added'.format(df_perpetual.shape[0],
df_cq.shape[0]))

def stream():
    last_perp = pd.DataFrame(client.futures_continuos_klines(pair='BTCUSDT',
contractType='PERPETUAL', interval='1m', limit=1))
    last_cq = pd.DataFrame(client.futures_continuos_klines(pair='BTCUSDT',
contractType='CURRENT_QUARTER', interval='1m', limit=1))

    last_perp.drop(columns = [6,7,9,10,11], inplace = True)
    last_perp.columns = ['timestamp', 'open', 'max', 'min', 'close', 'volume', 'number_of_trades']
    last_perp['timestamp']//=1000
    last_perp['funding_rate'] = client.futures_mark_price(symbol = 'BTCUSDT')['lastFundingRate']

    last_cq.drop(columns = [6,7,9,10,11], inplace = True)
    last_cq.columns = ['timestamp', 'open', 'max', 'min', 'close', 'volume', 'number_of_trades']
    last_cq['timestamp']//=1000
    last_cq['time_left'] = last_cq['timestamp'].apply(lambda x: get_quarter_time_left(x))

try:
    last_perp.to_sql('BTC_perpetual', engine, if_exists = 'append', index = False)
    last_cq.to_sql('BTC_current_quarter', engine, if_exists = 'append', index = False)
except BaseException as e:
    send_telegram_message('ERROR: {}'.format(e), config['telegram']['nikita'],
config['telegram']['token_error'])
    send_telegram_message('ERROR: {}'.format(e), config['telegram']['heorhii'],
config['telegram']['token_error'])

logging.info('ERROR: {}'.format(e))

def daily_notification():
    def format_timestamp(timestamp):
        x = datetime.fromtimestamp(int(timestamp))
        return '{}/{}/{} {}:{}'.format(str(x.day).zfill(2), str(x.month).zfill(2), str(x.hour).zfill(2),
str(x.minute).zfill(2))

    last_three_fr = pd.DataFrame.from_dict(client.futures_funding_rate(symbol = 'btcusdt', limit =
3)).iloc[:,1:]
    last_three_fr['fundingTime'] = (last_three_fr['fundingTime']//1000).apply(format)

    cq_bid_ask = run_sql_script('SELECT timestamp, bidPrice, askPrice FROM `exchange
data` .BTC_cq_bid_ask ORDER BY timestamp DESC LIMIT 1454')
    cq_bid_ask = pd.DataFrame(cq_bid_ask, columns = ['timestamp', 'bidPrice', 'askPrice'])

    perp_bid_ask = run_sql_script('SELECT timestamp, bidPrice, askPrice FROM `exchange
data` .BTC_perp_bid_ask ORDER BY timestamp DESC LIMIT 1454')
    perp_bid_ask = pd.DataFrame(perp_bid_ask, columns = ['timestamp', 'bidPrice', 'askPrice'])

```

```

df = pd.merge (perp_bid_ask, cq_bid_ask, on = 'timestamp')
df['askP-bidQ'] = df['askPrice_x'] - df['bidPrice_y']
df['askQ-bidP'] = df['askPrice_y'] - df['bidPrice_x']

df['askP-bidQ_perc'] = (df['askPrice_x'] - df['bidPrice_y'])/df['bidPrice_y']
df['askQ-bidP_perc'] = (df['askPrice_y'] - df['bidPrice_x'])/df['bidPrice_y']

first_max_spread_perc = df['askP-bidQ_perc'].max()
first_max_spread_abs = df[df['askP-bidQ_perc'] == first_max_spread_perc].iloc[0,5]
first_max_spread_timestamp = format_timestamp(df[df['askP-bidQ_perc'] ==
first_max_spread_perc].iloc[0,0])

first_min_spread_perc = df['askP-bidQ_perc'].min()
first_min_spread_abs = df[df['askP-bidQ_perc'] == first_min_spread_perc].iloc[0,5]
first_min_spread_timestamp = format_timestamp(df[df['askP-bidQ_perc'] ==
first_min_spread_perc].iloc[0,0])

second_max_spread_perc = df['askQ-bidP_perc'].max()
second_max_spread_abs = df[df['askQ-bidP_perc'] == second_max_spread_perc].iloc[0,6]
second_max_spread_timestamp = format_timestamp(df[df['askQ-bidP_perc'] ==
second_max_spread_perc].iloc[0,0])

second_min_spread_perc = df['askQ-bidP_perc'].min()
second_min_spread_abs = df[df['askQ-bidP_perc'] == second_min_spread_perc].iloc[0,6]
second_min_spread_timestamp = format_timestamp(df[df['askQ-bidP_perc'] ==
second_min_spread_perc].iloc[0,0])
message = textwrap.dedent("""
funding:
{:>8f} - {}
{:>8f} - {}
{:>8f} - {}
{:.8f} - total

askP-bidQ spread:
max:{:>9}% ({:>5}) - {}
min:{:>9}% ({:>5}) - {}

askQ-bidP spread:
max:{:>9}% ({:>5}) - {}
min:{:>9}% ({:>5}) - {}""".format(
    float(last_three_fr['fundingRate'][0]),
    format_timestamp(last_three_fr['fundingTime'][0]),
    float(last_three_fr['fundingRate'][1]),
    format_timestamp(last_three_fr['fundingTime'][1]),
    float(last_three_fr['fundingRate'][2]),
    format_timestamp(last_three_fr['fundingTime'][2]),
    round(last_three_fr['fundingRate'].astype(float).sum(),8),
)

```

```

        round(first_max_spread_perc, 6), round(first_max_spread_abs, 1),
first_max_spread_timestamp,
        round(first_min_spread_perc, 6), round(first_min_spread_abs, 1),
first_min_spread_timestamp,
        round(second_max_spread_perc, 6), round(second_max_spread_abs, 1),
second_max_spread_timestamp,
        round(second_min_spread_perc, 6), round(second_min_spread_abs, 1),
second_min_spread_timestamp,
    ))
send_telegram_message(message, config['telegram']['nikita'], config['telegram']['token_daily'])
send_telegram_message(message, config['telegram']['heorhii'], config['telegram']['token_daily'])

def weekly_notification():
    BTC_perpetual = pd.DataFrame(run_sql_script('SELECT * FROM `exchange
data` .BTC_perpetual ORDER BY timestamp DESC limit 10090'))
    BTC_perpetual.columns = ['timestamp', 'open', 'max', 'min', 'close', 'volume', 'number_of_trades',
'funding_rate']

    BTC_current_quarter = pd.DataFrame(run_sql_script('SELECT * FROM `exchange
data` .BTC_current_quarter ORDER BY timestamp DESC limit 10090'))
    BTC_current_quarter.columns = ['timestamp', 'open', 'max', 'min', 'close', 'volume',
'number_of_trades', 'time_left']

    data = BTC_perpetual.merge(BTC_current_quarter, on='timestamp', how='left',
suffixes=('_p','_q')).sort_values('timestamp', ascending=False)
    data = data.astype(float)
    data['spread'] = (
        (data['min_q']/data['min_p']
        + data['max_q']/data['max_p'])/2 - 1)*100
    message = '\n'.join(str(data['spread'].describe()).split('\n')[1:-1])

    send_telegram_message(message, config['telegram']['nikita'], config['telegram']['token_daily'])
    send_telegram_message(message, config['telegram']['heorhii'], config['telegram']['token_daily'])

def get_bid_ask_data():
def get_quarter_symbol():
    time = datetime.now()
    delivery = time + relativedelta(month = ((time.month-1)//3+1)*3) + relativedelta(day= 31,
weekday=FR(-1)) + relativedelta(hour=8, minute=0, second=0)
    delivery = pytz.utc.localize(delivery)

    if (delivery.timestamp() < time.timestamp()):
        time = (time + timedelta(weeks=2))
        delivery = time + relativedelta(month = ((time.month-1)//3+1)*3) + relativedelta(day= 31,
weekday=FR(-1)) + relativedelta(hour=8, minute=0, second=0)

    quarter_symbol = 'BTCUSDT_{0}{1}{2}'.format(delivery.year%100, str(delivery.month).zfill(2),
delivery.day)

```

```

    return quarter_symbol

    BTC_perp_bid_ask = pd.DataFrame.from_dict([client.futures_orderbook_ticker(symbol =
'BTCUSDT)]).iloc[:,1:-1]
    BTC_perp_bid_ask['timestamp'] = int(datetime.now().timestamp()//60*60)
    BTC_perp_bid_ask['funding_rate'] = client.futures_mark_price(symbol =
'BTCUSDT')['lastFundingRate']

    BTC_cq_bid_ask = pd.DataFrame.from_dict([
        client.futures_orderbook_ticker(
            symbol = get_quarter_symbol(datetime.now().timestamp())
        )
    ]).iloc[:,1:-1]
    BTC_cq_bid_ask['timestamp'] = int(datetime.now().timestamp()//60*60)
    BTC_cq_bid_ask['time_left'] = get_quarter_time_left(BTC_cq_bid_ask['timestamp'][0])

try:
    BTC_perp_bid_ask.to_sql('BTC_perp_bid_ask', engine, if_exists = 'append', index = False)
    BTC_cq_bid_ask.to_sql('BTC_cq_bid_ask', engine, if_exists = 'append', index = False)
except BaseException as e:
    send_telegram_message('ERROR: {}'.format(e), config['telegram']['nikita'],
config['telegram']['token_error'])
    send_telegram_message('ERROR: {}'.format(e), config['telegram']['heorhii'],
config['telegram']['token_error'])
    logging.info('ERROR: {}'.format(e))

def run_backup_and_stream():
    backup_from_last_record()
    try:
        schedule.every().minute.at(":00").do(get_bid_ask_data)
        schedule.every().day.at("08:10:30").do(daily_notification)
        schedule.every().minute.at(":10").do(stream)
        # schedule.every.friday.at("08:10:45").do(weekly_notification)
        while True:
            schedule.run_pending()
    except KeyboardInterrupt:
        pass
    finally:
        send_telegram_message('Script stopped!', config['telegram']['nikita'],
config['telegram']['token_error'])
        send_telegram_message('Script stopped!', config['telegram']['heorhii'],
config['telegram']['token_error'])

        logging.info("stopping scheduler")
        schedule.clear()

def main():
    try:

```

```

    send_telegram_message('Script started!', config['telegram']['nikita'],
config['telegram']['token_error'])
    send_telegram_message('Script started!', config['telegram']['heorhii'],
config['telegram']['token_error'])
    connect_ssh_tunnel(config['ssh']['address'], config['ssh']['username'], config['ssh']['password'])
    create_sql_engine()
    run_backup_and_stream()
except:
    send_telegram_message('Script stopped!, restarting in 300 seconds', config['telegram']['nikita'],
config['telegram']['token_error'])
    send_telegram_message('Script stopped!, restarting in 300 seconds',
config['telegram']['heorhii'], config['telegram']['token_error'])
    sleep(300)
    main()

if __name__ == '__main__':
    main()

```

rozmitka.py

```

#!/usr/bin/env python
# coding: utf-8

# ## import

# In[1]:


from src.config.config import Config
from src.db_writer.db import DB
from src.quarter_functions import get_delivery, get_quarter_time_left, get_quarter_symbol
import pandas as pd
import numpy as np
pd.set_option('display.max_columns', None)\

import matplotlib.pyplot as plt
import seaborn as sns
plt.rcParams["figure.figsize"] = (12.8, 7.2)

from tqdm import tqdm
from IPython.display import clear_output

# ## load data

```

In[]:

```
# BTC_current_quarter = pd.read_sql('SELECT * FROM `exchange data`.BTC_current_quarter;', con=db.con)
# BTC_current_quarter.to_csv('BTC_current_quarter.csv')

# BTC_perpetual = pd.read_sql('SELECT * FROM `exchange data`.BTC_perpetual;', con=db.con)
# BTC_perpetual.to_csv('BTC_perpetual.csv')

# BTC_cq_bid_ask = pd.read_sql('SELECT * FROM `exchange data`.BTC_cq_bid_ask;', con=db.con)
# BTC_cq_bid_ask.to_csv('data/BTC_cq_bid_ask.csv')

# BTC_perp_bid_ask = pd.read_sql('SELECT * FROM `exchange data`.BTC_perp_bid_ask;', con=db.con)
# BTC_perp_bid_ask.to_csv('data/BTC_perp_bid_ask.csv')
```

In[11]:

```
BTC_current_quarter = pd.read_csv('C:/Users/krotenko.n/Documents/GitHub/New folder/data/load_historical_quarterly_data_batches.csv', index_col=[0])
# BTC_current_quarter = pd.read_csv('C:/Users/krotenko.n/Documents/GitHub/New folder/data/BTC_current_quarter.csv', index_col=[0])
BTC_perpetual = pd.read_csv('C:/Users/krotenko.n/Documents/GitHub/New folder/data/load_historical_perentrentual_data_batches.csv', index_col=[0])
# BTC_perpetual = pd.read_csv('C:/Users/krotenko.n/Documents/GitHub/New folder/data/BTC_perpetual.csv', index_col=[0])
BTC_cq_bid_ask = pd.read_csv('C:/Users/krotenko.n/Documents/GitHub/New folder/data/BTC_cq_bid_ask.csv', index_col=[0])
BTC_perp_bid_ask = pd.read_csv('C:/Users/krotenko.n/Documents/GitHub/New folder/data/BTC_perp_bid_ask.csv', index_col=[0])
```

In[12]:

```
BTC_current_quarter = BTC_current_quarter\
    .sort_values(['timestamp', 'symbol'])\
    .drop_duplicates(subset=['timestamp'], keep='first')\
    .reset_index(drop=True)

BTC_perpetual.sort_values('timestamp', inplace=True)

# ### merge
```

In[13]:

```
candles = BTC_perpetual.merge(
    BTC_current_quarter,
    on='timestamp',
    how='left',
    suffixes=['_perp', '_cq'])

order_book = BTC_perp_bid_ask.merge(
    BTC_cq_bid_ask,
    on='timestamp',
    suffixes=['_perp', '_cq'])

candles = candles[candles['timestamp'] >= order_book['timestamp'].min()]

df_total = candles.merge(
    order_book,
    on='timestamp',
    how='left')
```

In[15]:

```
df_total['timestamp'] = df_total['timestamp'].astype(int)
df_total['time_left'] = df_total['timestamp'].apply(get_quarter_time_left)
```

Fill missing

In[17]:

```
df_total = df_total.interpolate()
```

EDA

time_left

In[19]:

```
fig, (ax0,ax1, ax2) = plt.subplots(1,3, figsize=(20, 4))
```

```
ax0.plot(df_total['timestamp'], df_total['time_left'])
ax1.plot(df_total['time_left'])
ax2.plot(df_total['timestamp'])
plt.show()
```

In[20]:

```
plt.figure(figsize=(12.8, 7.2))
temp = df_total['time_left'].diff()
plt.plot(temp)
```

In[21]:

```
plt.figure(figsize = (12.8, 7.2))
plt.plot(df_total['timestamp'], df_total['open_perp'])
plt.ylim([0, 32000])
# plt.plot(df_total['timestamp'][-1000:], df_total['open_cq'][-1000:])
# plt.plot(df_total['timestamp'][-1000:], df_total['open_perp'][-1000:])
# plt.plot(df_total['timestamp'][-1000:], df_total['open_perp'][-1000:])
```

other features overview

In[22]:

```
# plt.plot(df_total['funding_rate'])
plt.plot(df_total['funding_rate'].rolling(1000).mean())
plt.axhline(0, c = 'r')
```

In[23]:

```
df_total.head()
```

rozmitka

prep

In[26]:

```

def plot_selected_extremas(df, col, window=(0,400), color='red', figsize=(7.2,3.6)):
    plt.figure(figsize = figsize)
    plt.title(f'{col}:{window}')
    plt.plot(df.loc[window[0]:window[1], 'open_perp'])
    plt.scatter(df.loc[window[0]:window[1]].loc[df['is_extrema'], 'index'],
                df.loc[window[0]:window[1]].loc[df['is_extrema'], 'open_perp'],
                c='lime')
    plt.scatter(df.loc[window[0]:window[1]].loc[df[col], 'index'],
                df.loc[window[0]:window[1]].loc[df[col], 'open_perp'],
                c=color)
    plt.show()
    plt.close()

# profit = balance_start * (np.power(1-COMMISSION_RATE, 2) * (1 + LEVERAGE * (1 -
close_price/open_price)) - 1)

```

In[25]:

```

COMMISSION_RATE = 0.0008
LEVERAGE = 5

```

```

def get_balance(open_price, close_price, commision=COMMISSION_RATE, leverage =
LEVERAGE):
    return np.power(1 - commision, 2) * (1 + leverage * abs(1 - close_price/open_price))

def get_balance_2contracts(price0, price1, price2):
    return get_balance(price0, price1) * get_balance(price1, price2)

def get_balance_3contracts(price0, price1, price2, price3):
    return get_balance(price0, price1) * get_balance(price1, price2) * get_balance(price2, price3)

```

In[42]:

```

min_prof = 1.0075
min_prof_3 = 1.0075*1.0075

```

Step 1: mark extremas

In[43]:

```

df_total['diff'] = df_total['open_perp'].diff(1)
df_total['diff_shift1'] = df_total['diff'].shift(-1)

```

```
# if price didnt change from t-1, compare with t-2, t-3...
i = -1
print(i, (df_total['diff_shift1'] == 0).sum())
while ((df_total['diff_shift1'] == 0).sum()):
    i -= 1
    df_total.loc[df_total['diff_shift1'] == 0, 'diff_shift1'] = \
        df_total['diff'].shift(i)[df_total['diff_shift1'] == 0]
    print(i, (df_total['diff_shift1'] == 0).sum())
```

```
df_total['is_extrema'] = (df_total['diff'] * df_total['diff_shift1'] < 0)
df_total['ismaxima'] = (df_total['diff_shift1'] < 0) & df_total['is_extrema']
df_total['isminima'] = (df_total['diff_shift1'] >= 0) & df_total['is_extrema']
```

In[44]:

```
df_rozmitka = df_total[['timestamp', 'open_perp', 'is_extrema', 'ismaxima', 'isminima']].copy()
df_rozmitka.reset_index(inplace=True)
df_rozmitka.head()
```

In[27]:

```
plot_selected_extremas(df_rozmitka, 'is_extrema', (0,200))
```

Step 2: initial demarkation

In[46]:

```
iterator = iter(df_rozmitka.index[df_rozmitka['is_extrema']])
# skip first value because it wouldnt be an extrema
iterator.__next__()
(index_start, price_start) = df_rozmitka.loc[iterator.__next__(), ['index', 'open_perp']]
extremas = [index_start]
try:
    while True:
        index_first_contract, price_first_contract = df_rozmitka.loc[iterator.__next__(), ['index',
        'open_perp']]
        index_second_contract, price_second_contract = df_rozmitka.loc[iterator.__next__(), ['index',
        'open_perp']]
        index_third_contract, price_third_contract = df_rozmitka.loc[iterator.__next__(), ['index',
        'open_perp']]
```

```

balance1 = get_balance(price_start, price_first_contract)
balance123 = get_balance_3contracts(price_start, price_first_contract, price_second_contract,
price_third_contract)
balance__3 = get_balance(price_start, price_third_contract)

max_balance = max(balance1, balance123/min_prof_3, balance__3)

if (max_balance > min_prof):
    if(max_balance == balance1):
        extremas.append(index_first_contract)
        price_start = price_first_contract

    elif(max_balance == balance__3):
        iterator = iter(df_rozmitka.index[df_rozmitka['is_extrema'] &
(df_rozmitka.index>=index_third_contract)])

    elif(max_balance == balance123/min_prof_3):
        extremas.append(index_first_contract)
        iterator = iter(df_rozmitka.index[df_rozmitka['is_extrema'] &
(df_rozmitka.index>=index_first_contract)])
        (index_start, price_start) = df_rozmitka.loc[iterator.__next__(), ['index', 'open_perp']]
    else:
        pass
except StopIteration:
    pass

```

In[47]:

```

df_rozmitka.loc[extremas, 'extr1'] = True
df_rozmitka['extr1'].fillna(False, inplace=True)

```

In[48]:

```

print(f"total extremas: {df_rozmitka['is_extrema'].sum():>12,}")
print(f"extremas after step2: {df_rozmitka['extr1'].sum():,}")

```

In[28]:

```
plot_selected_extremas(df_rozmitka, 'extr1')
```

```
# ### Step 3: find global maxima/minima for intervals
```

```
# In[50]:
```

```
df_rozmitka['extr2'] = df_rozmitka['extr1'].copy()

all_extr = df_rozmitka.loc[df_rozmitka['is_extrema']].copy()
extr2 = df_rozmitka.loc[df_rozmitka['extr2']].reset_index(drop=True).copy()
non_extr2 = df_rozmitka.loc[(~df_rozmitka['extr2']) & df_rozmitka['is_extrema']].copy()

extr2.head()
```

```
# In[51]:
```

```
start = 0
try:
    while True:
        left = extr2.loc[start, ['index', 'open_perp']]
        mid = extr2.loc[start+1, ['index', 'open_perp']]
        right = extr2.loc[start+2, ['index', 'open_perp']]

        extrema_range = all_extr.loc[left['index']:right['index']]

        if left['open_perp'] > mid['open_perp']:
            index_extrema = extrema_range['open_perp'].idxmin()
        elif left['open_perp'] < mid['open_perp']:
            index_extrema = extrema_range['open_perp'].idxmax()
        else:
            extr2.drop(extr2.index[extr2['index'] == mid['index']])
            continue

        extrema_range = all_extr.loc[left['index']:right['index']]
        if index_extrema == mid['index']:
            start += 1
            continue
        elif index_extrema == right['index']:
            extr2 = extr2.drop(start+1).reset_index(drop=True)
            continue
        else:
            # print(f"replace extrema {mid['index']} on {index_extrema}")
            extr2.loc[start+1] = extrema_range.loc[index_extrema]
            start += 1
            continue

except:
    pass
```

In[52]:

```
df_rozmitka['extr2'] = np.where(df_rozmitka.index.isin(extr2['index']), True, False)
```

In[53]:

```
print(f"total extremas: {df_rozmitka['is_extrema'].sum():>12,}")
print(f"extremas after step2: {df_rozmitka['extr1'].sum():,}")
print(f"extremas after step3: {df_rozmitka['extr2'].sum():,}")
```

In[36]:

```
# plot_selected_extremas(df_rozmitka, 'extr1')
plot_selected_extremas(df_rozmitka, 'extr2', window=(400,600))
```

```
df_rozmitka['added'] = ~df_rozmitka['extr1'] & df_rozmitka['extr2']
plot_selected_extremas(df_rozmitka, 'added', window=(400,600))
df_rozmitka.drop(columns='added', inplace=True)
```

```
df_rozmitka['deleted'] = df_rozmitka['extr1'] & ~df_rozmitka['extr2']
plot_selected_extremas(df_rozmitka, 'deleted', window=(400,600))
df_rozmitka.drop(columns='deleted', inplace=True)
```

Step 4: delete excessive extremas

In[58]:

```
def price(idx):
    return df_rozmitka.loc[idx, 'open_perp']

def step3(df_raw, extr_col, new_extr_col):
    df = df_raw.copy()
    df[new_extr_col] = df[extr_col]
    left_index = 0
    while True:
        stop_trigger = True
        for contract0, contract1, contract2, contract3 in zip(
            df.loc[df[new_extr_col], 'index'].loc[left_index:], df.loc[:, 'index'].iloc[:-3],
            df.loc[:, 'index'].iloc[-3:-1], df.loc[:, 'index'].iloc[-1])):
            if contract0 == contract1 == contract2 == contract3:
                stop_trigger = False
                break
        if stop_trigger:
            break
```

```

df.loc[df[new_extr_col], 'index'].loc[left_index:].shift(-1).iloc[:-3],
df.loc[df[new_extr_col], 'index'].loc[left_index:].shift(-2).iloc[:-3],
df.loc[df[new_extr_col], 'index'].loc[left_index:].shift(-3).iloc[:-3]):


# balance1 = get_balance(1,
#     price(contract0),
#     price(contract3))
balance2 = get_balance(
    price(contract0),
    price(contract3))
balance3 = get_balance_3contracts(
    price(contract0),
    price(contract1),
    price(contract2),
    price(contract3))

if (balance2*min_prof_3 >= balance3):
    stop_trigger = False
    clear_output(wait=True)
    print([contract1, contract2])

    df.loc[[contract1, contract2], new_extr_col] = False
    left_index = contract3
    break
if stop_trigger:
    return df

```

In[59]:

```
df_rozmitka = step3(df_rozmitka, 'extr2', 'extr3')
```

In[60]:

```

print(f"total extremas: {df_rozmitka['is_extrema'].sum() > 12,}")
print(f"extremas after step1: {df_rozmitka['extr1'].sum(),}")
print(f"extremas after step2: {df_rozmitka['extr2'].sum(),}")
print(f"extremas after step3: {df_rozmitka['extr3'].sum(),}")

```

In[41]:

```

plot_selected_extremas(df_rozmitka, 'extr3', window=(400,600))
plot_selected_extremas(df_rozmitka, 'extr2', window=(400,600))

```

```
df_rozmitka['deleted'] = ~df_rozmitka['extr3'] & df_rozmitka['extr2']
plot_selected_extremas(df_rozmitka, 'deleted', window=(400,600))
df_rozmitka.drop(columns='deleted', inplace=True)
```

Step 5: check for missing extremas in-between pairs

In[65]:

```
def price(idx):
    return df_rozmitka.loc[idx, 'open_perp']

def step4(df_raw, extr_col, new_extr_col):
    df = df_raw.copy()
    df[new_extr_col] = df[extr_col]
    while(True):
        new_extremas = []
        for start, end in tqdm(zip(df.loc[df[new_extr_col], 'index'].iloc[:-1], df.loc[df[new_extr_col], 'index'].shift(-1))):
            order_short_long = (('isminima', 'ismaxima') if (price(start) > price(end)) else ('ismaxima', 'isminima'))
            profitable_combos = {}
            for first_contract in df.loc[start+1:end-1].loc[df[order_short_long[0]]].index:
                for second_contract in df.loc[first_contract+1:end-1].loc[df[order_short_long[1]]].index:
                    extr2_balance = get_balance(
                        price(start),
                        price(end))
                    extr3_balance = get_balance_3contracts(
                        price(start),
                        price(first_contract),
                        price(second_contract),
                        price(end))

                    if(extr3_balance > extr2_balance * min_prof_3):
                        profitable_combos.update({(first_contract, second_contract) : extr3_balance})
            if(len(profitable_combos) > 0):
                new_extremas.append(max(profitable_combos, key=profitable_combos.get))
    print(f"extremas before: {df[new_extr_col].sum()}")
    df.loc[pd.Series(new_extremas).explode().values, new_extr_col] = True
    print(f"extremas after: {df[new_extr_col].sum()}")
    if (len(new_extremas) == 0):
        return df
```

In[67]:

```
df_rozmitka = step4(df_rozmitka, 'extr3', 'extr4')
```

In[68]:

```
print(f"total extremas: {df_rozmitka['is_extrema'].sum():>12,}")
print(f"extremas after step1: {df_rozmitka['extr1'].sum():,}")
print(f"extremas after step2: {df_rozmitka['extr2'].sum():,}")
print(f"extremas after step3: {df_rozmitka['extr3'].sum():,}")
print(f"extremas after step4: {df_rozmitka['extr4'].sum():,}")
```

In[42]:

```
plot_selected_extremas(df_rozmitka, 'extr3', window=(400,800), figsize=(10,4))
plot_selected_extremas(df_rozmitka, 'extr4', window=(400,800), figsize=(10,4))
```

```
df_rozmitka['added'] = ~df_rozmitka['extr3'] & df_rozmitka['extr4']
plot_selected_extremas(df_rozmitka, 'added', window=(400,800), figsize=(10,4))
df_rozmitka.drop(columns='added', inplace=True)
```

Step 6: repeat steps 3,4 until number of points doesnt change

In[70]:

```
df_rozmitka = step3(df_rozmitka, 'extr4', 'extr5')
df_rozmitka = step4(df_rozmitka, 'extr5', 'extr6')
df_rozmitka = step3(df_rozmitka, 'extr6', 'extr7')
```

In[71]:

```
print(f"total extremas: {df_rozmitka['is_extrema'].sum():>12,}")
print(f"extremas after step1: {df_rozmitka['extr1'].sum():,}")
print(f"extremas after step2: {df_rozmitka['extr2'].sum():,}")
print(f"extremas after step3: {df_rozmitka['extr3'].sum():,}")
print(f"extremas after step4: {df_rozmitka['extr4'].sum():,}")
print(f"extremas after step5: {df_rozmitka['extr5'].sum():,}")
print(f"extremas after step6: {df_rozmitka['extr6'].sum():,}")
print(f"extremas after step7: {df_rozmitka['extr7'].sum():,}")
```

In[54]:

```
plot_selected_extremas(df_rozmitka, 'extr7')
```

In[59]:

```
df_rozmitka.loc[df_rozmitka['extr7'], 'timestamp'].diff().max()
```

In[109]:

```
contract_durations = df_rozmitka.loc[df_rozmitka['extr7'], 'timestamp'].diff()
print(contract_durations.shape)
# contract_durations = contract_durations[contract_durations<10000]
contract_durations = contract_durations.sort_values().reset_index(drop=True)
contract_durations.head()
```

In[122]:

```
plt.figure(figsize=(8,4))
# plt.title("Розподілення часу тривалості контрактів")
sns.histplot(contract_durations,bins=range(60,50000, 480))
```

In[97]:

```
contract_durations = df_rozmitka.loc[df_rozmitka['extr7'], 'timestamp'].diff()
print(contract_durations.shape)
contract_durations = contract_durations[contract_durations<10000]
contract_durations = contract_durations.sort_values().reset_index(drop=True)
contract_durations.head()
```

In[121]:

```
plt.figure(figsize=(8,4))
# plt.title("Розподілення часу тривалості контрактів")
sns.histplot(contract_durations,bins=range(60,10120, 120))
```

In[116]:

```
(contract_durations<=1800).mean()*100
```

In[118]:

```
(contract_durations<=3600).mean()*100
```

In[142]:

```
plt.figure(figsize=(8,4))
plt.plot(contract_durations, contract_durations.index/18324)
plt.yticks(np.arange(0,1.01,0.1))
plt.xticks(range(0,47501,2500), rotation=75)
plt.grid()
```

In[62]:

```
sns.displot(contract_durations)
```

Plots for examples

In[]:

```
get_balance_3contracts(10000, 11000, 9900, 10890),\
get_balance_3contracts(10000, 9000, 9900, 8910)
```

In[]:

```
fig, axs = plt.subplots(2,2, figsize=(9.6, 5.4))
axs[0,0].set_ylim([0,120])
axs[0,1].set_ylim([0,120])
axs[1,0].set_ylim([0,120])
axs[1,1].set_ylim([0,120])
```

```

axs[0,0].set_title('сценарій 1', fontsize=10)
axs[0,0].plot([0,1,2,3], [110, 58, 62, 10])
axs[0,0].scatter([0,1,2,3], [110, 58, 62, 10], c='blue')
axs[0,0].scatter([0,3], [110,10], color='red')

axs[0,1].set_title('сценарій 2', fontsize=10)
axs[0,1].plot([0,1,2,3], [110, 10, 18, 14])
axs[0,1].scatter([0,1,2,3], [110, 10, 18, 14])
axs[0,1].scatter([0,1], [110,10], color='red')

axs[1,0].set_title('сценарій 3', fontsize=10)
axs[1,0].plot([0,1,2,3], [110, 10, 100, 20])
axs[1,0].scatter([0,1,2,3], [110, 10, 100, 20], color='red')

# axs[1,1].set_title('сценарій 4')
# axs[1,1].plot([0,1,2,3], [110, 10, 100, 90])
# axs[1,1].scatter([0,1,2,3], [110, 10, 100, 90])
# axs[1,1].scatter([0,1,2], [110, 10, 100], color='red')

axs[1,1].set_title('сценарій 4', fontsize=10)
axs[1,1].plot([0,1,2,3], [110, 104, 107, 104])
axs[1,1].scatter([0,1,2,3], [110, 104, 107, 104])
axs[1,1].scatter([0], [110], color='red')
plt.tight_layout()
plt.show()

```

In[]:

```

plt.figure(figsize=(10,4))
plt.plot(['Y', 'Z', 'A', 'B', 'C'], [12, 8, 8, 8, 11])
plt.scatter(['Y', 'Z', 'A', 'B', 'C'], [12, 8, 8, 8, 11])
plt.ylim([7,13])

```

In[5]:

```

df_total = pd.read_csv('C:/Users/krotenko.n/Documents/GitHub/New folder/data/df_total.csv',
index_col=[0])
df_rozmitka = pd.read_csv('C:/Users/krotenko.n/Documents/GitHub/New
folder/data/df_rozmitka.csv', index_col=[0])

```

In[19]:

```
plot_selected_extremas(df_rozmitka, 'extr1', (1300,1400))
```

In[39]:

```
plot_selected_extremas(df_rozmitka, 'extr1', (1470,1527))
```

In[37]:

```
plot_selected_extremas(df_rozmitka, 'extr1', (11,40))
```

In[38]:

```
plot_selected_extremas(df_rozmitka, 'extr1', (225, 265))
```

features.py (converted from .ipynb)

```
#!/usr/bin/env python
# coding: utf-8
```

In[2]:

```
import pandas as pd
import numpy as np
pd.set_option('display.max_columns', None)
```

In[3]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In[5]:

```

df_total = pd.read_csv('C:/Users/krotenko.n/Documents/GitHub/New folder/data/df_total.csv',
index_col=[0])
df_rozmitka = pd.read_csv('C:/Users/krotenko.n/Documents/GitHub/New
folder/data/df_rozmitka.csv', index_col=[0])

df_fin = df_total.copy()
df_fin['extr7'] = df_rozmitka['extr7']

```

In[9]:

```

df_fin.loc[df_fin['extr7'], 'id_contract'] = range(18324)
df_fin['id_contract'] = df_fin['id_contract'].shift(-1).fillna(method='ffill')

```

In[10]:

```

df_fin['date'] = pd.to_datetime(df_fin['timestamp'], unit='s')
# df_fin['month'] = df_fin['date'].dt.month
df_fin['day_of_week'] = df_fin['date'].dt.day_of_week
df_fin['second_of_day'] = df_fin['timestamp']%86400
df_fin.drop(columns=['date'], inplace=True)

```

In[11]:

```

df_fin['open_close_diff_perp'] = df_fin['open_perp'] - df_fin['close_perp']
df_fin['high_low_diff_perp'] = df_fin['high_perp'] - df_fin['low_perp']
df_fin['avg_trade_volume_perp'] = df_fin['volume_perp'] / df_fin['trades_perp']
df_fin['bid_ask_spread_abs_perp'] = df_fin['askPrice_perp'] - df_fin['bidPrice_perp']
df_fin['bid_ask_spread_perp'] = (df_fin['askPrice_perp'] - df_fin['bidPrice_perp'])/df_fin['askPrice_perp']
df_fin['bid_ask_qty_spread_perp'] = (df_fin['askQty_perp'] - df_fin['bidQty_perp'])/df_fin['askQty_perp']
df_fin['bid_ask_qty_spread_abs_perp'] = df_fin['askQty_perp'] - df_fin['bidQty_perp']
df_fin['bid_ask_qty_sum_perp'] = df_fin['askQty_perp'] + df_fin['bidQty_perp']

```

```

df_fin['open_close_diff_cq'] = df_fin['open_cq'] - df_fin['close_cq']
df_fin['high_low_diff_cq'] = df_fin['high_cq'] - df_fin['low_cq']
df_fin['avg_trade_volume_cq'] = df_fin['volume_cq'] / df_fin['trades_cq']
df_fin['avg_trade_volume_cq'].fillna(0, inplace=True)
df_fin['bid_ask_spread_abs_cq'] = df_fin['askPrice_cq'] - df_fin['bidPrice_cq']

```

```

df_fin['bid_ask_spread_cq'] = (df_fin['askPrice_cq'] - df_fin['bidPrice_cq'])/df_fin['askPrice_cq']
df_fin['bid_ask_qty_spread_cq'] = (df_fin['askQty_cq'] - df_fin['bidQty_cq'])/df_fin['askQty_cq']
df_fin['bid_ask_qty_spread_abs_cq'] = df_fin['askQty_cq'] - df_fin['bidQty_cq']

df_fin['perp_cq_open_spread_abs'] = df_fin['open_perp'] - df_fin['open_cq']
df_fin['perp_cq_open_spread'] = (df_fin['open_perp'] - df_fin['open_cq'])/df_fin['open_perp']

df_fin['short_term_expectations_v1'] = (df_fin['open_perp'] - df_fin['open_cq'])/(df_fin['time_left'] + 86400)
df_fin['short_term_expectations_v2'] = (df_fin['open_perp'] - df_fin['open_cq'])/(df_fin['time_left'] + 86400*9)
df_fin['short_term_expectations_v3'] = (df_fin['open_perp'] - df_fin['open_cq'])/(df_fin['time_left'] + 86400*90)
df_fin['short_term_expectations_v4'] = np.power(df_fin['open_perp'] - df_fin['open_cq'], 2)/(df_fin['time_left'] + 86400)
df_fin['short_term_expectations_v5'] = np.power(df_fin['open_perp'] - df_fin['open_cq'], 2)/(df_fin['time_left'] + 86400*9)
df_fin['short_term_expectations_v6'] = np.power(df_fin['open_perp'] - df_fin['open_cq'], 2)/(df_fin['time_left'] + 86400*90)
df_fin['short_term_expectations_v7'] = np.power(df_fin['open_perp'] - df_fin['open_cq'], 2)/np.sqrt(df_fin['time_left'] + 86400)
df_fin['short_term_expectations_v8'] = np.power(df_fin['open_perp'] - df_fin['open_cq'], 2)/np.sqrt(df_fin['time_left'] + 86400*9)
df_fin['short_term_expectations_v9'] = np.power(df_fin['open_perp'] - df_fin['open_cq'], 2)/np.sqrt(df_fin['time_left'] + 86400*90)
df_fin['short_term_expectations_v10'] = np.sqrt(np.abs(df_fin['open_perp'] - df_fin['open_cq']))/(df_fin['time_left'] + 86400)
df_fin['short_term_expectations_v11'] = np.sqrt(np.abs(df_fin['open_perp'] - df_fin['open_cq']))/(df_fin['time_left'] + 86400*9)
df_fin['short_term_expectations_v12'] = np.sqrt(np.abs(df_fin['open_perp'] - df_fin['open_cq']))/(df_fin['time_left'] + 86400*90)
df_fin['short_term_expectations_v13'] = np.sqrt(np.abs(df_fin['open_perp'] - df_fin['open_cq']))/np.sqrt(df_fin['time_left'] + 86400)
df_fin['short_term_expectations_v14'] = np.sqrt(np.abs(df_fin['open_perp'] - df_fin['open_cq']))/np.sqrt(df_fin['time_left'] + 86400*9)
df_fin['short_term_expectations_v15'] = np.sqrt(np.abs(df_fin['open_perp'] - df_fin['open_cq']))/np.sqrt(df_fin['time_left'] + 86400*90)

df_fin = pd.concat([df_fin, pd.get_dummies(df_fin['symbol_cq'], prefix='symbol', drop_first=True)], axis=1)
df_fin = pd.concat([df_fin, pd.get_dummies(df_fin['day_of_week'], prefix='dayofweek', drop_first=True)], axis=1)

```

In[21]:

```
def total_delta(series):
    return series.iloc[-1] - series.iloc[0]
def minmax_rel_diff(series):
    return (series.max() - series.min()) / series.max()

features_grouped_by_contracts = df_fin.groupby('id_contract').agg({
    'open_perp': ['min', 'max', total_delta, minmax_rel_diff],
    'open_cq': ['min', 'max', total_delta, minmax_rel_diff],
    'volume_perp': ['sum', 'mean'],
    'trades_perp': ['sum', 'mean'],
    'volume_cq': ['sum', 'mean'],
    'trades_cq': ['sum', 'mean'],
    'bidPrice_perp': ['min', 'max', total_delta, minmax_rel_diff],
    'askPrice_perp': ['min', 'max', total_delta, minmax_rel_diff],
    'bidQty_perp': ['sum', 'mean'],
    'askQty_perp': ['sum', 'mean'],
    'bidPrice_cq': ['min', 'max', total_delta, minmax_rel_diff],
    'askPrice_cq': ['min', 'max', total_delta, minmax_rel_diff],
    'bidQty_cq': ['sum', 'mean'],
    'askQty_cq': ['sum', 'mean'],
    'time_left': lambda x: (x.max() - x.min())
})
features_grouped_by_contracts.columns =
features_grouped_by_contracts.columns.to_flat_index()
features_grouped_by_contracts.head()
```

In[22]:

```
print(features_grouped_by_contracts.isna().sum().sum())
df_fin.isna().sum()[df_fin.isna().sum() != 0]
```

In[23]:

```
df_fin.drop(columns=['timestamp', 'symbol_perp', 'symbol_cq', 'diff', 'diff_shift1'], inplace=True)
df_fin.dropna(inplace=True)
```

model.py

```
#!/usr/bin/env python
```

```
# coding: utf-8
```

```
# ## Imports
```

```
# In[1]:
```

```
# from src.config.config import Config
# from src.db_writer.db import DB
import pandas as pd
import numpy as np
pd.set_option('display.max_columns', None)
```

```
# In[2]:
```

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.rcParams["figure.figsize"] = (12.8, 7.2)
```

```
# ## Preparation
```

```
# In[3]:
```

```
df_fin = pd.read_csv('C:/Users/krotenko.n/Documents/GitHub/New folder/data/all_features.csv',
index_col=[0])
df_fin.head()
```

```
# Lets imagine our range of opening contract lies in range:
```

```
# $$\frac{\text{high} + \max(\text{open}, \text{close})}{2} : \frac{\text{low} + \min(\text{open}, \text{close})}{2}$$
```

```
# In[4]:
```

```
df_fin['low_border'] = (df_fin['low_perp'] + df_fin[['open_perp', 'close_perp']].min(axis=1)) / 2
df_fin['up_border'] = (df_fin['high_perp'] + df_fin[['open_perp', 'close_perp']].max(axis=1)) / 2
df_fin.loc[df_fin['ismaxima'] & df_fin['extr7'], 'target1'] = df_fin.loc[df_fin['ismaxima'] &
df_fin['extr7'], 'up_border']
df_fin.loc[df_fin['isminima'] & df_fin['extr7'], 'target1'] = df_fin.loc[df_fin['isminima'] &
df_fin['extr7'], 'low_border']
```

```
df_fin['min_over_upcoming_30min'] = df_fin['low_border'][::-1].rolling(window=30).min()[:-1]
```

```

df_fin['max_over_upcoming_30min'] = df_fin['up_border'][::-1].rolling(window=30).max()[:-1]
df_fin.loc[df_fin['ismaxima'] & df_fin['extr7'], 'target2'] = df_fin.loc[df_fin['ismaxima'] &
df_fin['extr7'], 'min_over_upcoming_30min']
df_fin.loc[df_fin['isminima'] & df_fin['extr7'], 'target2'] = df_fin.loc[df_fin['isminima'] &
df_fin['extr7'], 'max_over_upcoming_30min']

df_fin.loc[df_fin['ismaxima'] & df_fin['extr7'], 'contract_type'] = 1
df_fin.loc[df_fin['isminima'] & df_fin['extr7'], 'contract_type'] = 0

df_fin[['target1', 'target2', 'contract_type']] = df_fin[['target1', 'target2', 'contract_type']].shift(-1).fillna(method='bfill')
df_fin.dropna(inplace=True)

```

In[5]:

```
# df_fin.loc[~df_fin['extr7'], ['ismaxima', 'isminima']] = False
```

In[6]:

```

def custom_loss(row, target_col, pred_col):
    if row['extr7']:
        if row['contract_type'] == 1:
            if row[pred_col] < row['open_perp']:
                return np.sqrt(np.abs(row[target_col] - row['open_perp']))
            elif (row[pred_col] >= row['open_perp']) & (row[pred_col] <= row['high_perp']):
                return np.sqrt(np.abs(row[target_col] - row[pred_col]))
            else:
                return 10 * np.sqrt(np.abs(row[pred_col] - row[target_col]))

        elif row['contract_type'] == 0:
            if row[pred_col] > row['open_perp']:
                return np.sqrt(np.abs(row['open_perp'] - row[target_col]))
            elif (row[pred_col] <= row['open_perp']) & (row[pred_col] >= row['low_perp']):
                return np.sqrt(np.abs(row[target_col] - row[pred_col]))
            else:
                return 10 * np.sqrt(np.abs(row[target_col] - row[pred_col]))

    else:
        if row['contract_type'] == 1:
            if row[pred_col] <= row['high_perp']:
                return 10 * np.sqrt(np.abs(row[pred_col] - row[target_col]))
            else:

```

```

    return np.sqrt(np.abs(row[pred_col] - row[target_col]))

elif row['contract_type'] == 0:
    if row[pred_col] >= row['low_perp']:
        return 10 * np.sqrt(np.abs(row[target_col] - row[pred_col]))
    else:
        return np.sqrt(np.abs(row[target_col] - row[pred_col]))
```

Preprocessing

In[7]:

```
print(df_fin.shape)
df_fin.head()
```

In[8]:

```
X = df_fin.drop(columns= ['target1', 'target2'])
y = df_fin[['target1', 'target2']]
```

In[9]:

```
from sklearn.preprocessing import StandardScaler

X_scaled = StandardScaler().fit_transform(X)
y_scaled = StandardScaler().fit_transform(y)
```

In[10]:

```
X_scaled.shape, y_scaled.shape
```

LSTM model

In[11]:

```
import numpy as np

from sklearn.model_selection import TimeSeriesSplit, GridSearchCV
```

```

from sklearn import model_selection

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.activations import relu
from tensorflow.keras.layers import LSTM, Dense, Dropout, Conv1D, Input, Flatten, concatenate
from tensorflow.keras.regularizers import l1_l2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from keras.utils.vis_utils import plot_model

from keras_tuner.tuners import GridSearch
from keras_tuner.engine.tuner import Tuner
from keras_tuner.oracles import BayesianOptimizationOracle

tf.config.list_physical_devices('GPU')

```

prepare LSTM data

In[12]:

```

def create_lstm_data(X, y, look_back=60):
    arrays = [np.array(X[i : i+look_back]) for i in range(len(X) - look_back + 1)]
    lstm_data = np.stack(arrays)
    lstm_target = y[look_back - 1:]
    print('initial X shape:', np.array(X).shape, 'y shape:', y.shape, 'look_back:', look_back)
    print('output X shape:', lstm_data.shape, 'y shape:', lstm_target.shape)
    return lstm_data, lstm_target

```

In[13]:

```

# X_lstm, y_lstm = create_lstm_data(X_scaled, y)
# np.save('X_lstm_60.npy', X_lstm)
# np.save('y_lstm_60.npy', y_lstm)

X_lstm = np.load('X_lstm_60.npy')
# y_lstm = np.load('y_lstm_60.npy')
y_lstm = y_scaled[60 - 1:]

```

Train-test-val split

In[14]:

```
# validation will be last ~21k, the rest will be split for Nested Cross-Validation
X_tt, X_test = X_lstm[:350000], X_lstm[350000:]
y_tt, y_test = y_lstm[:350000], y_lstm[350000:]
```

Ensemble

In[15]:

First model

```
input_layer = Input(shape=(60, 110))
cnn_layers = []
for i in range(110):
    cnn_layer = Conv1D(filters=8, kernel_size=3, activation='relu')(input_layer[:, :, i:i+1])
    cnn_layers.append(cnn_layer)
flattened_layers = [Flatten()(cnn_layer) for cnn_layer in cnn_layers]
merged_layer1 = concatenate(flattened_layers)
dense_layer1 = Dense(units=32, activation='relu')(merged_layer1)
dense_layer2 = Dense(units=32, activation='relu')(dense_layer1)
output_layer1 = Dense(units=2, activation='linear')(dense_layer2)
model1 = Model(inputs=input_layer, outputs=output_layer1, name='1D-CNN_model')
```

Second model

```
lstm_layer = LSTM(32, input_shape=(X_tt.shape[1], X_tt.shape[2]))(input_layer)
dense_layer3 = Dense(32, activation='relu')(lstm_layer)
dense_layer4 = Dense(32, activation='relu')(dense_layer3)
dense_layer5 = Dense(32, activation='relu')(dense_layer4)
output_layer2 = Dense(2, activation='linear')(dense_layer5)
model2 = Model(inputs=input_layer, outputs=output_layer2, name='LSTM_model')
```

Ensemble model

```
# ensemble_input2 = Input(shape=(X_tt.shape[1], X_tt.shape[2]))
output1 = model1(input_layer)
output2 = model2(input_layer)
merged_output = concatenate([output1, output2])
dense_layer_last = Dense(32, activation='relu')(merged_output)
ensemble_output = Dense(2, activation='linear')(dense_layer_last)
ensemble_model = Model(inputs=input_layer, outputs=ensemble_output)
```

In[16]:

X_tt.shape[1], X_tt.shape[2]

In[21]:

```
plot_model(ensemble_model, show_layer_names=True, show_shapes=True)
```

In[18]:

```
ensemble_model.compile(optimizer='adam', loss='mse')
```

In[19]:

```
callbacks = [
    EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True),
    ModelCheckpoint('best_model.h5', monitor='val_loss', save_best_only=True)
]
```

In[20]:

```
history = ensemble_model.fit(X_tt, y_tt, validation_data=(X_test, y_test), epochs=100,
batch_size=4096, callbacks=callbacks)
```

In[23]:

```
plt.ylim([0, 0.04])
plt.plot(history.history['loss'], label = 'loss' )
plt.plot(history.history['val_loss'], label = 'val_loss' )
plt.legend()
```

1D-CNN model

In[18]:

```
# Input layer
input_layer = Input(shape=(60, 110))
```

Convolutional layers for each channel

```

cnn_layers = []
for i in range(3):
    cnn_layer = Conv1D(filters=8, kernel_size=3, activation='relu')(input_layer[:, :, i:i+1])
    cnn_layers.append(cnn_layer)

# Flatten the output of the convolutional layers
flattened_layers = [Flatten()(cnn_layer) for cnn_layer in cnn_layers]

# Merge the flattened layers into a single layer
merged_layer = concatenate(flattened_layers)

# Dense layers
dense_layer = Dense(units=32, activation='relu')(merged_layer)
dense_layer1 = Dense(units=32, activation='relu')(dense_layer)

# Output layer
output_layer = Dense(units=2, activation='linear')(dense_layer1)

# Create the model
model2 = Model(inputs=input_layer, outputs=output_layer)

# Compile the model
model2.compile(optimizer='adam', loss='mse')

```

In[19]:

```
plot_model(model2, show_shapes=True)
```

In[19]:

```

# Input layer
input_layer = Input(shape=(60, 110))

# Convolutional layers for each channel
cnn_layers = []
for i in range(110):
    cnn_layer = Conv1D(filters=8, kernel_size=3, activation='relu')(input_layer[:, :, i:i+1])
    cnn_layers.append(cnn_layer)

# Flatten the output of the convolutional layers
flattened_layers = [Flatten()(cnn_layer) for cnn_layer in cnn_layers]

# Merge the flattened layers into a single layer
merged_layer = concatenate(flattened_layers)

```

```
# Dense layers
dense_layer = Dense(units=32, activation='relu')(merged_layer)
dense_layer1 = Dense(units=32, activation='relu')(dense_layer)

# Output layer
output_layer = Dense(units=2, activation='linear')(dense_layer1)

# Create the model
model = Model(inputs=input_layer, outputs=output_layer)

# Compile the model
model.compile(optimizer='adam', loss='mse')
```

In[20]:

```
callbacks = [
    EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True),
    ModelCheckpoint('best_model.h5', monitor='val_loss', save_best_only=True)
]
```

In[21]:

```
history = model.fit(X_tt, y_tt, validation_data=(X_test, y_test), epochs=100, batch_size=2048)
```

In[26]:

```
plt.ylim([0, 0.04])
plt.plot(history.history['loss'], label = 'loss' )
plt.plot(history.history['val_loss'], label = 'val_loss' )
plt.legend()
```

```
# ## model
```

```
# ### tuner
```

In[]:

```
# Define the LSTM and Dense layer architecture
def create_model(hp):
```

```

model = Sequential()
input_shape = (X_tt.shape[1], X_tt.shape[2])

num_dense_layers = hp.Choice('num_dense_layers', values=[3])
num_units = hp.Choice('num_units', values=[32])
# use_regularization = hp.Choice('use_regularization', values=[True,False])
use_regularization = False
use_dropout = hp.Choice('use_dropout', values=[True,False])
learning_rate = hp.Float('learning_rate', 0, 1e-10)

model.add(LSTM(units=num_units, activation='relu', input_shape=(input_shape)))
for _ in range(num_dense_layers):
    if use_regularization:
        model.add(Dense(units=num_units, activation='relu', kernel_regularizer=l1_l2()))
    if use_dropout:
        model.add(Dropout(0.2))
    else:
        model.add(Dense(units=num_units, activation='relu'))
    if use_dropout:
        model.add(Dropout(0.2))
model.add(Dense(units=2, activation='linear'))

optimizer = Adam(learning_rate=learning_rate)
model.compile(optimizer=optimizer, loss='mse')
return model

```

In[]:

```

callbacks = [
    # EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True),
    ModelCheckpoint('best_model.h5', monitor='val_loss', save_best_only=True)
]

```

In[]:

```

class CVTuner(Tuner):
    def run_trial(self, trial, x, y, batch_size=4096, epochs=50):
        tscv = TimeSeriesSplit(5)
        val_losses = []
        for train_indices, test_indices in tscv.split(x):
            x_train, x_test = x[train_indices], x[test_indices]
            y_train, y_test = y[train_indices], y[test_indices]
            model = self.hypermodel.build(trial.hyperparameters)

```

```

    model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_data=(x_test,
y_test), callbacks=callbacks)
    val_losses.append(model.evaluate(x_test, y_test))
    self.oracle.update_trial(trial.trial_id, {'val_loss': np.mean(val_losses)})
    self.save_model(trial.trial_id, model)

```

In[]:

```

tuner = CVTuner(
    hypermodel=create_model,
    project_name = 'project4',
    oracle = BayesianOptimizationOracle(
        objective='val_loss',
        max_trials=40))

```

In[]:

```
tuner.search(X_tt, y_tt, batch_size=2048, epochs=10)
```

In[]:

```

best_hps = tuner.get_best_hyperparameters()[0]
best_model = tuner.hypermodel.build(best_hps)
print(best_model.summary())

```

model

In[]:

```

model = keras.models.Sequential()
model.add(LSTM(32, input_shape=(X_tt.shape[1], X_tt.shape[2])))
model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(2, activation='linear'))

```

In[]:

```
callbacks = [
    EarlyStopping(monitor='val_loss', patience=30, restore_best_weights=True),
    ModelCheckpoint('best_model.h5', monitor='val_loss', save_best_only=True)
]
```

In[]:

```
model.compile(loss='mean_squared_error', optimizer='adam')
history = model.fit(X_tt, y_tt, validation_data=(X_test, y_test), epochs=1000, batch_size=16384,
 callbacks = callbacks)
```

In[]:

```
plt.ylim([0, 0.1])
plt.plot(history.history['loss'][:200], label = 'loss' )
plt.plot(history.history['val_loss'][:200], label = 'val_loss' )
plt.legend()
```

In[]:

```
plot_model(model, show_shapes=True)
```

ДОДАТОК Б ПРЕЗЕНТАЦІЯ

Слайд 1

Дослідження ефективності алгоритмів машинного навчання для задачі управління активами на ринку криптовалют з урахуванням оцінки довіри користувачів до криптовалюти

Виконав:
студент 4 курсу, групи КА-93
Кротенко Нікіта Сергійович

Керівник:
Асистент кафедри ММСА
Канцедал Георгій Олегович

Слайд 2

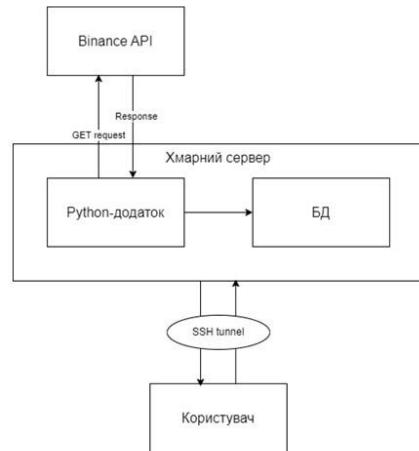
Об'єкт дослідження – дані про хвилинні свічки і глибину ринку (bid/ask)

Предмет дослідження – моделі для передбачення часових рядів.

Мета роботи – відтворення усього циклу реального дослідження, включаючи збір і підготовку даних, визначення цільових метрик та безпосередню розробку і налаштування моделей машинного навчання, для їх подальшого порівняння.

Слайд 3

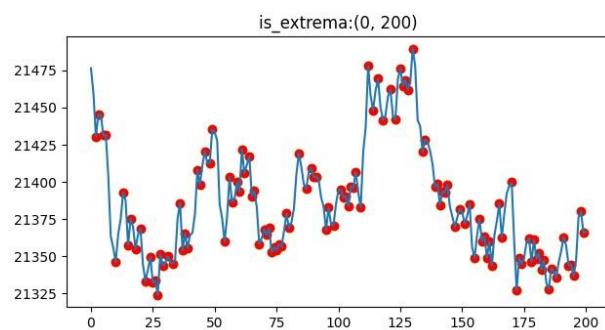
Блок-схема організації хмарного сервера для збору даних



Слайд 4

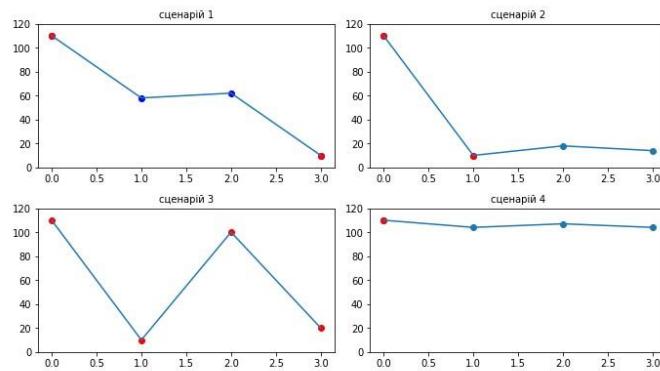
Створення розмітки даних

1. Виділення локальних екстремумів



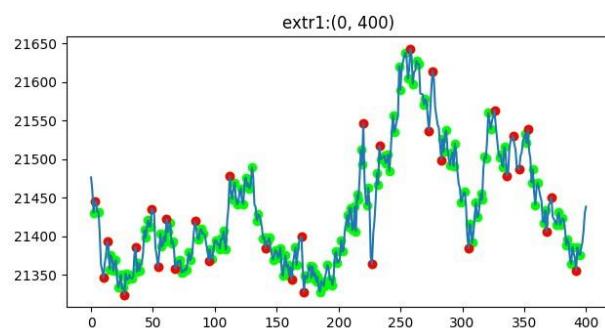
Слайд 5

Створення розмітки даних 2. Пошук початкових екстремумів



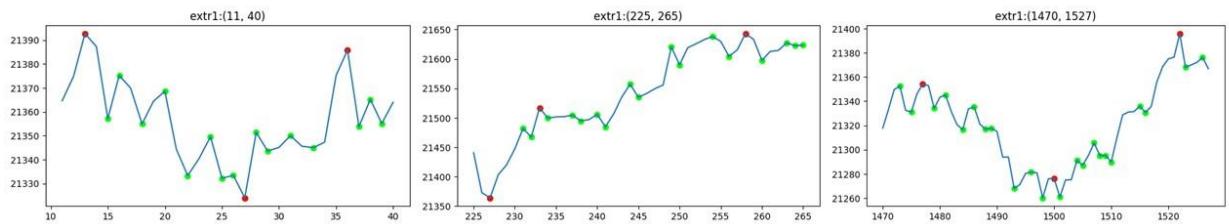
Слайд 6

Створення розмітки даних 2. Пошук початкових екстремумів



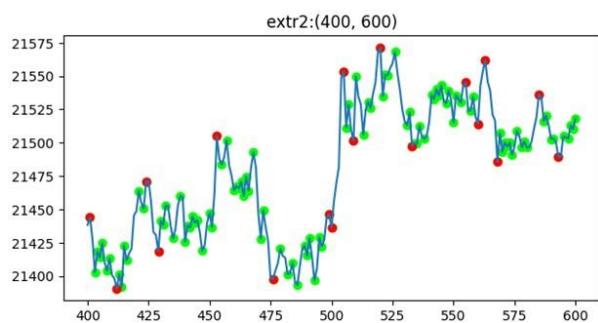
Слайд 7

Створення розмітки даних 3. Пошук глобальних екстремумів на відрізках



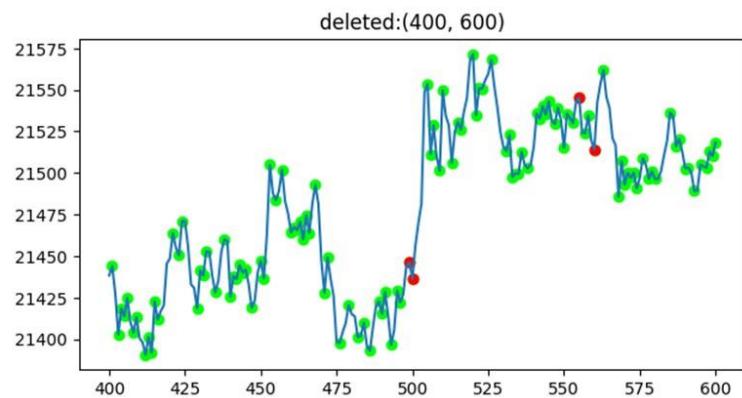
Слайд 8

Створення розмітки даних 3. Пошук глобальних екстремумів на відрізках



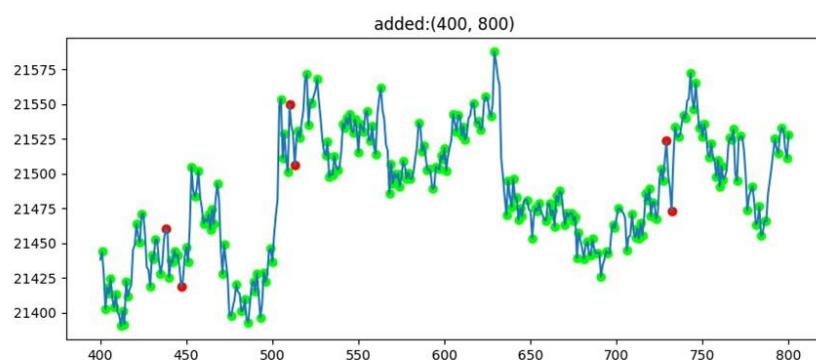
Слайд 9

Створення розмітки даних 4. Пошук неправильно визначених екстремумів



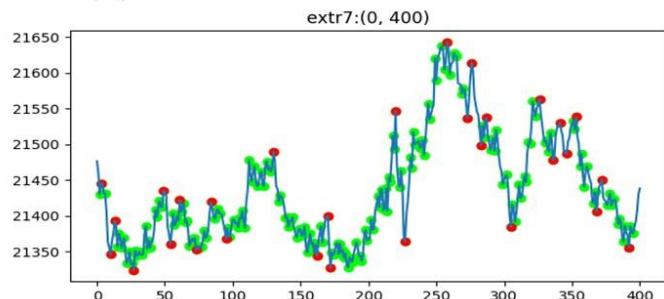
Слайд 10

Створення розмітки даних 5. Пошук пропущених екстремумів



Слайд 11

Створення розмітки даних Результати



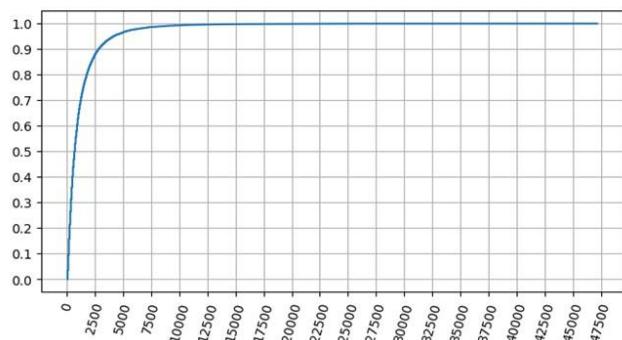
1. Екстремумів всього	188275
2. Екстремумів після виділення початкових	18620
3. Екстремумів після знайдення глобальних по відрізкам	16024
4. Екстремумів після видалення зайвих	15078
5. Екстремумів після додавання попередніх	18406
6. Екстремумів повтору кроків 4 і 5	18324

Слайд 12

Огляд створеної розмітки

Кумулятивний розподіл часу тривалості контрактів:

- 80.77% контрактів тривають до 30 хвилин;
- 93.48% контрактів тривають до 60 хвилин.



Слайд 13

Генерація нових ознак

1. По-хвилинні

$$high_low_diff_{perp} = high_{perp} - low_{perp}$$

$$open_close_diff_{perp} = open_{perp} - close_{perp}$$

$$bid_ask_spread_{perp} = \frac{askPrice_{perp} - bidPrice_{perp}}{askPrice_{perp}}$$

$$bid_ask_spread_abs_{perp} = askPrice_{perp} - bidPrice_{perp}$$

Та інші

Слайд 14

Генерація нових ознак

1. Ознаки групи «Короткострокові очікування»

$$short_term_expectations_1 = \frac{open_{perp} - open_{cq}}{time_left + 86400}$$

$$short_term_expectations_5 = \frac{(open_{perp} - open_{cq})^2}{time_left + 86400 * 9}$$

$$short_term_expectations_9 = \frac{(open_{perp} - open_{cq})^2}{\sqrt{time_left + 86400 * 90}}$$

$$short_term_expectations_{10} = \frac{\sqrt{|open_{perp} - open_{cq}|}}{time_left + 86400}$$

$$short_term_expectations_{14} = \frac{\sqrt{|open_{perp} - open_{cq}|}}{\sqrt{time_left + 86400 * 9}}$$

Та інші

Слайд 15

Генерація нових ознак

1. По-контрактні

Згрупувавши дані по контрактам, з колонок:

```
['volume_perp', 'trades_perp', 'volume_cq', 'trades_cq',
 'bidQty_perp', 'askQty_perp', 'bidQty_cq', 'askQty_cq']
```

беремо показники:

- sum – сума;
- mean – середнє;
- total_delta – різниця в ціні відкриття і ціні закриття контракту;
- minmax_rel_diff – відносна різниця в максимальній і мінімальній ціні що були зафіковані на проміжку доки було відкрито контракт.

Та інші.

Слайд 16

Вибір роду задачі: Регресія чи Класифікація

Недоліки задачі класифікації:

1. Модель не може отримати переваги від хвилинних коливань ціни і повинна закривати або відкривати контракти лише у фіксовану секунду хвилини і лише за ринковими ордерами.
2. Цільовий показник незбалансований, що ускладнює вибір метрики і потребує додаткових маніпуляцій над даними.

Переваги задачі регресії:

1. З можливістю відкривати лімітні ордери ми можемо отримати додатковий прибуток за рахунок коливання ціни контракту протягом хвилини.
2. За допомогою введення другої змінної ми можемо не відкривати контракти на інтервалах, на яких для отримання бажаного прибутку тривалість контракту перевищуватиме комфортні для нас 30 хвилин.

Враховуючи вищезазначені властивості було вирішено розглядати задачу як задачу регресії.

Слайд 17

Перший цільовий показник

Додатково визначимо 2 колонки:

$$up_border = \frac{high + max(open, close)}{2} \quad low_border = \frac{low + min(open, close)}{2}$$

Значення цільового показника 1:

Для мінімумів - low_border

Для максимумів - up_border

Слайд 18

Другий цільовий показник

Додатково визначимо 2 колонки:

- `min_over_upcoming_30min` – мінімальний показник `low_border`, який буде зафіксовано протягом наступних 30 хвилин;
- `max_over_upcoming_30min` – максимальний показник `up_border`, який буде зафіксовано протягом наступних 30 хвилин.

Значення цільового показника 2:

Для максимумів - `min_over_upcoming_30min`

Для мінімумів - `max_over_upcoming_30min`

Слайд 19

Фінальна обробка цільових показників

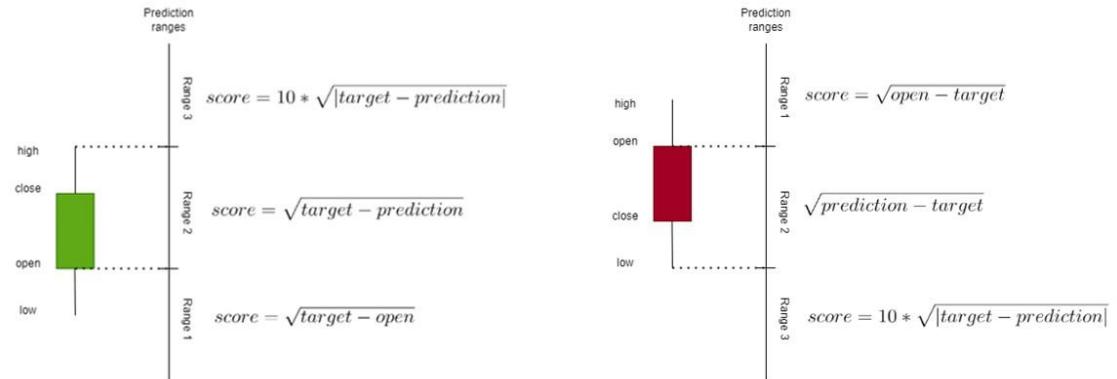
Тепер, маючи цільові показники для всіх рядків, що є екстремумами ми пересуваємо усі значення в цих колонках на клітинку назад і заповнюємо пропуски методом “backfill” – кожен пропуск заповнюється значенням наступної непорожньої клітинки.

Таким чином, після закриття кожної хвилинної свічки відкриватиметься новий лімітний ордер по ціні передбаченій в цільовому показнику 1.

У випадках, коли прибуток від контракту відкритого по ЦП1 і закритого по ЦП2 не перевищує 1.075% – актуальний контракт буде закрито по ціні ЦП1, але нового контракту відкрито не буде.

Слайд 20

Метрика



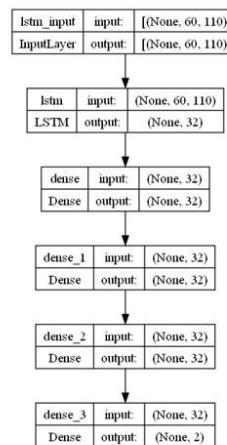
Слайд 21

Розбиття даних на тренувальну, тестову і валідаційну вибірки



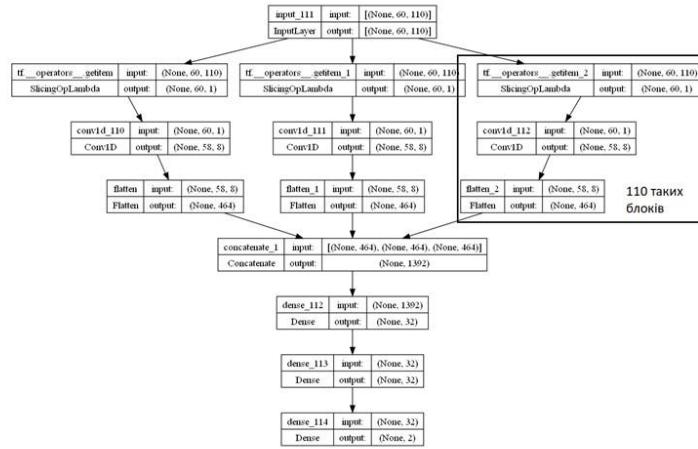
Слайд 22

Структура LSTM моделі



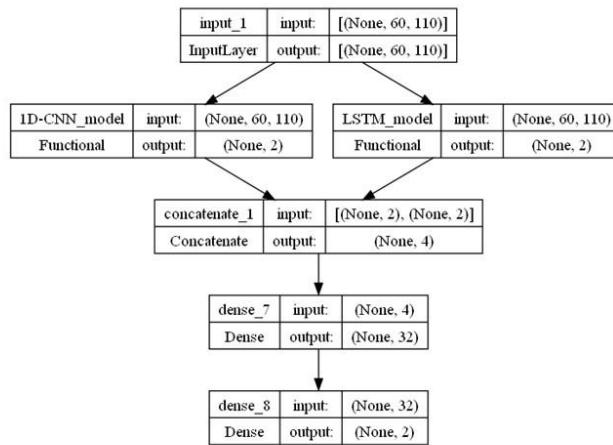
Слайд 23

Структура 1D-CNN моделі



Слайд 24

Структура ансамблевої моделі



Слайд 25

Порівняння моделей

Модель	Час навчання	Кількість епох	Похибка валідаційної вибірки	Похибка тренувальної вибірки
LSTM	5:33	140	6.184e-4	5.903e-5
1D-CNN	43:08	100	2.885e-4	1.358e-4
Ensemble	22:45	53	9.904e-4	1.841e-4

Слайд 26

Чому не можна напевно стверджувати, що знайдені архітектури мереж є оптимальними.

1. Множина можливих комбінацій гіпер-параметрів неповна (навіть в адекватних межах);
2. Початкові ваги нейронної мережі випадково ініціалізуються перед тренуванням. Ці ваги впливають на те, які важливі функції та особливості модель може навчитися розпізнавати. Так як ми тренували кожну мережу лише по одному разу - результати можуть значно змінитись при перетренуванні навіть вже існуючих мереж.

Слайд 27

Як можна покращити результати моделей

- 1) хоча й було налаштовано систему сповіщень, в даних все одно наявні пропуски в моментах, коли виникали помилки з боку Binance API або Python додатку. Так як пропуски даних виникали не часто і зазвичай не перевищували 20 хвилин - дані було заповнено методом інтерполяції;
- 2) реалізувати фільтрацію ознак;
- 3) пошук моментів відкриття/закриття був реалізований спираючись на ціну відкриття свічки. Оптимальніше було б це зробити спираючись на ознаки "low_border", "high_border", створені пізніше;
- 4) реалізувати автоматичну генерацію ознак за допомогою бібліотек tsfresh/tsfel;
- 5) реалізувати моделі на основі дерев рішень (RandomForest) і моделі на основі градієнтного бустингу (XGBoost, LightGBM, CatBoost) та інших моделей, спробувати інші ансамблі моделей

Слайд 28

Висновки

Незважаючи на те, що не всі заплановані підходи вдалося втілити, і не було повністю реалізовано всі потенційні можливості для зросту, в даній роботі було детально описано весь цикл такого виду досліджень. Він включав збір і підготовку даних, визначення цілей і ключових метрик, а також підготовку та налаштування моделей для подальшого порівняння.

Розглянута тема дуже широка і актуальна, а тому вона потребує подальшого дослідження. Це дипломна стане чудовою основою для майбутньої магістрської роботи.

Слайд 29

Дякую за увагу!