

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Застосування принципів AutoML для аналізу методів
прогнозування курсу криптовалют»**

Виконав студент групи ІПЗм-22-1
СІТАЙЛО Максим Сергійович

Керівник роботи:
МОРОЗОВ Андрій Васильович

Рецензент:
ЧИЖМОТРЯ Олексій Володимирович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
 ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
 ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
 КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії програмного
забезпечення

Тетяна ВАКАЛЮК

«18» вересня 2023 р.

ЗАВДАННЯ на кваліфікаційну роботу

Здобувач вищої освіти: СІТАЙЛО Максим Сергійович

Керівник роботи: МОРОЗОВ Андрій Васильович

Тема роботи: «Застосування принципів AutoML для аналізу методів прогнозування курсу криптовалют»,

затверджена Наказом закладу вищої освіти від «18» вересня 2023 р. № 479/с

Вихідні дані для роботи: **Предметом дослідження є принципи автоматизованого машинного навчання та методи прогнозування курсу криптовалют. Об'єктом дослідження є розробка системи для аналізу та порівняння методів прогнозування курсу криптовалют з застосуванням підходів автоматизованого машинного навчання.**

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Морозов А.В.	22.09.2022р.	22.09.2022р.
2	Морозов А.В.	22.09.2022р.	22.09.2022р.
3	Морозов А.В.	22.09.2022р.	22.09.2022р.

РЕФЕРАТ

Магістерська кваліфікаційна робота включає в себе наукове дослідження методів прогнозування курсу криптовалют й принципів автоматизованого машинного навчання, а також розробку системи для аналізу методів прогнозування. Робота складається з програмного комплексу системи аналізу методів прогнозування курсу криптовалют та пояснювальної записки.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел із 40 найменувань, 3 таблиць та 53 ілюстрацій по тексту. Повний обсяг роботи становить 100 сторінок, з них 80 основного тексту.

В першому розділі було проведено аналіз процесів та методів прогнозування курсу криптовалют.

У другому розділі було проведено моделювання розроблюваної системи.

Третій розділ містить опис реалізації та тестування створеного програмного комплексу.

У висновках наведено підсумки проведеного дослідження та отриманих результатів.

КЛЮЧОВІ СЛОВА: АВТОМАТИЗОВАНЕ МАШИННЕ НАВЧАННЯ, МЕТОД ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ, КОНВЕЄР, СИСТЕМА АНАЛІЗУ, AUTOML.

ABSTRACT

The Master's qualification work involves scientific research of cryptocurrency price forecasting methods and principles of automated machine learning, as well as the development of a system for analyzing forecasting methods. The work consists of a software package of a system for analyzing cryptocurrency rate forecasting methods and an explanatory note.

The qualification work consists of an introduction, three chapters, conclusions, a reference list with 40 sources cited, 3 tables, and 53 figures. The total volume of the work is 100 pages, with 80 pages comprising the main text.

In the first chapter, an analysis of processes and methods for cryptocurrency price forecasting was conducted.

In the second section, the modeling of the developed system was performed.

The third section contains a description of the implementation and testing of the created software application.

The conclusions contain a summary of the conducted research and the obtained results.

KEYWORDS: AUTOMATED MACHINE LEARNING, CRYPTOCURRENCY PRICE FORECASTING METHODS, PIPELINE, ANALYSIS SYSTEM, AUTOML.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ПРОЦЕСІВ ТА МЕТОДІВ ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ	10
1.1. Аналіз процесу прогнозування вартості криптоактивів	10
1.2 Аналіз існуючих досліджень методів прогнозування котирувань криптовалют	12
1.3. Дослідження алгоритмів, які застосовуються для вирішення задач прогнозування ...	15
1.4. Аналіз можливості застосування AutoML для порівняльного аналізу алгоритмів	20
1.5 Постановка завдання	23
Висновки до першого розділу	26
РОЗДІЛ 2. МОДЕЛЮВАННЯ СИСТЕМИ АНАЛІЗУ МЕТОДІВ ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ	27
2.1 Проектування системи з застосуванням особливостей конвеєру AutoML	27
2.2 Аналіз та розробка ETL процесу даних системи.....	33
2.3. Проектування архітектури системи.....	40
2.4. Розробка архітектури аналітичного компонента системи.....	44
Висновки до другого розділу	47
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ АНАЛІЗУ МЕТОДІВ ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ	49
3.1 Вибір та обґрунтування засобів реалізації	49
3.2 Проектування та розробка компонентів системи.....	52
3.3 Тестування розробленої системи	56
Висновки до третього розділу	74
ВИСНОВКИ.....	75
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	77

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АМН – автоматизоване машинне навчання

БД – База даних

API – Application Programming Interface

AutoML – Automated Machine Learning

ETL – Extract, Transform, Load

OLAP – Online Analytical Processing

OLTP – Online Transaction Processing

LSTM – Long-Short Term Memory

ВСТУП

Актуальність теми. Прогнозування вартості активів завжди було популярною й актуальною проблемою, що сприяє постійному дослідженню існуючих методів й інструментів, а також пошуку варіантів їх оптимізації та удосконалення. Можливість використання ЕОМ та поява й розвиток штучного інтелекту мали значний вплив на сферу досліджень у даній галузі. Це спричинило виникнення нового покоління алгоритмів та методів для прогнозування курсу активів завдяки швидкодії згаданих засобів й кращій точності нейронних мереж порівняно зі звичайними алгоритмами. Значно вища ефективність даних інструментів мотивувала науковців до дослідження процесу використання машинного навчання, що, як результат, сприяло широкому дослідженню даного підходу й появі нових варіацій та модифікацій існуючих алгоритмів. Проте, наявність великої кількості досліджень з одного боку дозволяє проаналізувати різноманітні засоби та пропозиції для вирішення певного практичного завдання, а з іншого боку відсутність їх систематизації вимагає самостійного ознайомлення з результатами досліджень й експериментальної перевірки їх ефективності.

Актуальним також є дослідження можливості використання автоматизованого машинного навчання (AutoML) для аналізу методів прогнозування курсу криптовалют. На момент проведення дослідження AutoML здебільшого використовується з метою спрощення процесу використання машинного навчання для вирішення певних практичних задач. Проте, автоматизоване машинне навчання (АМН) також може дозволити автоматизувати процес порівняння результатів різноманітних досліджень й значно спростить та пришвидшить їх систематизацію.

Метою роботи є проектування та розробка програмного забезпечення для аналізу та порівняння між собою методів прогнозування курсу криптовалют з використанням принципів АМН. Для досягнення поставленої мети необхідно

вирішити такі основні **завдання**:

- аналіз існуючих методів прогнозування курсу криптовалют та формування вимог до системи з метою забезпечення аналізу їх ефективності та можливості порівняння між собою;
- дослідження особливостей процесів АМН та конвеєру AutoML;
- проектування архітектури системи з врахуванням особливостей обраного виду машинного навчання;
- розробка програмного забезпечення з впровадженням принципів AutoML, а також створення аналітичного модулю для порівняльного аналізу досліджуваних методів.

Об'єктом дослідження є розробка системи для аналізу та порівняння методів прогнозування курсу криптовалют з застосуванням підходів автоматизованого машинного навчання.

Предметом дослідження є принципи автоматизованого машинного навчання та методи прогнозування курсу криптовалют.

Методи дослідження. У роботі було використано методи системного та кореляційно-регресійного аналізу, моделювання потоків даних та процесів їх обробки, а також методи об'єктно-орієнтованого проектування та програмування.

Наукова новизна отриманих результатів полягає у тому, що:

- спроектовано підхід для систематизації досліджень, які стосуються сфери прогнозування курсу криптовалют з використанням штучного інтелекту;
- досліджено можливість застосування автоматизованого машинного навчання для вирішення задачі аналізу й порівняння різних методів прогнозування котирувань криптоактивів.

Практичне значення полягає у тому, що:

- розроблено систему, яка дозволяє автоматизувати процес створення моделі прогнозування від етапу збору даних до отримання результатів;

- спроектовано та розроблено вебдодаток, який дозволяє реалізовувати різноманітні алгоритми та порівнювати їх ефективність, що спрощує процес дослідження наукових ідей;
- реалізовано різні методи системного аналізу, що дозволяє ефективно аналізувати та порівнювати ефективність досліджуваних алгоритмів.

Апробація результатів роботи. Матеріали дослідження за тематикою кваліфікаційної роботи:

1. Sitailo M. S., Morozov A. V. Research of the cryptocurrency price prediction approaches / M. Sitailo, A. Morozov. // Тези доповідей V Всеукраїнської науково-технічної конференції: “Комп’ютерні технології: інновації, проблеми, рішення” 01-02 грудня – С. 211–212.

2. Sitailo M. S., Morozov A. V. Popular machine learning algorithm’s types for cryptocurrencies price prediction / M. Sitailo, A. Morozov. // Тези доповідей VI Всеукраїнської науково-технічної конференції: “Комп’ютерні технології: інновації, проблеми, рішення” 29-30 листопада.

РОЗДІЛ 1. АНАЛІЗ ПРОЦЕСІВ ТА МЕТОДІВ ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ

1.1. Аналіз процесу прогнозування вартості криптоактивів

Прогнозування вартості активів є дуже популярним видом завдань пов'язаних з аналізом даних. Існує велика кількість підходів та математичних методів, які досліджуються й застосовуються з метою визначення майбутньої ціни певного активу. Криптовалюти є одним з найпопулярніших їх видів, вартість якого намагаються передбачити [1].

Зазвичай для прогнозування застосовуються 2 основні підходи: технічний та фундаментальний аналізи. Обидва ці методи мають власні переваги та недоліки, а їх використання залежить від певних факторів та особливостей активу. Спільним їх недоліком є те, що вони не враховують певні сторонні фактори, які можуть значно вплинути на ефективність прогнозування.

Зазвичай передбачення курсу криптовалют базується на використанні технічного методу [2, 3]. Він включає в себе аналіз графіків ціни, попиту та індикаторів з використанням відповідних методів технічного аналізу. Значно рідше застосовується фундаментальний аналіз, що спричинене відсутністю певних фундаментальних індикаторів. Це зумовлене тим, що часто криптоактиви не прив'язані до певних пасивів та не приносять дохід як компанії, отже складно визначити фундаментальну цінність кожного з них. Проте, даний вид цифрових активів має певні характеристики, які рідко враховуються при передбаченні їх вартості, а саме дані про їх використання у мережі блокчейн. Так як модель даних цих валют побудована на основі попиту й пропозиції, дані характеристики можуть бути дуже важливими у передбаченні зміни популярності певного активу, що часто корелює з його ціною [4, 5].

Окрім згаданих типів аналізу криптовалют треба враховувати й безліч

сторонніх факторів. Як було зазначено, поточна ціна криптовалют формується відношенням між попитом та пропозицією. Саме ця особливість підкреслює важливість циклів економіки, відношення суспільства до даного виду активів та інших соціально-економічних [6 - 8] факторів під час створення прогнозу для даних активів.

З метою врахування описаних вище особливостей та проведення найбільш об'єктивного аналізу методів прогнозування необхідним є збір усіх згаданих вище типів даних. Це дозволить врахувати технічні, фундаментальні та соціально-економічні фактори при дослідженні методів й визначити ефективність використання кожного з цих видів інформації для вирішення задачі прогнозування котирувань криптоактивів.

Розвиток штучного інтелекту та створення нових архітектур нейронних мереж позитивно впливає на еволюцію підходів та методів. Велика кількість науковців та фахівців у даній галузі проводять дослідження й пропонують власні ідеї щодо створення нових або покращення існуючих методів прогнозування ціни активів. Це сприяє застосуванню найбільш ефективних алгоритмів для вирішення досліджуваного виду задач. Проте, наявність великої кількості варіацій алгоритмів прогнозування, а також неможливість систематизації усіх існуючих досліджень у цій галузі ускладнює процес вибору найефективнішого методу для вирішення поставленої задачі.

Зазвичай процес прогнозування складається з наступних етапів:

- аналіз особливостей завдання;
- збір та підготовка даних;
- вибір алгоритму;
- навчання нейронної мережі;
- аналіз отриманих результатів.

Етапи цього процесу є подібними майже для усіх видів задач прогнозування

і фази вибору правильного алгоритму та його навчання часто є найдовшим етапом вирішення певного практичного завдання. Це спричинене тим, що навіть якісно проведений аналіз існуючих досліджень лише звужує кількість потенційних алгоритмів, а визначення найбільш ефективного методу потребує проведення експериментального дослідження, порівняння результатів та вибір найкращого рішення для обраного завдання.

Автоматизація процесу порівняння різних методів прогнозування курсу криптоактивів може значно спростити згадані процеси, що надасть змогу швидко й ефективно знаходити найкращий метод прогнозування, а забезпечення гнучкості налаштування кожного алгоритму й можливість проведення порівняння за різними метриками сприятиме покращенню результатів дослідження й отриманню найбільш об'єктивних результатів.

1.2 Аналіз існуючих досліджень методів прогнозування котирувань криптовалют

Проблема прогнозування майбутньої ціни цього виду цифрових активів сприяла появі великої кількості різноманітних досліджень. Спочатку дослідники намагалися застосувати популярні на той час та універсальні алгоритми штучного інтелекту. Проведені ними дослідження довели ефективність використання певних архітектур нейронних мереж порівняно зі звичайними математичними методами. Для порівняння двох видів алгоритмів використовувалися метрики середньої абсолютної похибки (mean absolute error, MAE) та середньоквадратичної похибки (mean squared error, MSE). Для обчислення MAE використовувалася наступна формула:

$$MAE = \frac{1}{n} \sum_{i=1}^n |Y_i - \hat{Y}_i|, \quad (1)$$

де n – кількість прогнозованих елементів, Y_i – реальна ціна крипто активу, $\hat{Y}_i$ – прогнозована ціна криптоактиву.

Розрахунок MSE відбувався наступним чином:

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2, \quad (2)$$

де n – кількість прогнозованих даних, Y_i – реальна ціна крипто активу, $\hat{Y}_i$ – прогнозована ціна криптоактиву.

Значення метрик MAE та MSE були меншими на 10-15% та 20-24% відповідно при застосуванні алгоритмів машинного навчання у порівнянні з методами математичного аналізу. Проте, точність прогнозів все ще була недостатньо високою для можливості їх застосування при вирішенні практичних задач, що вимагало подальшої видозміни та розвитку використаних алгоритмів.

На основі цього науковці та фахівці цієї галузі намагалися видозмінювати базові алгоритми з метою їх оптимізації саме для задачі прогнозування ціни активів. Було проведено велику кількість досліджень з метою визначення наборів даних та взаємозалежностей між цими даними й іншими активами, щоб згодом використовувати найбільш ефективні набори для навчання нейронних мереж.

Аналіз різних джерел та типів даних займає окремий напрямок у дослідженні даного виду задач й багато фахівців намагаються визначити найефективніші параметри для кожного окремого виду активів. Більшість досліджень, які відносяться до прогнозування курсу криптоактивів поділяються на наступні категорії:

- дослідження алгоритмів машинного навчання та їх оптимізація;
- аналіз та використання статистичної інформації блокчейну;
- аналіз соціальних та економічних факторів.

Аналіз різних варіацій алгоритмів машинного навчання є дуже популярним напрямком. Загалом науковці намагаються зрозуміти природу й особливості формування ціни криптоактивів та використати цю інформацію для прогнозування майбутньої ціни за допомогою штучного інтелекту. Є достатньо велика кількість досліджень, які орієнтуються на аналіз ефективності використання певних даних

та їх вплив на результат прогнозування. Наприклад, К. Ларасаті у роботі [9] досліджує ефективність застосування статистичних даних блокчейну для прогнозування вартості криптовалюти Bitcoin. Окрім цього, достатньо популярними є дослідження впливу соціальних факторів на курс даного виду активів. Т. Росмен та С. Якар у публікації [10] досліджують вплив настроїв користувачів Twitter на ціну та об'єм торгів певного активу шляхом сентиментального аналізу твітів.

Окремим важливим й популярним підходом до прогнозування курсу криптовалют є аналіз методів та алгоритмів, які можуть застосовуватися для ефективного вирішення досліджуваної проблеми. Найпростішим інструментом для прогнозування курсу активів є лінійна регресія. М. Алі та С. Шатабда у науковій роботі [11] дослідили процес прогнозування курсу Bitcoin з використанням лінійної регресії, а також застосування певної методології до створення набору даних для її навчання.

Окрім лінійної регресії, науковці часто аналізують застосування ансамблевих методів машинного навчання, модифікацію їх параметрів та порівняння ефективності й точності їх роботи між собою. Здебільшого у цих роботах досліджується зміна архітектури або параметрів алгоритму, використання різних алгоритмів оптимізації та гіперпараметрів мережі. Прикладом однієї з таких робіт є публікація Д. Ву та С. Гуо [12] в якій вони дослідили використання XGBoost для прогнозування курсу Bitcoin у короткостроковій перспективі.

Також достатньо популярним інструментом прогнозування, який досліджується і здобуває популярність останнім часом є ARIMA. Її застосовують для вирішення задач, які пов'язані з прогнозування часових рядів й передбачення майбутньої ціни активів, де криптоактиви не є виключенням. Дослідження застосування ARIMA для прогнозування курсу криптовалют були наведені у праці Й. Сі [13], в якій він використав ARIMA для передбачення вартості Bitcoin.

Розвиток глибинного машинного навчання сприяв вирішенню та автоматизації великої кількості завдань, тому даний інструмент знайшов застосування і у цій сфері. Багато дослідників аналізували й розглядали різні архітектури нейронних мереж для вирішення проблеми прогнозування. С. Бісвас та М. Павар у роботі [14] провели аналіз використання методу глибинного машинного навчання для прогнозування ціни криптоактивів. Найбільш популярною архітектурою цього виду навчання для прогнозування часових рядів є LSTM. Вона доводить свою ефективність у порівнянні з іншими архітектурами і при вирішенні нашого завдання. Це привертає все більшу увагу науковців, що мотивує їх шукати методи для досягнення кращих результатів за допомогою зміни моделі або параметрів нейронної мережі, які базуються на згаданій архітектурі або ж оптимізації вхідного набору даних. В. Хуан та В. Чжан у публікації [15] використали згадану архітектуру з метою дослідження ефективності використання сентиментальних даних, а Я. Яо у науковій роботі [16] дослідив залежність ефективності моделі на основі LSTM від розміру вхідного часового ряду.

1.3. Дослідження алгоритмів, які застосовуються для вирішення задач прогнозування

Системи прогнозування відносяться до окремого типу задач інтелектуального аналізу даних й мають власні методи та алгоритми, які є найбільш ефективними для їх вирішення [17].

Одним з основних підходів, який застосовується для задач прогнозування є регресія. Вона базується на встановленні залежності між незалежними змінними та певним вихідним значенням або результатом. Даний метод використовується для моделей прогнозування послідовних даних. Процес навчання моделей, які використовують регресію полягає у тому, щоб модель мала змогу згенерувати очікуваний результат на основі певних вхідних даних. Це досягається шляхом

оцінки залежності між вхідними параметрами та їх впливом на кінцевий результат.

В процесі дослідження були розглянуті наступні алгоритми:

1. Лінійна регресія.
2. Random Forest.
3. XG Boost.
4. ARIMA.
5. LSTM.

Лінійна регресія [18] – алгоритм машинного навчання, який знаходить лінійну залежність між залежною та незалежною змінними (рис. 1.1). Процес навчання полягає у оптимізації параметрів прямої, яка визначається наступною формулою:

$$Y = \beta_0 + \sum_{i=1}^n \beta_i x_i + \varepsilon, \quad (3)$$

де Y – значення залежної змінної, x_i – значення незалежної змінної, β_0 – точка перетину прямої з віссю ординат, β_i – кутовий коефіцієнт, значення якого змінюється у процесі навчання для побудови оптимальної прямої, ε – випадкова похибка.

Цей вид алгоритму є дуже популярним й простим у застосуванні для вирішення задач, проте використання прямої дуже часто спричиняє занадто консервативні прогнози щодо майбутньої ціни активу. Ми досліджуємо процес прогнозування курсу криптовалют, які є дуже волатильними активами й зміна їх котирувань часто залежить від певних подій. Мала чутливість до швидкої зміни значень призведе до гіршої точності прогнозів, що в свою чергу робить даний алгоритм не найбільш ефективним для вирішення нашої задачі. З іншого боку, даний алгоритм дозволяє позбутися шуму у числових рядах даних, а збільшення періоду інтервалів прогнозування сприятиме вищій ефективності.

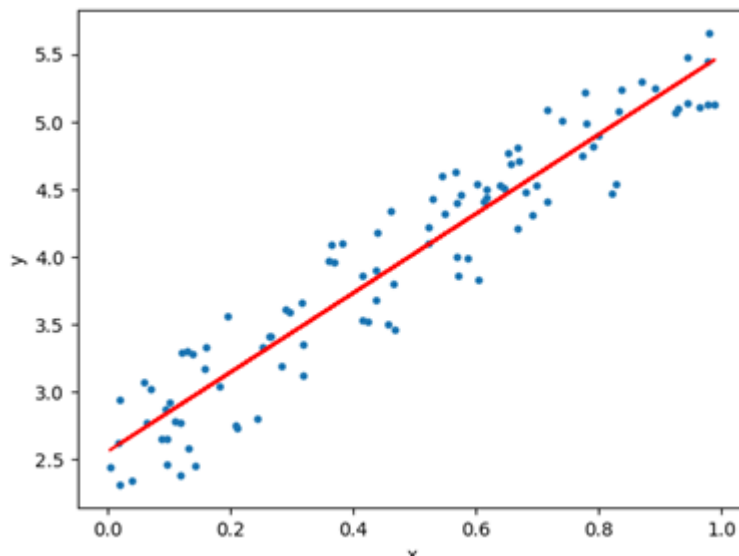


Рис. 1.1. Графічне зображення лінійної регресії

Random Forest – вид ансамблевого методу машинного навчання, що полягає у побудові певної кількості дерев прийняття рішень під час тренування моделі.

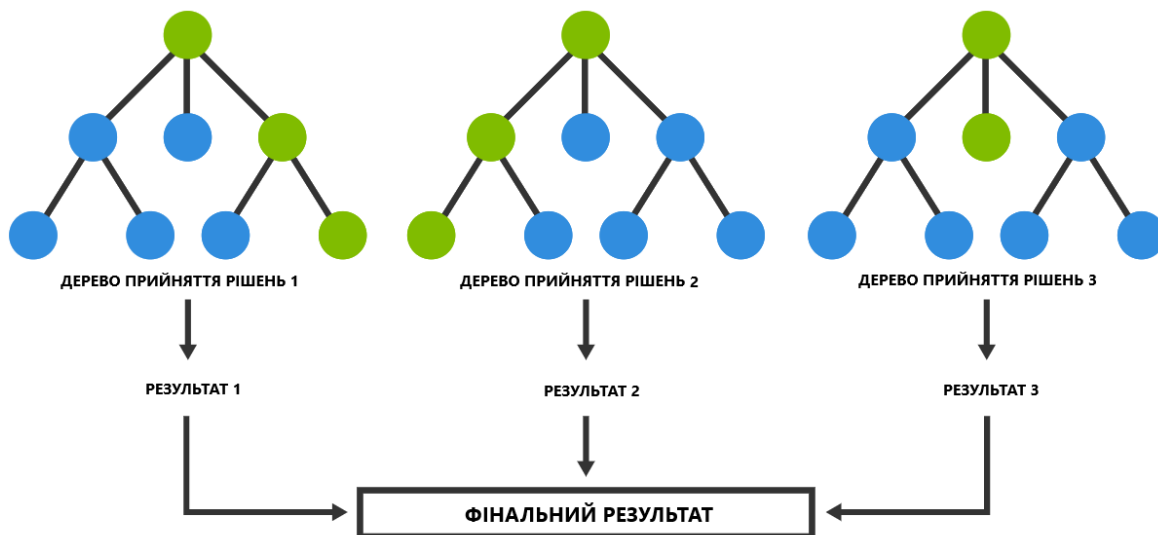


Рис. 1.2. Графічна репрезентація алгоритму Random Forest [19]

Процес прогнозування за допомогою даного алгоритму базується на генерації результату кожним зі створених дерев й визначення результату шляхом обробки отриманих даних (див. рис. 1.2). Є декілька варіантів обробки вихідних даних, проте в основу покладений метод, який здатний вирахувати певне число на основі всього набору результуючих даних. Дане число і є результатом прогнозування, а у нашому варіанті – майбутньою ціною певної криптовалюти.

Даний алгоритм зменшує ризик перенавчання й дозволяє просто визначити ефективність кожного параметру набору даних та використати найоптимальніші при побудові дерев прийняття рішень.

XG Boost – алгоритм, який також використовує дерева прийняття рішень, проте на відміну від Random Forest, він базується на ансамблі різних дерев, які створюються у процесі навчання. Даний алгоритм створює послідовні дерева, кожне з яких вчиться на помилках попередника (рис. 1.3). В результаті формується ансамбль дерев, який складається як з простих дерев прийняття рішень так і з тих, які були покращенні шляхом навчання на помилках інших. Результатом є значення вираховане на основі вихідного значення кожного елемента ансамблю.

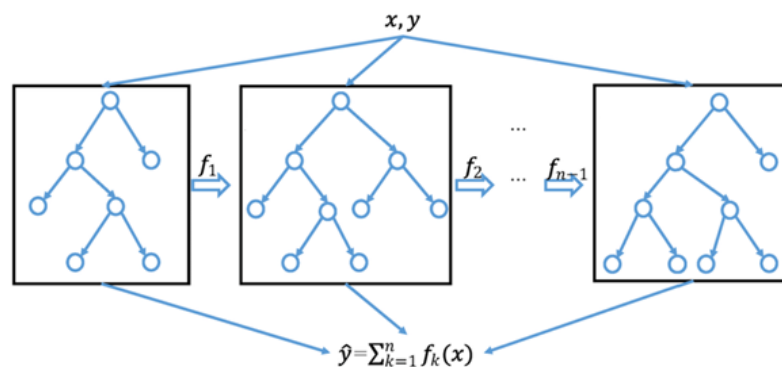


Рис. 1.3. Графічна репрезентація алгоритму XG Boost [20]

ARIMA – метод прогнозування майбутніх значень на основі попередніх з використанням методів статистичного аналізу рядів даних. Це дуже популярний метод прогнозування часових рядів й він часто застосовується для аналітики економічних та фінансових даних. В основу даного методу покладені наступні компоненти:

1. Авторегресія (AR) – модель авторегресії, яка використовує попередні значення для прогнозування майбутніх.

2. Інтеграція (I) – компонент, що забезпечує стаціонарність даних за допомогою використання різниці між послідовними даними.

3. Середня ковзна (МА) – включає залежність між різницею спостережень і ковзною середньою.

Прогнозовані дані обчислюються з врахуванням згаданих компонентів за наступною формулою:

$$Y_t = c + \sum_{i=1}^p \varphi_i Y_{t-i} + \sum_{i=1}^q \theta_i \varepsilon_{t-i} + \varepsilon_t, \quad (4)$$

де Y_t – значення прогнозованого (t-го) елемента часового ряду, c – константа, яка застосовується для врахування середнього значення даних часового ряду, φ_i – параметр компонента авторегресії, що відображає залежність значення поточного елемента від попередніх значень часового ряду, Y_{t-i} – попередній елемент часового ряду, θ_i – параметр компонента середньої ковзної, що відображає залежність значення поточного елемента від помилок прогнозування попередніх елементів часового ряду, ε_{t-i} – помилка прогнозування попередніх елементів часового ряду, ε_t – помилка прогнозування поточного елемента часового ряду, що визначається різницею між реальним і прогнозованим значеннями.

Даний метод дозволяє провести аналіз попередніх даних, визначати параметри кожного компонента й створити власні прогнози за допомогою згаданих статистичних компонентів (рис. 1.4). Це дозволяє застосовувати цей статистичний метод при прогнозуванні вартості криптоактивів. В той же час, точність отриманих результатів іноді є навіть краща ніж у лінійної регресії або ансамблевих методів прогнозування.

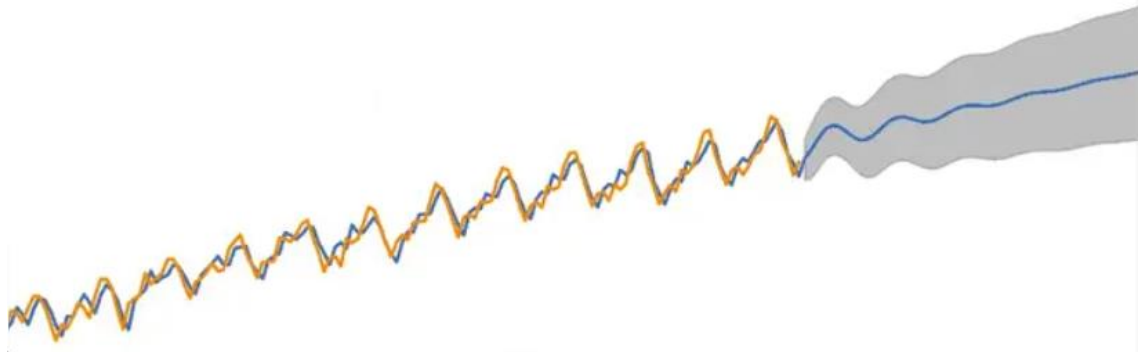


Рис. 1.4. Приклад результату використання алгоритму ARIMA [21]

LSTM – більш складний алгоритм у порівнянні з попередніми, проте він налічує значну кількість переваг. Ця архітектура є підтипом рекурентної нейронної мережі, які застосовують за необхідності враховувати порядок елементів при прогнозуванні послідовностей (рис. 1.5).

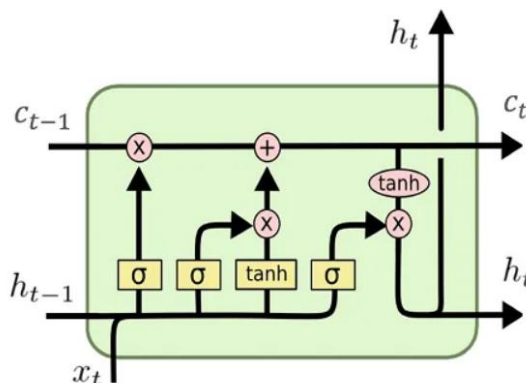


Рис. 1.5. Блок LSTM моделі [22]

Ідея даного виду мережі полягає у збереженні та використанні попередніх елементів послідовності для прогнозування наступних. Врахування попередніх даних є дуже важливою особливістю, що сприяє використанню даного виду нейронних мереж майже у всіх видах задач, які відносяться до часових рядів.

Ціна криптовалют являється результатом впливу великої кількості факторів. Врахування попередніх даних часового ряду допомагає аналізувати зміну кожного параметру та робити прогноз використовуючи знайдені закономірності та взаємозв'язки.

1.4. Аналіз можливості застосування AutoML для порівняльного аналізу алгоритмів

Одним із завдань нашого дослідження є аналіз можливості створення системи для проведення порівняльного аналізу ефективності методів прогнозування курсу криптовалют з використанням АМН. Основною метою даної системи є автоматизація процесу навчання нейронних мереж з можливістю зручного й швидкого їх налаштування та гнучкого вибору різних даних для

аналізу. Такі особливості системи допоможуть автоматизувати більшість процесів програмної реалізації та отримати простий на ефективний інструмент аналізу.

AutoML [23] – вид автоматизованого машинного навчання, що полягає у автоматизації застосування моделей машинного навчання для вирішення різних задач. Він включає в себе процеси вибору та обробки даних, ініціалізацію нейронних мереж, їх навчання та вимірювання ефективності. Це дозволяє користувачам, які не є фахівцями у сфері штучного інтелекту та машинного навчання, застосовувати новітні технології для вирішення певних практичних завдань. Взаємодія з даними системами зазвичай відбувається за допомогою графічного інтерфейсу вебсервісу, що не потребує написання коду або самостійного впровадження алгоритмів.

AutoML стає дуже популярним так як даний вид систем спрощує й автоматизує весь процес аналізу даних від збору даних до розгортання створеної моделі. Цей інструмент здобуває популярність як у початківців так і досвідчених аналітиків даних, оскільки він дозволяє швидко протестувати певні ідеї та підходи при вирішенні певних практичних завдань. Звичайний процес розробки нейронних мереж складається з багатьох етапів, які потребують створення певних програмних рішень, що вимагає часу та відповідного рівня компетентності. В той же час, автоматизоване машинне навчання зменшує кількість етапів й пришвидшує більшість з них, що дозволяє приділяти більше часу саме дослідженню та порівнянню різних алгоритмів, а також перевірці власних припущень.

АМН стрімко здобуває популярність й велика кількість компаній на ринку пропонують свої рішення з застосуванням цієї технології. Варто виділити наступні програмні продукти:

- Google AutoML;
- Amazon SageMaker Autopilot;

- Microsoft Azure AutoML.

Google AutoML [24] – інструмент розроблений компанією Google, що складається з трьох продуктів: AutoML Vision, AutoML Natural Language, AutoML Tables. AutoML Vision надає можливість автоматизувати процес комп'ютерного зору при роботі з зображеннями. AutoML Natural Language створений для обробки цифрових даних для аналізу текстів та аудіофайлів з метою визначення настроїв, голосових команд, верифікації за допомогою голосового зліпку, тощо. AutoML Tables призначений для роботи з табличними даними для вирішення задач класифікації та прогнозування.

Amazon SageMaker Autopilot [25] – програмне рішення від Amazon, яке представлено вебсервісом, який здатний автоматизувати процес створення та навчання моделей машинного навчання за допомогою автоматичного вибору алгоритму на основі наявних даних. Окрім автоматичного режиму, користувач може змінювати налаштування самостійно задля досягнення кращих результатів у вирішенні своєї проблеми.

Microsoft Azure AutoML [26] – інструмент від Microsoft, що дозволяє вирішувати велику кількість видів задач аналізу даних. Функціонал та можливості системи є досить подібними з Amazon SageMaker Autopilot, проте Microsoft Azure AutoML може взаємодіяти зі всією екосистемою Microsoft, наприклад Microsoft Azure. Це дозволяє отримати безліч переваг у створенні наборів даних, виконанні ETL процесів, а також інтеграції з іншими вебдодатками, які базуються на технологіях Microsoft.

Незважаючи на велику кількість інструментів АМН, більшість з них орієнтована на вирішення практичних завдань й не має достатнього функціоналу та можливостей для проведення дослідження та порівняння різних методів та алгоритмів машинного навчання. Цей факт спонукає до проведення дослідження можливості застосування даного виду машинного навчання у нашій системі.

Процес об'єктивного порівняння методів прогнозування вимагає створення, навчання та оцінки значної кількості варіацій нейронних мереж, що значно уповільнює й ускладнює цей процес. Проте, ця проблема може бути вирішеною за допомогою застосування принципів AutoML у нашій системі, що допоможе значно пришвидшити й спростити порівняння методів прогнозування курсу криптовалют за допомогою інструментів автоматизації. Для цього необхідно дослідити конвеєр АМН та спроектувати і реалізувати його основні етапи. Ці етапи будуть складатися з процесів збору та обробки даних, створення, навчання нейронних мереж та отримання й обробки результатів.

1.5 Постановка завдання

Завдання прогнозування ціни активів завжди було актуальним у сфері аналітики даних. Спочатку фахівці намагалися знаходити математичні закономірності у часових рядах даних використовуючи історичні дані зміни ціни або котирувань на біржі. Згодом цей напрям значно розвинувся і з'явилося поняття технічних індикаторів. Технічні індикатори - це певні математичні функції або метрики, які, наприклад, часто використовують у трейдингу. Поява і розвиток комп'ютерних технологій тільки популяризувало даний вид аналізу через те, що ЕОМ здатні неймовірно швидко виконувати розрахунки й відповідно трейдери здатні швидше здійснювати операції та отримувати більші прибутки.

Однак, розвиток комп'ютерних технологій, окрім автоматизації існуючих процесів, сприяв і подальшому розвитку даного виду аналітики. Розвиток штучного інтелекту спричинив велику кількість застосування даного виду технологій у різних сферах. Прогнозування вартості активів стало достатньо популярною проблемою, яку намагалися вирішити за допомогою машинного навчання.

З одного боку наявність великої кількості досліджень у даній галузі сприяє

розвитку й покращенню методів, але з іншого боку відсутність систематизації цих досліджень та взаємозалежності між отриманими результатами часто ускладнюють процес вирішення певного практичного завдання. Як результат, дослідник повинен самостійно проводити перевірку й експериментальне випробовування ідей запропонованих у цих роботах з метою створення найефективнішого вирішення власної проблеми. Дана ситуація породжує потребу автоматизації процесу пошуку найбільш оптимального рішення нашого завдання, а саме прогнозування курсу криптовалют.

Популярним типом вебсервісів у сфері машинного навчання в наш час стають сервіси AutoML. Вони надають можливість звичайним користувачам створювати нейронні мережі для вирішення певних бізнес-проблем за допомогою користувацького інтерфейсу. Тобто, вони неймовірно спрощують процес створення, навчання та розгортання систем аналізу даних й дозволяють користувачам з базовими знаннями про штучний інтелект використовувати даний інструмент у роботі та повсякденному житті.

Ріст популярності даного виду послуг у значній мірі спричинений простотою, та зручністю його використання. Це стало можливим саме завдяки можливості гнучкої конфігурації моделей нейронних мереж відповідно до власних потреб. При цьому система автоматично, тобто без зайвого втручання людини, здатна застосувати обрані користувачем параметри моделей нейронних мереж та налаштувати процес обробки даних від їх збору до генерації прогнозів та обробки результатів.

На основі описаних проблем та особливостей були визначені завдання, які мають бути вирішені в межах кваліфікаційної роботи. Для зручності ці завдання були поділені у дві групи, які представлені вебсервісом для створення, налаштування й аналізу методів прогнозування та компонентом для автоматизації процесів збору даних, створення, навчання нейронних мереж і генерації прогнозів.

Вебсервіс має наступні вимоги:

1. Забезпечення можливості налаштування нейронних мереж за допомогою користувацького інтерфейсу, що дозволить перевіряти різні ідеї та гіпотези досліджень без потреби змінювати програмний код.
2. Можливість порівняння алгоритмів між собою за певними метриками.
3. Наявність функціоналу для графічного аналізу вхідних даних та ефективності прогнозування досліджуваних методів.
4. Можливість проводити бек-тестинг для визначення потенційного прибутку від застосування досліджуваних алгоритмів.

Для реалізації можливості порівняння різних методів між собою необхідним є впровадження компоненту автоматизації, який буде вирішувати наступні завдання:

1. Проводити збір різних видів даних, які будуть доступні для використання при навчанні обраних моделей нейронних мереж.
2. Автоматизувати процес створення, навчання та порівняння методів прогнозування курсу криптовалют. Це має бути досягнутим за допомогою впровадження підходів AutoML у систему.
3. Виконувати прогнозування створеними моделями з метою подальшого використання отриманих результатів під час порівняння їх ефективності.
4. Забезпечувати трансформацію та підготовку даних для можливості їх використання у аналітичних системах.

В результаті виконання поставлених завдань має бути створена повноцінна система, яка дозволить ефективно порівнювати методи прогнозування курсу криптовалют та їх модифікації між собою.

Висновки до першого розділу

У даному розділі був проведений аналіз процесу прогнозування курсу криптовалют, який дозволив дослідити основні особливості даного виду задач й підходи, які використовуються для їх вирішення. Також був проведений аналіз актуальних досліджень, які стосуються передбачення майбутніх котирувань криптовалют й були розглянуті основні алгоритми, які застосовуються при прогнозуванні вартості активів.

Процес об'єктивного порівняння методів прогнозування вимагає створення, навчання та оцінки значної кількості варіацій нейронних мереж, що значно уповільнює й ускладнює цей процес. З цією метою було проведено дослідження можливості застосування принципів АМН у нашій системі, що допоможе значно пришвидшити й спростити порівняння досліджуваних методів за допомогою інструментів автоматизації.

У результаті проведеного дослідження процесу прогнозування курсу криптовалют, архітектур нейронних мереж та процесу автоматизованого машинного навчання були сформовані задачі, які мають вирішуватися розроблюваною системою для того, щоб вона надавала змогу проводити повноцінний та ефективний аналіз.

РОЗДІЛ 2. МОДЕЛЮВАННЯ СИСТЕМИ АНАЛІЗУ МЕТОДІВ ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ

2.1 Проектування системи з застосуванням особливостей конвеєру AutoML

Зазвичай AutoML рішення представляє собою конвеєр [27] (рис. 2.1), який складається з послідовних операцій, що дозволяють перетворити набір даних та певні вхідні параметри у очікуваний результат. Кожний етап може відрізнятися в залежності від поставленої задачі, проте основний підхід та порядок виконання процесів є подібним:

- збір даних;
- обробка даних;
- створення моделі нейронної мережі;
- навчання створеної моделі з використанням отриманих даних;
- вимірювання ефективності навченої нейронної мережі;
- розгортання моделі.

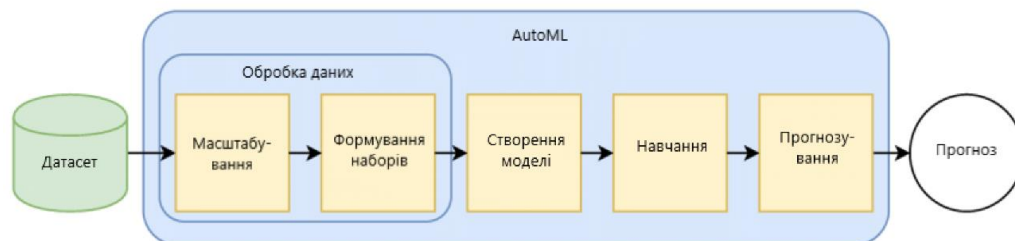


Рис. 2.1. Схема роботи конвеєру AutoML

Так як АМН може забезпечити нашій системі багато переваг шляхом оптимізації процесу порівняння методів прогнозування курсу криптовалют, важливим етапом нашого дослідження є аналіз можливості вирішення нашого завдання відповідно до вимог та особливостей даного виду машинного навчання.

Першим етапом розробки нейронної мережі є збір даних. У нашій системі цей процес може бути автоматизований та проведений за допомогою створення

відповідних програмних модулів, які дозволять зібрати дані з різноманітних джерел та зберегти їх у спільну БД для подальшого проведення технічного, фундаментального та сентиментального аналізів.

Наступним етапом конвеєру є обробка даних. Даний етап може поділятися на різну кількість підпроцесів, що залежить від типу даних та визначених цілей. Відповідно до особливостей нашої задачі було виокремлено 2 процеси: масштабування та формування наборів даних. Ці процеси будуть видозмінюватися відповідно до обраного алгоритму та його параметрів, проте набір процесів етапу обробки даних є незмінним.

Після обробки вхідної інформації відбувається формування тестових наборів даних. Наша система працюватиме з різними типами нейронних мереж й кожний з них має певні особливості й вимагає приведення вхідної інформації до певної структури. Загалом їх можна поділити на 2 групи:

- алгоритми, які приймають вектор, що являє собою скалярне значення кожного параметру моделі: лінійна регресія, Random Forest, XG Boost, ARIMA;
- алгоритми, які приймають проміжок даних, який містить в собі певну кількість наборів векторів параметрів моделі - LSTM.

Процес підготовки даних для алгоритмів першого типу являє собою простий їх розподіл на набори для тренування, валідації та тестування. В той же час алгоритми другого типу потребують додаткової трансформації вхідної інформації. У цьому випадку використовується відповідний параметр, який відповідає за розмір проміжку даних й відбувається генерація наборів усіх трьох груп даних з врахуванням цього параметру, а також особливостей їх структури для відповідної нейронної мережі. В результаті отримуються вхідні дані, які містять інформацію про попередні періоди, що дозволяє моделям прогнозування знайти певні закономірності в часових рядах та покращити свою ефективність.

Наступним кроком АМН є створення моделі. Як було згадано раніше,

однією з переваг обраної архітектури програмного забезпечення є можливість налаштування та керування усім процесом за допомогою графічного інтерфейсу. Задля цього, у нашій системі буде розроблений вебдодаток, який надасть можливість швидко й просто налаштувати параметри алгоритму, обрати архітектуру нейронної мережі, вид криптовалют, інтервал періодів, а також необхідні параметри кожного типу даних.

Важливим завданням у проектуванні та розробці системи відповідно до принципів АМН є створення системи, яка буде максимально проста й ефективна у використанні. Саме це дозволяє пришвидшити, спростити та автоматизувати певні рутинні або ресурсоємні процеси. В той же час, ця зручність не повинна впливати на гнучкість даної системи. Усі розглянуті алгоритми, використання яких повинно бути автоматизованим, повинні мати певні параметри й налаштування, які дозволяють модифікувати алгоритм й таким чином пристосувати його для вирішення певного завдання.

Для налаштування лінійної регресії були обрані наступні параметри:

- параметр регуляризації;
- епсілон SVR-моделі;
- tolerance.

Параметр регуляризації використовується при розрахунку похибки моделі, де похибка є L2 метрикою, яка використовується для зменшення перенавчання моделі та поліпшення здатності узагальнювати результат. Епсілон SVR-моделі – застосовується для визначення ширини “коридору” для функціональної межі SVR. Tolerance визначає допустиму похибку між прогнозованими і реальними даними.

Для можливості налаштування Random Forest важливим є проведення аналізу впливу параметрів на результат у контексті певної задачі. У випадку прогнозування курсу криптовалют ефективними виявилися наступні параметри:

- кількість дерев, які будуть використовуватися у ансамблі;

- максимальна глибина кожного дерева;
- мінімальна кількість елементів за якої відбувається внутрішній поділ вузла;

- мінімальна кількість елементів, яка має бути у листовому вузлі.

Для моделі на основі XGBoost були обрані такі параметри як:

- швидкість навчання;
- кількість оцінювачів;
- gamma.

Швидкість навчання відповідає за розмір кроку, який використовується при оновленні вагових коефіцієнтів з використанням методу градієнтного спуску. Кількість оцінювачів використовується для визначення кількості дерев, які будуть створені та змінені у процесі навчання. Параметр gamma застосовується для визначення мінімального зменшення значення цільової функції за якого відбувається поділ вузла.

На відміну від лінійної регресії, Random Forest та XGBoost також повинні мати можливість автоматичного пошуку найкращого набору параметрів. Це може бути досягнутим завдяки перебору можливих значень для кожного з них з відповідної сітки параметрів, вимірювання ефективності отриманих моделей на кожному етапі перебору та визначення найкращого набору параметрів.

Можливість налаштування параметрів є особливо важливим при створенні моделі ARIMA. Для того, щоб її можна було ефективно використовувати у системі на основі AutoML, важливим є крок вибору цих параметрів відповідно до вхідних даних. Необхідними для модифікації моделі є наступні параметри:

- p – вказує кількість попередніх спостережень, які використовуються для прогнозування;
- d – використовується при перетворенні вхідного часового ряду у стаціонарний;

- q – параметр для врахування зв'язку між поточними даними і попередніми випадковими коливаннями.

Процес створення моделі LSTM є найскладнішим з усіх попередньо згаданих й відповідно має більшу кількість параметрів та варіантів налаштування. Конфігурацію моделі на основі даної архітектури глибоких нейронних мереж можна розглядати у двох напрямках: зміна параметрів моделі та зміна структури глибокої нейронної мережі. Основними параметрами було обрано:

- кількість епох навчання;
- кількість епох для зупинки (навчання зупиняється, якщо впродовж визначеної кількості попередніх епох модель не зазнала покращень);
- параметр швидкості навчання;
- алгоритм оптимізації.

Окрім цього, важливим є використання функцій зворотного зв'язку, які дозволяють вчасно зупинити навчання за потреби або динамічно змінювати швидкість навчання. Вони допомагають автоматизувати контроль за змінами параметрів та метрик моделі прогнозування курсу криптовалют. Це дозволяє вирішити проблеми з перенавчанням навіть у автоматизованих системах, уникаючи потреби залучення користувача у цей процес. Налаштування структури моделі має відбуватися за допомогою клієнтського інтерфейсу у вигляді конструктора. Це дозволить додавати, редагувати або видаляти шари нейронної мережі (LSTM, Dense, Dropout), а також налаштовувати параметри кожного з них (кількість нейронів, функція активації, значення dropout, тощо). Це дозволяє зручно й швидко створювати різні будови нейронних мереж й порівнювати їх ефективність.

Отже, кожний з розглянутих алгоритмів має свої особливості, параметри та вимоги, проте можливість їх налаштування є невід'ємним етапом створення моделі у конвеєрі AutoML.

Після налаштування алгоритму за допомогою графічного інтерфейсу, він буде створюватися за допомогою відповідного програмного модулю. Кожна обрана архітектура буде реалізована й впроваджена таким чином, щоб вона підтримувала динамічну кількість даних й могла використовуватися іншими програмними модулями.

Отже, згаданий компонент системи звертатиметься до бази даних вебдодатку через мережевий інтерфейс та буде отримувати інформацію про нові або оновлені алгоритми. Після цього, даний модуль буде створювати моделі нейронних мереж, завантажувати та оброблювати дані й у результаті виконувати наступний крок конвеєру – навчання моделей.

Кожна модель має певні власні особливості та підтримувані набори даних, проте основна мета їх навчання – зменшення середньої квадратичної похибки, що сприяє покращенню точності результатів і, як наслідок, дозволяє генерувати більш точні прогнози майбутньої вартості курсу криптовалют. Етап навчання полягає у покращенні ефективності нейронної мережі шляхом зміни параметрів алгоритму задля зменшення похибки, а також контролю зміни цільових метрик даного алгоритму протягом цього процесу.

Після навчання вагові коефіцієнти й параметри певних нейронних мереж зберігатимуться й відбудуватиметься фінальний етап конвеєру – прогнозування. Прогнозування буде виконуватися для усіх нових або оновлених алгоритмів окремим програмним компонентом. Всі результати прогнозування зберігатимуться у базу даних, що дозволить потім використовувати ці дані для вимірювання ефективності кожного алгоритму та їх порівняння між собою.

Генерація результату є фінальним етапом AutoML. Вихідні дані конвеєру можуть використовуватися в залежності від потреб користувача. Так як наша система буде надавати можливість порівнювати різні методи прогнозування курсу криптовалют, необхідною вимогою до неї є наявність відповідного функціоналу

для графічного представлення результатів аналізу для більш наочного порівняння.

Отже, використання архітектури типу AutoML у нашій системі є можливим і дозволить зручно й швидко створювати різні методи прогнозування та конфігурувати їх відповідно до своїх потреб за допомогою вебдодатку. Це позбавить нас необхідності самостійно впроваджувати певні ідеї та підходи у програмному коді й спростить процес аналізу та порівняння різних видів алгоритмів між собою.

2.2 Аналіз та розробка ETL процесу даних системи.

Початковим етапом навчання розглянутих нейронних мереж є збір та обробка даних. Необхідний набір вхідної інформації залежить від методів, які будуть застосовуватися при прогнозуванні ціни криптоактивів. Для вирішення нашого завдання були обрані наступні підходи:

- технічний аналіз;
- фундаментальний аналіз;
- аналіз соціально-економічних факторів.

Технічний аналіз криптовалют базується на дослідженні історичних даних зміни вартості певної торгової пари. Так як поточна ціна криптоактиву формується шляхом відношення між попитом та пропозицією, ціна криптовалюти може відрізнятися в залежності від платформи, яка надає можливість торгівлі даним видом активів.

У нашому випадку було обрано платформу Binance, як постачальника історичних даних. Дана криптовалютна біржа є дуже популярною, бо її користувачі проводять велику кількість торгових операцій, що робить її однією з лідируючих у світі за об'ємом.

Згадана платформа надає доступ до своїх даних за допомогою API, що дозволяє швидко та просто отримувати дані з сервісу Binance (рис. 2.2). Окрім

цього, даний вебсервіс безкоштовно надає доступ до великого часового проміжку історичних даних та дозволяє отримувати оновлення інформації у режимі реального часу за допомогою вебсокетів.

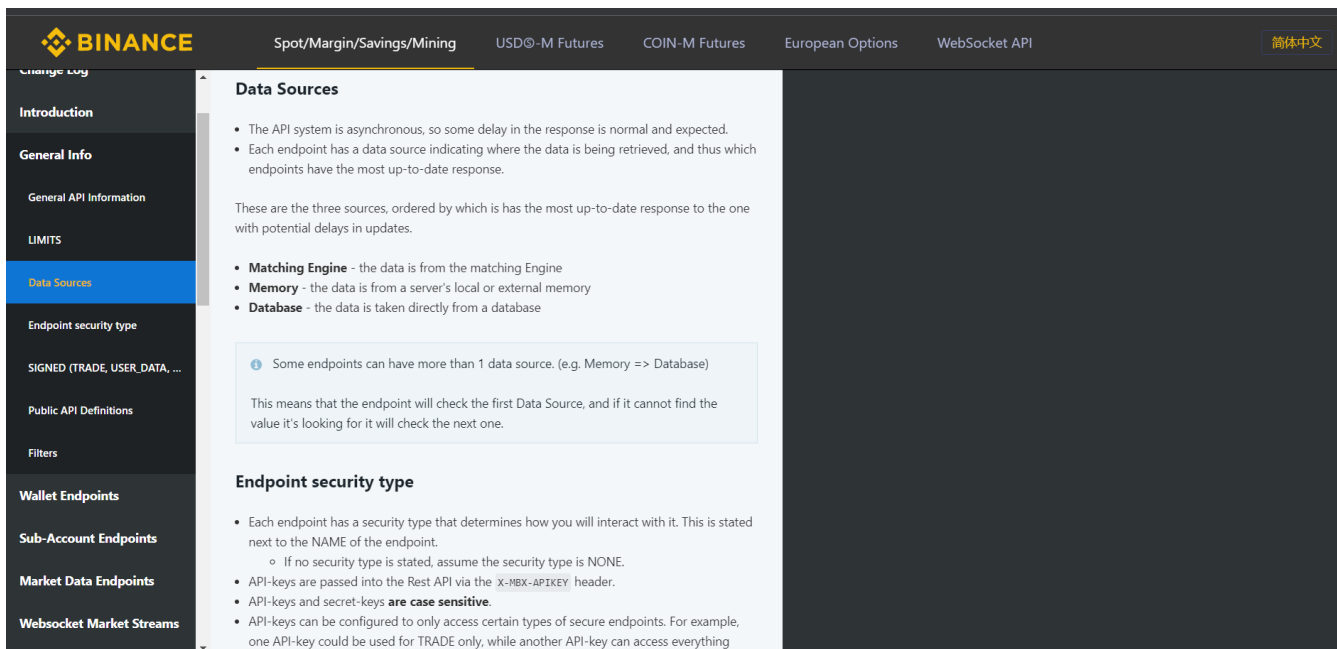


Рис. 2.2. Документація API для отримання даних з Binance [28]

Взаємодія зі згаданим вебсервісом була реалізована за допомогою модулю python-binance. Даний модуль є обгорткою для офіційного REST API Binance й дозволяє додаткам отримати доступ до історичних даних та додаткових функцій, які надаються користувачам криптобіржі.

Історичні дані зберігатимуться у БД та будуть використовуватися вебсервісом для навчання створених і налаштованих нейронних мереж та прогнозування вартості обраних криптоактивів. Назви полів історичних даних, їх тип та опис даних, які вони представляють, наведені у таблиці 2.1.

Таблиця 2.1

Структура історичних даних

Назва поля	Тип	Опис
trading_pair	varchar (32)	Назва торгової пари

Продовження таблиці 2.1

period	varchar (32)	Період вибірки даних
open	real	Ціна відкриття періоду
close	real	Ціна закриття періоду
high	real	Найвища ціна періоду
low	real	Найнижча ціна періоду
volume	real	Об'єм торгів представлений у вигляді загальної вартості операцій здійснених впродовж періоду
trades_amount	bigint	Кількість проведених торгових операцій впродовж періоду
close_time	integer	Час закриття представлений у форматі Unix

Аналіз фундаментальних факторів є дуже популярним підходом при дослідженні таких інвестиційних інструментів як акції та облігації компаній. В основу даного виду оцінки вартості активів покладено вивчення та використання інформації, яка містить дані про результати їх операційної діяльності, метрики прибутковості, тощо.

Криптоактиви відрізняються від згаданих видів активів й не можуть аналізуватися з використанням фундаментальних метрик через свої певні особливості. Проте, ми можемо певним чином видозмінити метод фундаментального аналізу й застосувати його для криптовалют.

Усі криптовалюти або токени базуються на технології блокчейн [29]. Ця технологія являє собою децентралізований цифровий реєстр транзакцій, що зберігається протягом тривалого часу і захищений певними методами шифрування. Дані цього реєстру розподіляються за допомогою мережі комп'ютерів. Користувачі цієї технології можуть безпосередньо взаємодіяти зі збереженими даними у режимі реального часу, не потребуючи посередника для підтвердження справжності транзакцій.

Зміни у блокчейн вносяться у вигляді транзакцій. Усі нові транзакції об'єднуються у блоки, після чого додаються до мережі шляхом приєднання блоку до останнього блоку-елемента ланцюга. Так як усі дані є загальнодоступними їх можна переглядати у вільному доступі.

Для прикладу, можна провести аналогію між блокчейном та банком. Чим більше у банку клієнтів, тим більше він починає заробляти на кредитах та комісіях з транзакцій, збільшуючи таким чином свій прибуток. Кожний блокчейн у свою чергу має певну кількість активних користувачів. Чим більше користувачів у мережі, тим більше власників монет, що позитивно впливає на її ринкову капіталізацію. Зміна вартості мережі прямопропорційно впливає на вартість криптовалюти, яка є нативною для цього блокчейну.

У наш час існує велика кількість вебсервісів, які дозволяють зручно переглядати блоки, транзакції та іншу інформацію про згадані системи (рис. 2.3). Ці сервіси є дуже зручними та корисними, проте вони є обмеженими у можливостях отримати дані за великі проміжки часу. Навіть якщо подібний вебресурс надає таку можливість, отримання цих даних вимагає придбання платного доступу.

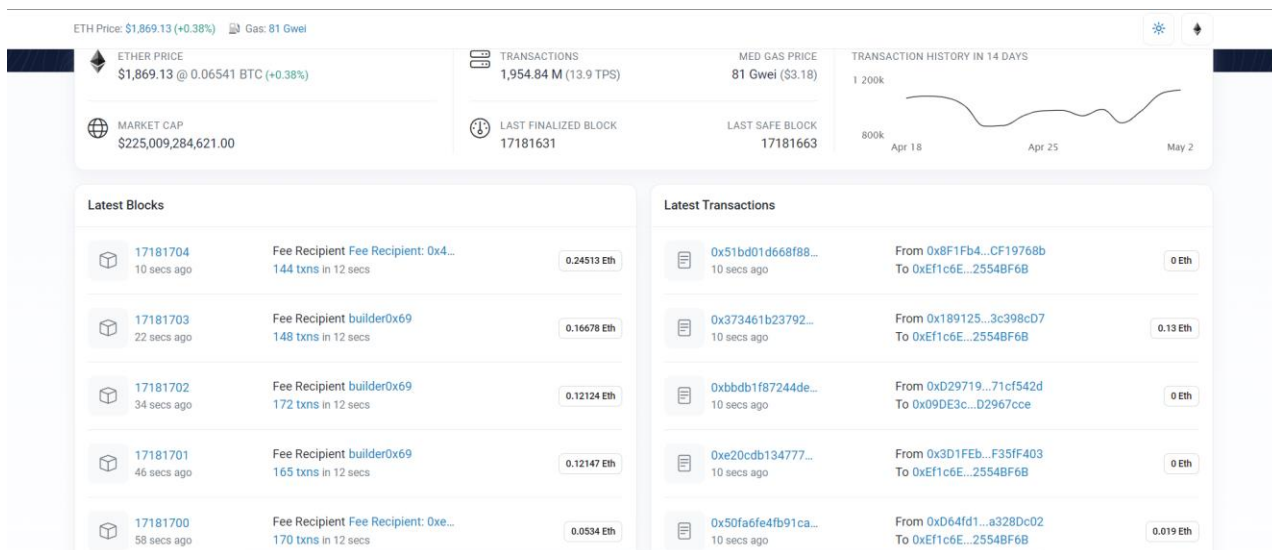


Рис. 2.3. Перегляд інформації про блокчейн Ethereum за допомогою вебресурсу Etherscan [30]

Як було загадано раніше, основною перевагою блокчейну є децентралізованість й загальнодоступність і саме це надає можливість отримати необхідні дані самостійно. Програмна реалізація відрізняється для кожного виду блокчейну, проте загальний концепт є ідентичним. Для впровадження процесу збору даних для нашої системи необхідно реалізувати програмні модулі для отримання даних з наступних мереж: Bitcoin, Ethereum та Binance Smart Chain.

Для збору даних Bitcoin було використано вебсервіс Blockchain Info, який індексує дані усього блокчейну й надає можливість їх отримати за допомогою API (рис. 2.4). На основі цього вебсервісу відбуватиметься завантаження інформації про блоки мережі та збір статистики на основі отриманих даних.

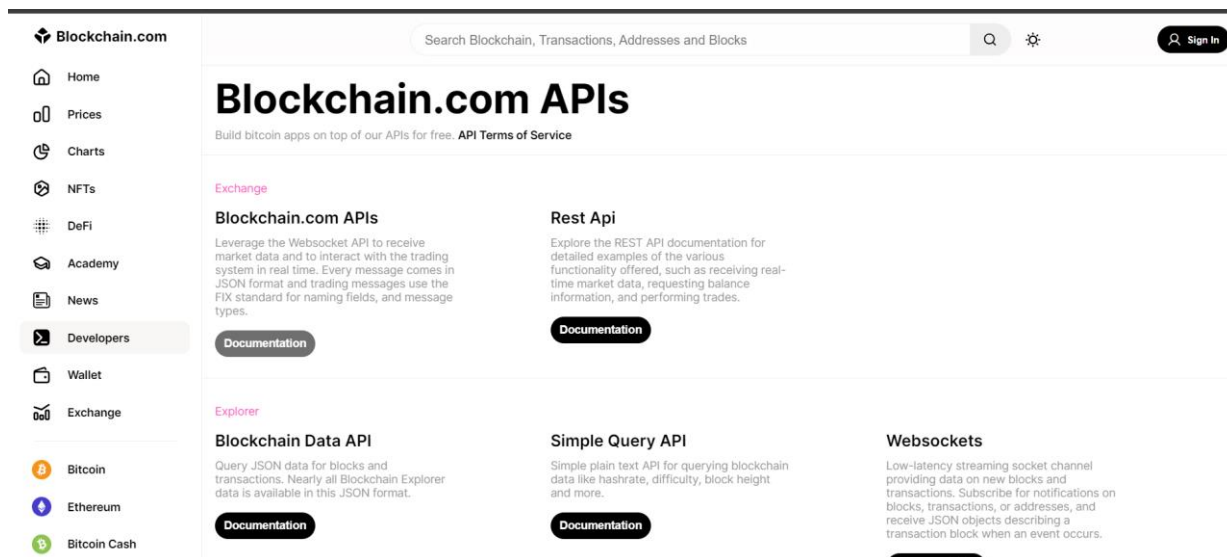


Рис. 2.4. API сервісу Blockchain Info [31]

За замовчуванням збір даних з блокчейнів Ethereum та Binance Smart Chain відбувається за допомогою взаємодії з відповідним блокчейном через протокол RPC [32]. Проте, розвиток технології Web3 [33] сприяв використанню особливостей та механізмів даної технології у великій кількості програмних продуктів, що спричинило створення модулю web3 для роботи з блокчейном.

Для взаємодії з мережею Ethereum було використано сервіс Infura, що дозволяє отримувати інформацію без розгортання блокчейну на локальному комп'ютері (рис. 2.5). В той же час, Binance Smart Chain спрощує процес доступу

до децентралізованої мережі надаючи відкриті сервіси для взаємодії з нею. Це позбавляє розробників необхідності розгортати копію блокчейну локально або шукати сервіси, які будуть це робити за них.

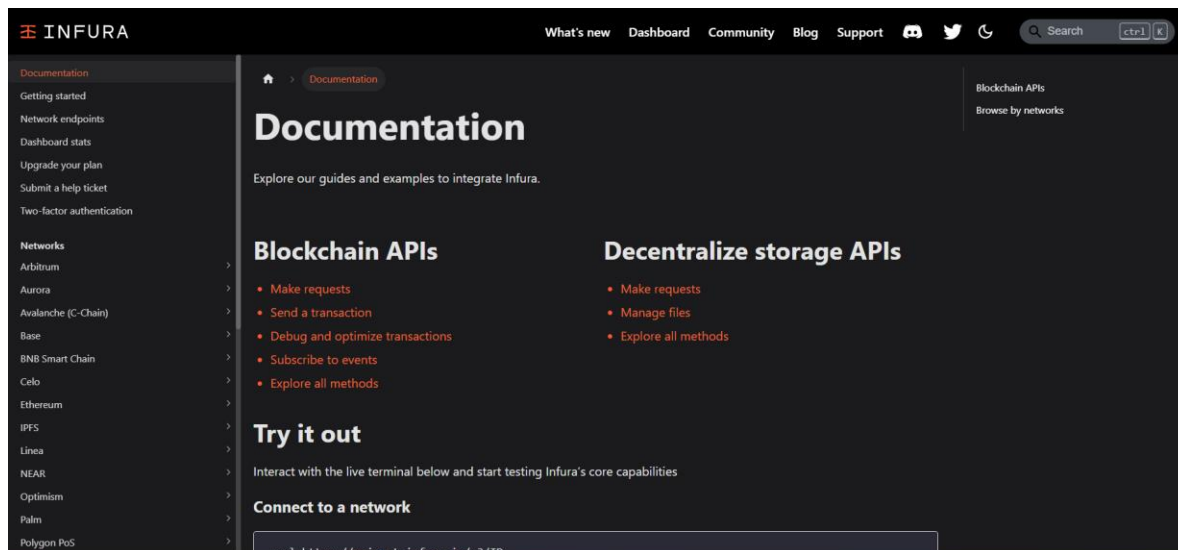


Рис. 2.5. Документація сервісу Infura [34]

У результаті виконання усіх процесів збору та обробки даних зі згаданих блокчейнів, опрацьовані дані були збережені у базу даних. Опис отриманої інформації наведений у таблиці 2.2.

Таблиця 2.2

Структура фундаментальних даних

Назва поля	Тип	Опис
ticker	varchar (32)	Назва криптовалюти або токена
blockchain_type	varchar (32)	Назва типу блокчейна
period	varchar (32)	Період вибірки даних
min_value	numeric	Мінімальна сума транзакції
max_value	numeric	Максимальна сума транзакції
average_amount	numeric	Середня сума транзакції
average_fee	numeric	Середня комісія транзакції
average_gas_price	numeric	Середня ціна газу (Ethereum та BSC)
gas_used	numeric	Кількість використаного газу (Ethereum та BSC)

Продовження таблиці 2.2

difficulty	numeric	Складність мережі
transaction_amount	bigint	Середня кількість транзакцій у блоці
time	integer	Час завершення періоду (Unix)

Криптовалюти та сфера децентралізованих технологій дуже тісно пов'язана з різними спільнотами. Даний вид сучасних технологій викликає цікавість у суспільства через свій значний потенціал розвитку нового етапу інтернет-технологій. Однак, популярність блокчейну, Web3 й як наслідок вартості криптовалют та токенів зростає за допомогою росту попиту з боку спільнот й оптимістичного погляду на майбутнє даної сфери.

Згадану особливість варто враховувати при аналізі та прогнозуванні курсу криптоактивів, отже було вирішено зібрати набір даних, який містить статистику щодо зміни настроїв відносно тієї чи іншої криптовалюти або токenu у суспільстві. Даний вид аналізу (сентиментальний аналіз) було вирішено проводити за допомогою обробки твітів з мережі X (Twitter).

За замовчуванням згадана мережа надає можливість безкоштовно аналізувати твіти тільки за невеликий проміжок часу, що не відповідає нашим потребам. Через це було прийняте рішення використовувати індексовану базу даних твітів, яка постійно збирає нові твіти та зберігає їх з метою оптимізації пошуку. Для цього було використано скрапер - бібліотеку `snsrape`. Сентиментальний аналіз твітів відбувався з використанням бібліотеки `nlTK`, шляхом обробки, аналізу та класифікації твітів у 3 категорії:

- негативні;
- нейтральні;
- позитивні.

Колекція твітів формується шляхом завантаження набору записів, які

містять відповідні хештеги, достатню кількість переглядів і підписників у автора, а також відповідають певним іншим критеріям. Після проведення оцінки емоційного забарвлення твітів, результат проведений аналіз зберігається у окрему таблицю бази даних з відповідною структурою. Детальний опис параметрів зібраної інформації наведено у таблиці 2.3.

Таблиця 2.3

Структура сентиментальних даних

Назва поля	Тип	Опис
keyword	varchar (256)	Ключове слово пошуку
period	varchar (32)	Період вибірки даних
positive	numeric	Кількість позитивних твітів
negative	numeric	Кількість негативних твітів
neutral	numeric	Кількість нейтральних твітів
time	integer	Час завершення періоду представлений у форматі Unix

В результаті проведення описаних операцій зі збору та обробки інформації було створено необхідний набір даних, який буде в подальшому використовуватися компонентами АМН для навчання та аналізу методів прогнозування курсу криптовалют.

2.3. Проектування архітектури системи

Ефективність впровадження системи залежить від правильності вибору підходів та розробки архітектури системи. Процес аналізу методів прогнозування курсу криптовалют включає в себе взаємодію різних програмних компонентів та технологій задля досягнення необхідної ефективності та продуктивності. Дана система повинна забезпечувати декілька функцій, які хоч і є взаємозалежними, проте вимагають різних засобів та технологій для їх впровадження. Через ці особливості необхідно було провести декомпозицію усієї системи на певні

компоненти, кожний з яких буде відповідати за виконання чітко визначеного набору завдань. В нашому вебсервісі важлива ефективність процесу аналізу, тому необхідним є заміна універсального набору технологій на більш вузькопрофільні задля досягнення кращої продуктивності. Отже, кожний компонент був спроектований відповідно до поставленого перед ним завдання. Схема взаємодії цих компонентів наведена на рисунку 2.6.

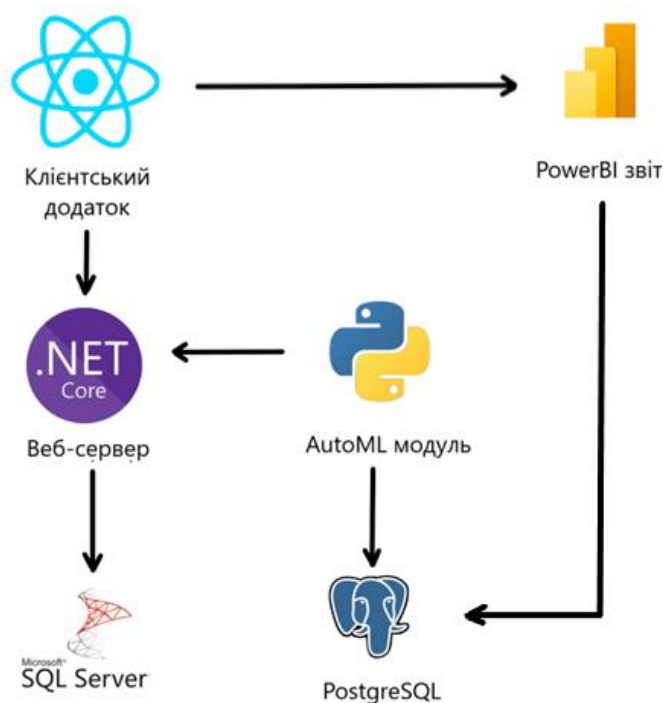


Рис. 2.6. Схема архітектури системи

Клієнтський додаток є єдиним компонентом з яким взаємодіє користувач. За допомогою нього можна налаштовувати кожний алгоритм відповідно до певних задач і потреб, проводити аналіз різноманітних методів, розраховувати потенційну ефективність кожного з них, а також здійснювати операції необхідні для конфігурації та контролю конвеєру AutoML. Даний компонент є вебдодатком доступ до якого користувачі отримують через браузер.

Вебсервер відіграє важливу роль у взаємодії всіх компонентів. Він обробляє усі запити, які надходять від клієнтської частини та зберігає інформацію у БД (SQL Server). Саме сервер відповідає за збереження інформації про створені та

налаштовані алгоритми й дозволяє користувачу конфігурувати різноманітні варіації моделей нейронних мереж. Окрім цього, вебсервер також надає інформацію спеціальному компоненту – AutoML модулю. Це дозволяє останньому отримувати актуальні дані про створені або оновлені алгоритми й відповідно виконувати конвеєр АМН для них.

AutoML модуль є основною частиною системи, що виконує всі етапи конвеєру й генерує результати. Ця результати в подальшому використовуються для порівняльного аналізу різних методів прогнозування курсу криптовалют. Згаданий компонент системи розподіляється на відповідні модулі, які відповідають за наступні завдання:

1. Збір даних.
2. Обробка даних і формування наборів для навчання, валідації та тестування.
3. Створення моделей прогнозування.
4. Навчання моделей прогнозування.
5. Генерація прогнозів готовими моделями для певного проміжку даних.

Збір даних відбувається відповідними методами, кожен з яких відповідає за певний тип або джерело даних. У нашій системі можна виділити наступні джерела даних: API, блокчейн та індексовані БД. Кожний тип отриманої інформації опрацьовується для того, щоб стандартизувати загальний масив даних для кожного часового проміжку та криптовалюти, що підтримуються системою.

Обробка даних включає в себе стандартизацію та нормалізацію. Перш за все, необхідно масштабувати дані, щоб певні їх параметри, які мають багатопорядкові значення через свої особливості, не мали занадто великий вплив на результати навчання та прогнозування. Додатково виконується нормалізація за першим елементом, що дозволяє представляти дані у вигляді їх змін відносно початкового елемента замість абсолютних значень. Також цей модуль відповідає

за формування наборів даних в залежності від типу обраного алгоритму.

Створення моделі прогнозування включає в себе програмне створення та ініціалізацію параметрів моделі відповідно до обраних користувачем налаштувань. Це досягається шляхом динамічного підставлення шаблонів необхідних алгоритмів, кожен з яких був впроваджений таким чином, щоб їх можна було змінювати у процесі виконання програми.

Навчання моделі відбувається з застосуванням методів оптимізації та використанням вбудованих у згадані шаблони функцій або за допомогою передачі об'єкту шаблону у функції відповідної бібліотеки. В результаті навчання відбувається оптимізація моделі з метою зменшення похибки прогнозування.

Генерація прогнозів відбувається відповідним модулем, який використовує навчені моделі для прогнозування значень на тестовому наборі даних. Окрім цього, отримані значення порівнюються з цільовими й на основі них вираховуються метрики для навченого алгоритму. Отримані прогнози, а також результати ефективності прогнозування зберігаються у базу даних.

AutoML компонент має окрему БД (PostgreSQL), де зберігаються усі дані, які використовуються для навчання алгоритмів прогнозування курсу криптовалют. Таке рішення було прийняте для більшої швидкодії при роботі з великими обсягами даних. Також у цьому сховищі зберігається інформація про результати прогнозування кожного з опрацьованих алгоритмів. Вона у подальшому трансформується окремим програмним модулем у певну структуру, яка може використовуватися аналітичними системами.

PowerBI – елемент системи, який надає можливість графічно представити як різноманітні види даних, так і отримані результати алгоритмів. Для цього був створений відповідний звіт на основі трансформованих даних. Цей звіт розгортається відповідним сервісом і вбудовується у наш клієнтський додаток для розширення можливостей аналізу різних методів прогнозування та дослідження

зібраних даних.

2.4. Розробка архітектури аналітичного компонента системи

Кожна система певним чином взаємодіє з даними виконуючи певні операції над ними. В залежності від типу даних система повинна мати доступ до них у відповідному форматі для забезпечення можливості виконання всіх необхідних операцій й при цьому вона має відповідати нефункціональним вимогам відповідного програмного забезпечення. Виділяють два підходи до організації та обробки даних в інформаційних системах: OLTP і OLAP.

OLTP (Online Transaction Processing) [35] – підхід до роботи з базою даних, при якому система виконує невеликі за розміром операції, які називаються транзакціями. Транзакції надходять й опрацьовуються паралельно, що зумовлює вищу відмовостійкість та швидкодію таких системи. OLTP-застосунки використовуються для різних задач у багатьох галузях. Дані системи проектуються, налаштовуються і оптимізуються задля ефективної обробки великої кількості транзакцій за короткі проміжки часу. Забезпечення згаданих переваг потребує дотримання відповідних вимог даного виду систем:

- нормалізованість моделей даних;
- обробка даних у режимі реального часу;
- дотримання принципів ACID [36].

Проте, даний вид систем має і свої недоліки. OLTP-системи оптимізуються для невеликих дискретних транзакцій, що негативно впливає на їх продуктивність при виконанні запитів для отримання комплексної інформації, що поєднує дані з різних таблиць. Даний недолік зумовлює надання переваги OLAP-системам замість OLTP при необхідності аналізувати великі масиви складних даних.

OLAP (Online Analytical Processing) [37] система відрізняється від попереднього типу тим, що надає можливість переглядати багатовимірні дані у

режимі реального часу, тобто без довгого очікування на результат запиту. Основною причиною появи даного виду систем був саме недолік OLTP виражений у неможливості швидкого отримання відповідей на різноманітні аналітичні запити.

OLAP системи отримали описані переваги за допомогою спеціальної структури – просторової моделі. Дана структура не є ефективною для внесення змін (обробки великої кількості транзакцій), але є чудовою для виконання складних запитів для отримання інформації. Проте, зазвичай інформація у вебсистемах структурована відповідно до принципів OLTP, тож це вимагає додаткової обробки даних у випадку необхідності їх використання для аналітики.

OLAP система потребує виконання ETL-процесу, за допомогою якого можна зробити миттєвий знімок реляційної БД (OLTP) та трансформувати дані у просторову модель. Саме з цією просторовою моделлю аналітична система здатна виконувати необхідні операції. Основною концепцією OLAP є багатовимірний куб [38], осями якого є числові або короткі лінгвістичні дані про предметну область.

OLAP-куб часто створюють за допомогою реляційних баз даних із застосуванням схем "зірка" або "сніжинка". В центрі "зірки" знаходиться таблиця, яка містить ключові факти відповідно до їх назв у сховищі. До таблиці фактів приєднується необхідна кількість таблиць-вимірів, які є "променями зірки" у схемі бази даних. Назви стовпчиків цих таблиць - це первинні дані на основі яких можуть виконуватися базові аналітичні операції.

В результаті збору даних з різних джерел та їх збереження у спільну БД, були отримані окремі таблиці для кожного типу інформації (рис. 2.7). Ці дані були збережені відповідно до вимог OLTP систем, тобто представляють собою нормалізовані моделі даних. Отже, їх необхідно трансформувати відповідно до описаних правил для можливості використання збереженої інформації в аналітичних системах.

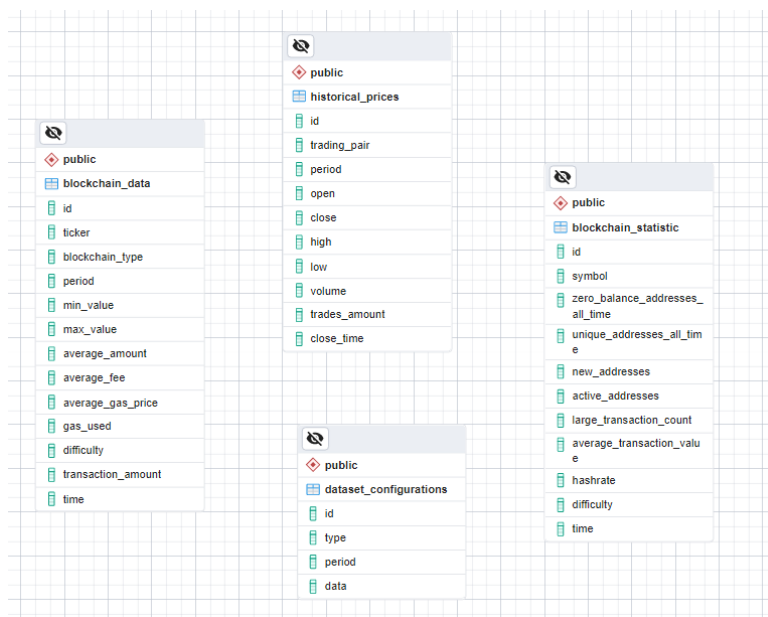


Рис. 2.7. Діаграма структури завантажених даних

Дотримуючись правил створення OLAP-кубу, в системі був впроваджений відповідний модуль трансформації. Він виконує обробку даних, які були зібрані для аналізу, а також результатів прогнозування. Цей процес включає в себе створення реплікації та виконання трансформації БД, що дозволяє перетворити початкову структуру даних у формат, який підтримується OLAP системами.

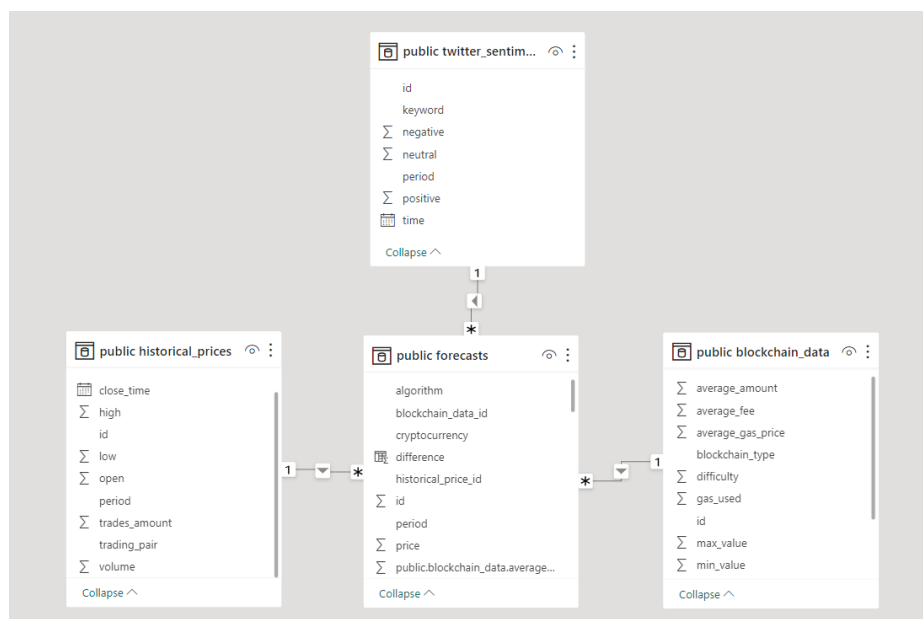


Рис. 2.8. Результат трансформації БД відповідно до схеми “зірка”

В результаті розробки та виконання згаданого модулю, вихідну структуру даних було перетворено (див. рис. 2.8) відповідно до визначених вимог, що дозволяє використовувати їх у аналітичному компоненті системи.

Висновки до другого розділу

У даному розділі було проведено проектування системи аналізу методів прогнозування курсу криптовалют. У нашому програмному забезпеченні аналіз та порівняння ефективності відбуватиметься за допомогою АМН. Задля цього був розглянутий основний підхід, який застосовується у системах такого типу й було спроектовано архітектуру вебсервісу з врахуванням отриманої інформації та особливостей нашого завдання.

Одним з основних процесів майже для усіх видів алгоритмів машинного навчання є збір даних. З цією метою було проведено дослідження кожного джерела інформації, визначено необхідні засоби для отримання доступу до них та розроблено програмні методи, які дозволяють впровадити процес їх отримання у нашій системі.

Планування архітектури програмного забезпечення дозволяє виокремити компоненти системи та визначити засоби їх взаємодії. Правильне проектування вебдодатку дозволяє ефективно використовувати обчислювальні ресурси й при цьому досягати високої продуктивності. У нашій інформаційній системі весь процес аналізу був поділений на підпроцеси й для кожного з них були обрані найоптимальніші засоби реалізації.

Об'єктивне порівняння різних методів прогнозування вартості криптовалют вимагає розробки та використання різноманітних засобів для їх аналізу та порівняння. Для цього був проведений аналіз особливостей структури даних підтримуваних OLAP системами й створений відповідний модуль для трансформації існуючих даних з метою їх подальшого використання у

аналітичному компоненті.

Отже, в результаті було повністю спроектовано систему аналізу методів прогнозування курсу криптовалют. Це включало планування як системи в цілому, так і певних процесів, які є невід'ємною її частиною. Було спроектовано конвеєр AutoML, налаштовано процес збору інформації, визначено задачі та особливості кожного програмного модулю, а також досліджено використання аналітичної системи й трансформовано дані відповідно до її вимог.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ АНАЛІЗУ МЕТОДІВ ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ

3.1 Вибір та обґрунтування засобів реалізації

Зазвичай розробка будь-якого програмного забезпечення починається з вибору засобів реалізації. Саме від правильності вибору та поєднання обраних засобів залежить ефективність системи під час виконання поставлених завдань. Система аналізу методів прогнозування курсу криптовалют складається з декількох компонентів, тому варто детально розглянути технології, які були використанні для впровадження кожного з них.

Одним з основних елементів системи є вебдодаток. За допомогою цього додатку користувач може налаштовувати різні варіації алгоритмів прогнозування курсу криптовалют та проводити їх порівняльний аналіз на основі отриманих результатів. Дана підсистема була реалізована на основі клієнт-серверної архітектури. Клієнтська частина буде створена за допомогою бібліотеки React (рис. 3.1) з використанням Redux (рис. 3.2) для керування станом додатку.

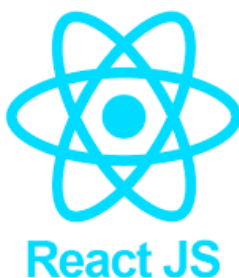


Рис. 3.1. Бібліотека React для розробки клієнтської частини



Рис. 3.2. Бібліотека Redux для керування станом додатку

Обробка запитів між клієнтським додатком та сервером була реалізована з використанням бібліотеки RxJs (рис. 3.3).



Рис. 3.3. Бібліотека RxJs

Серверна частина була впроваджена як вебсервіс за допомогою використання фреймворку .NET Core (рис. 3.4). Для взаємодії з базою даних використовувалася структура об'єктно-реляційного відображення EntityFramework.



Рис. 3.4. Фреймворк .NET Core

Компонент системи, який виконував усі спроектовані етапи конвеєру АМН був розроблений за допомогою мови програмування Python (рис. 3.5).



Рис. 3.5. Мова програмування Python

Для збору різних видів даних використовувалися бібліотеки Web3, binance-

connector та snsrape.

Web3 використовувалася для взаємодії з блокчейнами Ethereum та BSC. Вона базується на використанні RPC для роботи зі згаданими мережами. Особливостями цієї бібліотеки є те, що вона реалізує велику кількість функцій взаємодії з блокчейнами й дозволяє виконувати різноманітні операції – від читання даних до створення власних децентралізованих додатків.

Бібліотека binance-connector призначена для взаємодії з Binance API у додатках написаних мовою Python. Вона спрощує процес отримання даних зі згаданої платформи та виконання різних операцій, що робить її ефективним інструментом автоматизації.

Snsrape – популярна бібліотека, яка використовується для отримання та збору даних з соціальних мереж. Завдяки своїм особливостям вона значно спрощує процес пошуку та фільтрації даних, що робить її корисною у разі необхідності проводити сентиментальний аналіз. У нашій роботі також був використаний інструмент nltk у поєднанні з цією бібліотекою. Він застосовувався для роботи з природною мовою, що, як результат, дозволило аналізувати емоційне забарвлення отриманих даних й робити висновки про зміну відношення суспільства до певної криптовалюти або токєну.

Наша система підтримує велику кількість алгоритмів та архітектур нейронних мереж, що зумовлює використання значної кількості бібліотек для їх реалізації та навчання. Список використаних бібліотек та задач які для цього були використані виглядає наступним чином:

- scikit – створення та навчання моделей лінійної регресії та Random Forest, а також автоматичний підбір оптимальних параметрів для алгоритмів на основі дерев прийняття рішень;
- xgboost – створення та навчання методів прогнозування на основі архітектури XG Boost;

- statsmodels – бібліотека для статистичного аналізу даних шляхом створення моделі ARIMA;
- tensorflow – використовується для створення глибоких нейронних мереж й за допомогою неї були створені моделі прогнозування, які використовують LSTM.

Для взаємодії між БД та компонентом АМН використовувалася бібліотека psycopg2. Вона дозволяє виконувати різноманітні види операцій та повністю підійшла для наших завдань, незважаючи на великі обсяги даних та їх специфіку.

Отримані результати порівняння методів прогнозування курсу криптоактивів після опрацювання та трансформації відповідно до вимог аналітичних систем використовуються для створення звітів у Power BI (рис. 3.6) для наочного представлення результатів аналізу. Створення звіту та його публікація відбувалися за допомогою додатку Power BI Desktop.



Рис. 3.6. Інструмент Power BI для інтерактивної візуалізації даних

3.2 Проектування та розробка компонентів системи

Як було зазначено у попередніх розділах, система аналізу методів прогнозування курсу криптовалют складається з 3 основних компонентів: клієнтський додаток, серверний додаток та компонент, який реалізовує конвеєр АМН. Виокремлення цих компонентів відбувалося з врахуванням їх особливостей та задач, які вони мають виконувати.

Існують різноманітні архітектури, які використовуються при розробці вебдодатків. Кожна має свої переваги й недоліки, тому варто проводити самостійний аналіз та вибір архітектурного рішення в залежності від поставленого завдання. Для впровадження вебсервісу, розглядалися наступні архітектури:

- MVC;
- MVVM у поєднанні з Web API;
- Flux [39] у поєднанні з Web API.

Після проведення порівнянь згаданих варіантів та аналізу визначених вимог було обрано Flux у поєднанні з Web API. У нашій системі цей компонент є відносно простим й не вимагає реалізації складної логіки, так як вона відбувається або на вебсервері або у компоненті AutoML. Додаток користувача використовується тільки для налаштування алгоритмів, відображення вже готової інформації та аналізу результатів прогнозування й ефективності досліджуваних методів. Отже, важливими метриками для цього компоненту є зручність використання та швидкодія, що чудово забезпечується обраною архітектурою.

Flux – це архітектура, що застосовується у вебдодатках й основна її ідея полягає у використанні компонентів React з односпрямованим потоком даних (рис. 3.7).

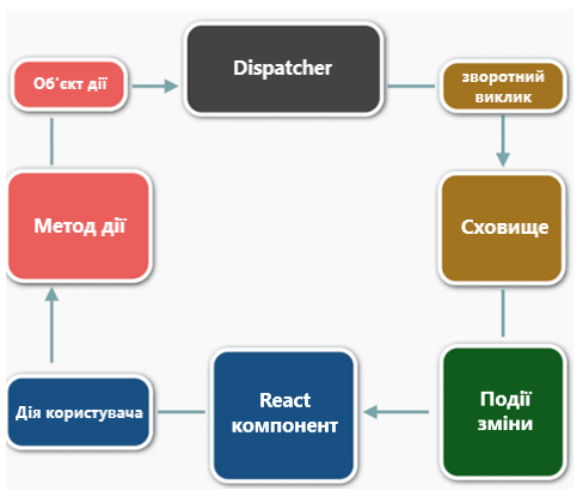


Рис. 3.7. Односпрямований потік даних у Flux додатках

Основними перевагами Flux є:

- ефективна декомпозиція структури та логіки додатку;
- простий і надійний підхід до передачі даних;
- структурований та організований програмний код;
- можливість повторного використання створених компонентів;
- вища продуктивність роботи додатку.

Даній архітектурі є властивий певний процес передачі даних, який складається зі заздалегідь визначених етапів. Кожний елемент має певні вимоги й їх реалізація має відповідати принципам Flux з метою дотримання односторонньої передачі інформації.

React-компоненти (React Views) – являють собою користувацький інтерфейс та елементи управління з якими взаємодіє користувач. Будь-яка взаємодія, яка вимагає зміни стану додатку вимагає використання методів дії. Методи дії (Actions) – допоміжні методи, що створюють об'єкти з певними параметрами та типом й передають їх у Dispatcher. Він, у свою чергу, за допомогою зворотних викликів передає створенні об'єкти у сховище (Store). Ці зворотні виклики заздалегідь реєструються у сховищі, отже воно може оновлювати дані після отримання об'єкту й генерувати події зміни. В той же час, React компоненти слухають ці події й отримують змінені дані зі сховища. Як результат, оновленими даними забезпечується все дерево компонентів.

Серверна частина, яка являє собою Web API, була спроектована з використанням багатошарової архітектури [40]. Ця архітектура значно спрощує процеси розробки та підтримки програмного забезпечення. Основна її концепція полягає у тому, що кожен її шар відповідає за виконання певного окремого виду операцій. Шари взаємодіють між собою, проте їх зони відповідальності повинні бути чітко розподілені й не пересікатися. Це дозволяє розробникам змінювати будь-який рівень без небажаного впливу на інші шари додатку.

Основними структурними елементами (рис. 3.8) даної архітектури є:

- контролер (Controller) - відповідає за обробку запитів, які надходять з клієнтської частини додатку або системи АМН;
- менеджер (Manager) – складається з методів, які реалізують бізнес-логіку додатку;
- провайдер (Provider) – відповідає за операції з даними й взаємодіє з базою даних за допомогою ORM;
- модель-сутність (Entity model) – модель, яка являє собою сутність в базі даних і використовується при операціях з даними між сервером та базою даних;
- модель-прив'язка (Binding model) – модель, яка використовується в контролері для отримання даних з запитів, їх валідації та обробки.



Рис. 3.8. Взаємодія компонентів багатoshарової архітектури

Архітектура компоненту АМН представляє собою набір послідовно з'єднаних модулів:

- модулі завантаження та обробки даних;
- модуль для роботи з базою даних;
- модулі для створення, ініціалізації та навчання моделей нейронних мереж;

- модуль прогнозування та вимірювання метрик ефективності алгоритмів;
- модуль комунікації між компонентом АМН та вебсервером;
- модуль трансформації структури даних для її подальшого використання у аналітичній системі.

Підтримка системою різних архітектур методів прогнозування, які відрізняються один від одного, ускладнювало процес автоматизації. З цією метою було розроблено відповідну структуру класів з використанням наступного базового класу:

```
class Network:
    def __init__(self, algorithm):
        self.algorithm = algorithm

    @abstractmethod
    def build(self, train_data):
        pass

    @abstractmethod
    def predict(self, test_data):
        pass

    @abstractmethod
    def pack_params(self):
        pass
```

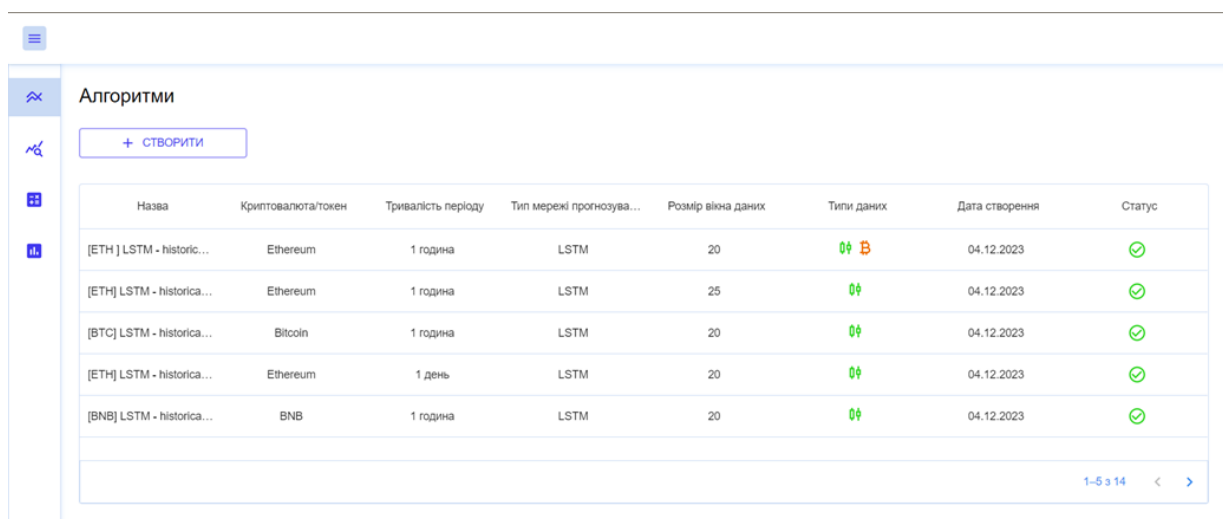
На основі цього класу було впроваджено усі архітектури нейронних мереж, які доступні у системі, що зробило рішення більш універсальним та спростило процес додавання нових типів алгоритмів за потреби.

3.3 Тестування розробленої системи

Результатом виконання кваліфікаційної роботи є створення системи аналізу

методів прогнозування курсу криптовалют. Взаємодія між користувачем та системою відбувається через клієнтський додаток, тож за допомогою нього можна протестувати створене програмне забезпечення шляхом виконання процесу аналізу алгоритму прогнозування від його створення до порівняння результатів.

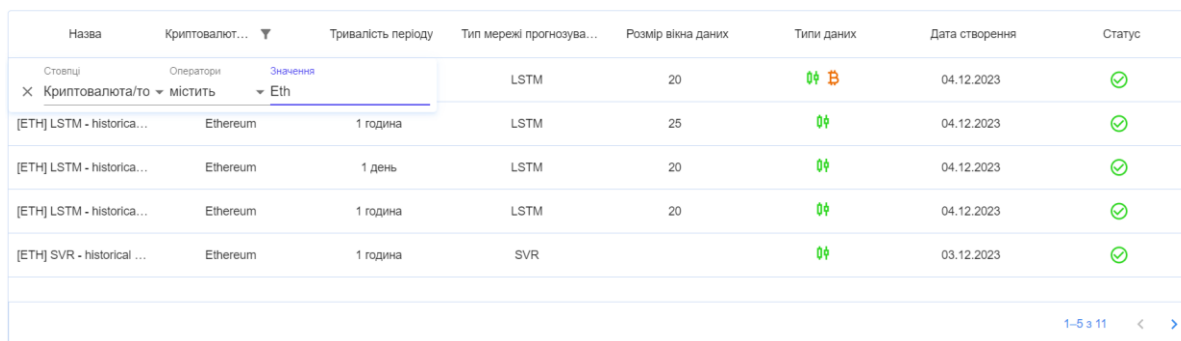
Відкривши головну сторінку додатку ми можемо побачити список усіх створених методів прогнозування (рис. 3.9). На цій сторінці користувач має змогу переглядати певну інформацію про ці алгоритми, а також статус їх обробки компонентом АМН.



Назва	Криптовалюта/токен	Тривалість періоду	Тип мережі прогнозува...	Розмір вікна даних	Типи даних	Дата створення	Статус
[ETH] LSTM - historica...	Ethereum	1 година	LSTM	20	0% B	04.12.2023	✓
[ETH] LSTM - historica...	Ethereum	1 година	LSTM	25	0%	04.12.2023	✓
[BTC] LSTM - historica...	Bitcoin	1 година	LSTM	20	0%	04.12.2023	✓
[ETH] LSTM - historica...	Ethereum	1 день	LSTM	20	0%	04.12.2023	✓
[BNB] LSTM - historica...	BNB	1 година	LSTM	20	0%	04.12.2023	✓

Рис. 3.9. Список створених методів прогнозування

Також цей список надає можливість фільтрації та сортування елементів за певними параметрами відповідно до потреб користувача (рис. 3.10). Це спрощує процес пошуку потрібних методів та дозволяє швидко отримати список усіх створених у системі алгоритмів за вказаними критеріями.



Назва	Криптовалют... ▼	Тривалість періоду	Тип мережі прогнозува...	Розмір вікна даних	Типи даних	Дата створення	Статус
Стовпці	Оператори	Значення	LSTM	20	0% B	04.12.2023	✓
X Криптовалюта/то	Містить	Eth					
[ETH] LSTM - historica...	Ethereum	1 година	LSTM	25	0%	04.12.2023	✓
[ETH] LSTM - historica...	Ethereum	1 день	LSTM	20	0%	04.12.2023	✓
[ETH] LSTM - historica...	Ethereum	1 година	LSTM	20	0%	04.12.2023	✓
[ETH] SVR - historical ...	Ethereum	1 година	SVR		0%	03.12.2023	✓

Рис. 3.10. Функціонал фільтрації елементів списку

Для створення нового алгоритму необхідно натиснути на кнопку “Створити”, яка розташована над списком. Після цього користувач перенаправляється на сторінку створення та налаштування нового методу прогнозування (рис. 3.11). Процес додавання нового алгоритму складається з 3 етапів:

1. Вибір архітектури нейронної мережі, тривалості періоду, криптовалюти та певних інших параметрів.
2. Вибір даних, які буду використовуватися при навчанні створених моделей та вимірюванні їх ефективності.
3. Налаштування параметрів моделі нейронної мережі (залежить від обраної архітектури).

Рис. 3.11. Сторінка створенні та налаштування методу прогнозування

Етап вибору даних полягає у визначенні певних параметрів зібраної інформації, що будуть використовуватися у алгоритмі. Вони розподілені у 3 групи відповідно до досліджуваних типів даних, які можуть застосовуватися при прогнозуванні вартості криптоактивів (рис. 3.12). Це дозволяє користувачу не просто обрати категорії даних, які він хоче застосувати у створюваному алгоритмі, а й використати певні поля й параметри, які він вважає найбільш ефективними для вирішення задачі прогнозування. Вибір параметрів даних доступний для усіх архітектур нейронних мереж, окрім ARIMA. Це спричинено тим, що даний вид алгоритмів не підтримує багатовимірні дані.

Вибір даних

Аналіз історичної ціни

- ☒ Ціна відкриття
- ☒ Ціна закриття
- ☒ Найвища ціна
- ☒ Найнижча ціна
- ☐ Об'єм торгів
- ☐ Кількість операцій обміну

Аналіз блокувину

- ☐ Мінімальна сума транзакції
- ☐ Максимальна сума транзакції
- ☐ Середня сума транзакції
- ☐ Середня комісія транзакцій
- ☐ Середня ціна газу
- ☐ Кількість використаного газу
- ☐ Складність мережі
- ☐ Середня кількість транзакцій у блоці

Аналіз Twitter

- ☐ Частка позитивних твітів
- ☐ Частка негативних твітів
- ☐ Частка нейтральних твітів

НАЗАД ДАЛІ

Конфігурація мережі

Рис. 3.12. Етап вибору параметрів даних

Останнім етапом створення алгоритму є налаштування нейронної мережі. Як було досліджено у попередніх розділах, кожна архітектура має свої певні особливості та параметри можливість налаштування яких має бути забезпечена системою. Отже, кожний з них має свій набір налаштувань, що відображається в залежності від обраного типу алгоритму.

Налаштування алгоритму лінійної регресії включає в себе вибір параметрів оптимізації прямої (рис. 3.13).

Створення алгоритму

← Налаштування алгоритму

✓ Вибір даних

1 Конфігурація мережі

2 Створення алгоритму

C параметр 1

Еррой параметр 0.1

Тол параметр 0.001

☐ Використовувати параметри за замовчуванням

НАЗАД СТОРИТИ

Рис. 3.13. Налаштування параметрів моделі лінійної регресії

На основі проведеного дослідження моделі XG Boost були визначені найбільш ефективні її параметри, можливість налаштування яких була додана у вебдодаток (рис. 3.14).

Рис. 3.14. Налаштування параметрів моделі XG Boost

ARIMA є статистичною моделлю, тому важливою є можливість зміни параметрів усіх її компонентів у клієнтському додатку (рис. 3.15).

Рис. 3.15. Налаштування параметрів моделі ARIMA

Random Forest є також ансамблевим методом, проте він має власний набір параметрів, які були визначені у процесі його дослідження (рис. 3.16).

Рис. 3.16. Налаштування параметрів моделі Random Forest

Архітектура LSTM є видом глибокої нейронної мережі, тому важливим для її ефективного навчання є вибір параметрів, які дозволяють контролювати процес оптимізації моделі прогнозування (рис. 3.17).

Рис. 3.17. Налаштування параметрів моделі LSTM

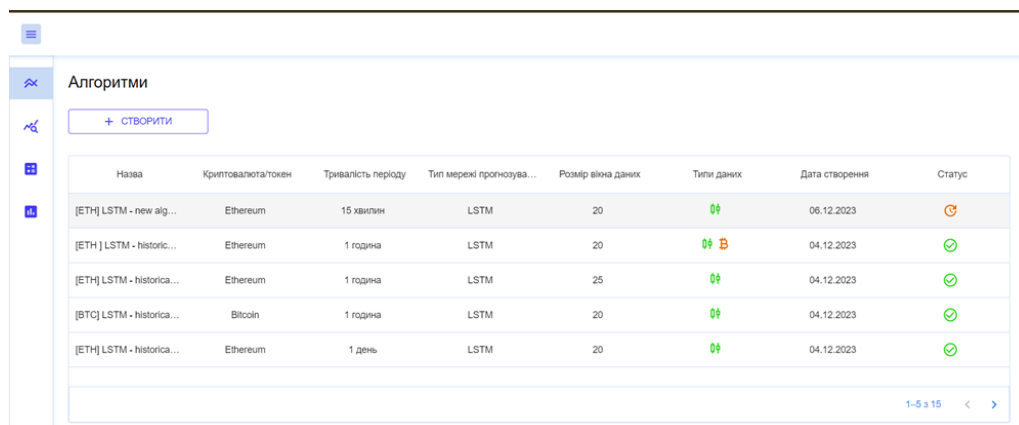
LSTM, на відміну від інших архітектур нейронних мереж, є алгоритмом глибоких нейронних мереж. Ця особливість свідчить про можливість та важливість процесу налаштування шарів цієї мережі та їх параметрів. Через це для даного виду архітектури було реалізовано спеціальний конфігуратор (рис. 3.18).

Рис. 3.18. Конфігуратор структури мережі LSTM

Створений конфігуратор надає можливість додавати, налаштовувати або

видаляти шари нейронної мережі. Цей функціонал надає гнучкості застосуванню даного виду алгоритмів й дозволяє порівнювати різні варіації структури глибинної нейронної мережі між собою.

Після вибору та налаштування усіх параметрів нового методу прогнозування він створюється й відображається у списку всіх алгоритмів (рис. 3.19). Звернувши увагу на створений елемент, можемо побачити що його статус свідчить про те, що він ще не був опрацьований компонентом AutoML.



Назва	Криптовалюта/токен	Тривалість періоду	Тип мережі прогнозува...	Розмір вікна даних	Типи даних	Дата створення	Статус
[ETH] LSTM - new alg...	Ethereum	15 хвилин	LSTM	20	0%	06.12.2023	✗
[ETH] LSTM - historic...	Ethereum	1 година	LSTM	20	0% B	04.12.2023	✓
[ETH] LSTM - historica...	Ethereum	1 година	LSTM	25	0%	04.12.2023	✓
[BTC] LSTM - historica...	Bitcoin	1 година	LSTM	20	0%	04.12.2023	✓
[ETH] LSTM - historica...	Ethereum	1 день	LSTM	20	0%	04.12.2023	✓

Рис. 3.19. Вигляд створеного алгоритму з відповідним статусом у списку

На цьому елемент створення нового методу прогнозування завершується. Далі розглянемо процес виконання автоматизованого навчання. Відповідний модуль АМН час від часу виконує запити на вебсервер для того, щоб отримати список створених або оновлених методів. Після цього кожний з отриманих методів прогнозування проходить всі етапи конвеєру. У нашому випадку ми створювали алгоритм на основі LSTM з власною будовою нейронної мережі. Створена будова була збережена у форматі JSON та отримана відповідним модулем, який відповідає за створення нейронної мережі. В результаті обробки отриманих даних та ініціалізації моделі створюється відповідний об'єкт, який представляє собою модель прогнозування. Інформація про структуру створеної глибинної нейронної мережі відображається у модулі АМН за допомогою чого ми можемо переглянути отриману будову мережі та впевнитися у правильності роботи описаного модуля (рис. 3.20).

```
Model: "model"
```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[None, 20, 4]	0
lstm (LSTM)	(None, 20, 256)	267264
dropout (Dropout)	(None, 20, 256)	0
lstm_1 (LSTM)	(None, 256)	525312
dropout_1 (Dropout)	(None, 256)	0
dense (Dense)	(None, 1)	257

```

Total params: 792833 (3.02 MB)
Trainable params: 792833 (3.02 MB)
Non-trainable params: 0 (0.00 Byte)

```

Рис. 3.20. Структура створеної мережі

Після етапу створення відбувається фаза оптимізації параметрів моделі у процесі навчання. Всі етапи цього процесу є автоматизованими, тому він відбувається відповідно до вказаних налаштувань без потреби втручання користувача (рис. 3.21).

```

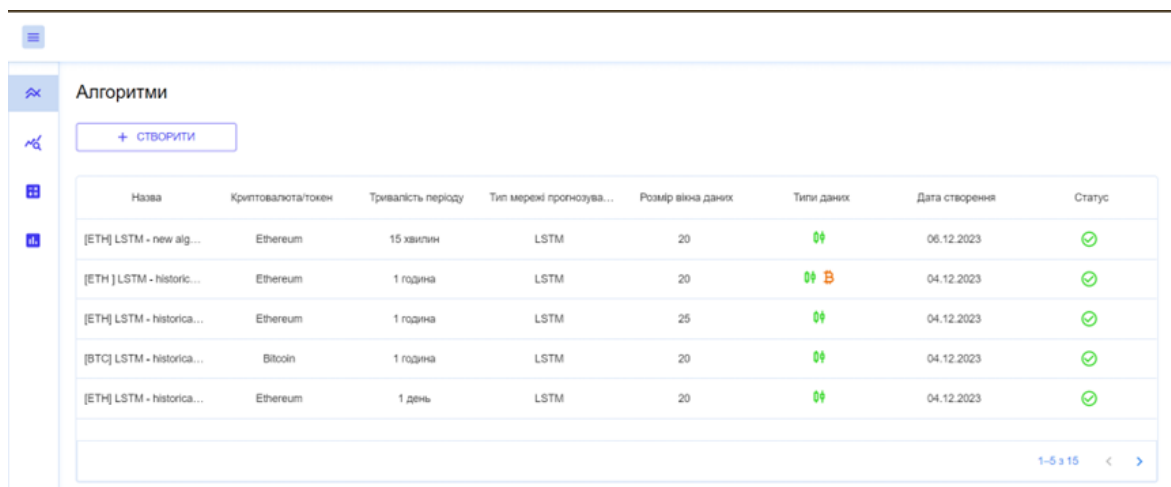
74/74 [=====] - 8s 53ms/step - loss: 7.6296e-05 - val_loss: 2.4831e-05 - lr: 0.0010
Epoch 2/100
74/74 [=====] - 4s 53ms/step - loss: 2.8208e-05 - val_loss: 1.0546e-05 - lr: 0.0010
Epoch 3/100
74/74 [=====] - 3s 46ms/step - loss: 2.5726e-05 - val_loss: 1.4149e-05 - lr: 0.0010
Epoch 4/100
74/74 [=====] - 3s 45ms/step - loss: 2.6856e-05 - val_loss: 1.2734e-05 - lr: 0.0010
Epoch 5/100
74/74 [=====] - 4s 50ms/step - loss: 2.2550e-05 - val_loss: 7.6955e-06 - lr: 0.0010
Epoch 6/100
74/74 [=====] - 4s 49ms/step - loss: 2.4577e-05 - val_loss: 1.2841e-05 - lr: 0.0010
Epoch 7/100
74/74 [=====] - 3s 43ms/step - loss: 2.6199e-05 - val_loss: 1.5320e-05 - lr: 0.0010
Epoch 8/100
74/74 [=====] - 4s 51ms/step - loss: 2.0548e-05 - val_loss: 7.1246e-06 - lr: 0.0010
Epoch 9/100
74/74 [=====] - 5s 64ms/step - loss: 2.1414e-05 - val_loss: 8.3706e-06 - lr: 0.0010
Epoch 10/100
74/74 [=====] - 3s 42ms/step - loss: 2.0084e-05 - val_loss: 7.1474e-06 - lr: 0.0010
Epoch 11/100
74/74 [=====] - 3s 42ms/step - loss: 1.8699e-05 - val_loss: 9.7410e-06 - lr: 0.0010

```

Рис. 3.21. Навчання створеної моделі

Після завершення навчання відбувається етап прогнозування даних створеним алгоритмом, вимірюються необхідні метрики та завершується виконання конвеєру AutoML. Значення метрик відправляються на вебсервер, а статус алгоритму змінюється. Після навчання можемо побачити, що створений алгоритм відображається у списку з відповідним статусом - опрацьований

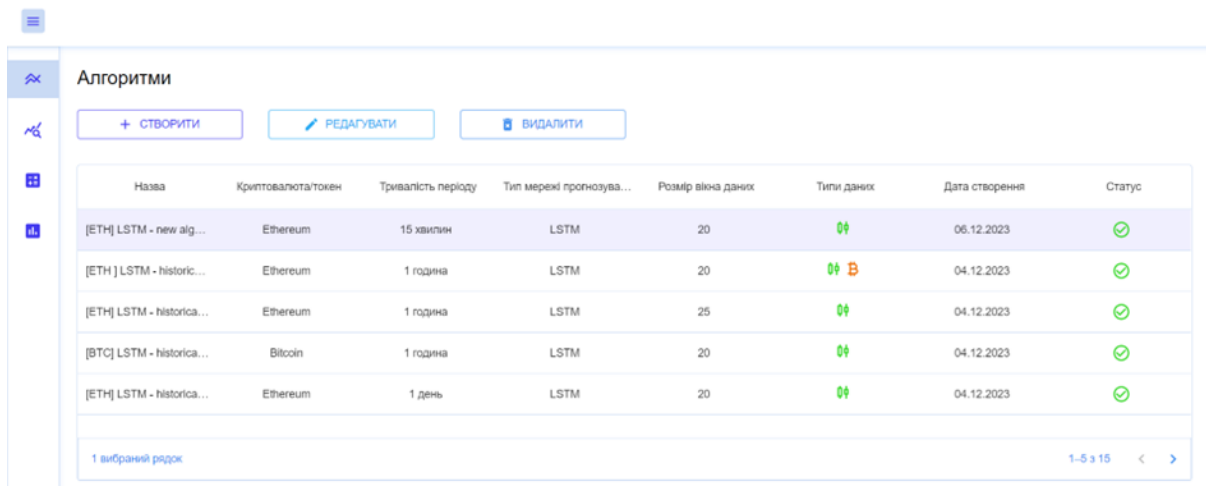
компонентом АМН (рис. 3.22).



Назва	Криптовалюта/токен	Тривалість періоду	Тип мережі прогнозува...	Розмір вікна даних	Типи даних	Дата створення	Статус
[ETH] LSTM - new alg...	Ethereum	15 хвилин	LSTM	20	00	06.12.2023	✓
[ETH] LSTM - historic...	Ethereum	1 година	LSTM	20	00 B	04.12.2023	✓
[ETH] LSTM - historica...	Ethereum	1 година	LSTM	25	00	04.12.2023	✓
[BTC] LSTM - historica...	Bitcoin	1 година	LSTM	20	00	04.12.2023	✓
[ETH] LSTM - historica...	Ethereum	1 день	LSTM	20	00	04.12.2023	✓

Рис. 3.22. Список алгоритмів після опрацювання компонентом АМН

За потреби користувач може редагувати або видаляти створені методи прогнозування. Для цього йому потрібно натиснути на рядок списку, який відноситься до потрібного алгоритму. Після цього з'являться додаткові кнопки “Редагувати” та “Видалити” (рис. 3.23).



Назва	Криптовалюта/токен	Тривалість періоду	Тип мережі прогнозува...	Розмір вікна даних	Типи даних	Дата створення	Статус
[ETH] LSTM - new alg...	Ethereum	15 хвилин	LSTM	20	00	06.12.2023	✓
[ETH] LSTM - historic...	Ethereum	1 година	LSTM	20	00 B	04.12.2023	✓
[ETH] LSTM - historica...	Ethereum	1 година	LSTM	25	00	04.12.2023	✓
[BTC] LSTM - historica...	Bitcoin	1 година	LSTM	20	00	04.12.2023	✓
[ETH] LSTM - historica...	Ethereum	1 день	LSTM	20	00	04.12.2023	✓

Рис. 3.23. Відображення елементів керування після вибору алгоритму

Обравши редагування, користувач буде перенаправлений на сторінку налаштування алгоритму, де він матиме змогу змінити параметри методу подібно до процесу створення (рис. 3.24). Статус алгоритму після оновлення зміниться, що дозволить системі АМН повторно опрацювати його з врахуванням оновлених параметрів.

Рис. 3.24. Сторінка редагування методу прогнозування

При видаленні елемента з'являється діалогове вікно з вимогою підтвердження операції, так як її не можна буде скасувати (рис. 3.25). Обраний метод прогнозування видалиться у разі підтвердження.

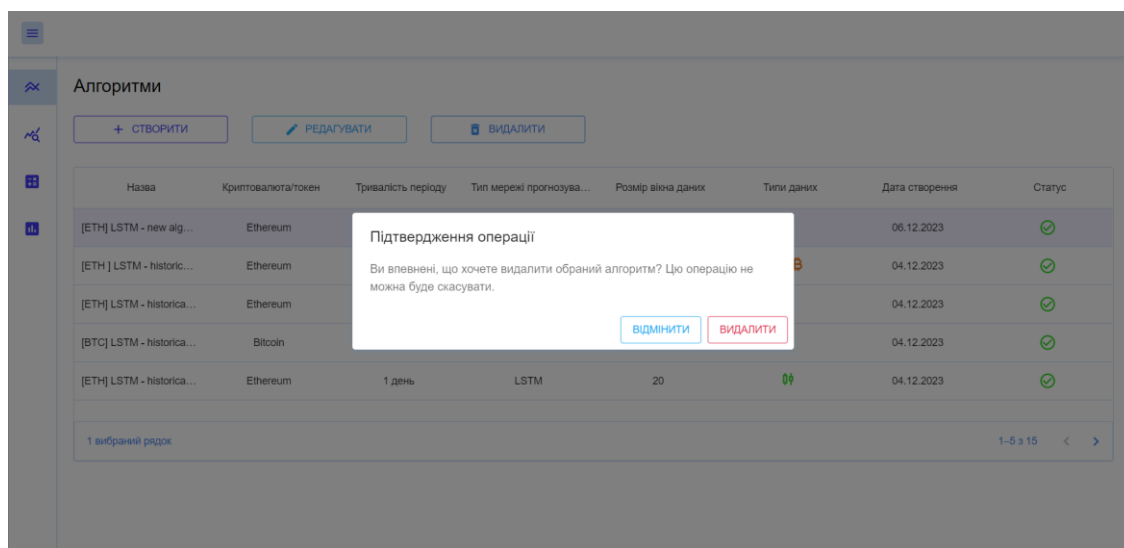
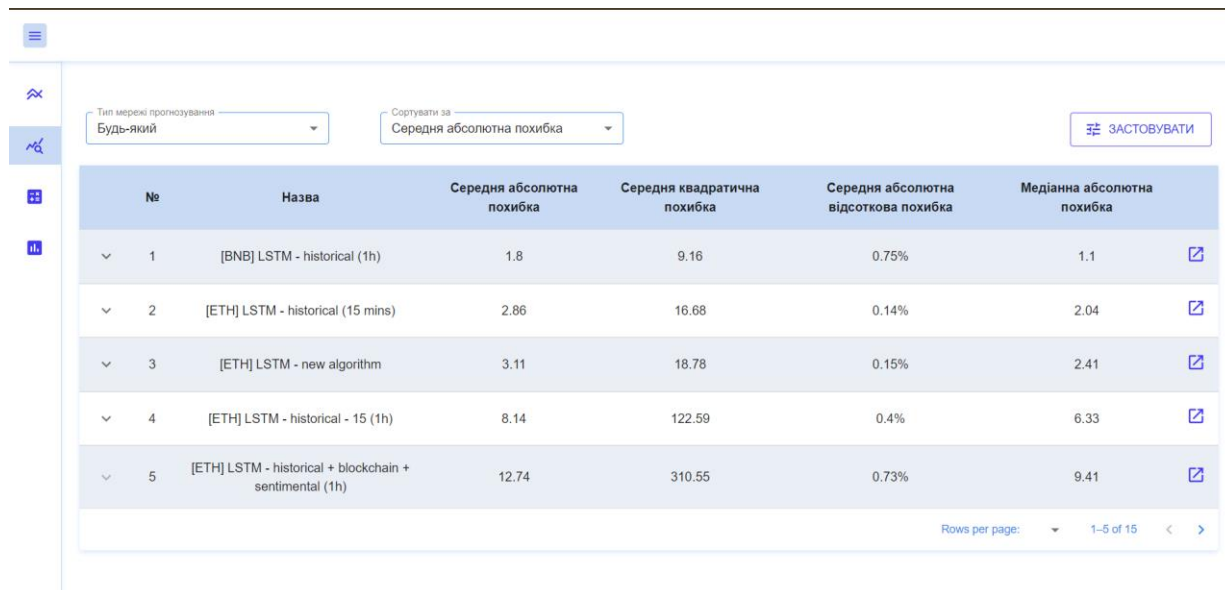


Рис. 3.25. Діалогове вікно при видаленні елемента

Створений метод прогнозування ціни криптовалют стає доступним для аналізу та порівняння після обробки модулем AutoML. У системі була реалізована можливість перегляду та сортування алгоритму за наступними метриками:

- середня абсолютна похибка;
- середня квадратична похибка;
- середня абсолютна відсоткова похибка;
- медіанна абсолютна похибка.

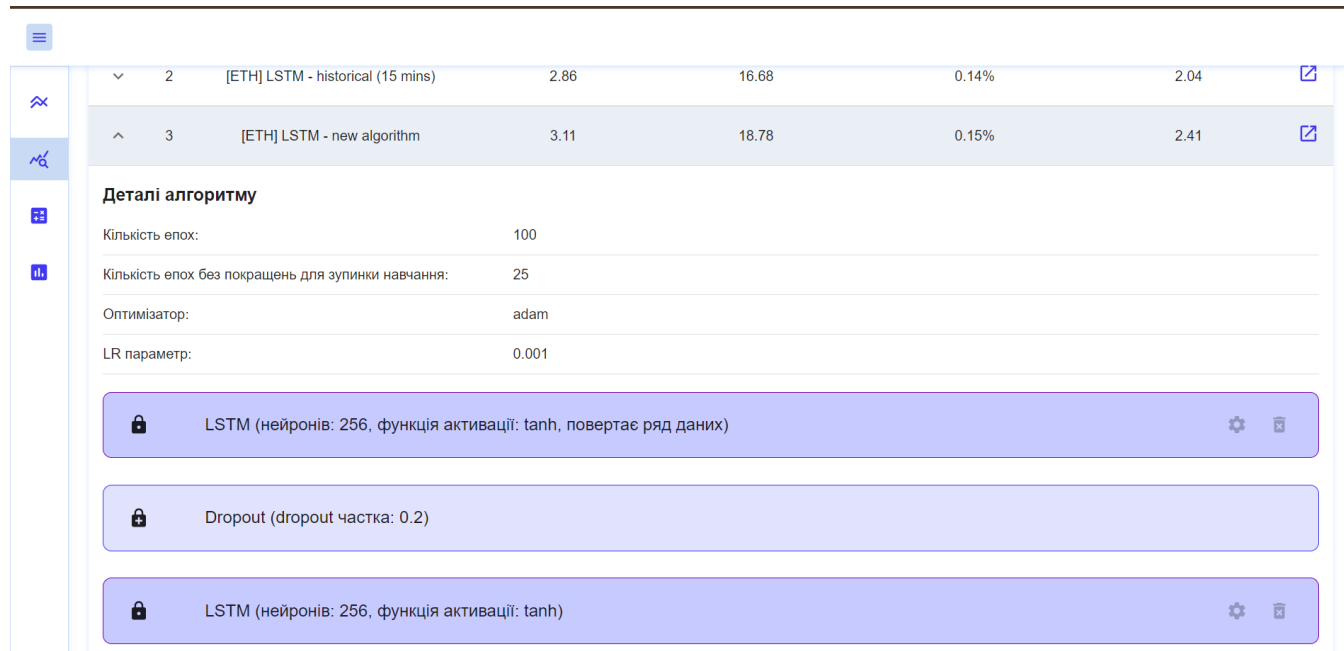
Ці дані представлені у вигляді таблиці й надають можливість фільтрації результатів за типом мережі (рис. 3.26) для зручності порівняння метрик ефективності подібних алгоритмів між собою.



№	Назва	Середня абсолютна похибка	Середня квадратична похибка	Середня абсолютна відсоткова похибка	Медіанна абсолютна похибка
1	[BNB] LSTM - historical (1h)	1.8	9.16	0.75%	1.1
2	[ETH] LSTM - historical (15 mins)	2.86	16.68	0.14%	2.04
3	[ETH] LSTM - new algorithm	3.11	18.78	0.15%	2.41
4	[ETH] LSTM - historical - 15 (1h)	8.14	122.59	0.4%	6.33
5	[ETH] LSTM - historical + blockchain + sentimental (1h)	12.74	310.55	0.73%	9.41

Рис. 3.26. Таблиця порівняння алгоритмів за визначеними метриками

У разі необхідності отримати більше інформації про параметри мережі, а також про будову нейронної мережі, користувач може розгорнути відповідний рядок таблиці (рис. 3.27).



№	Назва	Середня абсолютна похибка	Середня квадратична похибка	Середня абсолютна відсоткова похибка	Медіанна абсолютна похибка
2	[ETH] LSTM - historical (15 mins)	2.86	16.68	0.14%	2.04
3	[ETH] LSTM - new algorithm	3.11	18.78	0.15%	2.41

Деталі алгоритму

Кількість епох: 100

Кількість епох без покращень для зупинки навчання: 25

Оптимізатор: adam

LR параметр: 0.001

LSTM (нейронів: 256, функція активації: tanh, повертає ряд даних)

Dropout (dropout частка: 0.2)

LSTM (нейронів: 256, функція активації: tanh)

Рис. 3.27. Перегляд деталей про модель прогнозування

Після цього під елементом рядка з'явиться відповідний блок, який містить більш детальну інформацію. Також користувач може натиснути на піктограму у кінці рядку таблиці, що призведе до відкриття сторінки редагування обраного методу прогнозування у новій вкладці браузера. Це дозволяє переглянути усі можливі налаштування алгоритму й позбавляє необхідності шукати його у списку усіх наявних методів.

Окрім можливості порівняння алгоритмів між собою за метриками, у системі була релізована можливість детального аналізу результатів прогнозування за допомогою бек-тестингу та обчислення прибутковості для кожного алгоритму. Для цього виду дослідження необхідно обрати параметри певного методу прогнозування та перейти до його аналізу (рис. 3.28).

The screenshot shows a web interface for selecting an algorithm. On the left is a vertical sidebar with four icons: a menu icon, a line graph, a magnifying glass, and a plus sign. The main area has a header 'Оберіть алгоритм' (Select algorithm). Below it are three dropdown menus: 'Криптовалюта/токен' (Cryptocurrency/token) with 'Криптовалюта/токен' selected, 'Тривалість періоду' (Duration of period), and 'Алгоритм' (Algorithm). To the right of these is a button with a magnifying glass icon and the text 'дослідити' (research). Below the dropdowns is a large light blue box containing a blue icon of a hand pointing to a server rack, with the text 'Оберіть алгоритм для дослідження' (Select algorithm for research) below it.

Рис. 3.28. Форма вибору параметрів алгоритму для дослідження

Після вибору алгоритму користувач перенаправляється на сторінку його аналізу. На даній сторінці є можливість налаштування візуалізації та параметрів торгових операцій, що дозволяє порівняти реальні та прогнозовані дані, а також спрогнозувати можливий прибуток або втрати при використанні передбачень, які генеруються обраним алгоритмом (рис. 3.29).



Рис. 3.29. Сторінка бек-тестингу з можливістю аналізу та візуалізації

За допомогою інструментів налаштування користувач може пермикати вигляд відображення актуальної ціни (свічкова діаграма або проста лінія) (рис. 3.30), додавати параметри стоп-лосів (Stop Loss, SL) та тейк-профітів (Take Profit, TP), а також відображати їх на графіку (рис. 3.31). Цей інтерактивний елемент дозволяє провести глибший аналіз досліджуваного методу прогнозування, виявити сильні та слабкі сторони кожного алгоритму, а також перевірити ефективність цього алгоритму шляхом емуляції процесу трейдингу та застосування бек-тестингу.

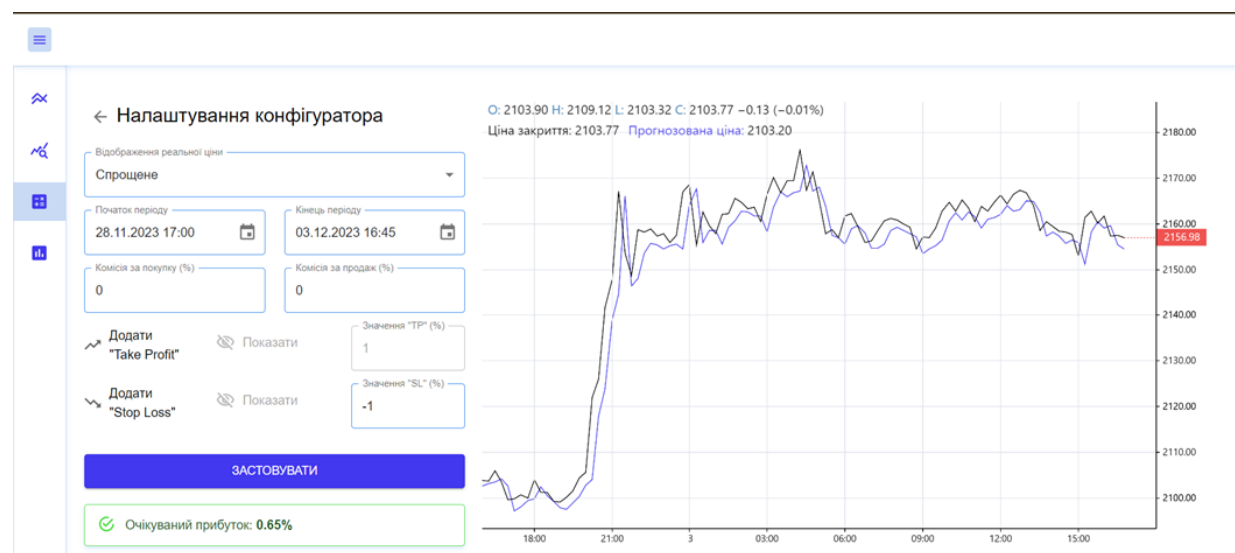


Рис. 3.30. Спрощене відображення реальної ціни

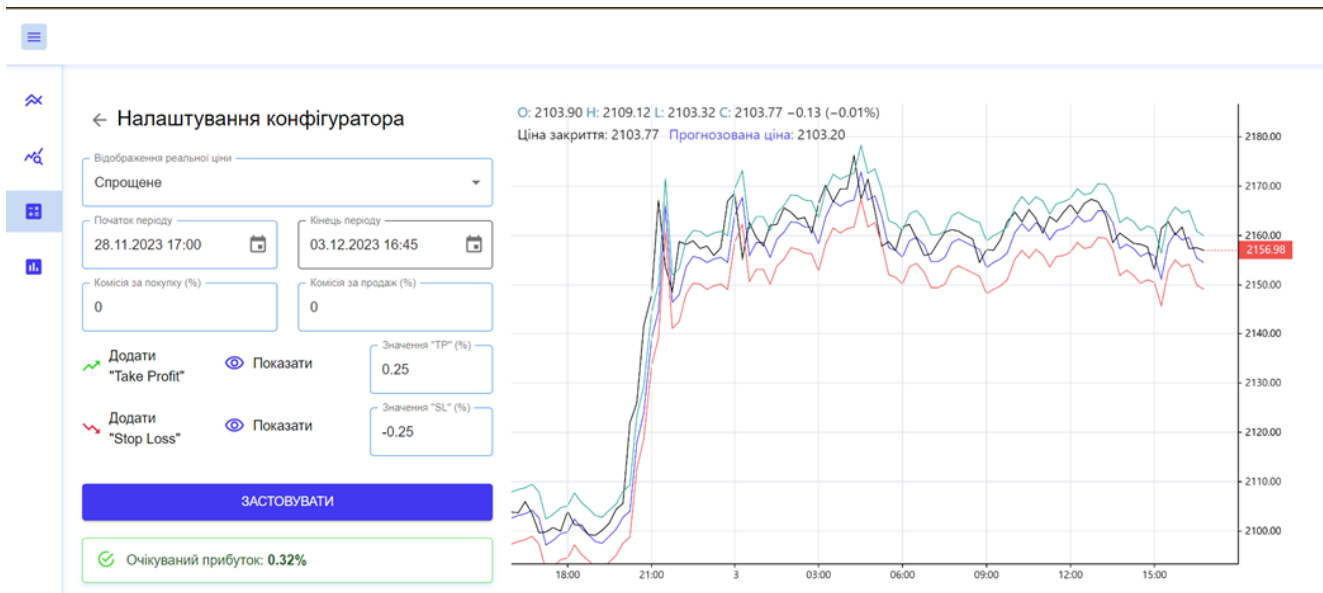


Рис. 3.31. Відображення SL та TP на графіку та їх застосування при розрахунку

Також його важливою перевагою є автоматичне обчислення потенційного прибутку або втрат, що у поєднанні з графічним відображенням інформації дозволяє визначити основні особливості створеного методу прогнозування й зробити висновки щодо можливості та доцільності його практичного використання.

Окрім цього, калькулятор надає змогу налаштовувати комісії біржі, які враховуються при обчисленні потенційних прибутків або втрат (рис. 3.32).

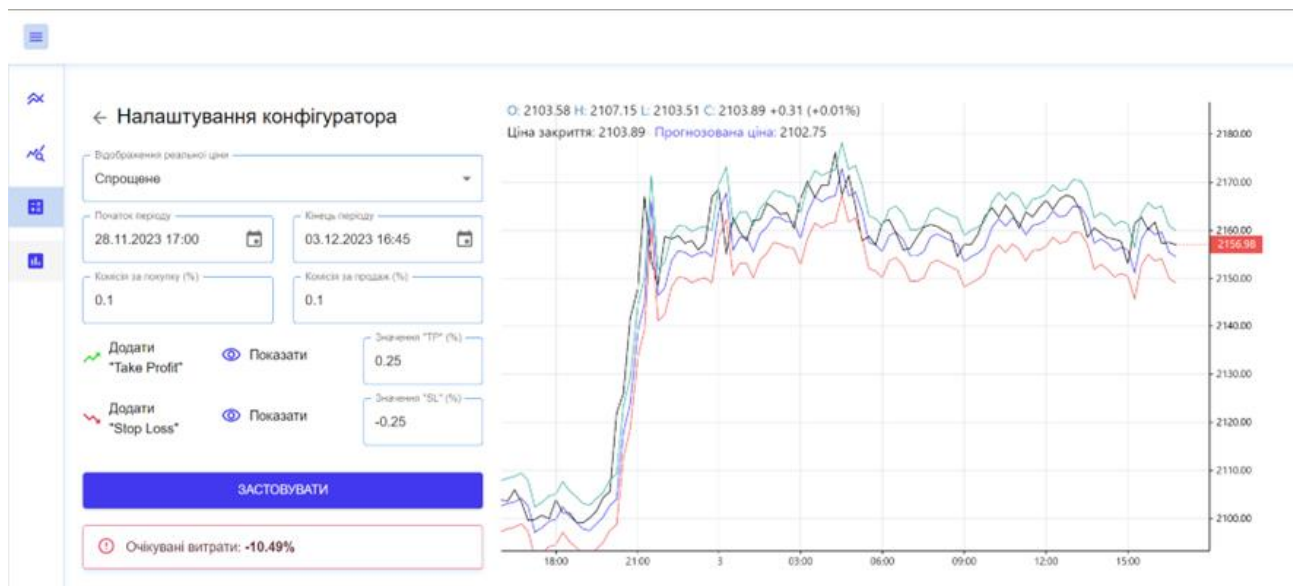


Рис. 3.32. Аналіз ефективності алгоритму з врахуванням комісій біржі

Комісії на придбання та продаж були розділені окремо, так як певні біржі мають різні значення для цих видів операцій. Це дозволяє провести більш об'єктивне дослідження й отримати більш практичні результати аналізу.

Як було згадано у попередніх розділах, важливим елементом порівняльного аналізу ефективності методів прогнозування курсу криптовалют є впровадження аналітичної системи. Для цього необхідна інформація була трансформована окремим модулем компоненту АМН та використана у Power BI звітах.



Рис. 3.33. Порівняльний аналіз прогнозів різних моделей прогнозування

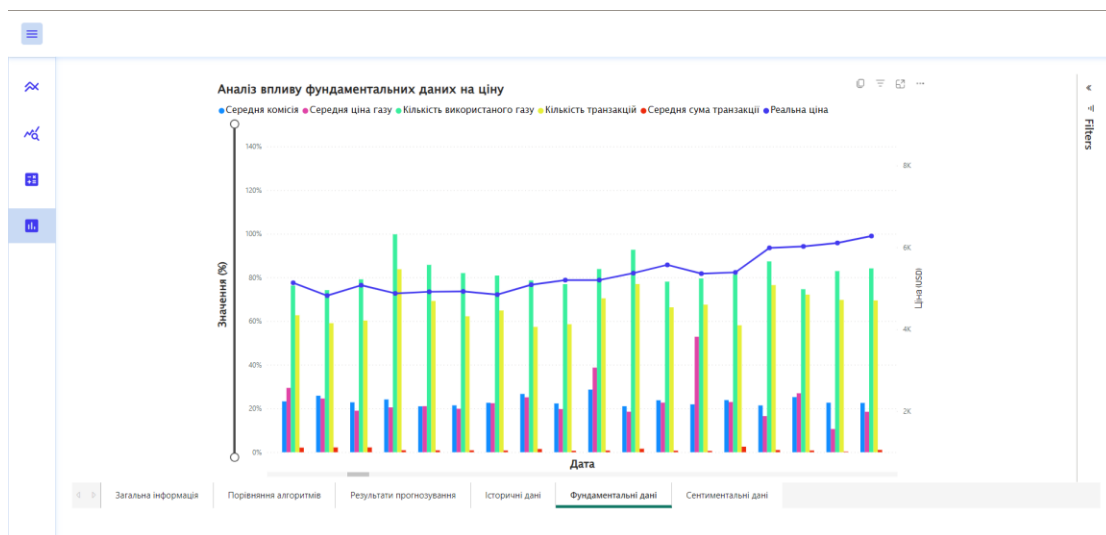


Рис. 3.34. Дослідження кореляції між параметрами даних та ціною криптоактиву

Ці звіти були інтегровані у клієнтський додаток, що дозволило проводити порівняння ефективності методів прогнозування між собою (див. рис. 3.33), а також оцінювати кореляцію зміни параметрів даних зі зміною котирувань досліджуваної криптовалюти (див. рис. 3.34).

Також варто звернути увагу на процес навчання алгоритмів, які дозволяють автоматичне визначення параметрів. Для використання цієї можливості користувач має обрати опцію “Автоматично визначити параметри” при створенні або редагуванні алгоритмів, які базуються на Random Forest або XG Boost (рис. 3.35).

Рис. 3.35. Налаштування алгоритму для автоматичного визначення параметрів

У цьому випадку, під час обробки налаштованого методу прогнозування ковесором AutoML виконається додатковий процес пошуку найкращих параметрів нейронної мережі (рис. 3.36). Визначений набір параметрів буде використаний при створенні моделі.

```
[CV] END max_depth=4, min_samples_leaf=4, min_samples_split=3, n_estimators=720; total time= 2.0s
[CV] END max_depth=1, min_samples_leaf=5, min_samples_split=7, n_estimators=980; total time= 2.3s
[CV] END max_depth=4, min_samples_leaf=4, min_samples_split=3, n_estimators=720; total time= 2.0s
[CV] END max_depth=1, min_samples_leaf=7, min_samples_split=7, n_estimators=380; total time= 0.8s
[CV] END max_depth=4, min_samples_leaf=4, min_samples_split=3, n_estimators=720; total time= 2.1s
[CV] END max_depth=1, min_samples_leaf=7, min_samples_split=7, n_estimators=380; total time= 0.8s
[CV] END max_depth=1, min_samples_leaf=4, min_samples_split=3, n_estimators=320; total time= 0.5s
[CV] END max_depth=1, min_samples_leaf=4, min_samples_split=3, n_estimators=320; total time= 0.6s
[CV] END max_depth=5, min_samples_leaf=4, min_samples_split=2, n_estimators=560; total time= 1.4s
[CV] END max_depth=5, min_samples_leaf=4, min_samples_split=2, n_estimators=560; total time= 1.5s
[CV] END max_depth=5, min_samples_leaf=4, min_samples_split=2, n_estimators=560; total time= 1.5s
[CV] END max_depth=2, min_samples_leaf=2, min_samples_split=3, n_estimators=760; total time= 1.5s
[CV] END max_depth=2, min_samples_leaf=2, min_samples_split=3, n_estimators=760; total time= 1.7s
[CV] END max_depth=2, min_samples_leaf=2, min_samples_split=3, n_estimators=760; total time= 1.8s
[CV] END max_depth=7, min_samples_leaf=1, min_samples_split=9, n_estimators=760; total time= 2.8s
[CV] END max_depth=7, min_samples_leaf=1, min_samples_split=9, n_estimators=760; total time= 2.8s
[CV] END max_depth=7, min_samples_leaf=1, min_samples_split=9, n_estimators=760; total time= 2.7s
The best parameters: {'n_estimators': 160, 'min_samples_split': 5, 'min_samples_leaf': 2, 'max_depth': 7}
```

Рис. 3.36. Процес пошуку оптимальних параметрів мережі

Після завершення усіх етапів конвеєру визначені значення надсилаються до серверу разом з обчисленими метриками й користувач може побачити ці значення параметрів у налаштуваннях методу (рис. 3.37).

Рис. 3.37. Перегляд інформації про визначені параметри у налаштуваннях методу

Як було багаторазово згадано, першим етапом АМН є збір та аналіз даних. Хоч і користувач не контролює процес отримання даних з різних джерел, ми можемо переглянути як це автоматично відбувається відповідними компонентами системи. Кожний тип інформації завантажується окремо спеціальними модулями. Завантаження історичних даних відбувається за допомогою використання API сервісу Binance (рис. 3.38). Як можемо побачити, програмний модуль визначає кількість часових інтервалів, які відсутні у базі даних й проводить їх завантаження.

```
[BTCUSDT] [fifteen-minutes] missing periods count 2000
[ETHUSDT] [fifteen-minutes] missing periods count 2000
[BNBUSDT] [fifteen-minutes] missing periods count 2000
[BTCUSDT] [hour] missing periods count 2000
[ETHUSDT] [hour] missing periods count 2000
[BNBUSDT] [hour] missing periods count 2000
[BTCUSDT] [day] missing periods count 176
[ETHUSDT] [day] missing periods count 176
[BNBUSDT] [day] missing periods count 176
```

Рис. 3.38. Завантаження історичних даних

Завантаження даних з блокчейну було реалізовано окремими модулями

через особливості обраних децентралізованих мереж. Доступ до статистичних даних про мережу Bitcoin був забезпечений за допомогою використання API, яке дозволяє переглядати блоки за їх ідентифікатором, а також отримувати список блоків, які були додані у певний проміжок часу (рис. 3.39).

```
Blocks set #1 block 1 of 143
Blocks set #1 block 2 of 143
Blocks set #1 block 3 of 143
Blocks set #1 block 4 of 143
Blocks set #1 block 5 of 143
Blocks set #1 block 6 of 143
Blocks set #1 block 7 of 143
Blocks set #1 block 8 of 143
Blocks set #1 block 9 of 143
Blocks set #1 block 10 of 143
Blocks set #1 block 11 of 143
Blocks set #1 block 12 of 143
Blocks set #1 block 13 of 143
Blocks set #1 block 14 of 143
Blocks set #1 block 15 of 143
Blocks set #1 block 16 of 143
Blocks set #1 block 17 of 143
Blocks set #1 block 18 of 143
```

Рис. 3.39. Завантаження даних з блокчейну Bitcoin через API

Обробка даних з блокчейнів Ethereum та BSC була подібна, проте отримання інформації було релазіовано за допомогою взаємодії з мережами через RPC (рис. 3.40).

```
[BNB] period: fifteen-minutes count: 1
[ETH] period: fifteen-minutes count: 1
[BNB] period: fifteen-minutes count: 2
[ETH] period: fifteen-minutes count: 2
[BNB] period: fifteen-minutes count: 3
[BNB] period: fifteen-minutes count: 4
[ETH] period: fifteen-minutes count: 3
[BNB] period: fifteen-minutes count: 5
[ETH] period: fifteen-minutes count: 4
[BNB] period: fifteen-minutes count: 6
[ETH] period: fifteen-minutes count: 5
[BNB] period: fifteen-minutes count: 7
[BNB] period: fifteen-minutes count: 8
[ETH] period: fifteen-minutes count: 6
[BNB] period: fifteen-minutes count: 9
[ETH] period: fifteen-minutes count: 7
[BNB] period: fifteen-minutes count: 10
[ETH] period: fifteen-minutes count: 8
[BNB] period: fifteen-minutes count: 11
[ETH] period: fifteen-minutes count: 9
[BNB] period: fifteen-minutes count: 12
```

Рис. 3.40. Завантаження даних з блокчейнів через Web3

Висновки до третього розділу

Початковим етапом розробки системи є вибір засобів реалізації. У результаті проектування нашого вебсервісу були виокремлені основні компоненти, які будуть вирішувати окремі завдання. Ефективність цих компонентів значено залежить від інструментів, які будуть обрані для їх реалізації. Отже, було проведено аналіз різних існуючих варіантів й були обрані найбільш оптимальні засоби.

Наступним етапом після вибору засобів є проектування та реалізація згаданих компонентів. Побудова правильної архітектури додатку дозволяє розробити систему, яка буде реалізовувати усі функціональні вимоги й при цьому відповідати нефункціональним. Був проведений аналіз існуючих підходів до розробки програмного забезпечення й прийняті відповідні архітектурні рішення для кожного компоненту системи.

Результатом реалізації будь-якої системи є програмний продукт, який вимагає проведення тестування з метою перевірки коректності роботи системи й підтвердження відповідності готового продукту до визначених вимог. Процес тестування полягав у створенні нового методу прогнозування, перевірці роботи компоненту АМН, а також аналізі та порівнянні отриманих результатів аналітичними модулями системи. Це дозволило переконатися у правильності роботи конвеєру AutoML, а також продемонструвати наскільки простішим та ефективнішим став процес дослідження різних методів прогнозування курсу криптовалют.

Отже, у межах цього розділу було реалізовано та протестовано систему аналізу методів прогнозування курсу криптовалют.

ВИСНОВКИ

У кваліфікаційній роботі було спроектовано та розроблено програмне забезпечення для аналізу та порівняння між собою методів прогнозування курсу криптовалют з використанням принципів АМН.

В основу системи було покладено використання різних методів прогнозування, тому важливим етапом було дослідження цих алгоритмів та аналіз можливості їх порівняння. Для цього був проведений огляд актуальних досліджень у цій сфері для того, щоб визначити провідні види нейронних мереж, які використовуються для задач прогнозування обраного виду активів. На основі проведеного дослідження був сформований список вимог до системи, щоб вона забезпечувала можливість ефективного їх порівняння та аналізу.

Наступним важливим етапом було дослідження систем АМН і конвеєру AutoML. Даний вид програмного забезпечення швидко здобуває популярність, проте є досить специфічним й вимагає врахування цих особливостей при проектуванні та розробці власних програмних продуктів. Для цього було проведене моделювання основних процесів системи: робота з даними, створення та навчання моделей прогнозування, генерація результатів та їх порівняння. Усі ці процеси було спроектовано відповідно до вимог конвеєру АМН, що підтвердило можливість розробки системи з використанням автоматизованого машинного навчання для вирішення наших завдань.

Після визначення вимог було проведене проектування системи, яка відповідатиме усім вимогам та поставленим завданням. В результаті був проведений поділ монолітної системи на окремі компоненти. Проведення такої декомпозиції дозволило простіше й ефективніше вибирати засоби реалізації, архітектуру кожного структурного елементу та розподіляти завдання між ними.

Важливою частиною роботи було проведення аналізу та впровадження модулів системи, які відповідають за виконання етапів конвеєру АМН.

Різноманітність підтримуваних алгоритмів і джерел даних вимагали пошуку універсальних рішень, бо від цього залежав рівень автоматизації системи. В результаті виконання цієї частини були створені архітектурні рішення й розроблені відповідні програмні модулі, що дозволило узагальнити вхідні дані й стандартизувати роботу з різними видами алгоритмів.

Система призначена для аналізу та порівняння різних варіацій алгоритмів прогнозування вартості криптовалют. Отже, заключним й дуже важливим елементом є компонент аналізу та порівняння отриманих результатів. Для цього у системі були задіяні різні підходи. Це дозволило користувачам порівнювати алгоритми між собою, аналізувати ефективність окремих методів прогнозування, а також досліджувати різні види даних і їх параметрів завдяки аналізу кореляції зміни їх значень зі зміною реальної ціни криптоактиву.

Отже, розроблена система відповідає усім визначеним вимогам й виконує поставлені завдання, що свідчить про досягнення мети роботи. Її основне практичне значення полягає у можливості стандартизації різноманітних досліджень сфери прогнозування курсу криптовалют. Відповідно до отриманих результатів, система аналізу методів прогнозування курсу криптовалют значно спрощує та пришвидшує цей процес, що дозволяє швидко перевірити ідеї досліджень на практиці та порівняти результати між собою.

Можливими напрямками подальших наукових досліджень є розширення джерел даних і набору підтримуваних алгоритмів нейронних мереж. Окрім цього, корисним може буде використання у дослідженні наступних метрик: кореляція цін різних криптовалют між собою, взаємозв'язок між ціною криптоактивів та іншими видами активів, використання індикаторів економіки та врахування біржових стаканів при прогнозування майбутньої ціни.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Sitailo M. S., Morozov A. V. Research of the cryptocurrency price prediction approaches / M. Sitailo, A. Morozov. // Тези доповідей V Всеукраїнської науково-технічної конференції: “Комп’ютерні технології: інновації, проблеми, рішення” 01-02 грудня – С. 211–212.
2. Jay P. Stochastic neural networks for cryptocurrency price prediction / P. Jay, V. Kalariya. // IEEE Access. – 2020. – №8. – С. 82804–82818.
3. Yiyang W. Cryptocurrency price analysis with artificial intelligence / W. Yiyang, Y. Yeze. – 2019. – С. 97–101.
4. Saad M. Toward characterizing blockchain-based cryptocurrencies for highly accurate predictions / M. Saad, J. Choi. – 2020. – С. 321–332.
5. Wang T. When blockchain meets ai: Optimal mining strategy achieved by machine learning [Електронний ресурс] / Wang. – 2021. – Режим доступу до ресурсу: <https://onlinelibrary.wiley.com/doi/abs/10.1002/int.22375>.
6. Cheuque G. Bitcoin price prediction through opinion mining / G. Cheuque, J. Reutter. – 2019. – С. 755–762.
7. Kraaijeveld O. The predictive power of public twitter sentiment for forecasting cryptocurrency prices / O. Kraaijeveld, J. De Smedt. – 2020. – С. 101–188.
8. Gupta I. Deep learning approach to determine the impact of socio economic factors on bitcoin price prediction / I. Gupta, N. Garg, A. Goel. – 2019. – С. 1–5.
9. Larasati K. Forecasting Bitcoin price based on blockchain information using long-short term method / K. Larasati, A. Primandari. – 2021. – С. 1–5.
10. Rothman T. Empirical analysis towards the effect of social media on cryptocurrency price and volume / T. Rothman, C. Yakar. – 2019. – С. 52.
11. Ali M. A data selection methodology to train linear regression model to predict bitcoin price / M. Ali, S. Shatabda. // 2nd International Conference on Advanced Information and Communication Technology (ICAICT). – 2020. – С. 330–335.

12. Wu J. Short term return prediction of cryptocurrency based on XGBoost algorithm / J. Wu, X. Guo. // International Conference on Big Data, Information and Computer Network (BDICN). – 2022. – С. 39–42.
13. Si Y. Using ARIMA model to analyse and predict bitcoin price / Yang Si. // DCP Business & Management. – 2022. – №34. – С. 1210–1216.
14. Biswas S. Cryptocurrency price prediction using neural networks and deep learning / S. Biswas, M. Pawar. // 7th International Conference on Advanced Computing and Communication Systems (ICACCS). – 2021. – №1. – С. 408–413.
15. Huang X. Lstm based sentiment analysis for cryptocurrency prediction [Електронний ресурс] / X. Huang, W. Zhang // arXiv.org. – 2021. – Режим доступу до ресурсу: <https://arxiv.org/abs/2103.14804>.
16. Yao Y. Combination of window-sliding and prediction range method based on LSTM model for predicting cryptocurrency [Електронний ресурс] / Y. Yao, L. Wang // arXiv.org. – 2021. – Режим доступу до ресурсу: <https://arxiv.org/abs/2102.05448>.
17. Sitailo M. S., Morozov A. V. Popular machine learning algorithm's types for cryptocurrencies price prediction / M. Sitailo, A. Morozov. // Тези доповідей VI Всеукраїнської науково-технічної конференції: “Комп’ютерні технології: інновації, проблеми, рішення” 29-30 листопада.
18. Linear Regression in Machine learning [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/ml-linear-regression/>.
19. What is a random forest? [Електронний ресурс] // Spotfire – Режим доступу до ресурсу: <https://www.spotfire.com/glossary/what-is-a-random-forest>.
20. XGBoost: A Powerful Machine Learning Framework [Електронний ресурс] // CloudThat. – 2023. – Режим доступу до ресурсу: <https://www.cloudthat.com/resources/blog/xgboost-a-powerful-machine-learning-framework>.

21. Using the ARIMA model and Python for Time Series forecasting [Электронный ресурс]. – 2022. – Режим доступа до ресурсу: <https://hands-on.cloud/using-the-arima-model-and-python-for-time-series-forecasting/>.
22. LSTM Gradients [Электронный ресурс]. – 2020. – Режим доступа до ресурсу: <https://towardsdatascience.com/lstm-gradients-b3996e6a0296..>.
23. AutoML [Электронный ресурс] // AutoML.org – Режим доступа до ресурсу: <https://www.automl.org/automl/>.
24. Google Cloud AutoML [Электронный ресурс] // Google – Режим доступа до ресурсу: <https://cloud.google.com/automl/>.
25. SageMaker Autopilot [Электронный ресурс] // AWS Documentation – Режим доступа до ресурсу: <https://docs.aws.amazon.com/sagemaker/latest/dg/autopilot-automate-model-development.html>.
26. Automated machine learning [Электронный ресурс] // Microsoft – Режим доступа до ресурсу: <https://azure.microsoft.com/en-us/products/machine-learning/automatedml/#overview>.
27. Use AutoML for building simple to complex machine learning pipelines [Электронный ресурс] // packthub – Режим доступа до ресурсу: <https://hub.packtpub.com/automl-build-machine-learning-pipeline-tutorial/>.
28. Binance API Documentation [Электронный ресурс] // Binance – Режим доступа до ресурсу: <https://binance-docs.github.io/apidocs/spot/en/#introduction>.
29. What is Blockchain Technology? How Does Blockchain Work? [Электронный ресурс] // simplilearn. – 2023. – Режим доступа до ресурсу: <https://www.simplilearn.com/tutorials/blockchain-tutorial/blockchain-technology>.
30. The Ethereum Blockchain Explorer [Электронный ресурс] – Режим доступа до ресурсу: <https://etherscan.io/>.

31. Blockchain.com APIs [Электронный ресурс] // Blockchain.com – Режим доступа до ресурсу: <https://www.blockchain.com/explorer/api>.
32. Remote Procedure Call (RPC) [Электронный ресурс] // TechTarget – Режим доступа до ресурсу: <https://www.techtarget.com/searchapparchitecture/definition/Remote-Procedure-Call-RPC>.
33. Introduction to Web3 [Электронный ресурс] // ethereum.org. – 2023. – Режим доступа до ресурсу: <https://ethereum.org/en/web3/>.
34. Infura Docs [Электронный ресурс] // Infura.io – Режим доступа до ресурсу: <https://www.infura.io/>
35. What is OLTP? [Электронный ресурс] // IBM – Режим доступа до ресурсу: <https://www.ibm.com/topics/oltp>.
36. What does ACID mean in Database Systems? [Электронный ресурс] // Database.Guide – Режим доступа до ресурсу: <https://database.guide/what-is-acid-in-databases/>.
37. Biscobing J. OLAP (online analytical processing) [Электронный ресурс] / Biscobing // TechTarget – Режим доступа до ресурсу: <https://www.techtarget.com/searchdatamanagement/definition/OLAP>.
38. What are OLAP cubes? [Электронный ресурс] // Snowflake.com – Режим доступа до ресурсу: <https://www.snowflake.com/trending/olap-cubes>.
39. An introduction to the Flux architectural pattern [Электронный ресурс] // FreeCodeCamp. – 2019. – Режим доступа до ресурсу: <https://www.freecodecamp.org/news/an-introduction-to-the-flux-architectural-pattern-674ea74775c9/>.
40. Multilayered Software Architecture [Электронный ресурс]. – 2020. – Режим доступа до ресурсу: <https://medium.com/@e0324913/multilayered-software-architecture-1eaa97b8f49e>.

ДОДАТКИ

Програмне впровадження процесу створення нового методу прогнозування

Form.tsx

```

const Form = ({ isLoading, networkArchitectures, algorithm, submitForm,
loadNetworkArchitectures }: FormType) => {
  const [activeStep, setActiveStep] = useState(0)
  const [currentAlgorithm, setCurrentAlgorithm] = useState<IAlgorithm>(algorithm ||
emptyAlgorithmObject);

  useEffect(() => {
    loadNetworkArchitectures();
  }, []);

  if (!networkArchitectures.length)
    return <Skeleton variant='rounded' className='page-skeleton' />

  return (
    <>
      <Stepper activeStep={activeStep} orientation='vertical'
className={styles.stepContainer}>
        <Step key='algorithm' className={styles.step}>
          <StepLabel>
            <Typography variant='h6'>
              {Texts.AlgorithmConfigurationStep}
            </Typography>
          </StepLabel>
          <StepContent>
            <AlgorithmConfigurationStep
              algorithm={currentAlgorithm}
              networkArchitectures={networkArchitectures}
              isLoading={isLoading}
              setActiveStep={setActiveStep}
              setCurrentAlgorithm={setCurrentAlgorithm}
            />
          </StepContent>
        </Step>
      </>
    )
  
```

```

        { currentAlgorithm.networkArchitecture?.name !==
NetworkArchitectures.ARIMA && (
            <Step key='data' className={styles.step}>
                <StepLabel>
                    <Typography variant='h6'>
                        {Texts.DataConfigurationStep}
                    </Typography>
                </StepLabel>
                <StepContent>
                    <DataConfigurationStep
                        algorithm={currentAlgorithm}
                        isLoading={isLoading}
                        setActiveStep={setActiveStep}
                        setCurrentAlgorithm={setCurrentAlgorithm}
                    />
                </StepContent>
            </Step>
        )}
        <Step key='network' className={styles.step}>
            <StepLabel>
                <Typography variant='h6'>
                    {Texts.NetworkConfigurationStep}
                </Typography>
            </StepLabel>
            <StepContent>
                <NetworkConfigurationStep
                    algorithm={currentAlgorithm}
                    isLoading={isLoading}
                    setActiveStep={setActiveStep}
                    submitForm={submitForm}
                />
            </StepContent>
        </Step>
    </Stepper>
</>
);
};

```

```

type FormType = {
  isLoading: boolean;
  networkArchitectures: INetworkArchitecture[];
  algorithm?: IAlgorithm;
  submitForm: (algorithm: IAlgorithm) => void;
  loadNetworkArchitectures: () => void;
};

const mapStateToProps = (
  state: IApplicationState,
) => ({
  isLoading: state.notification.loader.isLoading,
  networkArchitectures: state.page.networkArchitectures.list
});

export default connect(mapStateToProps, { loadNetworkArchitectures })(Form);

```

actions.ts

```

export const createAlgorithm = (algorithm: IAlgorithm) => ({
  type: CREATE_ALGORITHM,
  payload: {
    algorithm
  }
});

export type CreateAlgorithmActionType = ReturnType<typeof createAlgorithm>

```

epic.ts

```

const createAlgorithmEpic: Epic = (action$) => action$.pipe(
  filter(isOfType(CREATE_ALGORITHM)),
  switchMap((action: CreateAlgorithmActionType) =>
    createAlgorithmRequest(action.payload.algorithm).pipe(
      map(() => navigateTo(Routes.Algorithms.List)),
      catchError(error => of(showSnackBarMessage(error.response.message,
        SnackBarMessageTypes.Error))),
      startWith(changeLoading(true)),
      endWith(changeLoading(false))
    )),
);

```

```
);
```

AlgorithmController.cs

```
[HttpPost]
[Route("create")]
public object Create(AlgorithmModel model)
{
    var networkArchitecture =
        _networkArchitectureManager.GetById(model.NetworkArchitectureId);
    if (networkArchitecture == null)
        return NotFound(new NotFoundResponse("Не було знайдено обрану архітектуру
мережі"));

    var algorithm = model.ToEntity(networkArchitecture);
    _algorithmManager.Create(algorithm);

    return Ok();
}
```

BaseManager.cs

```
public virtual T Create(T entity)
{
    return _provider.Create(entity);
}
```

BaseProvider.cs

```
public virtual T Create(T entity)
{
    return WrapDataAccessExceptionAsync(async () =>
    {
        _context.Set<T>().Add(entity);
        int addedRowCount = await _context.SaveChangesAsync();
        return addedRowCount > 0 ? entity : null;
    }, "Error occurred during saving of the new entity", null);
}
```

Algorithm.cs

```
public class Algorithm : Entity
{
    [Required]
    [StringLength(100)]
    public string Name { get; set; }

    [Required]
    [StringLength(20)]
    public string Cryptocurrency { get; set; }

    [Required]
    [StringLength(30)]
```

```

public string Period { get; set; }

[Required]
public NetworkArchitecture NetworkArchitecture { get; set; }

[Range(1, 100)]
public byte? WindowSize { get; set; }

[Required]
public DateTime CreatedDate { get; set; }

public AlgorithmStatus Status { get; set; }

public decimal? MeanAbsoluteError { get; set; }

public decimal? MeanSquaredError { get; set; }

public decimal? MeanAbsolutePercentageError { get; set; }

public decimal? MedianAbsoluteError { get; set; }

[NotMapped]
public List<string> HistoricalDataFields { get; set; }

[NotMapped]
public List<string> BlockchainDataFields { get; set; }

[NotMapped]
public List<string> TwitterDataFields { get; set; }

public string HistoricalFields
{
    get => JsonConvert.SerializeObject(HistoricalDataFields);
    set => HistoricalDataFields = JsonConvert.DeserializeObject<List<string>>(value);
}

public string BlockchainFields
{
    get => JsonConvert.SerializeObject(BlockchainDataFields);
    set => BlockchainDataFields = JsonConvert.DeserializeObject<List<string>>(value);
}

public string TwitterFields
{
    get => JsonConvert.SerializeObject(TwitterDataFields);
    set => TwitterDataFields = JsonConvert.DeserializeObject<List<string>>(value);
}

public string Config { get; set; }

public Algorithm()
{
    HistoricalDataFields = new List<string>();
    BlockchainDataFields = new List<string>();
    TwitterDataFields = new List<string>();
}

```

```
}
```

AlgorithmModel.cs

```
public class AlgorithmModel
{
    [Required]
    [MaxLength(100)]
    public string Name { get; set; }

    [Required]
    [MaxLength(20)]
    public string Cryptocurrency { get; set; }

    [Required]
    [MaxLength(30)]
    public string Period { get; set; }

    [Required]
    public Guid NetworkArchitectureId { get; set; }

    [Range(1, 100)]
    public byte? WindowSize { get; set; }

    [Required]
    public bool UsesCustomData { get; set; }

    [DataFields(requiredFieldName: "close")]
    public List<string> HistoricalDataFields { get; set; }

    [DataFields]
    public List<string> BlockchainDataFields { get; set; }

    [DataFields]
    public List<string> TwitterDataFields { get; set; }

    public string Config { get; set; }

    public Entities.Algorithm ToEntity(NetworkArchitecture networkArchitecture)
    {
        return new Entities.Algorithm()
        {
            Name = Name,
            Cryptocurrency = Cryptocurrency,
            Period = Period,
            NetworkArchitecture = networkArchitecture,
            WindowSize = networkArchitecture.SupportsRangeData ? WindowSize : null,
            HistoricalDataFields = HistoricalDataFields,
            BlockchainDataFields = BlockchainDataFields,
            TwitterDataFields = TwitterDataFields,
            CreatedDate = DateTime.Now,
            Status = AlgorithmStatus.New,
            Config = Config
        };
    }
}
```

Програмна реалізація шаблону моделі LSTM та основних модулів AutoML

lstm.py

```

import os
import tensorflow as tf
from keras.callbacks import EarlyStopping, LearningRateScheduler
from keras.models import Sequential, Model
from keras.layers import Dense, Dropout, LSTM, Input
import numpy as np
from networks.network import Network
from common import prepare_data_frames, normalize_data
from data_provider import save_predictions
import json

def lr_scheduler(epoch, lr):
    if epoch >= 25 and epoch % 5 == 0:
        return lr * 0.85

    return lr

class LSTMNetwork(Network):
    def __init__(self, algorithm):
        super().__init__(algorithm)
        self.config = json.loads(algorithm.config)

    def build(self, train_data):
        train_data = normalize_data(train_data)
        X_train, X_test, y_train, y_test = prepare_data_frames(train_data,
self.algorithm.window_size)

        shape = np.array(X_train).shape
        if self.config['useDefaultParameters'] == True:
            print('Using default parameters for LSTM model')
            self.model = Sequential([
                LSTM(128, input_shape=(shape[1], shape[2]), return_sequences = True),
                Dropout(0.2),
                LSTM(256, return_sequences = True),
                Dropout(0.2),
                LSTM(256),
                Dropout(0.2),
                Dense(1)
            ])
            self.model.compile(loss = 'mse', optimizer = 'adam')

            es_callback = EarlyStopping(monitor = 'loss', patience = 30, restore_best_weights
= True)
            lr_callback = LearningRateScheduler(lr_scheduler)
            self.model.fit(np.asarray(X_train).astype('float32'), y_train, epochs = 200,
batch_size = 16, shuffle = True, callbacks=[lr_callback, es_callback], validation_data =
(np.asarray(X_test).astype('float32'), y_test))
        else:
            epochs = self.config['epochs']
            patience = self.config['patience']

```

```

optimizer = self.config['optimizer']
lr = self.config['learningRate']
print('LSTM model: epochs = %s, patience = %s, optimizer = %s, lr = %s' % (epochs,
patience, optimizer, lr))

self.model = self.build_model(self.config['layers'], (shape[1], shape[2]))

optimizer_parameter = tf.keras.optimizers.Adam(learning_rate = lr)
if optimizer == 'rmsprop':
    optimizer_parameter = tf.keras.optimizers.RMSprop(learning_rate = lr)
elif optimizer == 'sgd':
    optimizer_parameter = tf.keras.optimizers.SGD(learning_rate = lr)

self.model.compile(loss = 'mse', optimizer = optimizer_parameter)

es_callback = EarlyStopping(monitor = 'loss', patience = patience,
restore_best_weights = True)
lr_callback = LearningRateScheduler(lr_scheduler)
print(self.model.summary())

self.model.fit(np.asarray(X_train).astype('float32'), y_train, epochs = epochs,
batch_size = 16, shuffle = True, callbacks=[lr_callback, es_callback], validation_data =
(np.asarray(X_test).astype('float32'), y_test))

self.model.save('models/' + self.algorithm.id)

def build_model(self, layers, input_shape):
    inputs = Input(shape = input_shape)
    current_layer = inputs
    print(layers)
    for layer in layers:
        if layer['type'] == 'LSTM':
            current_layer = LSTM(layer['units'], activation = layer['activation'],
return_sequences = layer['returnSequences'])(current_layer)
        else:
            current_layer = Dense(layer['units'], activation =
layer['activation'])(current_layer)

    if layer['usesDropout'] == True:
        current_layer = Dropout(layer['dropout'])(current_layer)

    return Model(inputs = inputs, outputs = current_layer)

def predict(self, test_data):
    timestamps = test_data[test_data.columns[-1]]
    initial_price = test_data[0][0]

    test_data = test_data.drop(test_data.columns[-1], axis = 1)
    test_data = normalize_data(test_data)
    _, X_test, _, _ = prepare_data_frames(test_data, self.algorithm.window_size,
test_size = 1)
    predictions = self.model.predict(np.asarray(X_test).astype('float32'))
    predictions = (1 + np.squeeze(predictions, -1)) * initial_price

    save_predictions(self.algorithm, predictions, timestamps)

```

```
    return predictions
```

models_manager.py

```
import data_provider
import pandas as pd
from algorithm import Algorithm
import requests_helper
from sklearn.metrics import mean_absolute_error, mean_squared_error,
mean_absolute_percentage_error, median_absolute_error
from networks import svr, lstm, random_forest, xgboost, arima, prophet
import json

def calculate_metrics(algorithm, predictions):
    actual_price = pd.DataFrame(data_provider.load_price(algorithm.period,
algorithm.currency, offset = 1500 + algorithm.window_size, limit =
len(predictions)), columns = ['timestamp', 'price']).price.values

    return {
        'MeanAbsoluteError': mean_absolute_error(predictions, actual_price),
        'MeanSquaredError': mean_squared_error(predictions, actual_price),
        'MeanAbsolutePercentageError': mean_absolute_percentage_error(predictions,
actual_price),
        'MedianAbsoluteError': median_absolute_error(predictions, actual_price)
    }

def create_network(algorithm):
    if algorithm.network == 'SVR':
        return svr.SVRNetwork(algorithm)
    if algorithm.network == 'LSTM':
        return lstm.LSTMNetwork(algorithm)
    if algorithm.network == 'Random Forest':
        return random_forest.RandomForestNetwork(algorithm)
    if algorithm.network == 'XG Boost':
        return xgboost.XGBoostNetwork(algorithm)
    if algorithm.network == 'ARIMA':
        return arima.ARIMANetwork(algorithm)
    if algorithm.network == 'Prophet':
        return prophet.ProphetNetwork(algorithm)

    return None

data = requests_helper.load_new_algorithms()
for item in data:
    algorithm = Algorithm(item)
    network = create_network(algorithm)
    if network == None:
        continue

    data_provider.delete_existing_predictions(algorithm.id)
    data = data_provider.load_training_data(algorithm)
    if data == None:
        continue

    data = pd.DataFrame(data)
    network.build(data)
```

```

data = data_provider.load_test_data(algorithm)
if data == None:
    break

data = pd.DataFrame(data)

predictions = network.predict(data)
metrics = calculate_metrics(algorithm, predictions)

if 'useAutoParameters' in network.config and network.config['useAutoParameters'] ==
True:
    requests_helper.save_parameters(algorithm.id, network.pack_params())
    requests_helper.track_algorithm_processing(algorithm.id, metrics)

```

model_builder.py

```

import numpy as np

def train_test_split(data, test_size = 0.2):
    split_row = len(data) - int(test_size * len(data))
    train_data = data.iloc[:split_row]
    test_data = data.iloc[split_row:]
    return train_data, test_data

def extract_window_data(data, window_size):
    window_data = []
    data_length = len(data)
    for idx in range(data_length - window_size):
        window_data.append(data[idx: (idx + window_size)].copy())

    return window_data

def prepare_train_data(data, test_size = 0.2, delay_size = 1):
    train, test = train_test_split(data, test_size)

    X_train = train[:-delay_size]
    X_test = test[:-delay_size]
    y_train = train[0][delay_size:]
    y_test = test[0][delay_size:]

    return X_train, X_test, y_train, y_test

def prepare_data_frames(data, window_size = 5, test_size = 0.2):
    train_data, test_data = train_test_split(data, test_size)
    X_train = extract_window_data(train_data, window_size)
    X_test = extract_window_data(test_data, window_size)

    y_train = np.asarray(train_data[0][window_size:]).astype('float32')
    y_test = np.asarray(test_data[0][window_size:]).astype('float32')

    return X_train, X_test, y_train, y_test

def build_model(algorithm, data):
    model = None
    if algorithm.supports_ranges == True:

```

```

X_train, X_test, y_train, y_test = prepare_data_frames(data, algorithm.window_size)
if algorithm.network == 'LSTM':
    model = lstm.build_model(algorithm.id, X_train, y_train, X_test, y_test)
else:
    X_train, X_test, y_train, y_test = prepare_data(data)
    if algorithm.network == 'SVR':
        model = svr.build_model(X_train, y_train)
    elif algorithm.network == 'LinearRegression':
        model = lr.build_model(X_train, y_train)

return model

def predict(algorithm, data, model):
    if algorithm.supports_ranges == True:
        _, X_test, _, _ = prepare_data_frames(data, algorithm.window_size, test_size = 1)
        if algorithm.network == 'LSTM':
            return lstm.predict(model, X_test)

    _, X_test, _, _ = prepare_data(data, test_size = 1)
    if algorithm.network == 'SVR':
        return svr.predict(model, X_test)

    return lr.predict(model, X_test)

```

Програмний код компонентів збору даних з різних джерел

binance_price_loader.py

```

from binance.spot import Spot
import schedule
import time
from modules import database, helpers
import datetime

periods = ['fifteen-minutes', 'hour', 'day']

def map_data(raw_data):
    data = []
    for item in raw_data:
        data.append({
            'open': item[1],
            'close': item[4],
            'high': item[2],
            'low': item[3],
            'volume': item[5],
            'trades_amount': item[7],
            'close_time': (item[6] + 1) / 1000,
        })

    return data

def get_period_parameter(period):
    if period == 'fifteen-minutes':
        return '15m'
    elif period == 'hour':
        return '1h'

    return '1d'

def load_data():
    for period in periods:
        settings = database.load_dataset_settings('historical data', period)
        if settings == None or 'token_pairs' not in settings:
            continue

        token_pairs = settings['token_pairs']
        for token_pair in token_pairs:

            res = database.fetch_item("SELECT close_time FROM historical_prices WHERE
trading_pair='%s' AND period='%s' ORDER BY close_time DESC" % (token_pair, period))
            last_timestamp = 0 if res == None else res[0]
            missing_periods = helpers.get_missing_periods_count(last_timestamp, period)
            period_parameters = get_period_parameter(period)

            if missing_periods <= 0:
                continue

```

```

        print([' + token_pair + ']', '[' + period + ']', 'missing periods count',
              str(missing_periods))

        loaded_data = client.klines(token_pair, period_parameters, limit =
min(missing_periods, 1000), endTime = helpers.get_end_timestamp(period) * 1000 - 1)
        start_time = loaded_data[0][0]
        missing_periods -= 1000

        results = []
        if missing_periods > 0:
            results = map_data(client.klines(token_pair, period_parameters, limit =
missing_periods, endTime = start_time - 1))

        results.extend(map_data(loaded_data))

        connection = database.initialize_connection()
        cursor = connection.cursor()

        for result in results:
            result['token_pair'] = token_pair
            result['period'] = period
            close_time = result['close_time']
            ts_now = datetime.datetime.now().timestamp()
            ticks_per_period = helpers.get_ticks_per_period(period)
            if close_time > last_timestamp and ts_now > close_time and close_time %
ticks_per_period == 0:
                cursor.execute("INSERT INTO historical_prices (trading_pair, period, open,
close, high, low, volume, trades_amount, close_time) VALUES('{token_pair}', '{period}',
{open}, {close}, {high}, {low}, {volume}, {trades_amount},
{close_time})".format(**result))

        cursor.close()
        connection.commit()

client = Spot()

schedule.every(60).seconds.do(load_data, period = 'fifteen-minutes')
schedule.every(1).day.at('00:01:00').do(load_data, period = 'day')
schedule.every(1).hour.at('01:00').do(load_data, period = 'hour')

```

```

while True:
    schedule.run_pending()
    time.sleep(1)

```

btc_screener.py

```

import requests
import datetime
import time
from modules import helpers, database
import math

periods = ['fifteen-minutes', 'hour', 'day']

def get_period_end(period, timestamp):
    ticks_per_period = helpers.get_ticks_per_period(period)

```

```

full_periods_count = timestamp // ticks_per_period

return (full_periods_count + 1) * ticks_per_period

def get_blocks_per_day(timestamp):
    return requests.get('https://blockchain.info/blocks/' + str(int(timestamp * 1000))
+ '?format=json').json()

def get_block(hash):
    return requests.get('https://blockchain.info/rawblock/' + hash).json()

def process_block(block):
    total_fee = 0
    transactions_amount = block['n_tx']
    max_value = 0
    min_value = 10 ** 100
    total_amount = 0
    for transaction in block['tx']:
        total_fee += transaction['fee']
        value = 0
        for output in transaction['out']:
            value += int(output['value'])

        if max_value < value:
            max_value = value

        if min_value > value:
            min_value = value
        total_amount += value

    return block['time'], {
        'max_value': max_value,
        'min_value': min_value,
        'average_amount': total_amount / transactions_amount,
        'average_fee': total_fee / transactions_amount,
        'average_gas_price': 0,
        'gas_used': 0,
        'difficulty': 0,
        'transaction_amount': block['n_tx']
    }

def load_data():
    for period in periods:
        settings = database.load_dataset_settings('blockchain data', period)
        if settings == None or 'BTC' not in settings:
            return

        last_item = database.fetch_item("SELECT time FROM blockchain_data WHERE ticker='BTC'
and blockchain_type='BTC' and period = '%s' ORDER BY time DESC" % period)
        last_timestamp = 0 if last_item == None else last_item[0]
        missing_periods = helpers.get_missing_periods_count(last_timestamp, period)
        if missing_periods <= 0:
            continue

        if period == 'fifteen-minutes':

```

```

        missing_periods = math.ceil(missing_periods / 96)

    if period == 'hour':
        missing_periods = math.ceil(missing_periods / 24)

    timeframes_count = 0
    current_timestamp = helpers.get_end_timestamp(period)
    statistics = {}
    step = 1
    if period == 'hour':
        step = 5

    if period == 'day':
        step = 25

    while timeframes_count < missing_periods:
        blocks = get_blocks_per_day(current_timestamp)
        blocks_count = len(blocks)
        for i in range(0, blocks_count, step):
            print("Blocks set #" + str(timeframes_count + 1), "block", str(i + 1), "of",
blocks_count)
            loaded_block = None
            try:
                loaded_block = get_block(blocks[i]['hash'])
            except:
                print("Error occurred during block loading", blocks[i]['hash'])

            if loaded_block == None:
                continue

            time, block_statistic = process_block(loaded_block)
            statistics[time] = block_statistic

            if time <= last_timestamp:
                timeframes_count = missing_periods
                break

        current_timestamp -= helpers.get_ticks_per_period('day')
        timeframes_count += 1

    grouped_periods = {}
    for key in statistics.keys():
        end_timestamp = get_period_end(period, key)
        if end_timestamp not in grouped_periods:
            grouped_periods[end_timestamp] = []

        grouped_periods[end_timestamp].append(statistics[key])

    results = {}
    for key in grouped_periods.keys():
        items = grouped_periods[key]
        items_count = len(items)

        max_value = 0
        min_value = 0

```

```

average_amount = 0
average_fee = 0
transaction_amount = 0

for item in items:
    max_value += item['max_value']
    min_value += item['min_value']
    average_amount += item['average_amount']
    average_fee += item['average_fee']
    transaction_amount += item['transaction_amount']

results[key] = {
    'period': period,
    'max_value': max_value / items_count,
    'min_value': min_value / items_count,
    'average_amount': average_amount / items_count,
    'average_fee': average_fee / items_count,
    'average_gas_price': 0,
    'gas_used': 0,
    'difficulty': 0,
    'transaction_amount': round(transaction_amount / items_count),
    'time': key,
}

connection = database.initialize_connection()
cursor = connection.cursor()

for key in results.keys():
    if key > last_timestamp:
        cursor.execute("INSERT INTO blockchain_data (ticker, blockchain_type, period,
min_value, max_value, average_amount, average_fee, average_gas_price, gas_used,
difficulty, transaction_amount, time) VALUES('BTC', 'BTC', '{period}', {min_value},
{max_value}, {average_amount}, {average_fee}, 0, 0, 0, {transaction_amount},
{time})".format(**results[key]))

    cursor.close()
    connection.commit()

load_data()

```

blockchain_screener.py

```

from web3 import Web3
from web3.middleware import geth_poa_middleware
import datetime
import time
from modules import database, helpers
import requests
import threading

periods = ['fifteen-minutes', 'hour', 'day']
networks = ['ETH', 'BNB']

rpc_endpoints = {
    'ETH': 'https://mainnet.infura.io/v3/480f3dbbf32747da8822fc2c8e4f3706',
    'BNB': 'https://bsc-dataseed1.defibit.io/'
}

```

```

}

def initialize_summary_data(data, token_name, gas_used, total_difficulty):
    data[token_name] = {}
    data[token_name]['min_value'] = 10 ** 100
    data[token_name]['max_value'] = 0
    data[token_name]['average_amount'] = 0
    data[token_name]['average_fee'] = 0
    data[token_name]['average_gas_price'] = 0
    data[token_name]['gas_used'] = gas_used
    data[token_name]['difficulty'] = total_difficulty
    data[token_name]['transaction_amount'] = 0

def aggregate_data(data, token_name, transaction):
    transactions_amount = data[token_name]['transaction_amount']

    value = transaction['value'] / 10 ** 9
    if value < data[token_name]['min_value']:
        data[token_name]['min_value'] = value

    if value > data[token_name]['max_value']:
        data[token_name]['max_value'] = value

    data[token_name]['average_amount'] += transaction.value / 10 ** 9 / transactions_amount
    data[token_name]['average_fee'] += transaction.gas / 10 ** 9 / transactions_amount
    data[token_name]['average_gas_price'] += transaction.gasPrice / 10 ** 9 /
transactions_amount

def get_block_summary(block, network, tokens = {}):
    data = {}
    initialize_summary_data(data, network, block.gasUsed, block.totalDifficulty)

    transactions_amount = len(block.transactions)
    data[network]['transaction_amount'] = transactions_amount

    for transaction in block.transactions:
        from_address = str(transaction['from']).lower()
        to_address = str(transaction['to']).lower()
        for key in tokens.keys():
            address = tokens[key]
            if address == '' or (from_address != address and to_address != address):
                continue

            if key not in data.keys():
                initialize_summary_data(data, key, block.gasUsed, block.totalDifficulty)

            data[key]['transaction_amount'] += 1

    for transaction in block.transactions:
        aggregate_data(data, network, transaction)
        from_address = str(transaction['from']).lower()
        to_address = str(transaction['to']).lower()
        for key in tokens.keys():
            address = tokens[key]
            if address == '' or (from_address != address and to_address != address):
                continue

```

```

        aggregate_data(data, key, transaction)

    return data

def get_closest_block(network, timestamp):
    url = (scan_urls[network] % timestamp)
    block_id = None
    try:
        block_id = int(requests.get(url = url, headers = headers).json()['result'])
    except:
        print(network, timestamp, "Closest block was not found")

    return block_id

def get_period_steps(period):
    if period == 'fifteen-minutes':
        return 180

    if period == 'hour':
        return 600

    return 3600

def load_blocks(network, provider, current_timestamp, period, tokens):
    timestamps = []

    ticks_per_period = helpers.get_ticks_per_period(period)
    for i in range (current_timestamp - ticks_per_period, current_timestamp,
get_period_steps(period)):
        timestamps.append(i)

    block_ids = []
    for timestamp in timestamps:
        block_id = get_closest_block(network, timestamp)
        if block_id != None:
            block_ids.append(block_id)
        time.sleep(0.25)

    blocks_summary = []
    for id in block_ids:
        block = provider.eth.get_block(id, True)
        blocks_summary.append(get_block_summary(block, network, tokens))
        time.sleep(0.25)

    count = len(blocks_summary)
    for token in tokens:
        max_value = 0
        min_value = 0
        average_amount = 0
        average_fee = 0
        average_gas_price = 0
        gas_used = 0
        difficulty = 0
        transaction_amount = 0
        for block in blocks_summary:

```

```

        if token not in block:
            continue

        max_value += block[token]['max_value']
        min_value += block[token]['min_value']
        average_amount += block[token]['average_amount']
        average_fee += block[token]['average_fee']
        average_gas_price += block[token]['average_gas_price']
        gas_used += block[token]['gas_used']
        difficulty += block[token]['difficulty']
        transaction_amount += block[token]['transaction_amount']

def get_end_timestamp(period):
    ts_now = round(datetime.datetime.now().timestamp())
    ticks_per_period = helpers.get_ticks_per_period(period)

    full_periods_count = ts_now // ticks_per_period

    return full_periods_count * ticks_per_period

def load_data(provider, network):
    for period in periods:
        settings = database.load_dataset_settings('blockchain data', period)
        if settings == None or network not in settings:
            continue

        tokens = settings[network]
        last_item = database.fetch_item("SELECT time FROM blockchain_data WHERE ticker='%s'
AND blockchain_type = '%s' AND period='%s' ORDER BY time DESC" % (network, network,
period))
        last_timestamp = 0 if last_item == None else last_item[0]
        missing_periods = helpers.get_missing_periods_count(last_timestamp, period)
        if missing_periods <= 0:
            continue

        ticks_per_period = helpers.get_ticks_per_period(period)
        current_timestamp = get_end_timestamp(period)
        while missing_periods > 0 and current_timestamp > last_timestamp:
            print('[ ' + network + ']', 'period:', period, 'count:', 2001 - missing_periods)
            load_blocks(network, provider, current_timestamp, period, tokens)
            current_timestamp -= ticks_per_period
            missing_periods -= 1

def load_data_wrapper():
    threads = []
    for network in networks:
        provider = Web3(Web3.HTTPProvider(rpc_endpoints[network], request_kwargs={'timeout':
60}))
        if network == 'BNB':
            provider.middleware_onion.inject(geth_poa_middleware, layer=0)

        threads.append(threading.Thread(target = load_data, args = (provider, network)))
    threads[-1].start()

for item in threads:
    item.join()

```