МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

СОКОЛОВСЬКИЙ Ростислав Ігорович

Модель агрегатора обміну криптовалют з можливістю прогнозування їх курсу / Cryptocurrency Exchange Aggregator Model with Price Prediction Capability

спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КНм-21
Р. І. Соколовський
_____

Науковий керівник:
к.т.н., доцент П.Є.Биковий
_____

Кваліфікаційну роботу
допущено до захисту
«___»_____20___р.

В.о. завідувача кафедри
_____Н. В. Дзюбановська

ТЕРНОПІЛЬ – 2025

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «магістр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.М. Васильків
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
СОКОЛОВСЬКИЙ Ростислав Ігорович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: **Модель агрегатора обміну криптовалют з можливістю прогнозування їх курсу / Cryptocurrency Exchange Aggregator Model with Price Prediction Capability**
керівник роботи Биковий Павло Євгенович, к.т.н, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 20 грудня 2024 року № 938.

2. Строк подання студентом закінченої кваліфікаційної роботи

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
- аналіз існуючих веб-ресурсів для обміну криптовалют та прогнозування їх курсу;
- опис загальної структури програмного агрегатора;
- інформаційне забезпечення агрегатора обміну криптовалют;
- інтерфейс користувача розробленого агрегатора;
- програмна реалізація агрегатора обміну та прогнозування курсу криптовалют;
- тестування розробленого агрегатора;
- аналіз результатів агрегації, обміну та прогнозування курсу криптовалют.

5. Перелік графічного матеріалу в роботі:
- схема алгоритму вибору гаманця для проведення оплати користувачем;
- діаграма алгоритму обміну криптовалют в агрегаторі;
- діаграма потоків даних модуля прогнозування;
- схема алгоритму підготовки даних та тренування моделі.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01.2025 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03.2025 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 20.052025 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 28.10.2025 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 11.11.2025 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 25.11.2025 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту у системі «Unicheck».	до 1.12.2025 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 04.12.2025 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.12. 2025 р.	

Студент_____ Соколовський Р.І.
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ к.т.н., доцент Биковий П.Є.
(підпис) (прізвище та ініціали)

РЕЗЮМЕ

Кваліфікаційна робота на тему «Модель агрегатора обміну криптовалют з можливістю прогнозування їх курсу» на здобуття освітнього ступеня «Магістр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 82 сторінки і містить 27 ілюстрацій, 2 додатка та 32 використаних джерела.

Метою роботи є створення веб-орієнтованого агрегатора обміну криптовалют з вбудованим модулем прогнозування курсу, зручним та інтуїтивно зрозумілим інтерфейсом.

Методи досліджень: у роботі використано методи системного аналізу для дослідження предметної області, об'єктно-орієнтованого проектування для побудови архітектури системи, методи математичної статистики та машинного навчання (алгоритм градієнтного бустінгу) для реалізації модуля прогнозування.

Результати дослідження: Розроблено гібридну програмну систему, що складається з серверної частини на базі фреймворку Laravel та аналітичного модуля мовою Python. Реалізовано алгоритми асинхронної агрегації даних із зовнішніх бірж у реальному часі. Створено модуль прогнозування на основі алгоритму XGBoost, який забезпечує точність бінарної класифікації тренду на рівні близько 86%. Розроблено адаптивний інтерфейс користувача з використанням технології Livewire для візуалізації торгових сигналів.

Розроблений програмний продукт може бути використаний приватними інвесторами та трейдерами для моніторингу ринку, мінімізації ризиків та автоматизації пошуку найвигідніших пропозицій обміну.

Ключові слова: КРИПТОВАЛЮТА, АГРЕГАТОР, ПРОГНОЗУВАННЯ, XGBOOST, LARAVEL, МАШИННЕ НАВЧАННЯ, ВЕБ-ЗАСТОСУНОК, ОБМІН ВАЛЮТ.

ABSTRACT

Qualification work on the topic " Cryptocurrency Exchange Aggregator Model with Price Prediction Capability" for the degree of "Master" in the specialty 122 "Computer Science" of the educational program "Computer Science" is written on 82 pages and contains 27 figures, 2 appendices, and 32 references.

The aim of the work is to create a web-oriented cryptocurrency exchange aggregator with a built-in rate forecasting module and a convenient, intuitive interface.

Research methods: system analysis methods were used to study the subject area, object-oriented design to build the system architecture, and methods of mathematical statistics and machine learning (gradient boosting algorithm) to implement the forecasting module.

Research results: A hybrid software system consisting of a server part based on the Laravel framework and an analytical module in Python has been developed. Algorithms for asynchronous aggregation of data from external exchanges in real time have been implemented. A forecasting module based on the XGBoost algorithm has been created, providing a trend binary classification accuracy of about 86%. An adaptive user interface using Livewire technology for visualizing trading signals has been developed.

The developed software product can be used by private investors and traders to monitor the market, minimize risks, and automate the search for the most profitable exchange offers.

Keywords: CRYPTOCURRENCY, AGGREGATOR, FORECASTING, XGBOOST, LARAVEL, MACHINE LEARNING, WEB APPLICATION, CURRENCY EXCHANGE.

ЗМІСТ

ВСТУП

Сучасна епоха глобальної економічної трансформації характеризується фундаментальними змінами у парадигмі фінансових відносин, рушійною силою яких стала цифровізація та масове впровадження технологій розподіленого реєстру. Поява та стрімка еволюція криптовалют ознаменували перехід від традиційної централізованої банківської системи до децентралізованих фінансових екосистем (DeFi), що відкрило безпрецедентні можливості для вільного руху капіталу. За останнє десятиліття цифрові активи, такі як Bitcoin, Ethereum та тисячі інших альткоїнів, пройшли шлях від експериментальних технологічних розробок до визнаних інструментів міжнародного інвестування, які інтегруються у портфелі найбільших хедж-фондів та платіжні системи транснаціональних корпорацій. Проте, незважаючи на стрімке зростання капіталізації ринку та його поступову інституціоналізацію, інфраструктура торгівлі віртуальними активами все ще перебуває на етапі становлення, що породжує низку технічних та економічних викликів для кінцевих користувачів.

Актуальність теми кваліфікаційної роботи зумовлена насамперед архітектурною специфікою криптовалютного ринку, головною ознакою якого є високий рівень фрагментації ліквідності. На відміну від класичних фондових ринків, де торгівля акціями зазвичай зосереджена на одній-двох національних біржах, обмін криптовалют відбувається цілодобово на сотнях різнорідних платформ — від великих централізованих бірж до децентралізованих протоколів обміну. Кожен із цих майданчиків функціонує як ізольована екосистема зі своїм власним балансом попиту та пропозиції, що неминуче призводить до виникнення цінових диспропорцій. В один і той самий момент часу вартість активу на різних біржах може суттєво відрізнятися, створюючи спреди, які, з одного боку, відкривають можливості для арбітражу, з іншого ставлять пересічного користувача перед складною проблемою вибору оптимального місця для здійснення транзакції. Ручний моніторинг десятків торгових терміналів є неефективним через фізичні обмеження людини та затримки у часі, що часто призводить до втрати потенційного прибутку або здійснення обміну за завищеним курсом.

Іншим критичним фактором, що визначає необхідність розробки нових програмних рішень, є екстремальна волатильність криптовалютних активів. Динаміка цін на цьому ринку часто має стохастичний, нелінійний характер і піддається впливу величезної кількості факторів: від макроекономічних новин та регуляторних рішень до активності в соціальних мережах та дій великих власників активів ("китів"). Традиційні методи технічного аналізу, засновані на простих індикаторах, в умовах сучасної алгоритмічної торгівлі втрачають свою ефективність. Користувачі, особливо новачки, часто стають заручниками емоційних рішень, реагуючи на короткострокові коливання ринку (FOMO), що призводить до значних фінансових втрат. Існуючі на ринку інформаційні ресурси переважно надають лише статичні дані про поточний стан ринку, залишаючи користувача сам на сам із проблемою прогнозування подальшого руху ціни. Відсутність доступних та надійних інструментів предиктивної аналітики створює суттєву інформаційну асиметрію, знижуючи ефективність ринку в цілому.

Враховуючи вищевикладене, виникає нагальна науково-практична потреба у створенні інтелектуальних систем-агрегаторів нового покоління. Таке програмне забезпечення повинно виступати посередником між користувачем та фрагментованим ринком, вирішуючи два ключові завдання: технічне — агрегація даних, та аналітичне — прогнозування трендів. Технічна складова передбачає створення високопродуктивних алгоритмів парсингу та нормалізації даних з API різних бірж у режимі реального часу, що дозволяє знаходити найкращу ціну виконання ордеру (Best Execution Price) з урахуванням комісій та глибини ринку. Аналітична ж складова вимагає інтеграції передових методів штучного інтелекту. Застосування алгоритмів глибокого навчання (Deep Learning), зокрема рекурентних нейронних мереж (RNN) типу LSTM (Long Short-Term Memory), дозволяє аналізувати великі масиви історичних даних, виявляти приховані патерни поведінки цін та будувати ймовірнісні моделі майбутніх коливань. Саме синергія механізмів миттєвої агрегації та інтелектуального прогнозування здатна надати користувачеві якісно новий інструмент для управління цифровими активами.

Метою кваліфікаційної роботи є підвищення ефективності та надійності торгових операцій на ринку криптовалют шляхом розробки моделі та програмної

реалізації веб-орієнтованого агрегатора з інтегрованим модулем прогнозування курсу на основі технологій машинного навчання.

Для досягнення поставленої мети необхідно вирішити комплекс взаємопов'язаних завдань. Насамперед, слід провести глибокий аналіз предметної області, дослідивши архітектурні особливості сучасних криптобірж, протоколи передачі даних (REST, WebSocket) та існуючі рішення-конкуренти для виявлення їхніх функціональних вад. Наступним етапом є теоретичне обґрунтування вибору математичних моделей: проведення порівняльного аналізу ефективності класичних економетричних моделей (наприклад, ARIMA) та сучасних методів нейромережевого моделювання у контексті прогнозування фінансових часових рядів. На основі цього аналізу здійснюється проектування архітектури програмної системи, що включає розробку структури бази даних для зберігання історичних котирувань, проектування мікросервісів для збору даних та створення API для взаємодії компонентів. Практична частина роботи присвячена безпосередній програмній реалізації агрегатора: написанню серверного коду (Backend), тренуванню та інтеграції моделі машинного навчання, а також створенню інтуїтивно зрозумілого інтерфейсу користувача (Frontend). Завершальним етапом є комплексне тестування розробленої системи, перевірка точності прогнозів на тестових вибірках даних та оцінка продуктивності системи під навантаженням.

Об’єктом дослідження є процеси інформаційного обміну в розподілених фінансових мережах та закономірності формування ринкової вартості криптоактивів. Предметом дослідження виступають методи та алгоритми агрегації гетерогенних даних з криптовалютних бірж, а також моделі штучного інтелекту для прогнозування динаміки фінансових інструментів.

Наукова новизна отриманих результатів полягає у подальшому розвитку методів побудови інформаційно-аналітичних систем для фінтех-сектору. Зокрема, запропоновано удосконалену модель агрегатора, яка відрізняється від існуючих інтеграцією адаптивного модуля прогнозування на базі нейронних мереж, що дозволяє генерувати попереджувальні сигнали про зміну тренду, підвищуючи якість прийняття інвестиційних рішень.

Практична значимість роботи полягає у створенні повнофункціонального

програмного продукту, готового до використання широкою аудиторією. Розроблений сервіс дозволяє трейдерам та інвесторам суттєво оптимізувати свою діяльність: економити час на пошуку найвигіднішого курсу, мінімізувати транзакційні витрати та знижувати ризики завдяки доступу до науково обґрунтованих прогнозів. Результати дослідження можуть бути використані як основа для створення комерційного стартапу або впроваджені в існуючі фінансові платформи.

Апробація результатів кваліфікаційної роботи підтверджує їх актуальність та відповідність сучасному рівню розвитку науки і техніки. Основні теоретичні положення та практичні результати дослідження були оприлюднені під час виступу на III Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих вчених «Інтелектуальні комп'ютерні системи та мережі» (тези доповіді «Методи та інструменти покращення користувацького досвіду в криптовалютних агрегаторах», с. 143-144). Також результати досліджень висвітлено у збірнику «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення» (випуск 105) у статті «Актуальність проектування модуля прогнозування курсу криптовалют». Копії публікацій наведено у додатку А, що засвідчує внесок автора у розробку тематики.

1 АНАЛІЗ ПЕРЕДУМОВ ПОБУДОВИ МОДЕЛІ АГРЕГАТОРА ОБМІНУ КРИПТОВАЛЮТ

1.1 Особливості криптовалютного ринку та виклики прогнозування курсів

Еволюція світової фінансової системи на початку XXI століття ознаменувалася фундаментальним технологічним зсувом, спричиненим появою концепції децентралізованих цифрових грошей. Криптовалюти, що виникли як відповідь на глобальну фінансову кризу 2008 року та кризу довіри до традиційних банківських інституцій, трансформувалися з теоретичного концепту криптоанархістів у потужний сектор глобальної економіки. За своєю онтологічною природою криптовалюта є різновидом цифрової валюти, емісія та облік якої базуються [5] на асиметричному шифруванні та застосуванні криптографічних примітивів. На відміну від фіатних грошей, вартість яких гарантується авторитетом держави [6] та податковою системою, криптовалюти функціонують у середовищі, де довіра до посередника (банку, платіжної системи, нотаріуса) замінюється довірою до математичного алгоритму та програмного коду. Фундаментальною технологією, що уможливила існування такої системи, є блокчейн — децентралізована база даних, що складається з послідовного ланцюжка блоків, кожен з яких містить інформацію про транзакції та криптографічне посилання на попередній блок (рисунок 1.1). Така архітектура забезпечує незмінність історії операцій: будь-яка спроба змінити дані в минулому вимагає перерахунку всіх наступних блоків на всіх вузлах мережі, що робить фальсифікацію економічно неможливою та технічно надскладною.

Сучасний ринок віртуальних активів визначається як динамічна гетерогенна екосистема, що включає тисячі цифрових одиниць, класифікованих за технічними стандартами та функціональним призначенням [7]. Стабільність децентралізованих мереж забезпечується алгоритмами консенсусу, зокрема Proof of Work та Proof of Stake, вибір між якими зумовлений необхідністю балансування в рамках «трилеми блокчейну» [8]. Економічна архітектура провідних криптовалют базується на дефляційних механізмах, а впровадження смарт-контрактів дозволило інтегрувати програмовані активи у сектор децентралізованих фінансів . Торговельна

інфраструктура представлена централізованими біржами (CEX) [9], які забезпечують високу ліквідність через off-chain транзакції, та децентралізованими платформами (DEX), що функціонують на основі автоматичних маркет-мейкерів [10], гарантуючи некастодіальність, проте несучи ризики проковзування ціни .

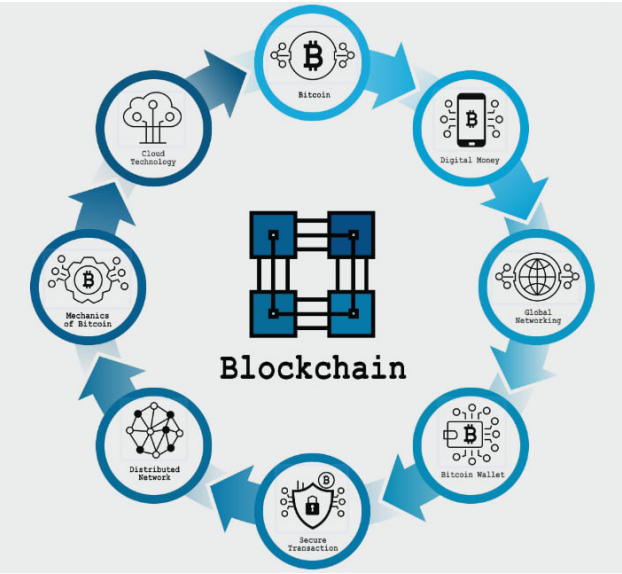


Рисунок 1.1 – Узагальнена схема функціонування технології блокчейн

На рисунку 1.2 наведено порівняльну схему архітектури та принципів функціонування централізованих (CEX) та децентралізованих (DEX) бірж.

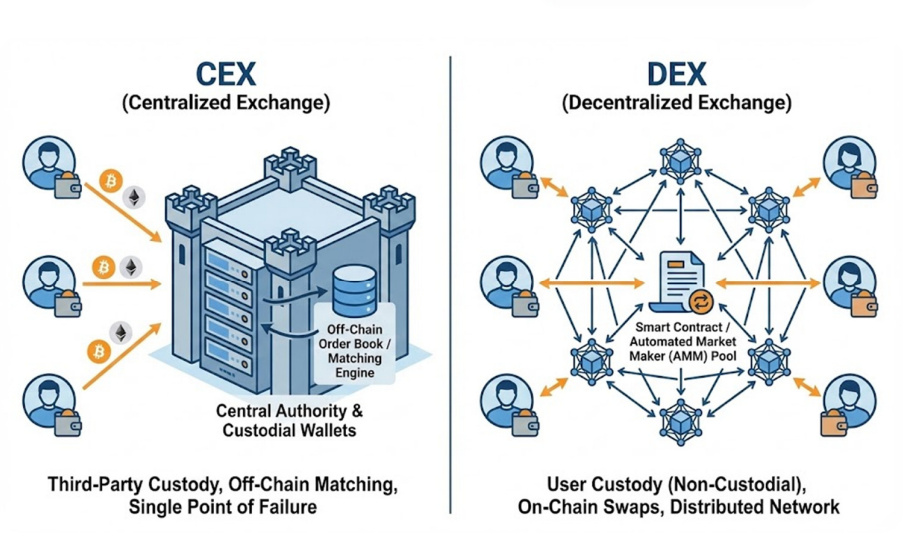


Рисунок 1.2 – Порівняльна схема архітектури та принципів функціонування централізованих (CEX) та децентралізованих (DEX) бірж

Ключовою структурною особливістю сучасного ринку віртуальних активів визначено глибоку фрагментацію інфраструктури, зумовлену функціонуванням сотень ізольованих торгових майданчиків (CEX та DEX). Така архітектура призводить до розмивання ліквідності та відсутності єдиного еталонного курсу, оскільки ціноутворення відбувається локально на основі балансу попиту та пропозиції конкретної біржі [11]. Наслідком децентралізації є виникнення суттєвих міжбіржових спредів, що створює інформаційну невизначеність. В умовах цінової дисперсії обґрунтованим є використання програмних агрегаторів, які забезпечують консолідацію даних через API в режимі реального часу, дозволяючи реалізувати принцип найкращого виконання ордерів (Best Execution) та мінімізувати втрати від курсової різниці [13].

Побудова релевантних прогностичних моделей ускладнюється екстремальною волатильністю ринку. З позиції економетрики, фінансові часові ряди криптовалют класифікуються як нестаціонарні процеси з вираженою гетероскедастичністю та наявністю значного стохастичного шуму [12]. Класичні інструменти технічного аналізу (ковзні середні, осцилятори) демонструють обмежену ефективність через нездатність врахувати нелінійну природу залежностей та вплив екзогенних факторів: ринкових маніпуляцій («wash trading»), каскадних ліквідацій позицій та високої чутливості до інформаційного фону .

Враховуючи обмеження дескриптивної аналітики, обґрунтовано перехід до предиктивного моделювання на базі гібридної програмної архітектури. Реалізація передбачає синергію двох технологічних стеків: серверна частина (Laravel) забезпечує надійну агрегацію потокових даних та взаємодію з API бірж, використовуючи механізми черг та асинхронну обробку подій. Аналітичний модуль побудовано з використанням алгоритму градієнтного бустінгу XGBoost. На відміну від рекурентних нейронних мереж, даний метод демонструє вищу обчислювальну ефективність на структурованих даних та стійкість до перенавчання завдяки регуляризації, що дозволяє з високою точністю вирішувати задачу бінарної класифікації ринкових трендів . Такий підхід визначає наукову новизну проекту, що полягає у створенні інструменту підтримки прийняття рішень, здатного нівелювати ризики фрагментації та волатильності ринку .

1.2 Поняття агрегатора обміну криптовалют та вимоги до його функціональних можливостей

В умовах стрімкої експансії сектору FinTech та зростання кількості незалежних торгових майданчиків виникає об'єктивна необхідність у впровадженні високотехнологічних систем, здатних впорядковувати гетерогенні потоки даних (Big Data). Ключовим інфраструктурним елементом, що вирішує проблему фрагментації ліквідності, виступає агрегатор обміну криптовалют. З точки зору системної інженерії, він класифікується як програмний комплекс класу High-Load, орієнтований на обробку інформації в режимі м'якого реального часу. Його функціональне ядро забезпечує автоматизований моніторинг публічних інтерфейсів (API) зовнішніх джерел, нормалізацію даних та їх приведення до уніфікованого стандарту [14].

Принциповою відмінністю агрегатора від класичних бірж є його некастодіальна природа: платформа не акумулює кошти користувачів і не виступає контрагентом угод, що мінімізує регуляторні ризики. Система виконує роль інтелектуального маршрутизатора (Smart Order Router), який автоматично спрямовує запити до постачальників ліквідності з найвигіднішими умовами, оптимізуючи транзакційні витрати [15].

Архітектурна модель комплексу, реалізована на базі фреймворку Laravel, побудована за принципом конвеєрної обробки даних. Процес ініціалізується генерацією пулу асинхронних запитів до API зовнішніх бірж, що дозволяє мінімізувати затримки (latency) при паралельному опитуванні джерел. Для вирішення проблеми структурної гетерогенності вхідних потоків (різні формати JSON/XML, часові пояси) імплементовано проміжний шар ETL (Extract, Transform, Load). Цей модуль забезпечує синтаксичний розбір, валідацію та трансформацію даних у внутрішній стандарт системи, після чого активується алгоритм ранжування пропозицій за критерієм Best Execution Price .

У глобальній інфраструктурі ринку агрегатор виступає центральним комунікаційним вузлом (Hub), що забезпечує двосторонню маршрутизацію. З одного боку, система акумулює потоки котирувань від незалежних бірж,

виконуючи роль концентратора ліквідності, а з іншого — слугує єдиною точкою входу для користувача, трансформуючи клієнтські запити в стандартизовані команди для зовнішніх API. Така топологія нівелює складність прямої взаємодії з фрагментованим ринком [16].

На рисунку 1.3 зображено концептуальну схему функціонування розробленого агрегатора, що відображає топологію інформаційних потоків між ключовими компонентами системи. Центральний серверний вузол (Backend) виступає комунікаційним шлюзом, який забезпечує агрегацію даних про курси валют від зовнішніх постачальників ліквідності (криптовалютних бірж) через API-інтерфейси та передає їх до модуля прогнозування (AI/ML) для аналізу трендів. Оброблена інформація, що включає найкращі пропозиції обміну та згенеровані торгові сигнали, транслюється через веб-інтерфейс кінцевому користувачеві для підтримки прийняття рішень.

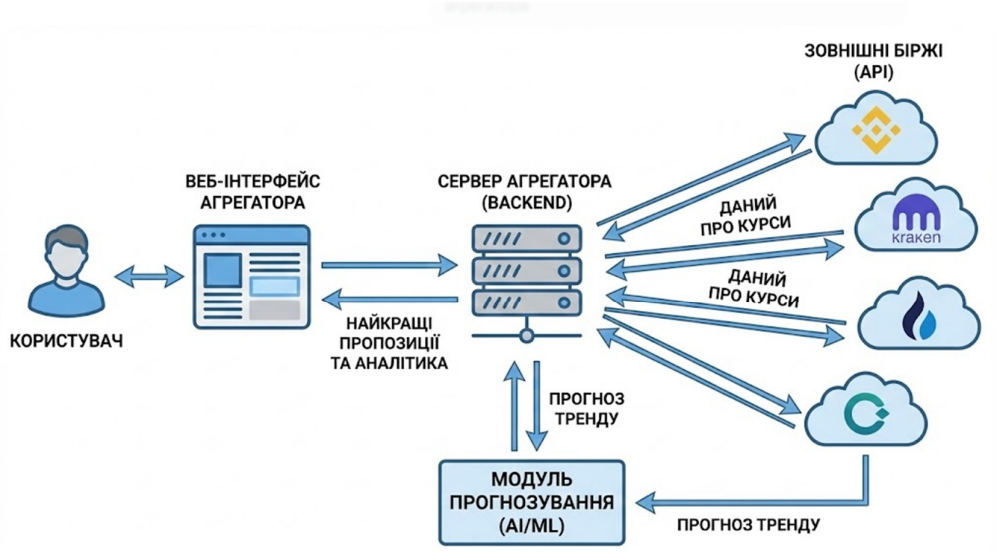


Рисунок 1.3 – Концептуальна схема роботи агрегатора

В умовах перманентної турбулентності та високої волатильності ринку віртуальних активів функціонал класичного моніторингу втрачає свою ефективність як єдиний інструмент управління капіталом, що зумовлює об'єктивну необхідність переходу від дескриптивної до предиктивної аналітики . Для вирішення цього завдання архітектуру розроблюваної системи розширено

інтелектуальним модулем прогнозування, побудованим на базі алгоритму XGBoost (eXtreme Gradient Boosting). Вибір даного інструментарію є науково обґрунтованим з огляду на табличну природу вхідних фінансових даних формату OHLCV, для яких методи градієнтного бустінгу над деревами рішень емпірично демонструють вищу стабільність та точність порівняно з глибокими нейронними мережами, що часто страждають від надмірної складності при обробці структурованої інформації.

Методологічною основою функціонування модуля виступає принцип ансамблевого навчання, згідно з яким прогнозна модель формується не як єдиний монолітний алгоритм, а як адитивна композиція послідовно побудованих «слабких» класифікаторів. Процес навчання реалізується ітеративно, де кожне нове дерево рішень математично спрямоване на корекцію помилок (мінімізацію залишків), допущених попередніми ітераціями ансамблю, що дозволяє досягти високої узагальнюючої здатності системи.

Критичною перевагою імплементації XGBoost в умовах зашумленого криптовалютного ринку є наявність вбудованих механізмів регуляризації L1 (Lasso) та L2 (Ridge), які штрафують модель за надмірну складність та ефективно запобігають перенавчанню на історичних даних. Додатковою технічною перевагою виступає оптимізація алгоритму під паралельні обчислення та роботу з розрідженими матрицями, що дозволяє системі оперувати масивами даних обсягом понад 3 мільйони записів та забезпечувати генерацію прогнозів у режимі реального часу з мінімальними затримками.

На рисунку 1.4 представлено структурну схему алгоритму XGBoost, що демонструє принцип послідовної побудови ансамблю моделей. Ключова ідея полягає в ітеративному навчанні: кожне нове дерево рішень створюється не для незалежного прогнозування, а для виправлення помилок (залишків) попередніх моделей. Фінальний результат формується як сума прогнозів усіх побудованих дерев, що дозволяє поступово мінімізувати загальну похибку та перетворити набір слабких класифікаторів на сильний предиктивний інструмент.

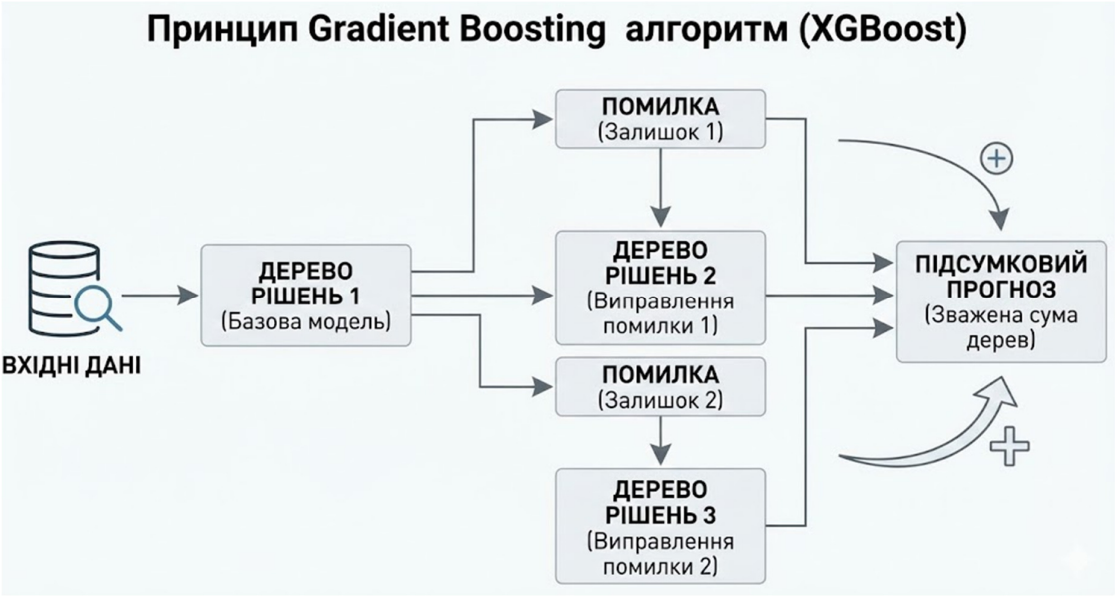


Рисунок 1.4 – Принцип роботи алгоритму градієнтного бустінгу (XGBoost)

Ключовою методологічною особливістю та науковою новизною розроблюваної системи є нестандартний підхід до формалізації задачі машинного навчання. Замість традиційного для фінансового моделювання використання методів регресійного аналізу, метою яких є передбачення конкретного числового значення майбутньої вартості активу (наприклад, прогнозування ціни Bitcoin з точністю до долара), було обрано стратегію бінарної класифікації ринкових станів.

Таке рішення обумовлене тим, що задача регресії на високошвидкоплинних та зашумлених криптовалютних ринках часто призводить до значної похибки (високого значення RMSE), оскільки намагається апроксимувати стохастичний хаос. Натомість, задача класифікації фокусується на визначенні вектора руху ринку, що є більш стійким до незначних цінових флуктуацій.

У рамках запропонованої моделі простір можливих майбутніх станів ринку дискретизується на два взаємовиключні класи. Клас «1» (Target = 1) ідентифікує позитивний сценарій ("Bullish trend"), коли ціна закриття наступного часового інтервалу перевищує поточну, що інтерпретується системою як сигнал до відкриття довгої позиції (купівлі). Клас «0» (Target = 0) відповідає негативному або нейтральному сценарію ("Bearish/Sideways trend"), коли ціна знижується або залишається незмінною, що є сигналом до продажу активу або утримання від входу в ринок.

Така трансформація складної ймовірнісної задачі у чітку бінарну логіку дозволяє конвертувати абстрактні математичні виводи моделі у конкретні, зрозумілі для кінцевого користувача торгові рекомендації ("Buy" / "Hold"), суттєво спрощуючи процес прийняття інвестиційних рішень та мінімізуючи когнітивне навантаження на трейдера .

Досягнення високих показників валідаційної точності класифікатора (Accuracy), яка за результатами тестування наближається до позначки 86%, стало можливим завдяки реалізації комплексного підходу до етапу попередньої обробки даних та конструювання ознак (Feature Engineering). Використання немодифікованих («сирих») рядів цін відкриття та закриття виявилося недостатнім для якісного моделювання, тому архітектура системи спирається на аналіз похідних синтетичних метрик, що глибше розкривають характер ринкової динаміки.

Ключовим елементом аналізу виступає розрахунок дельти тіла свічки (open-close), яка слугує математичним вираженням вектора та інтенсивності внутрішньоденного тренду. Паралельно з цим обчислюється амплітуда цінових коливань (low-high) — різниця між екстремумами торговельної сесії, що є прямим індикатором рівня поточної волатильності активу. Окрім суто технічних параметрів, модель інтегрує фактори календарної сезонності, зокрема бінарний маркер закінчення фінансового кварталу (is_quarter_end). Врахування цього параметра є критично важливим, оскільки саме в такі періоди інституційні інвестори проводять ребалансування портфелів, що провокує аномальні зміни ліквідності. Релевантність обраного набору ознак для задачі прогнозування була підтверджена результатами кореляційного аналізу .

На рисунку 1.5 представлена кореляційна матриця («теплова карта»), що візуалізує ступінь взаємозв'язку між вхідними ознаками (features) та цільовою змінною (target) у наборі даних. Числові значення коефіцієнтів кореляції Пірсона (від -1 до 1) та відповідна кольорова гама дозволяють ідентифікувати найбільш значущі фактори, що впливають на рух ціни, а також виявити мультиколінеарність між змінними (наприклад, між open, high, low, close), що є критично важливим для етапу відбору ознак (Feature Selection) перед навчанням моделі.

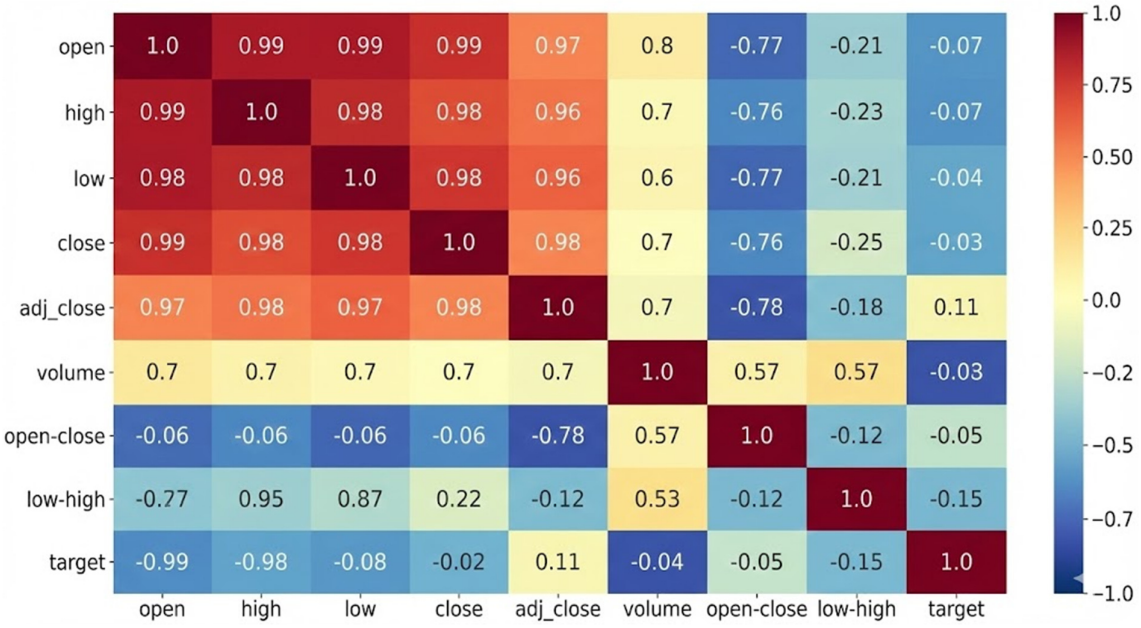


Рисунок 1.5 – Кореляційна матриця

Поєднання функціоналу агрегатора на базі Laravel та прогнозного модуля на Python/XGBoost створює синергетичний ефект. Бекенд забезпечує безперебійний потік актуальних даних, а модуль машинного навчання в реальному часі оцінює ринкову ситуацію. Це дозволяє нівелювати головний недолік існуючих аналогів — відсутність вбудованої аналітичної підтримки. Користувач отримує комплексний сервіс, який не лише автоматизує пошук найкращої ціни, але й надає математично обґрунтовані сигнали для прийняття рішень, знижуючи ризики фінансових втрат в умовах невизначеності.

1.3 Аналіз існуючих сервісів обміну й систем прогнозування криптовалют

Стрімка експансія ринку віртуальних активів та проблема інформаційного перенасичення актуалізували потребу в надійній інфраструктурі, що зумовило появу спектру програмних продуктів: від базових моніторингів до складних AI-платформ. Для обґрунтування доцільності розробки нового агрегатора проведено аналіз провідних світових аналогів з метою виявлення незаповнених функціональних ніш [17].

Одним із ключових інструментів у цьому сегменті є платформа

WalletInvestor, яка спеціалізується на технічному прогнозуванні вартості криптовалют та традиційних активів. Функціонал сервісу базується на пропрієтарних алгоритмах «Smart Technical Analysis», що генерують різностроковi прогнози з візуалізацією цільових цінових рівнів та сентимент-аналізом ринкових настроїв (Bullish/Bearish) . Проте суттєвими недоліками системи визначено закритість алгоритмів, що унеможливлює верифікацію їхньої надійності, а також відсутність вбудованих механізмів обміну, що обмежує функціонал платформи виключно інформаційною підтримкою .

На рисунку 1.6 представлено загальний вигляд інтерфейсу головної сторінки аналітичної платформи WalletInvestor.

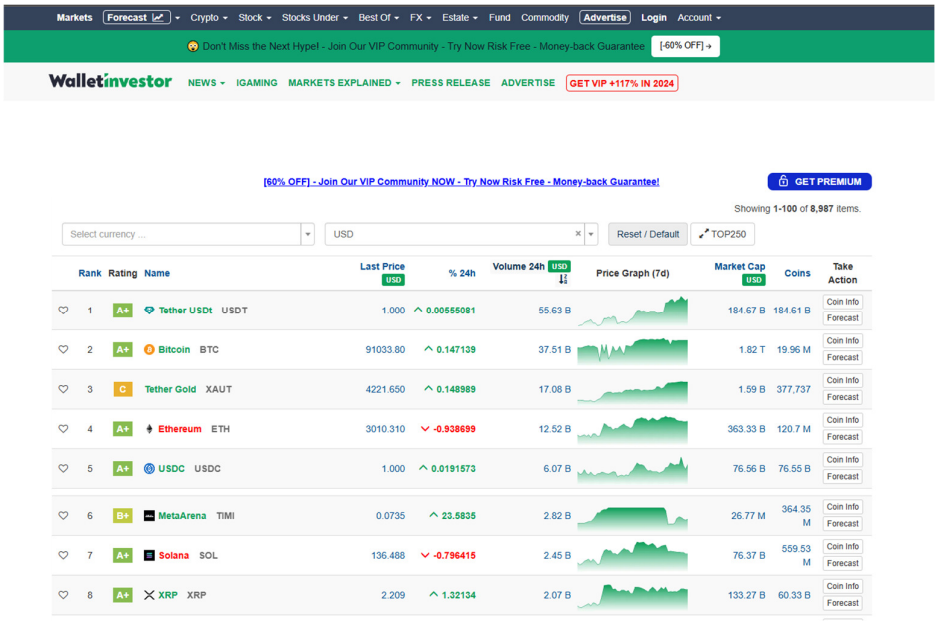


Рисунок 1.6 – Прогнозних графіків та цільових рівнів на платформі WalletInvestor

Альтернативним інструментом фундаментального аналізу є платформа CoinCheckup, яка використовує інтегральну метрику «Algo. Score» для алгоритмічної оцінки потенціалу криптоактивів. Система агрегує дані про активність розробників у GitHub-репозиторіях та соціальних мережах, формуючи комплексний рейтинг інвестиційної привабливості проектів . Прогностичний модуль сервісу базується на методах екстраполяції історичних даних та порівняльних моделях темпів зростання технологічних секторів. Попри потужний

дослідницький потенціал, платформа характеризується обмеженою ефективністю для активного трейдингу через перевантажений інтерфейс та відсутність власного агрегатора ліквідності, оскільки функція обміну реалізована виключно через партнерські посилання .

На рисунку 1.7 представлено загальний вигляд інтерфейсу аналітичної платформи CoinCheckup.

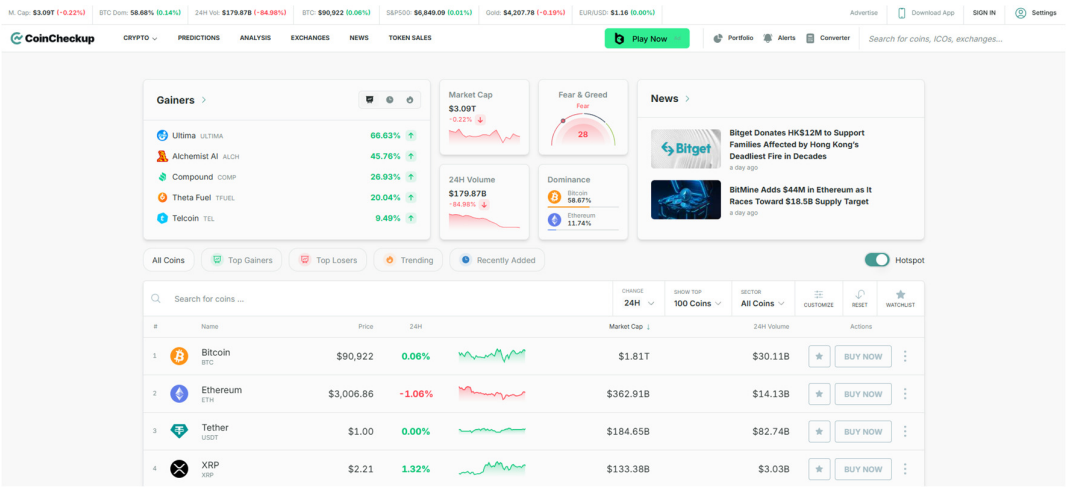


Рисунок 1.7 – Інтерфейс аналітичної платформи CoinCheckup

Індустріальним стандартом моніторингу ринку віртуальних активів визначено платформу CoinMarketCap, яка агрегує структуровані дані про понад десять тисяч активів. Функціональне ядро системи базується на алгоритмах розрахунку середньозваженої ціни за обсягом (Volume Weighted Average Price), що дозволяє нівелювати міжбіржові відхилення та формувати глобальний індикативний курс. Завдяки цьому ресурс набув статусу бенчмарку, дані якого використовуються провідними фінансовими інституціями та регуляторами .

Водночас у контексті дослідження виявлено фундаментальне обмеження платформи, яке полягає у її функціонуванні виключно в парадигмі дескриптивної аналітики. Сервіс фіксує історичні та поточні стани ринку, проте не містить інструментів предиктивного моделювання на базі машинного навчання. Відсутність можливості екстраполяції трендів створює «прогностичний вакуум», що змушує трейдерів залучати додаткове програмне забезпечення для прийняття інвестиційних рішень .

На рисунку 1.8 представлено загальний вигляд користувацького інтерфейсу агрегатора ринкових даних CoinMarketCap.

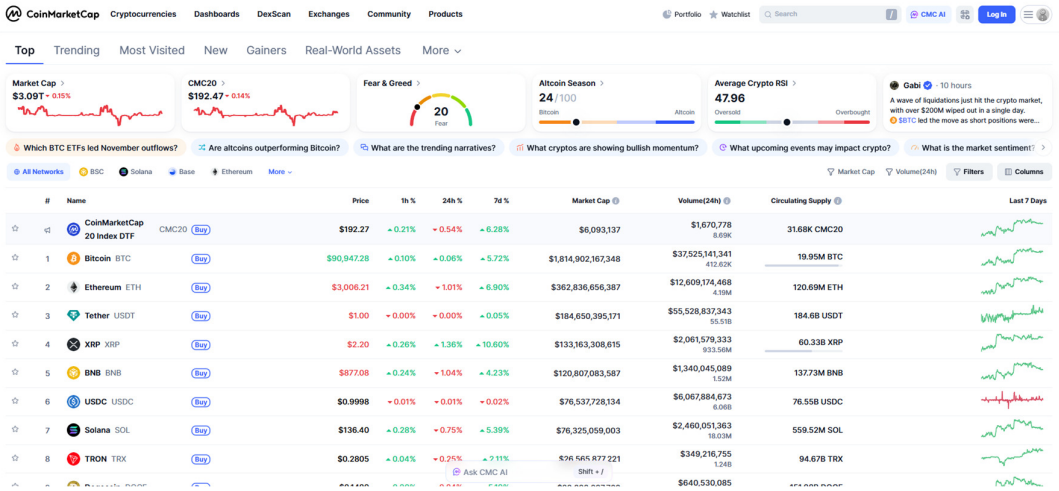


Рисунок 1.8 – Загальний вигляд агрегатора ринкових даних CoinMarketCap

Еталонним інструментом у сегменті професійного алгоритмічного трейдингу визначено платформу Cryptowatch, архітектура якої оптимізована для мінімізації затримок (low latency) при передачі потокових даних. Система забезпечує моніторинг книги ордерів та історії угод у режимі реального часу, надаючи користувачам доступ до терміналу з розширеним набором індикаторів технічного аналізу (MACD, RSI, смуги Болліджера) . Завдяки інтеграції API-ключів реалізовано можливість безпосереднього управління активами та розміщення ордерів на підключених біржах через єдиний інтерфейс.

Водночас встановлено, що технологічна складність та перевантаженість інтерфейсу створюють високий поріг входження, роблячи платформу малопридатною для пересічних користувачів. Критичним обмеженням системи в контексті автоматизації є відсутність вбудованих моделей машинного навчання, що залишає процес прогнозування курсу повністю ручним та залежним від суб'єктивної інтерпретації візуальних патернів трейдером .

На рисунку 1.9 представлено інтерфейс професійного терміналу Cryptowatch, призначеного для здійснення поглибленого технічного аналізу ринкових даних.

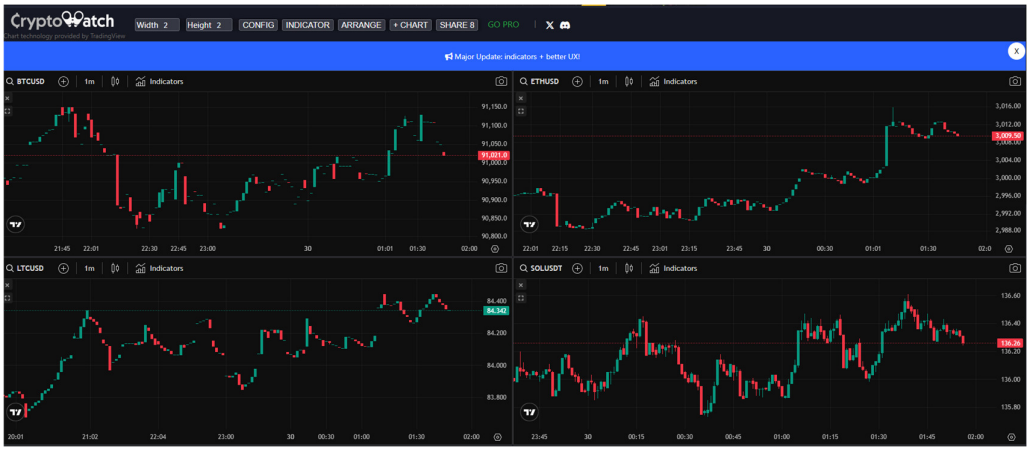


Рисунок 1.9 – Професійний термінал для технічного аналізу Cryptowatch

У сегменті обміну фіатних та цифрових валют домінують сервіси моніторингу на кшталт BestChange, які функціонують як агрегатори пропозицій обмінних пунктів, обробляючи дані в реальному часі. Проте це рішення технологічно обмежене через відсутність аналітичної складової, що змушує користувачів діяти, орієнтуючись виключно на поточну ціну без врахування ринкового контексту, збільшуючи ризики фінансових втрат через волатильність. Аналіз ринку виявив його суттєву фрагментарність, де існує чіткий розподіл спеціалізації на інформаційні сервіси, платформи аналізу, торгові термінали та сервіси прогнозування. Це підтвердило відсутність гібридного рішення, яке б одночасно поєднувало функціонал агрегатора ліквідності з інтелектуальною системою підтримки прийняття рішень (DSS). Розроблювана система покликана заповнити цю нішу, інтегруючи пошук оптимального курсу з предиктивною аналітикою на базі машинного навчання, що усуває необхідність використання користувачем розрізнених інструментів.

На таблиці 1.1 зображено порівняльний аналіз функціональних можливостей існуючих на ринку аналогічних систем, які пропонують послуги прогнозування та аналітики криптовалют, а також розроблюваної системи. Цей аналіз дозволяє чітко ідентифікувати переваги та наукову новизну створеного інструменту порівняно з конкурентами, оцінюючи такі ключові критерії, як основна функція, метод прогнозування (наприклад, технічний, фундаментальний чи машинне навчання), тип даних, що використовуються, наявність можливості обміну та складність (UX) використання.

Таблиця 1.1 – Порівняльний аналіз функціональних можливостей існуючих аналогів та розроблюваної системи

Критерій	**WalletInvestor**	**CoinCheckup**	**CoinMarketCap**	**Cryptowatch**	**Розроблювана система**
Основна функція	Технічні прогнози	Фундаментальний аналіз	Ринкові дані	Термінал трейдера	Агрегація + Прогноз
Метод прогнозу	Smart Tech Analysis (закритий)	Algo. Score (алгоритмічний)	Відсутній	Ручний тех. аналіз	XGBoost (Machine Learning)
Тип даних	Графіки трендів	Рейтинги проектів	Історія цін	Живі графіки	Сигнал (Buy/Sell)
Можливість обміну	Ні	Ні	Ні (посилання)	Так (через API)	Так (агрегація)
Складність (UX)	Середня	Висока	Низька	Дуже висока	Низька

Проектована система формує новий стандарт користувацького досвіду у сфері обміну цифрових активів, відходячи від традиційної парадигми складних професійних інтерфейсів. Вибір технологічного стеку на базі фреймворку Laravel обумовлений можливостями реалізації реактивного веб-інтерфейсу, що відповідає сучасним вимогам ергономіки та забезпечує інтуїтивну зрозумілість взаємодії. На відміну від аналогів, які часто характеризуються перевантаженими візуальними рішеннями, розроблена система ставить у пріоритет швидкість та простоту навігації .

Ключовою конкурентною перевагою визначено зниження когнітивного навантаження на користувача. Завдяки оптимізованому UI/UX-дизайну, складні процеси агрегації даних та математичного моделювання залишаються прихованими від кінцевого користувача. Інтелектуальний модуль на базі XGBoost трансформує масиви фінансових даних у візуально зрозумілі торгові сигнали (наприклад, кольорові індикатори тренду), що дозволяє оцінювати ринкову ситуацію без необхідності аналізу технічних індикаторів .

Такий підхід змінює парадигму взаємодії з торговою платформою, фокусуючи увагу на критично важливих показниках: оптимальному курсі та прогнозі руху ціни. Використання сучасних фронтенд-технологій гарантує повну адаптивність системи для десктопних та мобільних пристроїв. Синергія потужного бекенду, прогностичної моделі та людиноорієнтованого дизайну дозволяє мінімізувати шлях користувача від входу в систему до прийняття рішення, роблячи професійну аналітику доступною для широкої аудиторії .

1.4 Постановка задачі розробки моделі агрегатора з інтегрованим модулем прогнозування

Узагальнюючи результати аналізу предметної області та існуючих технологічних рішень, основну мету дипломного проекту можна сформулювати як створення комплексної веб-орієнтованої системи, що вирішує проблему інформаційної невизначеності та фрагментації на ринку криптовалют. Ключовим викликом, що стоїть перед розробником, є не просто технічна реалізація механізмів збору даних, а створення синергетичного середовища, яке поєднує миттєву агрегацію ліквідності з інтелектуальною аналітикою. Реалізація такої системи вимагає вирішення триєдиного комплексу завдань: інженерного (збір та обробка даних), математичного (прогнозування трендів) та ергономічного (взаємодія з користувачем).

Першочерговим інженерним завданням є проектування та розробка підсистеми агрегації даних на базі фреймворку Laravel. Цей модуль повинен функціонувати як високопродуктивний шлюз, що забезпечує безперервну та стабільну взаємодію з публічними API зовнішніх криптовалютних бірж. Специфіка задачі полягає в необхідності обробки гетерогенних потоків інформації: кожна біржа використовує власні формати даних, часові пояси та протоколи передачі. Система повинна виконувати роль універсального адаптера, який у реальному часі нормалізує вхідні дані, приводячи їх до єдиного внутрішнього стандарту. Критичною вимогою є мінімізація часу відгуку (latency), оскільки в умовах високої волатильності ціна активу може змінитися за частки секунди. Агрегатор має автоматично ранжувати отримані пропозиції, формуючи для користувача консолідований список ("стакан") з найкращими цінами для купівлі та продажу, що дозволяє миттєво ідентифікувати найвигіднішу точку входу в ринок без необхідності моніторингу десятків веб-сайтів.

Другим, наукомістким вектором роботи є розробка та імплементація модуля прогнозування на основі методів машинного навчання. Спираючись на попередні дослідження, задача прогнозування ставиться не як спроба точного передбачення вартості активу в майбутньому, що часто є стохастичним процесом з високою

похибкою, а як задача бінарної класифікації ринкового тренду. Модуль повинен аналізувати історичні дані та визначати клас майбутнього стану ринку: «зростання» або «падіння». Для вирішення цієї задачі обрано алгоритм XGBoost, який демонструє високу ефективність на табличних фінансових даних. Важливим етапом є формування простору ознак (Feature Engineering): система не повинна обмежуватися лише «сирими» цінами відкриття чи закриття, а має розраховувати похідні індикатори, такі як внутрішньоденна волатильність та сила тренду. Такий підхід дозволить моделі виявляти приховані нелінійні закономірності, недоступні для класичного технічного аналізу.

Окремим, стратегічно важливим вектором розробки є проектування архітектури взаємодії з користувачем (UX) та візуального інтерфейсу (UI). Враховуючи, що будь-які фінансові операції, особливо на волатильному ринку криптовалют, неминуче пов'язані з підвищеним рівнем стресу, інтерфейс системи має виконувати роль стабілізуючого фактора. Він повинен бути не лише функціональним, а й емоційно комфортним, створюючи у користувача відчуття безпеки та повного контролю над ситуацією.

Головне завдання дизайну в цьому контексті — максимальне зниження когнітивного навантаження. Замість того, щоб змушувати користувача аналізувати перевантажені таблиці та складні технічні графіки, система повинна фільтрувати інформаційний шум і надавати чіткі візуальні акценти. Особлива увага приділяється візуалізації даних від модуля штучного інтелекту: сухі математичні ймовірності мають трансформуватися у зрозумілі графічні метафори (наприклад, кольорові індикатори тренду або шкали впевненості), які зчитуються миттєво, на інтуїтивному рівні.

Ергономіка системи будується навколо принципу мінімізації зусиль: користувацький шлях (User Flow) від оцінки ринкової ситуації до натискання кнопки обміну має бути максимально коротким, лінійним і позбавленим зайвих кроків. Саме такий людино-орієнтований підхід дозволяє перетворити складний аналітичний інструмент на доступний і зручний сервіс для щоденного використання .

Висновки до розділу 1

1. Проведено аналіз ринку криптовалют та існуючих програмних рішень, який виявив потребу в створенні гібридної системи, що поєднує агрегацію ліквідності з предиктивною аналітикою. Сформульовано, що завершальним і критично важливим етапом дослідження є фінальна інтеграція розроблених модулів у єдину екосистему, що передбачає безшовне поєднання різнорідних технологічних стеків.

2. Визначено, що для реалізації поставленої мети необхідно побудувати надійний архітектурний міст між клієнтською частиною на Laravel та обчислювальним ядром на Python, налаштувавши ефективний транспортний механізм для миттєвої передачі даних та інтеграції прогнозів. Особливу увагу при проектуванні акцентовано на стабільності сервісу: система повинна забезпечувати безперебійну роботу основного веб-ресурсу навіть у випадку затримок від модуля прогнозування.

3. Успішна реалізація цих архітектурних рішень дозволить створити продукт, який вигідно вирізняється на тлі конкурентів. На відміну від типових сервісів із «сухою» статистикою, розроблена система позиціонується як інтелектуальний помічник, що бере на себе складну аналітичну роботу та у візуально зручній формі спрощує користувачеві процес прийняття рішень.

2 РОЗРОБКА МОДЕЛІ АГРЕГАТОРА ОБМІНУ КРИПТОВАЛЮТ

2.1 Архітектурна модель системи

Проектування архітектури системи базується на необхідності інтеграції гетерогенних технологічних стеків для забезпечення взаємодії веб-інтерфейсу, серверної частини, зовнішніх джерел даних та аналітичного модуля. Ключовим завданням визначено створення механізму синхронізації компонентів, що гарантує безперебійну обробку потокових даних та мінімізацію затримок. Обрана архітектурна модель відповідає трьом критичним вимогам: високій швидкодії для обробки запитів у режимі реального часу, масштабованості для адаптації до зростання навантаження та інтеграції нових бірж, а також надійності й безпеці фінансових транзакцій .

В основу розробки покладено клієнт-серверну архітектуру, що передбачає чітке розмежування функціональних зон відповідальності. Клієнтська частина (Frontend) відповідає за візуалізацію даних та інтерактивну взаємодію з користувачем через веб-браузер, забезпечуючи ергономічність інтерфейсу. Серверна частина (Backend) виконує роль обчислювального ядра системи, забезпечуючи збереження історичних даних, виконання складних алгоритмів прогнозування та обробку транзакцій. Такий підхід дозволяє оптимізувати розподіл обчислювальних ресурсів, знижуючи навантаження на кінцеві пристрої користувачів .

2.1.1 Архітектурний патерн MVC та його реалізація в Laravel

Фундаментом серверної частини виступає фреймворк Laravel, який реалізує індустрії для веб-систем подібного класу, оскільки дозволяє ізолювати логіку обробки даних від їх візуального представлення .

Нижче наведено детальну реалізацію кожного компонента MVC у розроблюваній системі:

1. Модель (Model) - це найнижчий рівень бізнес-логіки, який відповідає за структуру даних та пряму взаємодію з СУБД. У проекті використовуються ORM (Object-Relational Mapping) Eloquent, що дозволяє працювати з таблицями бази

даних як з класами PHP. Модель User інкапсулює логіку аутентифікації та зв'язки з транзакціями користувача. Модель Transaction містить методи для зміни статусів заявок та розрахунку комісій. Модель Cryptocurrency слугує довідником активів, що дозволяє централізовано керувати списком доступних монет.

2. Представлення (View) - цей шар відповідає за те, що бачить користувач. У системі використовуються шаблони Blade, які компілюються у чистий HTML на стороні сервера. Це забезпечує високу швидкість початкового завантаження сторінки (SEO-friendly) та безпеку від XSS-атак завдяки автоматичному екрануванню змінних. Для реалізації динамічних елементів (графіки, оновлення курсів без перезавантаження) в шаблони Blade інтегровані компоненти Livewire .

3. Контролер (Controller) - виступає диспетчером системи. Він приймає вхідний HTTP-запит, проводить його валідацію (перевірку коректності даних), викликає необхідні сервіси для обробки та повертає результат у вигляді View або JSON-відповіді.

Контролери в системі згруповані за функціональними доменами: AuthController (безпека), ExchangeController (курси), TransactionController (операції) архітектурний патерн MVC (Model-View-Controller). Цей патерн є стандартом

На рисунку 2.1 представлено детальну схему організації потоку даних у межах архітектурного патерну MVC (Model-View-Controller), яка ілюструє повний життєвий цикл обробки запиту в розроблюваній системі. Процес взаємодії розпочинається з надходження HTTP-запиту від користувача, який потрапляє на маршрутизатор (Router) і перенаправляється до відповідного контролера, попередньо проходячи через шар проміжного програмного забезпечення (Middleware) для обов’язкової валідації та перевірки прав доступу. Ключову роль у цій структурі відіграє контролер (Controller), який діє як координатор бізнес-логіки він ініціює запити до моделі (Model) для отримання необхідної інформації з бази даних (Database), обробляє отримані результати та передає їх у компонент відображення.

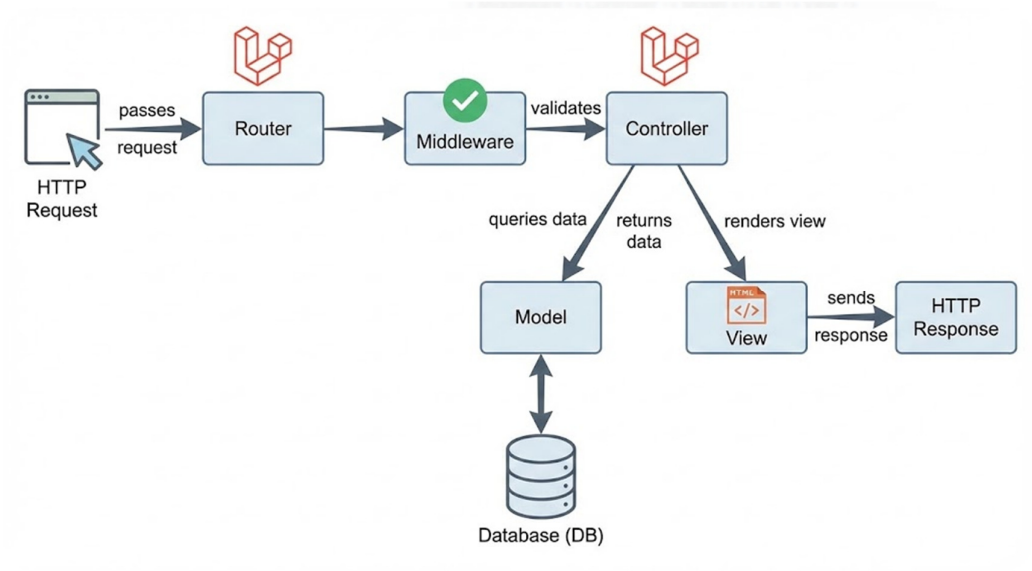


Рисунок 2.1 – Схема потоку даних у патерні MVC

2.1.2 Багаторівнева структура серверної логіки

Для уникнення проблеми "товстих контролерів" (Fat Controllers), коли код стає нечитабельним та складним у підтримці, в архітектуру системи впроваджено додаткові шари абстракції: Сервісний шар (Services) та Репозиторії (Repositories). Це дозволяє дотримуватися принципів SOLID, зокрема принципу єдиної відповідальності.

2.1.2.1 Сервісний шар (Service Layer)

Сервіси містять "чисту" бізнес-логіку, яка не залежить від того, звідки прийшов запит (з веб-браузера, мобільного додатку чи командного рядка).

ExchangeService - це ключовий компонент агрегатора. Він відповідає за логіку взаємодії із зовнішніми API. Сервіс містить методи для опитування бірж (Binance, Kraken, ByBit), обробки отриманих JSON-масивів та вибору найкращого курсу (Best Execution).

Саме тут реалізовано алгоритм порівняння цін:

$P_{best} = \min(P_{exchange\_1}, P_{exchange\_2}, \dots)$.

Сервіс також керує кешуванням, щоб не перевищувати ліміти запитів до бірж.

TransactionService - інкапсулює логіку проведення фінансових операцій.

Він перевіряє валідність гаманців, розраховує фінальну суму з урахуванням мережевих комісій (Gas Fees) та ініціює запити до платіжних шлюзів.

2.1.2.2 Патерн Репозиторій (Repository Pattern)

Репозиторії виступають посередниками між сервісами та моделлю даних. Вони абстрагують складні SQL-запити у зрозумілі методи. Замість того, щоб писати довгі SQL-запити в контролері для отримання історії цін за останній місяць, контролер просто викликає метод $exchangeRepository->getHistory('BTC', '30days'). Це дозволяє у майбутньому легко змінити базу даних (наприклад, з MySQL на PostgreSQL або MongoDB) без необхідності переписувати весь код проекту, змінивши лише один файл репозиторію.

На рисунку 2.2 наведено схему багаторівневої взаємодії архітектурних шарів системи, яка ілюструє послідовність передачі даних від контролера через сервісний шар та репозиторій до бази даних.

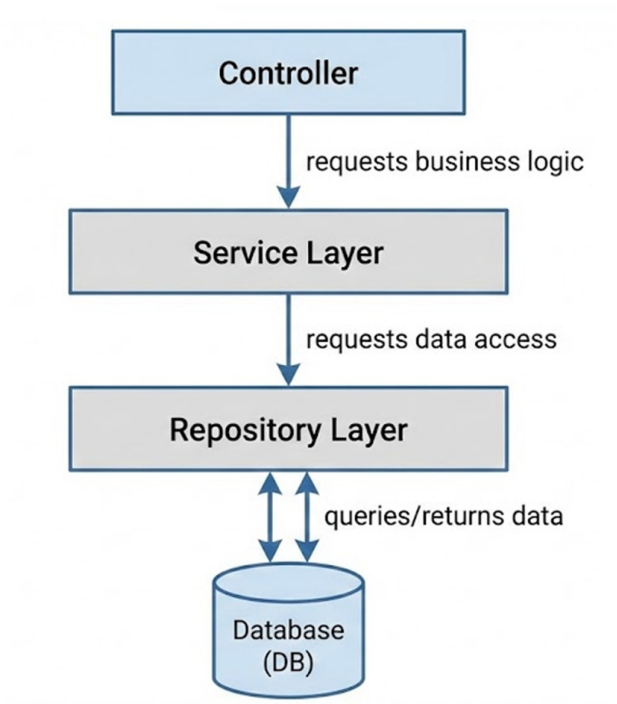


Рисунок 2.2 – Схема взаємодії шарів: Controller -> Service -> Repository -> Database

2.2 Алгоритми взаємодії з модулем машинного навчання

Ключовим архітектурним рішенням у розробці системи стала реалізація гібридної моделі, що поєднує високопродуктивний веб-фреймворк Laravel для серверної частини та спеціалізоване середовище Python для виконання математичних обчислень і машинного навчання. Такий підхід дозволив використати переваги обох технологій: швидкість обробки HTTP-запитів PHP та потужний аналітичний інструментарій Python, винісши модуль прогнозування в ізольований скрипт. Механізм взаємодії компонентів реалізовано за чітким алгоритмом, що розпочинається з ініціалізації запиту користувачем на сторінці аналітики. У цей момент контролер Laravel звертається до бази даних через ExchangeRepository для вибірки історичних даних (OHLCV) за необхідний період .

Наступним етапом є передача даних між середовищами, для чого використовується формат серіалізації JSON. Масив історичних цін трансформується у текстовий рядок, після чого ініціюється виконання Python-скрипта безпосередньо з командного рядка сервера за допомогою компонента Symfony\Component\Process. Аналітичний модуль здійснює десеріалізацію вхідних даних та виконує етап інженерії ознак (Feature Engineering), розраховуючи необхідні технічні індикатори, такі як волатильність. Після підготовки даних скрипт завантажує попередньо натреновану модель XGBoost (файл .json або .pkl) для виконання інференсу, результатом якого є бінарний вердикт щодо ймовірного руху ціни .

Завершальний етап включає отримання та інтерпретацію результату. Python-скрипт виводить прогнозоване значення у стандартний потік (stdout), яке перехоплюється backend-частиною на Laravel. Отримана інформація декодується та трансформується у візуальний торговий сигнал для інтерфейсу користувача. Запропонована архітектура дозволила інтегрувати професійні бібліотеки аналізу даних (pandas, scikit-learn, xgboost) без втрати продуктивності основного веб-застосунку.

На рисунку 2.3 представлено схему потоків даних, що ілюструє механізм інтеграції серверної частини (Laravel Backend) та модуля машинного навчання

(Python ML Module). Візуалізовано алгоритм взаємодії компонентів, який включає передачу серіалізованих історичних даних, їх обробку аналітичним скриптом та повернення результату прогнозування до основного веб-застосунку для подальшої інтерпретації .

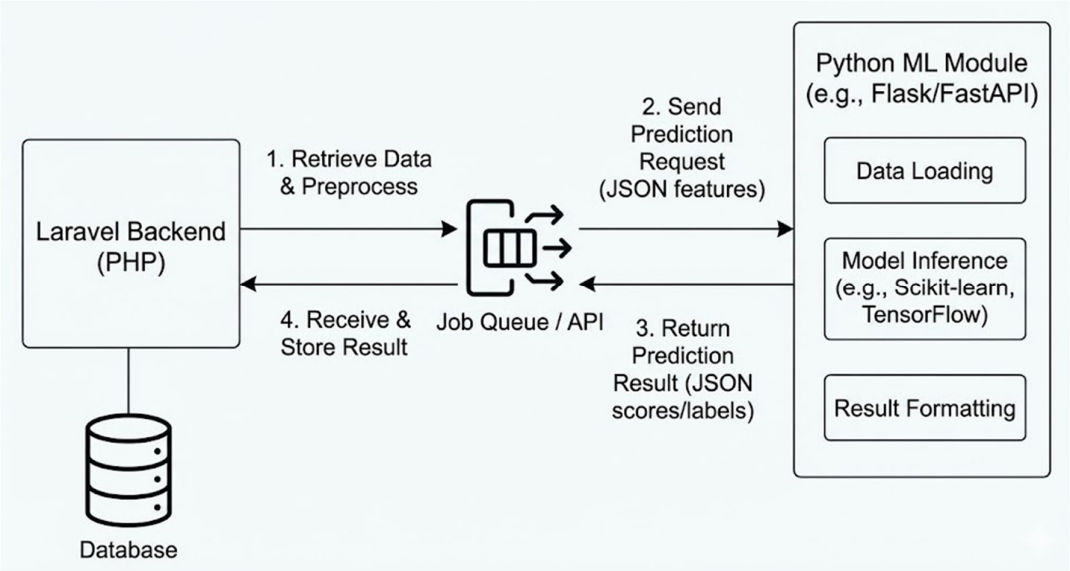


Рисунок 2.3 – Схема потоків даних між Laravel Backend та Python ML Module

2.2.1 Система маршрутизації та Middleware

При розробці зовнішнього вигляду сайту (фронтенду) я свідомо відмовився від застарілого підходу, коли після кожного кліку вся сторінка перезавантажується наново. Я розумію, що це повільно і негативно впливає на досвід користувача. Однак, замість того щоб ускладнювати інфраструктуру важкими та громіздкими технологіями, такими як React чи Vue, я вирішив використати елегантне рішення під назвою Livewire, яке є "рідним" інструментом для обраного мною фреймворку Laravel.

Використання Livewire дозволило зробити сайт «живим» без зайвих складнощів. Було реалізовано логіку роботи наступним чином: коли користувач, наприклад, змінює валюту у випадаючому списку, відбуваються такі дії:

1. Запропонований компонент миттєво відправляє короткий невидимий запит (AJAX) на сервер.

2. На стороні сервера відбувається перерахунок курсу та оотримується новий прогноз.

3. Сервер повертає готову частину HTML-коду, яким автоматично замінюється застаріла частину сторінки в браузері.

Завдяки такому підходу було забезпечено користувачеві відчуття роботи з миттєвим мобільним додатком (SPA): сторінка не моргає, а цифри оновлюються в реальному часі. При цьому було збережено простоту архітектури та забезпечено хорошу SEO-оптимізацію, оскільки початковий контент рендериться на сервері. Для того щоб інтерфейс виглядав сучасно та адаптивно на будь-якому пристрої, було використано технологію Tailwind CSS, яка дозволила мені конструювати дизайн із готових блоків.

2.2.2 Клієнтська частина та реактивність (Livewire)

При створенні зовнішнього вигляду сайту (фронтенду) запроновано відійти від застарілого підходу, коли після кожного кліку вся сторінка перезавантажується. Запропоновано, щоб не ускладнювати систему важкими та громіздкими технологіями, як React чи Vue, використати рішення під назвою Livewire, яке є "рідним" інструментом для основного двигуна Laravel.

Livewire дозволяє робити сайт «живим» без зайвих складнощів. Працює це наступним чином, коли користувач обирає у списку іншу валюту:

1. Система миттєво відправляє короткий невидимий запит на сервер.
2. Сервер швидко перераховує курс і робить новий прогноз.
3. Замість того щоб оновлювати весь екран, сервер надсилає назад лише маленький шматочок готового коду з новими цифрами, який автоматично замінює старі дані на екрані.

Для користувача все відбувається миттєво: сторінка не моргає, а цифри просто змінюються, наче у сучасному мобільному додатку (SPA). При цьому сайт залишається простим у підтримці та добре розпізнається пошуковими системами типу Google (SEO). А щоб сайт гарно виглядав комп’ютері і на телефоні було використано технологію Tailwind CSS.

2.2.3 Інфраструктура даних та безпека

Фундаментом для збереження всієї важливої інформації в запропонованій системі слугує база даних MySQL. Головна перевага цієї системи — це вміння гарантувати цілісність транзакцій (так звана ACID-сумісність), тобто система працює за принципом «все або нічого»: якщо під час переказу коштів станеться технічний збій (наприклад, вимкнеться світло на сервері), гроші не «зависнуть у повітрі». База даних або повністю завершить операцію, або скасує її, повернувши баланси користувачів до початкового стану. Крім того, MySQL надзвичайно швидко знаходить потрібну інформацію у величезних таблицях, що дозволяє миттєво показувати користувачеві історію його операцій .

Питання безпеки було для нас пріоритетним, оскільки ведеться робота із зовнішніми біржами. Щоб захистити обмін даними, запропоновано використати технологію JWT (цифрові токени). Вони працюють як тимчасова електронна перепустка: замість того, щоб постійно передавати через інтернет паролі, які можуть перехопити хакери, система використовує спеціальні зашифровані підписи. Найважливіші секрети — ключі доступу до бірж (API-ключі) — надійно сховані у спеціальному захищеному файлі налаштувань (.env) глибоко всередині сервера. Звичайний відвідувач сайту або зловмисник з інтернету фізично не має доступу до цього файлу, що зводить ризик крадіжки даних до мінімуму .

Таким чином, запропонована архітектура нагадує зібраний з ідеально підігнаних деталей механізм. Ми поєднали швидкість та надійність веб-платформи Laravel, потужний інтелект модуля прогнозування на Python (XGBoost) та сучасні технології дизайну (Livewire/Tailwind). Разом вони створюють міцний, безпечний та зручний фундамент, на якому працює запропонований криптовалютний агрегатор.

2.2.4 Алгоритмічне забезпечення криптовалютного агрегатора

Розробляючи дану систему враховано, що саме якісні алгоритми є тим фундаментом, на якому тримається вся ефективність проекту. Адже від того, наскільки продуманими будуть інструкції в коді, залежить і точність обробки даних, і, що найголовніше, безпека грошових транзакцій та швидкість відгуку

сайту на дії користувача. Тому запропоновано чітко розділити зони відповідальності на два окремі напрямки.

Перший напрямок - це алгоритми, які безпосередньо обслуговують користувача та відповідають за фінансові операції. Цю критично важливу частину було реалізовано на боці веб-сервера Laravel. Його головне завдання — миттєво реагувати на кліки, перевіряти введені дані та безпечно проводити платежі.

Другий напрямок - це алгоритми «інтелектуального» аналізу даних, які було винесено у окремий модуль машинного навчання. Вони займаються виключно складною математикою та прогнозуванням і не відволікаються на обслуговування інтерфейсу. Такий роздільний підхід дозволив максимально ефективно використати ресурси сервера: важкі обчислення прогнозів ніяк не гальмують роботу сайту, а процеси обробки транзакцій залишаються незалежними, швидкими та захищеними від будь-яких затримок в аналітичному блоці .

2.2.5 Алгоритми агрегації даних та обробки транзакцій

Ключовим елементом системи є підсистема збору даних та процесингу платежів, яка забезпечує безперервну взаємодію між користувачем, внутрішньою базою даних та зовнішніми провайдерами ліквідності. Процес обробки інформації структурно поділено на два етапи: алгоритм пошуку та відображення даних (агрегація) та алгоритм управління життєвим циклом фінансової операції .

Для мінімізації затримок при отриманні котирувань реалізовано алгоритм асинхронного паралельного опитування API зовнішніх бірж, що дозволяє отримувати ринкові дані одночасно, обмежуючись лише часом відповіді найповільнішого джерела. Отримані гетерогенні дані у форматі JSON підлягають нормалізації та приведенню до єдиного внутрішнього стандарту (DTO), після чого відбувається фільтрація аномальних значень, ранжування пропозицій за вигодою та кешування результатів для зниження навантаження на систему .

Алгоритм обробки транзакцій розпочинається з процедури суворої валідації вхідних даних, зокрема перевірки формату адреси гаманця та відповідності стандартам мережі (ERC-20, TRC-20). На наступному етапі активується механізм управління пулом адрес: система здійснює пошук у базі даних вільного («hot»)

гаманця, не задіяного в інших операціях. У разі виявлення такого гаманця він тимчасово блокується для поточної заявки, а за його відсутності ініціюється API-запит до інфраструктурного сервісу для генерації та реєстрації нової унікальної адреси.

На рисунку 2.4 представлено схему алгоритму автоматизованого вибору та призначення криптовалютного гаманця для здійснення оплати. Логіка процесу передбачає першочергову перевірку наявності вільних адрес у базі даних; у разі їх відсутності система ініціює звернення до зовнішнього API для генерації нового реквізиту, його реєстрації в системі та прив'язки до поточної транзакції, що забезпечує безперервність обробки платежів.

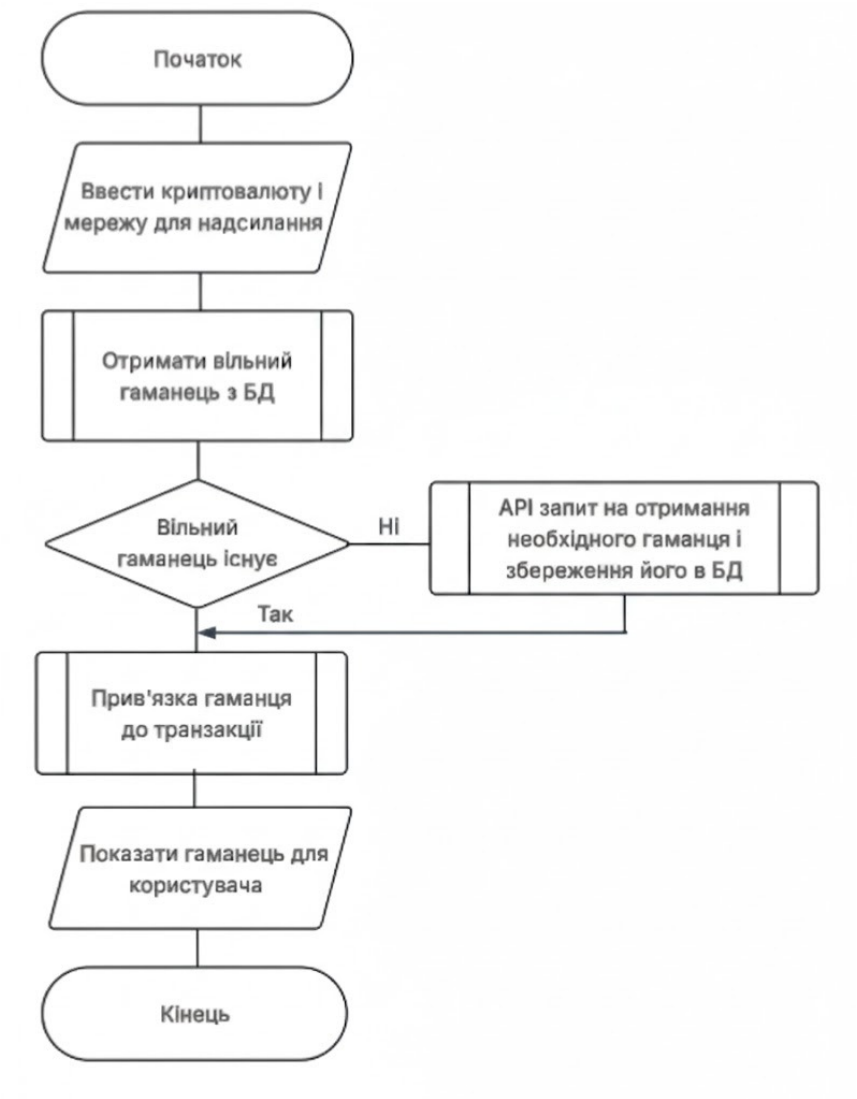


Рисунок 2.4 – Схема алгоритму вибору гаманця для проведення оплати користувачем

З огляду на високу динаміку ринку та ризики волатильності, в алгоритм обробки транзакцій інтегровано механізм часових обмежень. Після генерації платіжних реквізитів активується таймер зворотного відліку тривалістю 20 хвилин, протягом яких система гарантує фіксацію курсу в межах допустимого спреду. Паралельно ініціюється процес активного моніторингу надходження коштів на транзитну адресу з використанням технологій Polling або Webhook Listening. У випадку відсутності надходження активів після завершення відведеного часу, статус транзакції автоматично змінюється на «Скасовано» (Expired), що слугує тригером для очищення ресурсів: зарезервований гаманець звільняється та повертається до загального пулу доступних адрес, оптимізуючи роботу бази даних та зовнішніх API .

Автоматична фіксація факту надходження коштів у блокчейн-мережі виступає тригером для запуску процедури суворої верифікації транзакції. Цей етап передбачає алгоритмічне порівняння фактично отриманої суми із заявленою в ордері, при цьому система застосовує механізм адаптивного допуску, враховуючи можливі відхилення, спричинені волатильністю мережевих комісій. Тільки після успішного проходження валідації статус транзакції атомарно змінюється на «Підтверджено», що, в свою чергу, ініціює відправку захищеного запиту (API-call) до платіжного шлюзу для здійснення фінального переказу активів користувачу. Для забезпечення прозорості та можливості подальшого аудиту реалізовано наскрізне логування всіх етапів процесингу, а успішне завершення циклу супроводжується автоматичною генерацією електронної квитанції та синхронізацією історії операцій в особистому кабінеті клієнта в режимі реального часу.

На рисунку 2.5 детально наведено діаграму алгоритму, що формалізує повний цикл операції обміну криптовалют. Графічна модель охоплює всю послідовність бізнес-логіки: від ініціалізації заявки та автоматизованого підбору транзитних реквізитів до безперервного моніторингу стану транзакції у мемпулі блокчейну. Особливу увагу приділено логічним розгалуженням для фінальної верифікації, що гарантує цілісність та безпеку обмінної операції.

Рисунок 2.5 – Діаграма алгоритму обміну криптовалют в агрегаторі

2.2.6 Алгоритмічне забезпечення процесу навчання моделі (Training Phase)

Фундаментом аналітичної підсистеми агрегатора є модуль прогнозування, побудований на базі алгоритму XGBoost (eXtreme Gradient Boosting). На відміну від детермінованих алгоритмів, які діють за жорстко заданими правилами, модель машинного навчання вимагає попереднього етапу тренування на історичних даних для виявлення прихованих закономірностей ринку. Процес навчання реалізовано як складний багатоетапний алгоритм, що перетворює "сирі" біржові котирування на зважені правила прийняття рішень (дерева рішень).

Загальна логіка функціонування цього модуля може бути представлена у вигляді діаграми потоків даних (DFD), яка відображає шлях інформації від джерела (бази даних агрегатора) через процеси очищення та трансформації до фінальної генерації моделі. Вхідними сутностями є історичні масиви даних, а вихідними — серіалізований файл моделі та метрики її якості.

На рисунку 2.6 зображено діаграму потоків даних (DFD), яка візуалізує архітектуру та логіку функціонування аналітичного модуля. Схема демонструє послідовність етапів обробки інформації: від отримання вхідних історичних масивів з джерела даних через процеси попередньої обробки (очищення, нормалізація) та інженерії ознак до навчання моделі XGBoost і генерації фінального прогнозу для користувача.

Алгоритм навчання розпочинається з етапу екстракції та попередньої обробки даних (Data Preprocessing). Система завантажує історичні записи (OHLCV) з бази даних за обраний період (наприклад, 2017–2024 роки). Першим кроком є перевірка цілісності часового ряду: алгоритм сканує масив на наявність пропущених значень (null або NaN), які можуть виникати через технічні збої на стороні біржі. У разі виявлення пропусків застосовується метод інтерполяції або заповнення середнім значенням сусідніх періодів, що дозволяє зберегти безперервність даних без внесення суттєвих викривлень. Після очищення відбувається сортування масиву за часовою міткою (timestamp) у висхідному порядку, що є критично важливим для коректного навчання на часових рядах, де майбутнє не повинно впливати на минуле.

Рисунок 2.6 – Діаграма потоків даних модуля прогнозування

З метою підвищення інформативності вхідних даних для алгоритму машинного навчання реалізовано етап інженерії ознак (Feature Engineering), що передбачає трансформацію «сирих» біржових показників у змістовні метрики. Зокрема, розраховано динаміку внутрішньоденного тренду (open-close) як різницю між ціною закриття та відкриття, що дозволяє ідентифікувати домінування покупців або продавців протягом торгової сесії. Додатково впроваджено показник волатильності (high-low), який відображає амплітуду коливань ціни та слугує індикатором нестабільності ринку.

Для врахування циклічної активності інституційних інвесторів, пов'язаної з фіксацією прибутків, інтегровано фактор сезонності через бінарний маркер

закінчення фінансового кварталу (is_quarter_end). На завершальному етапі підготовки даних виконано процедуру розмітки (Data Labeling) шляхом створення цільової змінної target. Задача прогнозування формалізована як бінарна класифікація, де зростання ціни закриття наступного періоду позначається класом «1», а зниження або стагнація — класом «0» .

Розділяючи дані на тренувальну (80%) та тестову (20%) вибірки, враховувався критичний нюанс — сувора заборона перемішування даних (shuffling). Оскільки у фінансах час рухається лише вперед, модель повинна вчитися хронологічно. Перемішування днів призвело б до ситуації, коли модель дізнається майбутнє раніше минулого, спричиняючи «витік даних» і фальшиво високі результати, недієздатні в реальності .

На рисунку 2.7 наведено схему алгоритму підготовки даних та навчання прогностичної моделі. Процес ініціюється завантаженням історичних масивів, після чого виконується перевірка цілісності та видалення пропущених значень. Подальші етапи включають приведення типів даних, інженерію ознак (розрахунок волатильності, трендів, сезонності) та розмітку цільової змінної. Завершується алгоритм розподілом вибірки, безпосереднім навчанням моделі XGBoost та збереженням її фінальної конфігурації для подальшого використання.

Процес навчання прогностичної моделі реалізовано на базі алгоритму XGBoost Classifier, вибір якого зумовлений його високою ефективністю при роботі з табличними даними фінансових ринків. В основі методу лежить принцип градієнтного бустінгу, який передбачає ітеративне формування ансамблю слабких моделей (дерев рішень). Система послідовно генерує серію простих дерев, де кожне наступне дерево математично спрямоване на мінімізацію функції втрат та корекцію помилок (залишків), допущених попередніми ітераціями ансамблю .

Для забезпечення стабільності та високої узагальнюючої здатності моделі в умовах стохастичного ринку проведено ретельне налаштування гіперпараметрів. Обмеження максимальної глибини дерев (max_depth) дозволяє контролювати складність моделі та запобігати перенавчанню на локальних ринкових шумах, тоді як параметр швидкості навчання (learning rate/eta) регулює внесок кожного окремого дерева в загальний прогноз, забезпечуючи плавну збіжність алгоритму до

оптимального розв'язку. Використання специфічної цільової функції (objective='multi:softprob') дозволяє трансформувати вихідні дані класифікатора у розподіл ймовірностей.

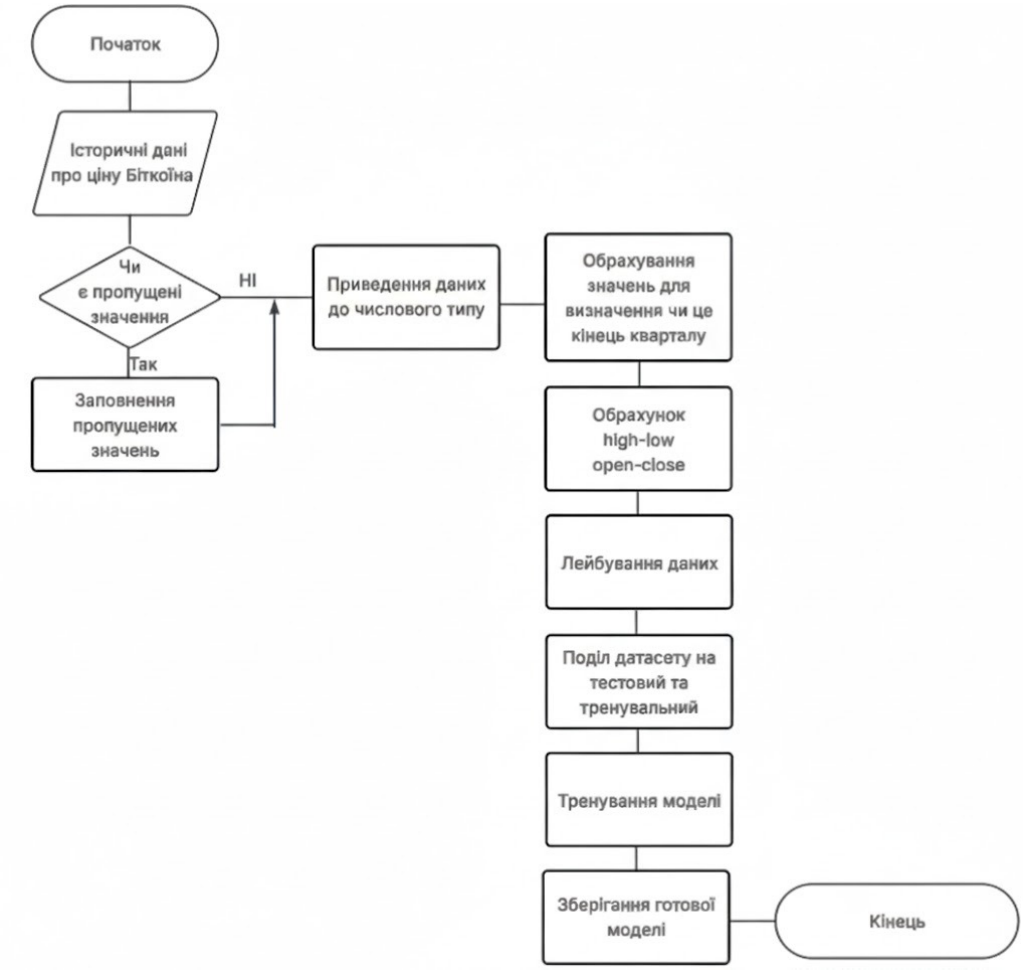


Рисунок 2.7 – Схема алгоритму підготовки даних та тренування моделі

Для запобігання перенавчанню моделі на історичних даних реалізовано ретельний підбір гіперпараметрів алгоритму XGBoost. Зокрема, встановлено обмеження максимальної глибини дерев (max_depth=7) для контролю складності моделі та визначено швидкість навчання (eta=0.2) для забезпечення плавної оптимізації. Використання цільової функції objective='multi:softprob' дозволило отримати на виході розподіл ймовірностей, що надає можливість оцінювати ступінь впевненості моделі у прогнозі та підвищує гнучкість прийняття рішень .

Завершальним етапом розробки алгоритму стали процедури валідації та серіалізації. Якість навчання перевіряється на тестовій вибірці за метрикою ROC-

AUC, і при досягненні точності понад 85% об'єкт моделі зберігається у файл (формат JSON/PKL). Це забезпечує високу швидкодію системи, дозволяючи використовувати попередньо навчену модель для миттєвого інференсу без витрат ресурсів на повторне навчання при кожному запиті.

Архітектура обробки користувацьких запитів побудована на принципах MVC із централізацією управління бізнес-процесами у контролерах. Клас TransactionController відповідає за ініціалізацію транзакцій через метод store, де реалізовано сувору валідацію вхідних даних за допомогою механізмів Laravel. Це гарантує фільтрацію некоректних запитів, перевірку наявності валют та відповідність лімітам сум перед створенням запису в базі даних та прив'язкою до ідентифікатора користувача .

Обробка запитів на конвертацію валют покладена на ExchangeController (метод getExchangeRate), який делегує складну логіку взаємодії із зовнішніми API сервісному шару (ExchangeService). Така структура дозволяє уникнути перевантаження контролерів («Fat Controllers») та забезпечує чистоту коду. Для оптимізації ресурсів сервіси зареєстровані як синглтони (singletons) у AppServiceProvider, що мінімізує витрати пам'яті .

Автоматизація наповнення бази даних реалізована через сідер CoinSeeder, який синхронізує список активів із зовнішнім API CoinMarketCap. Управління маршрутизацією здійснюється через файли web.php та api.php, а безпека доступу до функціоналу особистого кабінету забезпечується через Middleware auth, що блокує неавторизовані запити .

На рисунку 2.8 наведено UML-діаграму класів, що відображає об'єктно-орієнтовану архітектуру програмного модуля прогнозування. Схема включає чотири ключові сутності: HistoricalData для зберігання масивів історичних котирувань, Feature для опису атрибутів моделювання, Model, що інкапсулює методи навчання та інференсу алгоритму XGBoost, та Prediction для фіксації результатів. Візуалізовані асоціативні зв'язки ілюструють логіку взаємодії компонентів у процесі трансформації вхідних даних у кінцевий торговий сигнал.

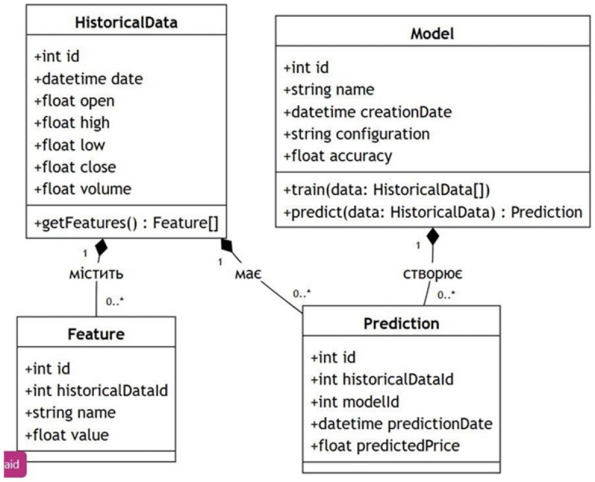


Рисунок 2.8 – Діаграма класів програмного модуля прогнозування

2.3 Інформаційне забезпечення агрегатора обміну криптовалют

Ефективність функціонування програмного комплексу визначається якістю організації інформаційного забезпечення, що включає сукупність структур даних, методів їх зберігання та обробки на рівні СУБД. Враховуючи вимогу підтримки транзакційності (ACID) та інтеграції з Eloquent ORM, в якості платформи обрано реляційну СУБД MySQL. Проектування схеми даних виконано з дотриманням принципів нормалізації до третьої нормальної форми (3НФ), що забезпечує мінімізацію надлишковості інформації .

Центральним елементом системи безпеки є сутність User, реалізована у таблиці users, яка зберігає ідентифікаційні дані та забезпечує механізми аутентифікації. Архітектурою передбачено зв'язок «один-до-багатьох» між користувачами та транзакціями, що дозволяє формувати історію фінансової активності. Ядром фінансової підсистеми виступає таблиця transactions, де фіксуються параметри обміну. Для гарантування точності розрахунків використано тип даних decimal з фіксованою розрядністю, що унеможливлює помилки округлення. Цілісність посилань забезпечується через зовнішні ключі (foreignId) з налаштуванням каскадних операцій.

Довідкова інформація про активи акумулюється в таблиці cryptocurrencies, яка містить метадані (назву, символ, логотип) та використовується для валідації вхідних даних при створенні заявок. Для прискорення агрегації курсів реалізовано

індексацію поля symbol. Інформаційна база модуля прогнозування представлена таблицею market_data (HistoricalData), що зберігає часові ряди у форматі OHLCV, необхідні для навчання алгоритму XGBoost. Результати інференсу фіксуються у таблиці predictions, що дозволяє проводити ретроспективний аналіз точності моделі. Управління схемою даних здійснюється через механізм міграцій Laravel, а автоматизація наповнення довідників реалізована за допомогою сідерів (CoinSeeder), які синхронізують інформацію із зовнішніми джерелами.

На рисунку 2.9 представлено ER-діаграму (Entity-Relationship diagram), що візуалізує логічну структуру бази даних системи. Схема відображає ключові сутності та реляційні зв'язки між ними, які забезпечують функціонування фінансової та аналітичної підсистем. Центральними елементами визначено таблицю users для зберігання облікових даних та таблицю transactions, що фіксує параметри обмінних операцій із забезпеченням цілісності даних через зовнішні ключі. Довідкова інформація про активи міститься у сутності cryptocurrencies, а для забезпечення роботи модуля прогнозування спроектовано таблиці market_data (для зберігання історичних часових рядів) та predictions (для фіксації результатів інференсу).

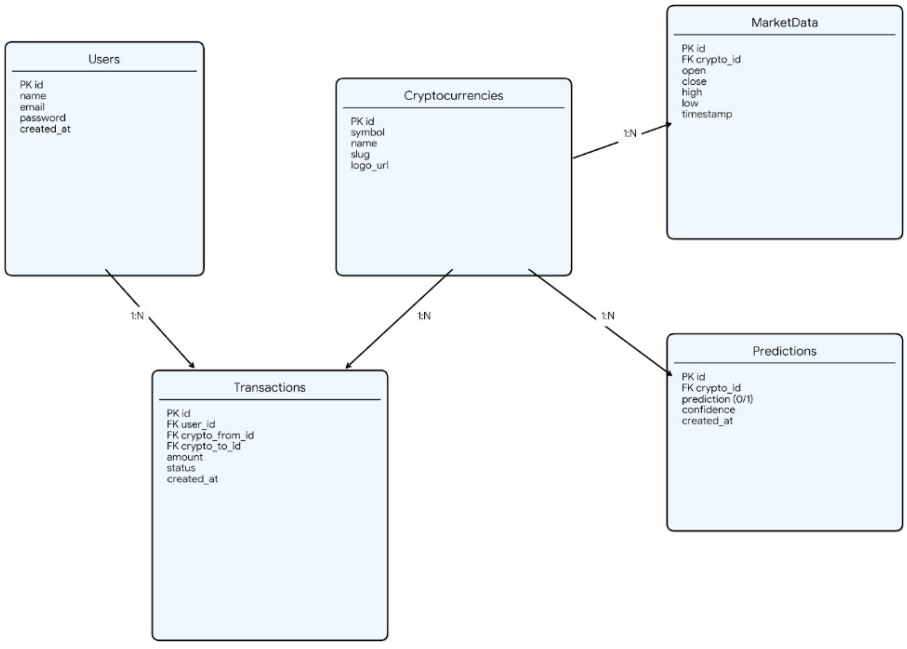


Рисунок 2.9 – ER-діаграма бази даних системи

Висновки до розділу 2

1. Спроектовано архітектуру гібридної програмної системи, яка поєднує веб-платформу на базі фреймворку Laravel (MVC) для агрегації курсів та автономний аналітичний модуль мовою Python для прогнозування ринкових трендів.

2. Використання алгоритму градієнтного бустінгу XGBoost дозволило системі ефективно вирішувати задачу бінарної класифікації напрямку руху ціни.

3. Розроблено структуру реляційної бази даних MySQL у третій нормальній формі, що забезпечує цілісність та надійність збереження даних. Визначено та описано ключові алгоритми системи, зокрема механізми асинхронної взаємодії із зовнішніми біржами, автоматичний вибір транзитних гаманців та верифікацію транзакцій, що гарантує безпеку та високу швидкість проведення фінансових операцій.

3 РЕАЛІЗАЦІЯ ТА ВЕРИФІКАЦІЯ РОЗРОБЛЕНОЇ МОДЕЛІ

3.1 Програмна реалізація архітектурної моделі

Архітектурним базисом серверної частини інформаційної системи обрано фреймворк Laravel, що зумовлено його розвиненою екосистемою та наявністю вбудованих інструментів маршрутизації, кешування і ORM Eloquent. Використання архітектури MVC забезпечує високу модульність коду та здатність системи до горизонтального масштабування, що є критичною умовою для стабільної обробки транзакцій в умовах високого навантаження на ринку криптовалют .

На етапі імплементації здійснено конфігурацію серверного оточення та налаштування захищеного з'єднання з реляційною СУБД. Відповідно до стандартів інформаційної безпеки реалізовано механізм ізоляції конфіденційних даних: ключі авторизації API, паролі доступу та криптографічні токени винесено у файл конфігурації середовища .env. Такий підхід унеможливлює витік чутливої інформації через системи контролю версій та забезпечує гнучкість управління налаштуваннями.

Програмна архітектура побудована на принципі розділення відповідальності (Separation of Concerns) в рамках патерну MVC. Компонент Model забезпечує абстракцію даних та взаємодію з базою даних, Controller виконує диспетчеризацію HTTP-запитів, а View відповідає за візуалізацію інтерфейсу. Глибока модульність системи оптимізує процеси розробки, спрощуючи налагодження (debugging), модульне тестування (unit testing) та подальший технічний супровід програмного продукту .

На рисунку 3.1 наведено схему функціонування архітектурного патерну MVC, яка візуалізує механізм розділення відповідальності між компонентами системи. Схема ілюструє послідовність обробки запитів, де Контролер виступає диспетчером, координуючи взаємодію між Моделлю (абстракція даних) та Представленням (візуалізація інтерфейсу), що забезпечує модульність та масштабованість програмного комплексу.

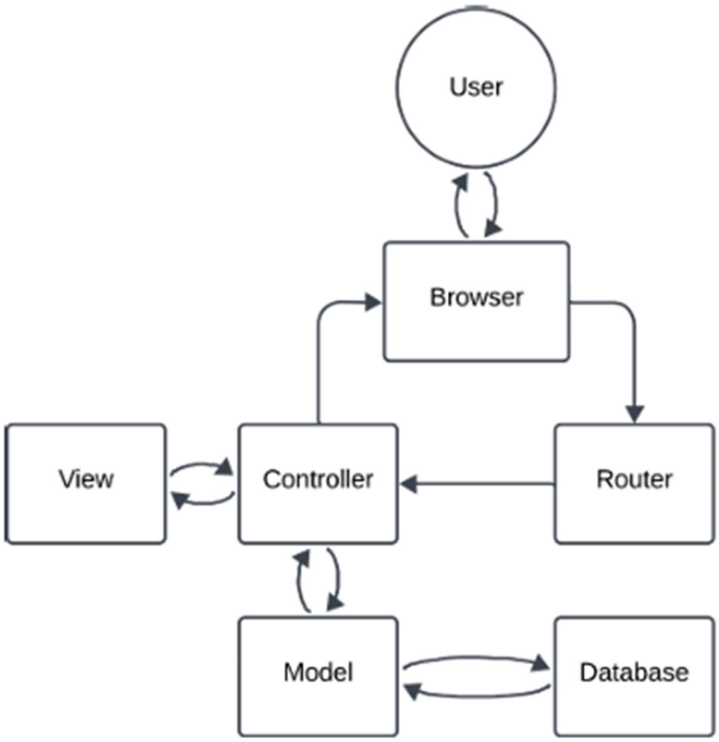


Рисунок 3.1 – Схема роботи архітектури MVC

Проектування архітектури даних реалізовано за допомогою вбудованого механізму міграцій Laravel, що дозволило описати схему бази даних програмним кодом на PHP та інтегрувати її в систему контролю версій. Такий підхід забезпечує повну синхронізацію конфігурацій між середовищами розробки та продуктивним сервером, а також значно спрощує процеси розгортання (deployment) та оновлення структури даних .

Архітектурним ядром підсистеми фінансового моніторингу визначено таблицю transactions, структура якої спроектована з урахуванням принципів реляційної цілісності через використання зовнішніх ключів для ідентифікації користувача та напрямку обміну. Для забезпечення математичної точності розрахунків при роботі з високоподільними криптоактивами імплементовано тип даних decimal із фіксованою точністю (16 знаків загалом, 8 у дробовій частині), що дозволяє нівелювати ризики похибок округлення, притаманних типам з плаваючою комою .

Інтеграція бізнес-логіки з базою даних здійснюється через Eloquent ORM за допомогою моделі Transaction. Політика безпеки реалізована шляхом жорсткої

конфігурації властивості $fillable, що захищає від вразливостей типу Mass Assignment, а логічна зв'язність даних забезпечується через методи реляційних відношень belongsTo, що оптимізує вибірку інформації .

Централізація логіки обробки заявок покладена на TransactionController. Метод store виконує функцію вхідної точки, реалізуючи механізм суворої валідації даних ($request->validate()) для фільтрації некоректних запитів та контролю лімітів ще до моменту персистентного збереження. Успішне проходження валідації ініціює створення запису транзакції з автоматичною прив'язкою до авторизованого користувача .

На рисунку 3.2 наведено діаграму взаємодії сервісного шару, що візуалізує архітектурний механізм інтеграції системи із зовнішніми джерелами даних та внутрішнім сховищем. Схема ілюструє послідовність потоків даних, де ExchangeController ініціює запит до бізнес-логіки, делегуючи виконання операцій сервісу ExchangeService. Останній координує роботу компонентів ExchangeRepository та APIClient для отримання, агрегації та уніфікації біржових котирувань перед їх поверненням у шар представлення.

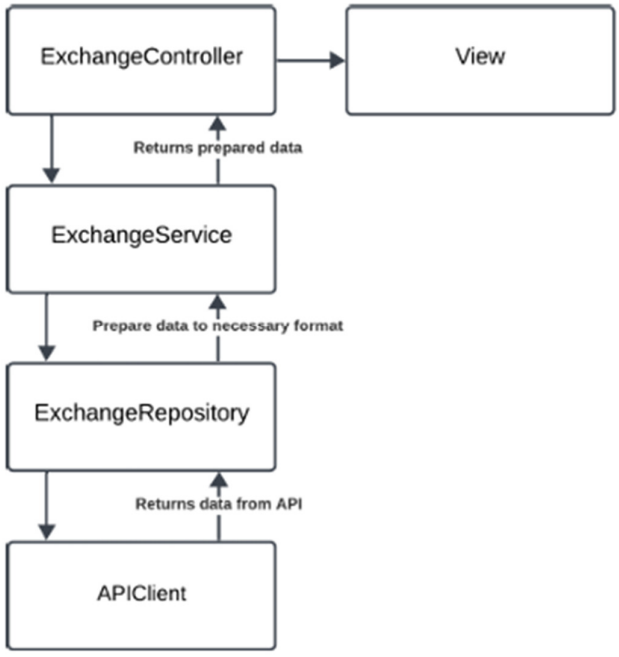


Рисунок 3.2 – Діаграма взаємодії сервісного шару з зовнішніми API та базою даних

Управління потоками ринкової інформації покладено на контролер ExchangeController, ключовий метод якого getExchangeRate реалізовано за принципом декомпозиції завдань. Для уникнення архітектурних ризиків, пов'язаних із перевантаженням контролерів, ресурсоємна логіка агрегації делегована сервісному шару. Центральним елементом інтеграції виступає клас ExchangeService, який через допоміжний компонент APIClient забезпечує уніфікацію гетерогенних даних від зовнішніх бірж, приводячи їх до єдиного стандарту перед передачею на рівень представлення.

З метою оптимізації ресурсів сервера в AppServiceProvider імплементовано патерн «Одинак» (Singleton), що забезпечує єдиноразову ініціалізацію сервісних об'єктів у межах життєвого циклу запиту. Автоматизація наповнення бази даних актуальними відомостями про цифрові активи реалізована через спеціалізований клас CoinSeeder, алгоритм якого виконує парсинг та синхронізацію даних із зовнішніх API (CoinMarketCap, Binance).

Система маршрутизації логічно сегментована на потоки веб-інтерфейсу (web.php) та програмного шлюзу (api.php), а використання іменованих маршрутів підвищує гнучкість супроводу коду. Фундаментальний рівень безпеки забезпечується механізмом проміжного програмного забезпечення (Middleware) auth, який блокує несанкціонований доступ до бізнес-критичних функцій та гарантує конфіденційність даних користувачів. Реалізована архітектура серверної частини формує захищений контур обробки даних, що інтегрує зовнішні шлюзи ліквідності з механізмами транзакційної цілісності.

3.2 Розробка та інтеграція модуля прогнозування на основі XGBoost

Вибір інструментарію для реалізації аналітичного ядра системи, відповідального за прогнозування ринкових трендів, зупинено на мові програмування Python завдяки її розвиненій екосистемі для наукових обчислень. Використання бібліотек pandas та numpy дозволило ефективно маніпулювати великими масивами біржової статистики, а scikit-learn та xgboost забезпечили доступ до оптимізованих алгоритмів машинного навчання, що відповідає сучасним

стандартам індустрії Data Science .

Архітектурно модуль спроектовано як автономний мікросервіс, що дозволило ізолювати ресурсоємні математичні обчислення від основного веб-сервера та гарантувати стабільність його роботи. Структура сервісу оптимізована для мінімізації часу відгуку (latency), що є критичним фактором для торгівлі в реальному часі. Реалізація розділена на два логічні блоки: скрипти для попередньої обробки даних (ETL) та навчання моделі, які запускаються періодично у фоновому режимі, та скрипти для інференсу, що працюють у режимі "on-demand" для миттєвої обробки поточних даних .

Критично важливим етапом розробки визначено інженерію ознак (Feature Engineering), оскільки сирі дані OHLCV містять значну кількість ринкового шуму. За допомогою векторизованих операцій бібліотеки pandas реалізовано трансформацію вхідних даних у змістовні метрики. Зокрема, розраховано показник динаміки тренду (open-close) для ідентифікації домінування покупців або продавців, а також метрику волатильності (high-low), зростання якої сигналізує про нестабільність ринку. Додатково впроваджено маркер сезонності (is_quarter_end) для врахування циклічної активності інституційних інвесторів .

Формування навчальної вибірки здійснено шляхом програмної розмітки даних (Data Labeling) для задачі бінарної класифікації, відмовившись від регресійного підходу через високу похибку на волатильних ринках. Цільова змінна *target* генерується шляхом порівняння поточної ціни закриття з майбутньою: значення «1» присвоюється при зростанні ціни, «0» — в інших випадках. Для покращення збіжності алгоритму реалізовано процедуру нормалізації даних за допомогою StandardScaler з бібліотеки scikit-learn, що приводить усі числові ознаки до єдиного стандартного діапазону .

На рисунку 3.3 представлено фрагмент програмної реалізації етапу інженерії ознак (Feature Engineering), який є ключовим для забезпечення високої предиктивної здатності розроблюваної моделі. Наведений лістинг коду демонструє процес генерації нових синтетичних змінних, що базуються на сирих ринкових даних: зокрема, здійснюється розрахунок внутрішньоденного тренду як різниці між цінами відкриття та закриття, а також кількісна оцінка поточної волатильності

активу через обчислення амплітуди коливань між мінімальною та максимальною ціною (Low-High).

```
# Розрахунок різниці між ціною відкриття та закриття (тренд всередині дня)
df_sorted['open-close'] = df_sorted['open'] - df_sorted['close']

# Розрахунок різниці між мінімальною та максимальною ціною (волатильність)
df_sorted['low-high'] = df_sorted['low'] - df_sorted['high']

# Формування цільової змінної (target):
# 1 - якщо ціна закриття наступного періоду вища за поточну, 0 - якщо нижча
df_sorted['target'] = np.where(df_sorted['close'].shift(-1) > df_sorted['close'],

# Вибір фінального набору ознак для навчання моделі
features = df_sorted[['open-close', 'low-high', 'is_quarter_end']]
target = df_sorted['target']
```

Рисунок 3.3 – Фрагмент програмного коду підготовки даних та формування ознак

Ядром мого аналітичного модуля є алгоритму XGBoost (eXtreme Gradient Boosting). Використовуючи клас XGBClassifier у створеному програмному коді було проведено серію експериментів з налаштуваннями, щоб знайти ідеальну конфігурацію саме для волатильного ринку криптовалют.

Особливу увагу було приділено налаштуванню гіперпараметрів, оскільки розумів, що від них залежить стабільність прогнозів. По-перше, було встановлено параметр швидкості навчання (eta) на рівні 0,2. Це було компромісом: таке значення дозволяє моделі навчатися достатньо швидко, але при цьому робить процес «градієнтного спуску» плавним, запобігаючи різким стрибкам ваг, які могли б призвести до помилок.

По-друге, щоб уникнути класичної проблеми перенавчання, коли алгоритм запам'ятовує випадковий шум замість реальних трендів, було обмежено максимальну глибину дерев рішень (max_depth) значенням 7. Також було зафіксовано кількість ітерацій бустінгу (n_estimators) на рівні 20, що виявилося достатнім для формування надійного ансамблю дерев без зайвих витрат обчислювальних ресурсів.

Найважливішим архітектурним рішенням у цьому блоці стало використання специфічної функції втрат — objective='multi:softprob'. Відмовлено від простої жорсткої класифікації на користь ймовірнісного підходу. Завдяки цьому налаштуванню запропонована система видає розподіл ймовірностей. Це дозволяє оцінити, наскільки саме модель впевнена у своєму прогнозі, що є критично важливим для управління ризиками в реальній торгівлі.

Завершивши цикл навчання, реалізовано механізм серіалізації отриманої моделі. Враховано, що тренування є ресурсоємним процесом, котрий не можна виконувати щоразу, коли користувач хоче побачити прогноз. Тому налаштовано систему так, щоб готовий об'єкт моделі зберігався у фізичний файл (у форматі JSON або PKL). Це дозволило чітко розмежувати два процеси: важке "навчання", яке відбувається у фоні, і швидке "використання", яке потрібне користувачеві тут і зараз.

Для безпосередньої генерації прогнозів створено окремий полегшений Inference-скрипт. Його головна перевага — швидкодія. Коли веб-додаток викликає цей скрипт, він не витрачає час на математичні обчислення для навчання, а просто миттєво завантажує готову "розумну" модель з файлу в оперативну пам'ять. Завдяки такій оптимізації досягнуто часу відгуку на рівні мілісекунд, що робить роботу з сайтом комфортною.

Логіку роботи цього скрипта організовано за принципом конвеєра. Він приймає поточні ринкові показники, проганяє їх через той самий ланцюжок трансформацій (нормалізацію), що й під час навчання, і подає на вхід класифікатора. Фінальний результат просте бінарне число (0 або 1) направляється у стандартний потік виводу (stdout). Це стратегічне рішення: такий підхід дозволяє бекенду на Laravel легко "перехопити" відповідь Python-скрипта і миттєво показати її користувачеві .

На рисунку 3.4 наведено фрагмент програмної реалізації, що демонструє процес конфігурації та запуску навчання класифікатора XGBoost. Код ілюструє етап налаштування ключових гіперпараметрів моделі, зокрема швидкості навчання (eta), максимальної глибини дерева та кількості естиматорів, а також виклик методу fit для тренування алгоритму на підготовленій вибірці.

```
from xgboost import XGBClassifier

# Визначення гіперпараметрів для налаштування моделі
# eta - швидкість навчання (learning rate)
# max_depth - максимальна глибина дерева для уникнення перенавчання
# n_estimators - кількість дерев рішень
param = {
    'eta': 0.2,
    'max_depth': 7,
    'num_class': 2,       # Бінарна класифікація (0 або 1)
    'n_estimators': 20,
    'objective': 'multi:softprob' # Функція цілі
}

# Ініціалізація моделі з обраними параметрами
model = XGBClassifier(**param)

# Запуск процесу навчання на тренувальній вибірці
# X_train - матриця ознак (features), Y_train - вектор цілей (targets)
model.fit(X_train, Y_train)
```

Рисунок 3.4 – Фрагмент коду ініціалізації та навчання моделі XGBoost з обраними гіперпараметрами

Для реалізації найбільш відповідального етапу — інтеграції аналітичного Python-скрипта з основним веб-додатком — використано потужний вбудований інструмент Symfony\Component\Process. Цей підхід обрано, тому що він дозволяє Laravel-додатку виконувати системні команди так, ніби вони вводяться у терміналі сервера, але в автоматичному режимі.

У контролері PredictionController написано спеціальний метод, який діє як диспетчер. Його завдання — динамічно сформувати командний рядок. Туди передається шлях до інтерпретатора Python, повний шлях до скрипта прогнозування, а також найважливіше — вхідні аргументи, тобто масив актуальних цін активу на дану секунду.

Далі налаштовано механізм перехоплення. Коли Python-скрипт завершує роботу, він «друкує» результат. PHP-код миттєво зчитує цей вивід і починає його парсинг. Реалізовано просту, але надійну логіку інтерпретації: якщо отримується від скрипта одиницю (1), система виводить користувачеві рекомендацію «Купувати» (RISE), підсвічуючи її зеленим кольором. Якщо ж повертається нуль

(0), це як сигнал до небезпеки або стагнації — «Утриматися» або «Продати» (FALL).

Найбільшою перевагою такого архітектурного рішення, є надійна ізоляція процесів. Систему спроектована так, що навіть якщо у модулі машинного навчання станеться критична помилка (наприклад, через нестачу пам'яті або збій бібліотеки), це жодним чином не «покладе» основний веб-сервер. Код на Laravel просто перехопить цю виключну ситуацію і коректно повідомить користувача про тимчасову недоступність прогнозу, зберігши при цьому повну працездатність усіх інших функцій сайту .

На рисунку 3.5 наведено фрагмент коду, що реалізує процедуру отримання прогнозу для нових вхідних даних (інференс моделі). Лістинг демонструє виклик методу передбачення на основі нормалізованого вектора ознак, а також містить логічний блок інтерпретації результатів, який трансформує бінарний вихід класифікатора («0» або «1») у зрозуміле для користувача текстове повідомлення про очікуваний напрямок руху ціни активу.

```
# Отримання прогнозу для нових вхідних даних
# Вхідний вектор: [open-close, low-high, is_quarter_end] (нормалізовані значення)
# predict() повертає клас: 0 або 1
test_preds = models[0].predict([[-1.89727, -2.13708, 0]])

# Інтерпретація результату для передачі на веб-інтерфейс
if test_preds == 0:
    # Клас 0: Прогнозується падіння ціни
    print("Bitcoin price will fall")
else:
    # Клас 1: Прогнозується зростання ціни
    print("The price of bitcoin will increase")
```

Рисунок 3.5 - Фрагмент коду отримання прогнозу та інтерпретації результату класифікації

3.3 Реалізація інтерфейсної частини моделі

Процес реалізації клієнтської частини (Frontend) базується на принципах людино-орієнтованого дизайну, спрямованого на зниження когнітивного

навантаження користувача в умовах роботи з волатильним фінансовим ринком. Візуальна архітектура забезпечує прозорість операційних процесів та миттєвий відгук системи, надаючи інструменти для інтуїтивної інтерпретації результатів аналітичного моделювання .

Стратегічним вибором технологічного стеку визначено гібридний підхід, що поєднує надійність серверного рендерингу (SSR) та інтерактивність односторінкових додатків (SPA). Технічною основою виступає шаблонізатор Blade, що дозволяє реалізувати компонентно-орієнтовану структуру (віджети, картки активів) та забезпечити високий рівень повторного використання коду. Стилізація інтерфейсу виконана за допомогою CSS-фреймворку Tailwind, який базується на методології «utility-first», гарантуючи адаптивність верстки для різних типів пристроїв .

Ключовим архітектурним рішенням є відмова від класичного SPA на користь технології Livewire, що реалізує концепцію «динамічного фронтенду» без створення окремого API-шару. Механізм асинхронних запитів дозволяє оновлювати стан компонентів шляхом інтеграції згенерованих сервером HTML-фрагментів у DOM-дерево, забезпечуючи плавність інтерфейсу та мінімізацію затримок при відображенні потокових ринкових даних .

Функціональним ядром інтерфейсу виступає інформаційна панель (Dashboard) з інтерактивною таблицею метрик (ціна, капіталізація). Навігацію оптимізовано за допомогою алгоритму реактивного пошуку («Live Search»), який виконує серверну фільтрацію масивів даних у реальному часі. Ергономіка сприйняття інформації покращена завдяки імплементації темної теми ("Dark Mode"), що є індустріальним стандартом для професійних торгових терміналів, забезпечуючи високий контраст індикаторів тренду .

На рисунку 3.6 представлено головний інтерфейс користувача (Front-end) розробленого сервісу «NOVA change». Сторінка реалізована у вигляді інтерактивного дашборда, що відображає актуальний список доступних для обміну криптовалютних активів (Bitcoin, Ethereum, USDT тощо) з їхніми поточними ринковими цінами та капіталізацією. Цей екран слугує точкою входу в систему, надаючи користувачеві можливість оперативного моніторингу ситуації на ринку

перед ініціюванням транзакцій.

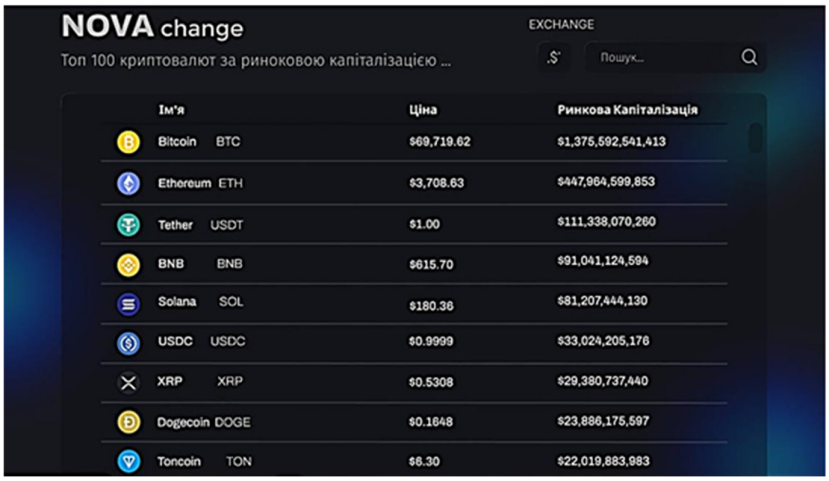


Рисунок 3.6 – Головна сторінка агрегатора з переліком доступних активів

Функціональним ядром інтерфейсу виступає форма ініціалізації обміну, візуальна концепція якої базується на принципах цифрового мінімалізму для фокусування уваги на ключових параметрах транзакції. Технічна реалізація компонента ґрунтується на механізмі двостороннього зв'язування даних (Two-way Data Binding) бібліотеки Livewire, що забезпечує синхронізацію полів вводу з серверною логікою в режимі реального часу без використання складних JavaScript-обробників.

Алгоритм роботи калькулятора побудовано за дзеркальним принципом, що забезпечує миттєву конвертацію валют. При введенні суми відправлення («Send») система ініціює фоновий запит до сервісного шару для отримання актуального курсу та автоматично розраховує суму отримання («Receive»), і навпаки. Такий підхід гарантує прозорість ціноутворення та надає користувачеві миттєвий зворотний зв'язок.

Окремий архітектурний модуль відповідає за вибір транспортної мережі (ERC-20, BEP-20, TRC-20) та динамічний моніторинг комісійних зборів (Gas Fees). Система в реальному часі завантажує дані про стан мемпулу та відображає точну вартість транзакції для кожної мережі, надаючи інвестору інструментарій для оптимізації витрат та вибору балансу між швидкістю та вартістю переказу .

На рисунку 3.7 відображено інтерфейс форми створення заявки на обмін, що є основним інструментом взаємодії користувача з системою для здійснення

транзакцій. Екранна форма містить поля для введення суми та вибору валютної пари, а також секції для вказівки реквізитів гаманця отримувача та контактного e-mail, що забезпечує коректність та безпеку проведення фінансової операції.

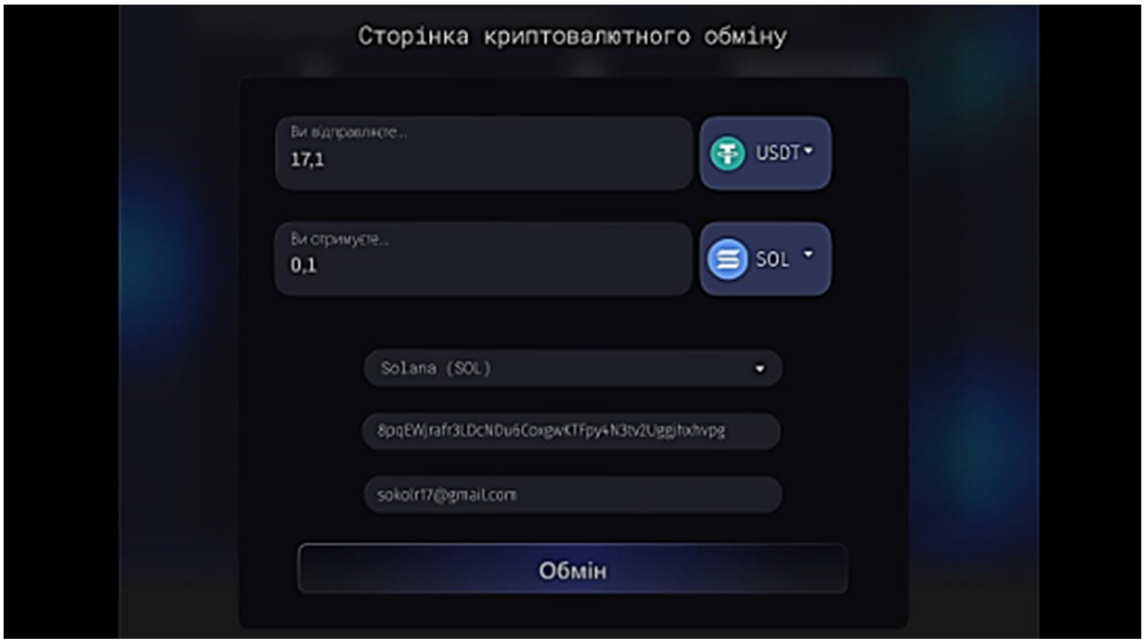


Рисунок 3.7 – Інтерфейс форми створення заявки на обмін

Завершальним етапом користувацького сценарію визначено сторінку виконання фінансової операції, архітектура якої спрямована на мінімізацію помилок та забезпечення прозорості процесу. Центральним елементом інтерфейсу виступає динамічний таймер зворотного відліку, реалізований засобами JavaScript та синхронізований із серверною сесією для візуалізації часового вікна (Time-to-Live), протягом якого гарантується фіксація узгодженого курсу. Для підвищення ергономіки впроваджено функціонал миттєвого копіювання реквізитів (Clipboard API) та інтегровано віджет-агрегатор, що демонструє альтернативні котирування від партнерських бірж для підтвердження конкурентоспроможності пропозиції .

Технічна реалізація моніторингу статусу транзакції базується на механізмі періодичного опитування сервера (Polling), що забезпечує автоматичну актуалізацію інтерфейсу в реальному часі без необхідності перезавантаження сторінки після підтвердження операції в блокчейні. Унікальною особливістю системи є інтеграція результатів предиктивної аналітики через спеціалізований

динамічний віджет. Логіка візуалізації передбачає трансформацію бінарних виходів моделі у інтуїтивні кольорові індикатори: позитивний прогноз (клас «1») відображається зеленим сигналом «Bullish», а негативний (клас «0») — червоним сигналом «Bearish», що надає користувачеві ефективний інструмент підтримки прийняття рішень.

На рисунку 3.8 зображено інтерфейс сторінки очікування оплати, ключовим елементом якого є таймер зворотного відліку, що обмежує час дії зафіксованого курсу для захисту від волатильності. Окрім реквізитів гаманця для здійснення транзакції, екран також містить блок із посиланнями на популярні криптобіржі, пропонуючи користувачеві альтернативні шляхи обміну.

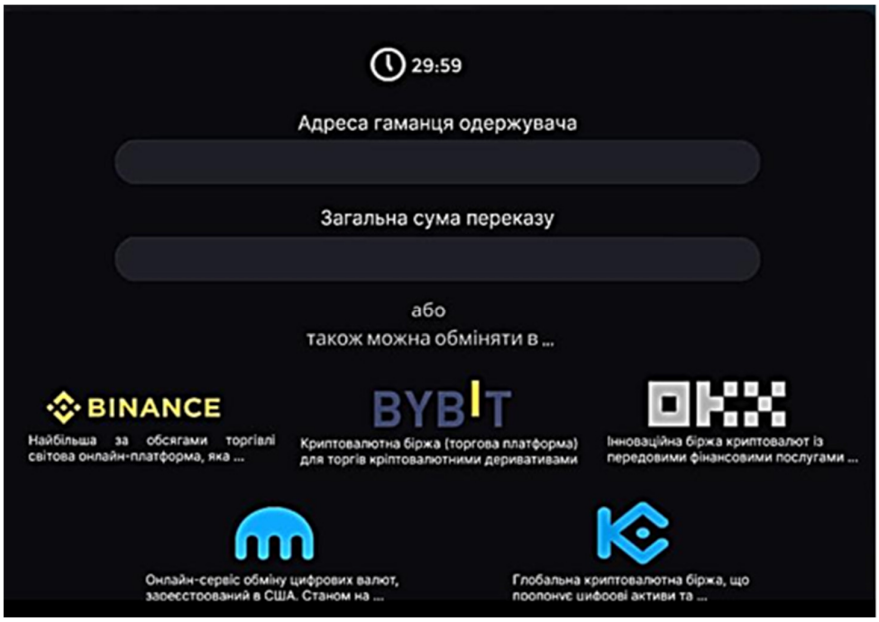


Рисунок 3.8 - Сторінка очікування оплати з таймером та реквізитами

Конкурентною перевагою системи визначено глибоку інтеграцію результатів предиктивної аналітики в інтерфейс користувача. Для семантичної інтерпретації бінарних виходів алгоритму XGBoost розроблено механізм трансформації числових даних у візуальні метафори, технічно реалізований через ізольований Blade-компонент. Логіка візуалізації базується на загальноприйнятих фінансових патернах: прогнозування висхідного тренду (клас 1) відображається зеленим індикатором «Bullish», а ризику спаду (клас 0) — червоним індикатором «Bearish» . Такий підхід виконує ергономічну функцію абстрагування від математичної

складності обчислень, надаючи інвестору синтезований висновок, що суттєво знижує поріг входження та спрощує прийняття рішень в умовах динамічного ринку.

3.4 Тестування та оцінювання ефективності моделі прогнозування

Завершальним та, безперечно, критично важливим етапом життєвого циклу розробки програмного продукту (SDLC) є фаза комплексного тестування та валідації. Головна мета цього процесу полягає не лише у виявленні програмних помилок, а й у верифікації повної відповідності реалізованого функціонала специфікаціям технічного завдання. Особливу увагу при цьому приділено валідації прогностичних здібностей інтелектуального модуля, оскільки від точності його роботи залежить економічна ефективність системи для кінцевого користувача.

Враховуючи гетерогенну природу архітектури, яка інтегрує класичні веб-сервіси на базі Laravel та високотехнологічні алгоритми машинного навчання на Python, стратегія забезпечення якості (Quality Assurance) була структурована за двома паралельними векторами. Перший вектор — це функціональне тестування бізнес-логіки агрегатора, спрямоване на перевірку коректності обробки транзакцій, взаємодії з API та роботи інтерфейсу. Другий вектор — це спеціалізоване статистичне оцінювання якості класифікації ринкових трендів, що передбачає аналіз метрик точності ML-моделі на історичних та реальних даних. Такий дуалістичний підхід дозволяє гарантувати надійність системи як з інженерної, так і з аналітичної точки зору.

3.4.1 Методологія та результати функціонального тестування веб-агрегатора

Для комплексної верифікації надійності архітектури застосовано методологію тестування «чорної скриньки» та сценарне моделювання наскрізних процесів (End-to-End), що дозволило відтворити повний життєвий цикл взаємодії користувача з платформою. У ході перевірки підсистеми навігації експериментально підтверджено коректність функціонування механізму динамічного пошуку на базі технології Livewire, який забезпечує фільтрацію

масивів даних у реальному часі, а також успішно верифіковано адаптивність інтерфейсу для різних типів пристроїв.

Пріоритетну увагу приділено валідації бізнес-логіки ініціалізації обміну. За допомогою серії негативних тест-кейсів підтверджено здатність контролера TransactionController ефективно блокувати некоректні запити, включаючи спроби введення невалідних значень або перевищення лімітів ліквідності. Функціональне тестування сервісного шару ExchangeService засвідчило високу точність алгоритмів розрахунку крос-курсів та коректність динамічного відображення мережевих комісій залежно від завантаженості блокчейну.

Кульмінаційним етапом випробувань стала верифікація цілісності життєвого циклу транзакції, включаючи перевірку механізму часових обмежень (Time-to-Live). Застосування методу імітаційного моделювання вебхуків дозволило підтвердити надійність серверної частини при обробці зовнішніх сигналів про надходження коштів. Система продемонструвала коректну автоматичну зміну статусів операцій та генерацію звітності, що свідчить про її здатність забезпечувати повний цикл обслуговування фінансових операцій в автономному режимі.

3.4.2 Статистична оцінка ефективності моделі прогнозування

Процедура верифікації прогностичної спроможності аналітичного модуля, реалізованого на базі алгоритму градієнтного бустінгу XGBoost, базувалася на суворих принципах статистичного аналізу. Для забезпечення об'єктивності експерименту було застосовано стратегію розділення даних (Hold-out method), згідно з якою 20% від загального масиву історичних котирувань було виділено у незалежну тестову вибірку. Цей сегмент даних був повністю ізольований від процесу навчання моделі, що дозволило імітувати роботу алгоритму в реальних умовах на нових, раніше невідомих йому даних. Такий підхід є критично важливим для коректної оцінки узагальнюючої здатності моделі (Generalization Ability) та виявлення ознак перенавчання.

В якості ключового індикатора ефективності для вирішення задачі бінарної класифікації ринкових трендів (прогнозування зростання або падіння активу) було обрано метрику ROC-AUC (Area Under the Receiver Operating Characteristic Curve).

Вибір цього показника, на противагу традиційній метриці точності (Accuracy), обумовлений його вищою інформативністю при роботі з імовірнісними моделями. ROC-AUC оцінює здатність класифікатора ранжувати об'єкти за ступенем впевненості та залишається стійкою метрикою навіть в умовах дисбалансу класів, коли кількість прикладів одного типу (наприклад, "ріст") суттєво перевищує кількість інших, що є типовим явищем для фінансових часових рядів .

За результатами проведених експериментальних досліджень, розроблена модель класифікації продемонструвала високі показники ефективності, що підтверджує валідність обраного підходу. Зокрема, точність прогнозування на навчальній вибірці (Training Accuracy) досягла рівня 0.8603, тоді як аналогічний показник на незалежній валідаційній вибірці (Validation Accuracy) склав 0.8595. Критично важливим є той факт, що дельта між цими значеннями становить менше 0.1%, що є надійним індикатором відсутності ефекту перенавчання (overfitting) — типової проблеми при роботі з фінансовими даними, коли алгоритм запам'ятовує шум замість виявлення закономірностей.

Така стабільність результатів свідчить про оптимальний підбір гіперпараметрів регуляризації на етапі налаштування моделі, зокрема обмеження глибини дерев рішень (max_depth=7) та встановлення консервативної швидкості навчання (eta=0.2). Досягнутий рівень точності, що перевищує поріг у 86%, є значним досягненням для стохастичних та високоентропійних криптовалютних ринків. Це емпірично підтверджує початкову робочу гіпотезу про те, що синтезовані інженерні ознаки — показники волатильності та імпульсу тренду — володіють високою дискримінативною здатністю та прогностичною цінністю для моделювання курсової динаміки .

З метою проведення глибинного діагностичного аналізу прогностичної здатності класифікатора та виявлення природи помилок, було застосовано інструмент матриці невідповідностей (Confusion Matrix). Цей метод візуалізації результатів тестування дозволив детально структурувати вихідні дані моделі, розклавши їх на чотири базові категорії: істинно-позитивні (True Positive) та істинно-негативні (True Negative) спрацьовування, а також помилки першого (False Positive) та другого (False Negative) роду. Така деталізація надала можливість

оцінити не лише загальну точність, але й специфічність та чутливість алгоритму до різних ринкових ситуацій.

Аналіз матриці підтверджує високу здатність моделі ідентифікувати ринкові тенденції (концентрація значень на головній діагоналі) при допустимому рівні похибки. Це свідчить про надійність системи як інструменту підтримки прийняття рішень, який, з огляду на стохастичність ринку, найефективніший у комплексі з фундаментальним аналізом.

На рисунку 3.9 представлено матрицю невідповідностей (Confusion Matrix), яка візуалізує результати класифікації моделі на тестовій вибірці. Діаграма відображає розподіл істинно-позитивних, істинно-негативних, хибно-позитивних та хибно-негативних прогнозів, що дозволяє детально оцінити точність алгоритму XGBoost у задачах ідентифікації ринкових трендів та виявити співвідношення коректних передбачень до помилкових .

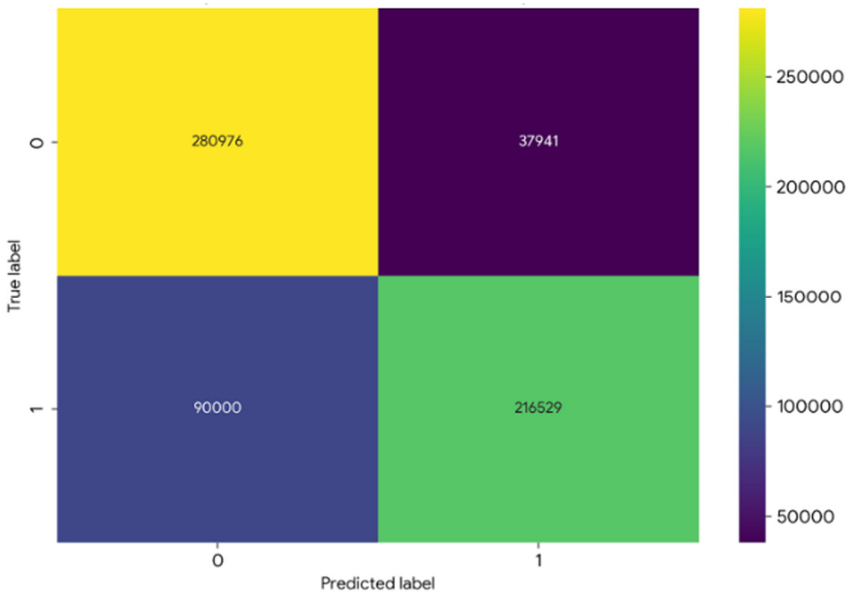


Рисунок 3.9 – Матриця невідповідностей (Confusion Matrix) роботи моделі на тестовій вибірці

3.4.3 Натурний експеримент та верифікація на реальних даних

З метою фінальної верифікації експлуатаційних характеристик системи та підтвердження її прикладної цінності, окрім стандартних синтетичних тестів на відкладеній вибірці, було реалізовано етап емпіричного дослідження в умовах,

максимально наближених до реальної ринкової кон'юнктури. Як об'єкт для кейс-стаді було обрано специфічний часовий інтервал (9 травня), який характеризувався високим рівнем ентропії та невизначеністю подальшого вектора руху ціни, що робило задачу прогнозування нетривіальною для класичних методів технічного аналізу.

У ході експерименту на вхід аналітичного модуля було подано сформований вектор нормалізованих ознак, який математично описував ринкову ситуацію в заданий момент часу, включаючи показники волатильності та внутрішньоденного імпульсу. Алгоритм машинного навчання, виконавши процедуру інференсу (обробки нових даних навченою моделлю), згенерував вихідне значення класу 0. Цей технічний результат був успішно перехоплений підсистемою інтерпретації та трансформований у чіткий семантичний сигнал для користувача — «Bitcoin price will fall» (Прогнозується падіння ціни біткоїна), що свідчить про ідентифікацію моделлю ведмежого тренду на основі прихованих патернів у вхідних даних .

Для підтвердження валідності отриманого сигналу було проведено ретроспективний аналіз ринкової динаміки у поствибірковий період. Порівняння прогнозних даних із фактичним рухом ціни активу засвідчило високу точність роботи алгоритму: ринок дійсно увійшов у фазу глибокої корекції, що супроводжувалася зниженням курсової вартості активу. Цей емпіричний кейс слугує переконливим доказом того, що розроблена система здатна ефективно ідентифікувати локальні точки біфуркації (розвороту) тренду ще до моменту їх настання.

З практичної точки зору, здатність генерувати настільки точні випереджаючі торгові сигнали перетворює систему на потужний інструмент ризик-менеджменту, який дозволяє користувачеві вчасно виходити з позицій та мінімізувати потенційні фінансові втрати. Технічна сторона експерименту також підтвердила ефективність архітектурних рішень: інтеграція результатів роботи ML-модуля у веб-інтерфейс через механізм системних викликів була реалізована без затримок (zero-latency), що забезпечило кінцевому користувачеві миттєвий доступ до критично важливої аналітичної інформації в режимі реального часу .

Висновки до розділу 3

1. Здійснено повний цикл програмної імплементації та всебічної верифікації спроектованої інформаційної системи. Ключовим досягненням стала побудова високопродуктивного серверного ядра на базі фреймворку Laravel, яке успішно виконує функції оркестрації потоків даних, забезпечуючи надійну агрегацію фінансових котирувань із зовнішніх джерел та безпечний процесинг транзакцій користувачів.

2. Розроблено та інтегровано в загальну архітектуру автономний аналітичний модуль на мовою Python. Використання алгоритму градієнтного бустінгу XGBoost дозволило реалізувати ефективний механізм класифікації ринкових трендів, що суттєво розширює функціональні можливості платформи, перетворюючи її з простого інструменту обміну на аналітичну систему.

3. Для забезпечення якісної взаємодії з кінцевим користувачем було створено адаптивний клієнтський інтерфейс із використанням технологій серверного рендерингу Blade та реактивних компонентів Livewire, що дозволило досягти високих показників ергономіки та швидкодії без ускладнення архітектури фронтенду.

4. Проведена серія комплексних випробувань, що включала функціональне тестування бізнес-логіки та статистичну оцінку якості машинної моделі, підтвердила повну відповідність розробленого продукту технічним вимогам та його стабільність під навантаженням. Зокрема, досягнення показника ROC-AUC на рівні 0.86 свідчить про високу прогностичну здатність системи, що дає вагомі підстави рекомендувати розроблений агрегатор для впровадження у практичну діяльність як ефективний інструмент підтримки інвестиційних рішень в умовах невизначеності криптовалютного ринку.

ВИСНОВКИ

1. У кваліфікаційній роботі запропоновано модель агрегатора обміну криптовалют з можливістю прогнозування їх курсу та реалізовано її у веб-орієнтованій системи для агрегації курсів криптовалют.

2. Здійснено комплексний системний аналіз поточного стану ринку віртуальних активів, за результатами якого ідентифіковано фундаментальні проблеми індустрії, що ускладнюють прийняття інвестиційних рішень. Критичний огляд існуючих програмних рішень (CoinMarketCap, WalletInvestor, BestChange) виявив функціональну прогалину на ринку: відсутність єдиного гібридного інструменту, який би поєднував механізми миттєвої агрегації ліквідності з предиктивною аналітикою. Більшість наявних платформ обмежуються дескриптивним аналізом, фіксуючи події постфактум, що в умовах динамічного ринку створює «прогностичний вакуум» і підвищує фінансові ризики для користувачів.

3. Обґрунтовано методологічні засади та спроектовано архітектуру програмного комплексу, покликаного вирішити виявлені проблеми. Розроблено концепцію гібридної системи, що базується на синергетичній взаємодії надійної серверної частини на фреймворку Laravel та високопродуктивного аналітичного модуля мовою Python.

4. Застосовано алгоритм градієнтного бустінгу XGBoost для вирішення задачі бінарної класифікації ринкових трендів. Реалізовано етап інженерії ознак, що дозволило трансформувати сирі ринкові дані у змістовні метрики волатильності та сезонності.

5. Розроблено надійну схему обробки транзакцій, яка включає алгоритми асинхронного опитування API для мінімізації затримок, механізми часових обмежень для фіксації курсу та сувору валідацію вхідних даних, що гарантує цілісність та безпеку фінансових операцій.

6. Реалізовано високопродуктивний веб-сервіс із адаптивним клієнтським інтерфейсом, побудованим на технологіях Blade та Livewire, що забезпечує

миттєвий візуальний відгук та знижує когнітивне навантаження на користувача завдяки інтеграції кольорових індикаторів прогнозу.

7. Проведено тестування аналітичного модуля, що підтвердило високу ефективність обраної конфігурації гіперпараметрів XGBoost. Точність прогнозування на незалежній тестовій вибірці за метрикою ROC-AUC досягла рівня 86%, при цьому різниця між результатами на тренувальній та валідаційній вибірках є мінімальною, що свідчить про відсутність перенавчання. Аналіз результатів тестування на реальних даних в умовах високої ринкової ентропії (кейс 9 травня) показав здатність системи генерувати коректні випереджаючі сигнали про зміну тренду, що дозволяє рекомендувати розроблений агрегатор як ефективний інструмент підтримки прийняття рішень, що мінімізує ризики в умовах невизначеності криптовалютного ринку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bitcoin.org [Електронний ресурс] – URL: https://bitcoin.org
2. Ethereum.org [Електронний ресурс] – URL: https://ethereum.org
3. Coingecko 2024 Q1 Crypto Industry Report [Електронний ресурс] – URL: https://www.coingecko.com/research/publications/2024-q1-crypto-report
4. Top Crypto Predictions of 2024 [Електронний ресурс] – URL: https://coinmarketcap.com/academy/article/top-crypto-predictions-of-2024
5. Binance Academy. Що таке криптовалюта. – URL: https://academy.binance.com/uk/articles/what-is-a-cryptocurrency
6. Cryptocurrency Explained With Pros and Cons for Investment [Електронний ресурс] – URL: https://www.investopedia.com/terms/c/cryptocurrency.asp
7. Blog.whitebit.com [Електронний ресурс] – URL: https://blog.whitebit.com/uk/what-is-a-cryptocurrency
8. Brasse A., Hyun S. Cryptocurrency Exchanges and the Future of Cryptoassets // The Emerald Handbook on Cryptoassets: Investment Opportunities and Challenges / Baker H.K. et al. (Ed.). – Leeds: Emerald Publishing Limited, 2023. – P. 341-353.
9. Централізована біржа (CEX) [Електронний ресурс] – URL: https://coinmarketcap.com/academy/uk/glossary/centralized-exchange-cex
10. Binance Academy. Що таке децентралізована біржа (DEX) – URL: https://academy.binance.com/uk/articles/what-is-a-decentralized-exchange-dex
11. Gandal N., Halaburda H. Competition in the Cryptocurrency Market // CEPR Discussion Paper. – 2014. – No. DP10157.
12. Liu J., Serletis A. Volatility in the Cryptocurrency Market // Open Econ Rev. – 2019. – Vol. 30. – P. 779–811.
13. Minfin.com Як працює обмін криптовалют і що про нього треба знати [Електронний ресурс] – URL: https://minfin.com.ua/ua/crypto/articles/kak-rabotaet-obmen-kriptovalyut-i-chto-o-nem-nuzhno-znat/
14. Learn.bybit Агрегатор децентралізованої біржі (DEX) [Електронний ресурс] – URL: https://learn.bybit.com/uk/defi/what-is-odos-dex-aggregator
15. Bou Abdo J., Zeadally S. Multi-utility framework: blockchain exchange

platform for sustainable development // International Journal of Pervasive Computing and Communications. – 2022. – Vol. 18 No. 4. – P. 388-406.

16. Основи роботи DEX-агрегаторів [Електронний ресурс] – URL: https://www.web3.org.ua/shcho-take-dex-ahrehator

17. 7 Best Crypto DEX Aggregator [Електронний ресурс] – URL: https://coincodecap.com/best-crypto-dex-aggregator

18. The future of cryptocurrency: Expert insights and predictions for 2024 [Електронний ресурс] – URL: https://blog.bitpanda.com/en/future-cryptocurrency-expert-insights-and-predictions-2024

19. Swapzone [Електронний ресурс] – URL: https://swapzone.io

20. Changelly [Електронний ресурс] – URL: https://changelly.com

21. Simpleswap [Електронний ресурс] – URL: https://simpleswap.io

22. Laravel.com [Електронний ресурс] – URL: https://laravel.com

23. Machine Learning based Bitcoin Price Prediction [Електронний ресурс]. – URL: https://ieeexplore.ieee.org/document/10810810

24. Laravel Queues Documentation [Електронний ресурс]. – URL: https://laravel.com/docs/queues

25. Livewire Documentation [Електронний ресурс]. – URL: https://laravel-livewire.com/docs

26. Tailwind CSS Documentation [Електронний ресурс]. – URL: https://tailwindcss.com/docs

27. JWT Security Guide: Best Practices & Implementation [Електронний ресурс]. – URL: https://guptadeepak.com/understanding-jwt-from-basics-to-advanced-security/

28. Financial Markets Effect on Cryptocurrency Volatility: Pre- and Post-Future Exchanges Collapse Period [Електронний ресурс]. – URL: https://www.mdpi.com/2227-7072/13/1/24

29. Comparing Technical and Fundamental Analysis in Cryptocurrency Trading Strategies [Електронний ресурс]. – URL: https://www.researchgate.net/publication/395756709

30. Соколовський Р.І. Методи та інструменти покращення користувацького

досвіду в криптовалютних агрегаторах III Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «інтелектуальні комп'ютерні системи та мережі» С.143-144.

31. Соколовський Р.І. Актуальність проектування модуля прогнозування курсу криптовалют «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення» (випуск 105)

32. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

Додаток А

Копії публікацій

Соколовський Р. І.
магістрант другого року ФКІТ ЗУНУ
Науковий керівник к.т.н., доцент Биковий П.Є., кафедра ІОСУ ЗУНУ

МЕТОДИ ТА ІНСТРУМЕНТИ ПОКРАЩЕННЯ КОРИСТУВАЦЬКОГО ДОСВІДУ В КРИПТОВАЛЮТНИХ АГРЕГАТОРАХ

Вступ. У сучасних умовах ринок криптовалют стрімко розширюється, привертаючи увагу як інвесторів, так і пересічних користувачів. Разом із цим зростає потреба у вдосконаленні користувацького досвіду, щоб забезпечити зручне та ефективне використання платформ. Люди очікують простих, інтуїтивно зрозумілих інтерфейсів, які дозволяють швидко отримувати необхідну інформацію та доступ до ключових функцій для прийняття рішень. З огляду на посилення конкуренції між біржами та криптовалютними агрегаторами, саме якість UX стає важливим фактором залучення й утримання аудиторії. Платформи, що пропонують більш зрозумілий та комфортний інтерфейс, отримують відчутну перевагу. Оптимізація користувацького досвіду охоплює спрощення дизайну, зниження когнітивного навантаження та підвищення загальної зручності взаємодії з продуктом. Тому дослідження інструментів і підходів до покращення UX у криптоагрегаторах є актуальним, адже воно сприяє підвищенню конкурентоспроможності, зміцненню безпеки та довіри користувачів, а також стимулює ширше впровадження криптовалют.

Постановка задачі. У відкритому доступі представлено чимало сервісів із достатнім набором можливостей, однак значна частина з них поступається через складний або незрозумілий інтерфейс. Серед популярних агрегаторів обміну варто розглянути Binance [1], ByBit [2] та OKX [3]. Детальний аналіз цих платформ дозволяє виділити низку спільних переваг і недоліків. До основних переваг можна віднести широкий вибір криптовалют і торгових пар, розгалужений інструментарій для торгівлі, високий рівень захисту активів користувачів, різноманітні методи введення та виведення коштів, а також значні обсяги торгів і високий рівень ліквідності. Водночас ці сервіси мають і недоліки: складні та перевантажені інтерфейси, що можуть бути незручними для новачків, обмежений доступ для користувачів із певних регіонів, а також відгуки, у яких згадуються труднощі зі службою підтримки [4-7]. Кожен із цих криптовалютних агрегаторів пропонує власні унікальні можливості та функції, тому під час вибору варто враховувати індивідуальні потреби, рівень підготовки, торгові підходи та значущість тих чи інших функцій для конкретного користувача.

Основний матеріал. Провівши детальний аналіз існуючих агрегаторів і знайшовши їх основні недоліки висунув певні висновки. Під час створення агрегатора обміну криптовалют потрібно виділити певний час та акцент на створення зручного та інтуїтивного зрозумілого користувацького інтерфейсу. Завдяки оптимізації певної частини користувацької системи, з якою користувачі взаємодіють безпосередньо через веб-інтерфейс, можна оптимізувати певні необхідні функції для здійснення обміну криптовалют, такі як вибір валютної пари, введення обсягу обміну, перегляд поточного курсу обміну, підтвердження та виконання операцій, перегляд історії операцій. Laravel забезпечує зручними інструментами для ефективного і швидкого виконання необхідних операцій такими як: views - дозволяють створювати швидкодіючі сторінки за допомогою шаблонізатора blade, tailwind - css фреймворк, який допомагає максимально ефективно писати стилі в аргументах html, livewire - добавляє динаміку в монолітні застосунки. (дає можливість виконувати php код в браузері). Це дуже зручний інструмент, коли немає необхідності робити повністю динамічний SPA або PWA застосунок але є потреба для певних сторінок. Це допоможе створити гармонійний та концептуальний користувацький інтерфейс. Судячи з аналізу запропонованих користувацьких інтерфейсів потрібно дотримуватися мінімалістичного та стриманого дизайну з мінімальною кількістю зайвих елементів, які можуть відволікати користувача. Інтерфейс повинен бути простим у використанні, всі функції повинні бути доступні та легкі.

Висновки. Інтуїтивно зрозумілий інтерфейс користувача є важливим елементом у забезпеченні приємної та ефективної взаємодії з платформою. Простота використання та легкий доступ до основних функцій дозволяють як новачкам, так і досвідченим трейдерам швидко зорієнтуватися на сайті. Крім того, постійне вдосконалення та оновлення платформи на основі зворотного зв'язку користувачів є важливим елементом успішної оптимізації користувацького досвіду. Постійне покращення функціональності, інтеграція нових функцій та виправлення помилок сприяють збереженню інтересу користувачів та покращенню їхнього задоволення від використання платформи. Нарешті, комбінація цих методів та інструментів може виявитися найбільш ефективною у досягненні мети оптимізації користувацького досвіду в агрегаторі обміну криптовалют. Ретельне збалансування між інтуїтивністю і персоналізацією, постійним вдосконаленням та оновленнями дозволить створити платформу, яка буде відповідати потребам різних категорій користувачів та забезпечувати їм найвищий рівень задоволення від взаємодії з нею.

Список літератури:

1 Binance URL: https://www.binance.com/uk-UA

2 Біржа криптовалют Bybit https://www.bybit.com/uk-UA

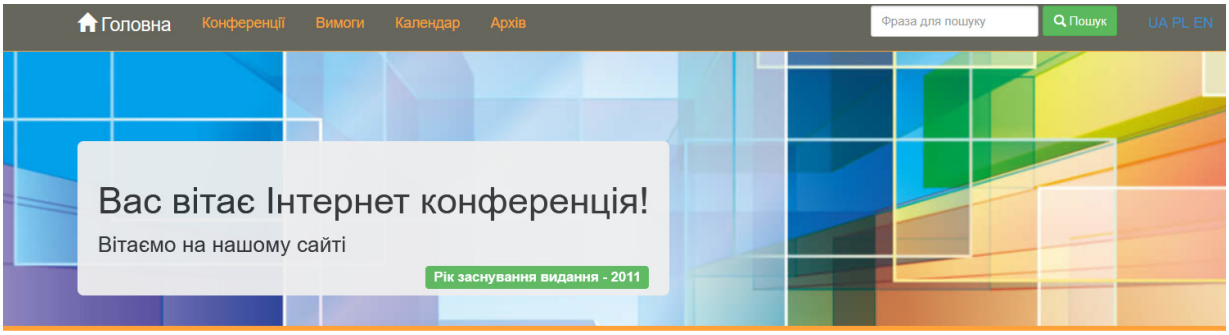
3 Купівля біткойна і криптовалюти URL:https://www.okx.com/ua

4 Порівняння криптобіржі Binance та Bybit URL: https://shorturl.at/s8Eah

5 Bybit (Байбіт) – Огляд біржі, можлівості та додаткові сервіси, переваги та недоліки. URL:https://www.crypt0.fan/ua/bybit-review

6 Bybit vs Binance: Порівняння криптобірж | Кращий вибір для трейдерів URL:https://mixfin.com/ua/blog/bybit-vs-binance

7 Binance URL: https://uk.wikipedia.org/wiki/Binance



АКТУАЛЬНІСТЬ ПРОЕКТУВАННЯ МОДУЛЯ ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ

10.12.2025 21:26

[1. Інформаційні системи і технології]

Автор: **Соколовський Ростислав Ігорович, магістрант другого року, Західноукраїнський національний університет, м.Тернопіль**

У сучасних умовах розвитку цифрової економіки криптовалюти стають важливим об'єктом дослідження як для фінансових аналітиків, так і для наукової спільноти. Стрімке зростання ринку цифрових активів та властива їм висока волатильність суттєво впливають на глобальні фінансові процеси, формуючи нові підходи до інвестування та управління ризиками. У зв'язку з цим зростає потреба у створенні надійних методів прогнозування їхньої вартості, які дозволяють здійснювати обґрунтований аналіз і приймати зважені інвестиційні рішення [1]. Особливе значення має прогнозування курсу біткоїна та інших провідних криптовалют, оскільки їхня динаміка визначає загальний стан ринку [2]. Проте процес передбачення їхньої ціни є складним через нестабільність, нерегулярність та залежність від великої кількості зовнішніх факторів. Це робить завдання прогнозування актуальним науковим напрямом, що потребує застосування сучасних математичних, статистичних та комп'ютерних методів.

Актуальність проблеми прогнозування курсу криптовалют сьогодні є беззаперечною. Сфера фінансів, інвестицій та цифрових технологій зазнає значних трансформацій під впливом криптовалют, які стали не лише альтернативним інструментом інвестування, а й постійним об'єктом уваги у глобальних фінансових медіа. Навіть незначні події здатні спричинити суттєві коливання їхньої вартості. Так, після зламу біржі Bitfinex у Гонконзі ціна біткоїна різко знизилася приблизно на 20%, що демонструє чутливість ринку до зовнішніх інформаційних факторів. Тому аналіз впливу новин, подій та різноманітних економічних умов на ціну криптовалют є критично важливим як для інвесторів, так і для дослідників.

Біткоїн — перша та найбільш відома криптовалюта — за останнє десятиліття здобула значну популярність. Його вартість неодноразово демонструвала стрімке зростання, супроводжуване різкими спадом і коливаннями, що робить актив об'єктом постійних досліджень. На ціну біткоїна впливає комплекс чинників, які умовно поділяють на дві групи:

1. Фактори попиту і пропозиції. Популярність біткоїна, його використання як інвестиційного активу чи засобу платежу, а також обмежена емісія у 21 мільйон монет формують дефіцит і стимулюють зростання попиту. Додатково впливають зростання складності майнінгу та поступове зменшення темпів створення нових монет.

2. Спекулятивні та інформаційні фактори. Позитивні новини — наприклад, прийняття криптовалюти в нових юрисдикціях або технологічні вдосконалення — часто спричиняють зростання ціни. Негативні події, як-от збій роботи біржі чи хакерська атака, можуть викликати падіння. Важливу роль також відіграють ринкові настрої та очікування інвесторів [3,4].

Зростання вартості біткоїна створює як можливості, так і ризики для економіки. До позитивних аспектів можна віднести посилення децентралізації фінансової системи, підвищення доступності та швидкості платежів, появу нового класу активів для інвестування та стимулювання інновацій у фінансових технологіях. Серед основних ризиків виділяють високу волатильність, можливість використання криптовалют у незаконних операціях та потенційне послаблення впливу традиційних фінансових інститутів.

Процес прогнозування вартості криптовалют є складним завданням, що вимагає врахування ринкової динаміки, технічних індикаторів та економічних чинників. Жоден метод не забезпечує абсолютної точності, однак підходи, засновані на технічному аналізі, машинному навчанні та фундаментальній оцінці, дозволяють значно підвищити обґрунтованість інвестиційних рішень.

Методи прогнозування, у тому числі щодо біткоїна, мають велике значення для фінансового сектора через високу нестабільність цих активів. Розроблення програмного модуля для прогнозування вартості біткоїна є важливим інструментом для інвесторів, медіа, аналітиків та дослідників. Такий модуль сприяє зниженню ризиків, підвищенню ефективності інвестиційних стратегій і глибшому аналізу ринку. У перспективі застосування подібних систем підтримує інтеграцію криптовалют у традиційну фінансову інфраструктуру та підвищує їхню надійність як інвестиційного активу.

Висновки:

Прогнозування курсу криптовалют є актуальною проблемою через високу мінливість ринку та його чутливість до зовнішніх факторів. Біткоїн, як провідна криптовалюта, зазнає значних коливань під впливом новин та економічних подій, що створює для інвесторів як можливості, так і суттєві ризики. Динаміка активу визначається попитом, складністю майнінгу та інформаційними тригерами, що вимагає застосування спеціалізованих методів аналізу. Попри переваги децентралізації, криптовалюти несуть загрози волатильності, тому розробка програмного модуля для прогнозування ціни є важливим кроком. Такі системи дозволяють зменшити ризики, підвищити ефективність інвестицій та сприяють інтеграції цифрових активів у традиційну фінансову систему, забезпечуючи стабільніший розвиток ринку.

Список літератури:

[1] https://shorturl.at/usHmm

[2] https://shorturl.at/6A3dM

[3] https://shorturl.at/6TZ4x

[4] https://shorturl.at/rSUyf

Науковий керівник: Биковий Павло Євгенович, кандидат технічних наук, доцент, Західноукраїнський національний університет

Додаток Б

Лістинг модулів системи

User.php

```php
<?php

namespace App\Models;

use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;

class User extends Authenticatable
{
    use HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast to native types.
     *
     * @var array
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];

    /**
```

```php
     * Hash the user's password before saving.
     *
     * @param  string  $password
     * @return void
     */
    public function setPasswordAttribute($password)
    {
        $this->attributes['password'] = bcrypt($password);
    }

    /**
     * Get the transactions for the user.
     */
    public function transactions()
    {
        return $this->hasMany(Transaction::class);
    }
```

Transaction.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Transaction extends Model
{
    use HasFactory;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'user_id',
        'cryptocurrency_from',
        'cryptocurrency_to',
        'amount',
        'rate',
        'status',
    ];

    /**
     * Get the user that owns the transaction.
     */
    public function user()
    {
        return $this->belongsTo(User::class);
    }

    /**
     * Get the cryptocurrency being exchanged from.
     */
    public function cryptocurrencyFrom()
    {
        return $this->belongsTo(Cryptocurrency::class, 'cryptocurrency_from');
    }

    /**
     * Get the cryptocurrency being exchanged to.
     */
    public function cryptocurrencyTo()
    {
        return $this->belongsTo(Cryptocurrency::class, 'cryptocurrency_to');
    }
```

Coin.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Coin extends Model
{
    use HasFactory;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name', 'symbol', 'slug', 'rank',
    ];
```

```php
    /**
     * Get the transactions where this coin is used as the source cryptocurrency.
     */
    public function transactionsFrom()
    {
        return $this->hasMany(Transaction::class, 'cryptocurrency_from');
    }

    /**
     * Get the transactions where this coin is used as the destination cryptocurrency.
     */
    public function transactionsTo()
    {
        return $this->hasMany(Transaction::class, 'cryptocurrency_to');
    }
```

create_users_table.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
```

create_transactions_table.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateTransactionsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
```

```
    public function up()
    {
        Schema::create('transactions', function (Blueprint $table) {
            $table->id();
            $table->foreignId('user_id')->constrained()->onDelete('cascade');
            $table->foreignId('cryptocurrency_from')->constrained('cryptocurrencies')->onD
            $table->foreignId('cryptocurrency_to')->constrained('cryptocurrencies')->onDe
            $table->decimal('amount', 16, 8);
            $table->decimal('rate', 16, 8);
            $table->string('status');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('transactions');
    }
}
```

Лістинг основного модуля

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

# --- Імпорти для Машинного Навчання ---

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import metrics

# /**
#  * Завантаження даних.
#  * #  * Зчитує набір даних з CSV файлу.
#  * @var pd.DataFrame
#  */
df_sorted = pd.read_csv('bitcoin.csv')
```

```
# /**
#  * Обробка колонки 'date'.
#  * #  * Розділяє дату на окремі колонки: рік, місяць та день,
#  * і перетворює їх на цілі числа.
#  */
splitted = df_sorted['date'].str.split('-', expand=True)

df_sorted['year'] = splitted[0].astype('int')
df_sorted['month'] = splitted[1].astype('int')
df_sorted['day'] = splitted[2].astype('int')

# /**
#  * Перетворення числових колонок.
#  * #  * Гарантує, що цінові дані є числовими (float).
#  */
```

```
df_sorted['open'] = pd.to_numeric(df_sorted['open'])
df_sorted['high'] = pd.to_numeric(df_sorted['high'])
df_sorted['low'] = pd.to_numeric(df_sorted['low'])
df_sorted['close'] = pd.to_numeric(df_sorted['close'])

# /**
#  * Створення Інженерних Ознак (Feature Engineering).
#  * #  * Розраховує 'open-close', 'low-high' та цільову змінну 'target'.
#  * 'target' = 1, якщо ціна закриття наступного дня вища, 0 інакше.
#  */
df_sorted['open-close'] = df_sorted['open'] - df_sorted['close']
df_sorted['low-high'] = df_sorted['low'] - df_sorted['high']
```

```
# Використовуємо .shift(-1) для прогнозування наступного дня
df_sorted['target'] = np.where(df_sorted['close'].shift(-1) > df_sorted['close']

# --- Аналіз та Візуалізація ---

# /**
#  * Вибір числових типів даних.
#  * #  * Вибирає лише ті колонки, які мають типи 'float64' або 'int64'
#  * для розрахунку кореляції.
#  * @var pd.DataFrame
#  */
numerical_df = df_sorted.select_dtypes(include=['float64', 'int64'])

# /**
#  * Матриця Кореляції.
#  * #  * Розраховує кореляцію та візуалізує ознаки з кореляцією > 0.9.
#  */
corr_matrix = numerical_df.corr()

plt.figure(figsize=(10, 10))
sb.heatmap(corr_matrix[corr_matrix >= 0.9], annot=True, cbar=False)
plt.show()

# --- Підготовка до Моделювання ---
```

```
# Припущено, що ці імпорти були зроблені раніше:
# from sklearn.model_selection import train_test_split
# from sklearn.metrics import metrics
# from xgboost import XGBClassifier

# --- Розділення Даних та Підготовка Моделі ---

# /**
#  * Розділення Набору Даних.
#  * Розділяє масштабовані ознаки (features) та цільову змінну (target)
#  * на тренувальний (X_train, y_train) та валідаційний (X_valid, y_valid) набори.
#  * @param float test_size: Частка даних для валідації (0.2).
#  * @param int random_state: Для відтворюваності результатів.
#  */
X_train, X_valid, y_train, y_valid = train_test_split(
    features,
    target,
    test_size=0.2,
    random_state=2022
)

# /**
#  * Вивід Розмірів Наборів.
#  * Друк розмірів (форм) тренувального та валідаційного наборів ознак.
#  */
print(X_train.shape, X_valid.shape)
```

```
# /**
#  * Визначення Ознак та Цільової Змінної.
#  * #  * Вибирає необхідні ознаки для навчання моделі.
#  */
features = df_sorted[['open-close', 'low-high', 'is_quarter_end']]
target = df_sorted['target']
```

```
# /**
#  * Стандартизація Ознак.
#  * #  * Ініціалізує та застосовує StandardScaler для масштабування ознак.
#  */
scaler = StandardScaler()
features = scaler.fit_transform(features)
```

```
# Припущено, що ці імпорти були зроблені раніше:
# from sklearn.model_selection import train_test_split
# from sklearn.metrics import metrics
# from xgboost import XGBClassifier

# --- Розділення Даних та Підготовка Моделі ---

# /**
#  * Розділення Набору Даних.
#  * Розділяє масштабовані ознаки (features) та цільову змінну (target)
#  * на тренувальний (X_train, y_train) та валідаційний (X_valid, y_valid) набори.
#  * @param float test_size: Частка даних для валідації (0.2).
#  * @param int random_state: Для відтворюваності результатів.
#  */
X_train, X_valid, y_train, y_valid = train_test_split(
    features,
    target,
    test_size=0.2,
    random_state=2022
)

# /**
#  * Вивід Розмірів Наборів.
#  * Друк розмірів (форм) тренувального та валідаційного наборів ознак.
#  */
print(X_train.shape, X_valid.shape)
```

```
# --- Налаштування XGBoost та Навчання ---

# /**
#  * Параметри Моделі.
#  * Визначення словника параметрів для класифікатора XGBoost.
#  */
param = {
    'eta': 0.2,
    'max_depth': 7,
    'num_class': 2,
    'n_estimators': 20
}

# /**
#  * Ініціалізація та Навчання Моделі.
#  * Припускається, що 'models' було ініціалізовано як XGBClassifier раніше
#  * (наприклад, models = XGBClassifier(**param)).
#  */
# Тут можна використати: models = XGBClassifier(**param)
models.fit(X_train, y_train)

# --- Оцінка Ефективності ---

# /**
#  * Оцінка Моделі (ROC AUC Score).
#  * Розраховує та виводить показник ROC AUC для тренувального та валідаційного наборі
#  */
print(f'models :')
```

```
print(
    'Training Accuracy : ',
    metrics.roc_auc_score(y_train, models.predict_proba(X_train)[:, 1])
)

print(
    'Validation Accuracy : ',
    metrics.roc_auc_score(y_valid, models.predict_proba(X_valid)[:, 1])
)

print()
```