ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Кафедра інформаційних систем

Магістерська робота

на тему:

Інтелектуальна система прогнозування динаміки курсів криптовалют з використанням фрактального аналізу

Виконав: студент групи ПМІМ-22

спеціальності

122 "Комп'ютерні науки"

(шифр і назва спеціальності)

Бордун М.І.

(підпис) (прізвище та ініціали)

Керівник Соколовський Я.І.

(підпис) (прізвище та ініціали)

Рецензент Станкевич О.М.

(підпис) (прізвище та ініціали)

Львів – 2022

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет *прикладної математики та інформатики*
Кафедра *інформаційних систем*
Спеціальність *122 — комп'ютерні науки*
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри *проф. Шинкаренко Г.А*
"*5*" *вересня* **2022 року**

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Бордуну Михайлу Ігоровичу
(прізвище, ім'я, по батькові)

1 .Тема роботи *Інтелектуальна система прогнозування динаміки курсів криптовалют з використанням фрактального аналізу*

керівник роботи *Соколовський Ярослав Іванович, доктор технічних наук, професор*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від "*13*" *вересня* 20*22* року №*1*
2. Строк подання студентом роботи *12.12.2022 р.*

3. Вихідні дані до роботи *Література та інтернет ресурси за тематикою роботи,датасет криптовалютних даних*

4. Зміст магістерської роботи (перелік питань, які потрібно розробити)
1. Аналіз існуючого стану
2. Інформаційне забезпечення
3. Математичне та алгоритмічне забезпечення
4.Програмне забезпечення

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Схеми, діаграми, презентація дипломної роботи

6. Консультанти розділів роботи

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

7. Дата видачі завдання___*05.09.2022 р.*_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	*Аналіз існуючого стану*	*Вересень*	*Виконано*
2	*Інформаційне забезпечення*	*Вересень*	*Виконано*
3	*Математичне та алгоритмічне забезпечення*	*Жовтень*	*Виконано*
4	*Програмне забезпечення*	*Листопад*	*Виконано*
5	*Аналіз результатів*	*Листопад*	*Виконано*
6	*Оформлення роботи*	*Грудень*	*Виконано*
7			

Студент _____ *Бордун М.І.*
(підпис) **(прізвище та ініціали)**

Керівник роботи _____ *проф. Соколовський Я.І.*
(підпис) **(прізвище та ініціали)**

АНОТАЦІЯ

Магістерська робота студента Бордуна Михайла Ігоровича на тему: "Інтелектуальна система прогнозування динаміки курсів криптовалют з використанням фрактального аналізу" містить 55 сторінок, 12 джерел за списком використаної літератури та розділ додатків.

У цій роботі створено програмний продукт, фронтенд частиною якого відіграє Телеграм-бот. Запити обробляються бекенд-частиною застосунка з викликом до окремої програмної компоненти на мові програмування R для моделювання та прогнозування найпопулярніших криптовалют з використанням фрактальної моделі ARFIMA. Адаптовано алгоритм для ідентифікації фрактальних параметрів моделі. Реалізовано хмарну інтеграцію з платформою Microsoft Azure для забезпечення щоденного оновлення криптовалютних даних і зберігання їх в сховищі Blob storage.

Ключові слова: криптовалюта, часовий ряд, авторегресійна модель, ARIMA, довга пам’ять, фрактальна модель, ARFIMA, Python, R language, телеграм-бот, Microsoft Azure, API.

Master Thesis of student Bordun Mykhailo Ihorovych on the topic: "Intelligent system for predicting the dynamics of cryptocurrency rates using fractal analysis" contains 55 pages, 12 sources from the list of references and a section of appendices.

In this work, a software product was created, the frontend of which is based on the Telegram bot. Requests are processed by the backend part of the application with a call to a separate software component in the R programming language for modeling and forecasting the most popular cryptocurrencies using the ARFIMA fractal model. The algorithm for identifying the fractal parameters of the model has been adapted. Cloud integration with the Microsoft Azure platform has been implemented to ensure daily updating of cryptocurrency data and its storage in Blob storage.

Keywords: cryptocurrency, time series, autoregressive model, ARIMA, long memory, fractal model, ARFIMA, Python, R language, Telegram bot, Microsoft Azure, API.

ЗМІСТ

АНОТАЦІЯ 4

ТЕХНІЧНЕ ЗАВДАННЯ 8

ВСТУП 9

1. АНАЛІЗ ІСНУЮЧОГО СТАНУ 12

1.1 Короткі відомості з фрактального аналізу 12

1.2 Аналіз предметної області 12

1.3 Аналіз моделей прогнозування часових рядів (авторегресійні, фрактальні) 13

1.4 Аналіз програмних засобів 14

1.5 Висновки 15

2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ 16

2.1 Датасет для прогнозування динаміки курсів криптовалют 16

2.2 Дослідження даних для найпопулярніших криптовалют 17

2.3 Висновки 20

3. МАТЕМАТИЧНЕ ТА АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ 21

3.1 Дослідження наявності тренду, сезонності та білого шуму в часових рядах 21

3.2 Дослідження показників Херста, стаціонарності часових рядів та довгої пам'яті 22

3.3 Математичний опис фрактальної моделі ARFIMA 24

3.4 Ідентифікація оптимального параметра фрактального диференціювання моделі ARFIMA 25

3.5 Висновки 28

4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ 29

4.1 Програмна реалізація фрактальної моделі ARFIMA 29

4.2 Особливості реалізації задачі прогнозування для моделі ARFIMA 31

4.3 Реалізація програмного продукту у вигляді Telegram bot 34

4.4 Застосування хмарних технологій 39

4.4.1 Google Colab 39

4.4.2 Microsoft Azure 41
4.5 Висновки 44
ВИСНОВКИ 45
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 47
ДОДАТКИ 49
Додаток А. Реалізація R/S аналізу для знаходження показника Херста 49
Додаток Б. Реалізація фрактального диференціювання 50
Додаток В. Реалізація MA та AR моделей 52
Додаток Г. Реалізація інкрементального завантаження криптовалютних даних 54

ТЕХНІЧНЕ ЗАВДАННЯ

1. Підібрати набір історичних даних найбільш популярних криптовалют та проаналізувати обмежену вибірку з них для розуміння загальних тенденцій ринку та правильної очистки даних для подальших кроків.
2. Вибрати криптовалюту та провести декомпозицію її часового ряду та проаналізувати його на білий шум, стаціонарність та наявність довгої пам’яті.
3. З огляду на результати в другому пункті, провести підбір найбільш відповідних параметрів обраної фрактальної моделі для максимізації точності з огляду на метрику RMSE.
4. Адаптувати алгоритм для визначення фрактальних характеристик моделі.
5. Порівняти ефективність створеної фрактальної моделі з такою ж моделлю з автоматичним підбором параметрів, а також з найбільш відповідною авторегресійною моделлю на різних розмірах тренувальних та тестових даних. Проаналізувати результати.
6. Виконати виміри з використанням хмарних технологій та порівняти час моделювання фрактальними методами.
7. Виконати реалізацію програмного продукту для інтелектуальної системи з використанням Телеграм-боту.
8. Інтегрувати хмарні обчислення в програмний продукт для забезпечення безперебійного оновлення вхідних даних, що використовуються фрактальною моделлю.

ВСТУП

Актуальність завдання. В останні декілька років криптовалюти набувають все більшої популярності, особливо це стосується Bitcoin, який є ключовим рушієм для всього крипторинку. Криптовалюта є більш безпечною альтернативою фізичній валюті, яку ми зараз використовуємо. Це все було здобуто завдяки шифруванню передачі конфіденційних даних за допомогою криптографічних протоколів, які є надзвичайно складними кодовими системами.

Серед основних переваг використання криптовалютних транзакцій є, як правило, швидкість та простота використання. Наприклад, Bitcoin можна перевести з одного цифрового гаманця в інший, використовуючи тільки смартфон або комп'ютер. Кожна така транзакція записується у загальнодоступному списку, який називається Blockchain, що дає змогу простежити історію транзакцій та дозволяє зробити децентралізацію всіх транзакцій, що також означає, що будь-яке втручання (державне чи приватне) в Blockchain неможливе. Зацікавленість в даній сфері додають прогресивні компанії, які починають приймати криптовалюти як спосіб онлайн-оплати, серед яких нараховуються Microsoft, Overstock, StarBucks, Tesla та інші. З огляду на перспективність цього ринку, все більше людей вбачають в ньому потужний інвестиційний інструмент. Проте ринок криптовалют досить складний і не має конкретних регуляційних важелів, що і зумовлює також інтерес до його аналізу та прогнозування.

Для того, щоб проаналізувати та прогнозувати історичні дані, а особливо фінансові, найчастіше використовують авторегресійні моделі. Однак, аналіз наукових публікацій свідчить про те, що фрактальні моделі користуються популярністю в науковій спільноті, оскільки дозволяють врахувати наявність довготривалої пам'яті у процесах. Проте варто сказати, що зацікавленість до фрактального аналізу зростає, і це вже питання часу, коли в таких популярних

мовах програмування як Python з’являться готові рішення для такого роду аналізу, що однозначно знизить вхідний поріг і збільшить його популярність.

Об’єктом дослідження є дані курсів криптовалют, які представлені у вигляді впорядкованих послідовностей точок даних, розподілених на певний період часу.

Предметом дослідження є власне програмне та алгоритмічне забезпечення для аналізу динаміки курсів криптовалют та їх прогнозування на підставі адаптованих алгоритмів фрактального аналізу.

Отже, **метою роботи** є розроблення програмного та алгоритмічного забезпечення інформаційної системи, достатньо надійної для того, щоб допомогти інвестору в аналізі та прогнозуванні динаміки обраної ним криптовалюти.

Для досягнення поставленої мети у роботі виконано такі **завдання**:

- Підібрано набір даних найбільш популярних криптовалют та проаналізувано обмежену вибірку для правильної фільтрації даних;
- Синтезовано математичну модель фрактального часового ряду криптовалют, проведено декомпозицію ряду, проаналізовано його на білий шум, стаціонарність і наявність довгої пам’яті;
- Проведено підбір відповідних параметрів фрактальної моделі для максимізації точності з використанням метрики RMSE;
- Адаптовано алгоритм для ідентифікації фрактальних параметрів моделі ARFIMA;
- Порівняно ефективність створеної фрактальної моделі з такою ж моделлю з автоматичним підбором параметрів і авторегресійною моделлю на різних розмірах тренувальних та тестових даних. Виконано виміри з використанням хмарних технологій і порівняно час моделювання фрактальними методами;
- Реалізовано програмний продукт для інтелектуальної системи з використанням Телеграм-боту. Інтегровано хмарні обчислення для забезпечення щоденного оновлення криптовалютних даних.

Практичне значення роботи полягає в реалізації системи, яка буде використовуватися інвесторами-аматорами та звичайними людьми для прогнозування вибраної їм криптовалюти з використанням сучасного підходу фрактального моделювання.

Наукова новизна роботи. Синтезовано математичну модель фрактального часового ряду криптовалют, який дозволяє враховувати ефекти довготривалої пам’яті у процесі прогнозування; Адаптовано алгоритм для ідентифікації фрактальних параметрів моделі ARFIMA.

Апробація.

[1] Бордун М. Інтелектуальна система прогнозування динаміки курсів криптовалют з використанням фрактального аналізу / М. Бордун // Міжнародна студентська наукова конференція з прикладної математики та комп’ютерних наук (МСНКПМКН-2022), 5-6 травня 2022 р. – Львів:2022. – С. 79- 83. Режим доступу: https://ami.lnu.edu.ua/wp-content/uploads/2022/05/ISSCAMCS-2022.pdf

[2] Bordun M. DEVELOPMENT OF SOFTWARE AND ALGORITHMIC SECURITY FOR FORECASTING THE CRYPTOCURRENCY COURSE USING FRACTAL ANALYSIS METHODS / M. Bordun, Y. Sokolovskyy, M. Levkovych // XXX International Ukrainian-Polish Conference CAD IN MACHINERY DESIGN IMPLEMENTATION AND EDUCATIONAL ISSUES, 1 - 2 December, 2022, Lviv, Ukraine. – P. 10.

[3] Ya. I. Sokolovskyy, M. I. Bordun, M. V. Levkovych. DEVELOPMENT OF SOFTWARE AND ALGORITHMIC SECURITY FOR FORECASTING THE CRYPTOCURRENCY COURSE USING FRACTAL ANALYSIS METHODS// Computer Design Systems. Theory and Practice , Vol. 4, No. 1, 2022. P.81-94.

1. АНАЛІЗ ІСНУЮЧОГО СТАНУ

1.1 Короткі відомості з фрактального аналізу

Фрактальний аналіз — відносно молода і швидко прогресуюча галузь сучасної математики, яка засобами теорії мір дробових порядків, метричних розмірностей, операторів дробового інтегрування та диференціювання вивчає властивості математичних об'єктів зі складною локальною будовою. Виявлено тісні зв'язки деяких критичних показників складних систем з її фрактальними властивостями. Сьогодні нехтувати мікроструктурами і мікрофлуктуаціями реальних об'єктів, процесів і явищ — це, по меншій мірі, спотворювати істинну природу речей. Це особливо стосується фінансового сектору, який є досить нестаціонарний, тобто хаотичний, і стандартні методи глибинного навчання не можуть зафіксувати всі закономірності для подальшого ефективного прогнозування [4].

1.2 Аналіз предметної області

Предметною областю виступає ринок криптовалют, який досить недавно почав бути у всіх на слуху. Вперше термін «криптовалюта» почав використовуватися після появи платіжної системи Bitcoin, яка була розроблена в 2009 році людиною або групою осіб під псевдонімом Сатосі Накамото і використовує хешування SHA-256 і систему proof-of-work. Цікавим є те, що система ціни криптовалют забезпечена потужностями, які використовуються для її видобутку і всією іншою інфраструктурою, створеною для обслуговування транзакцій. Окрім цього, вона забезпечена попитом інвесторів. Щодо кількості криптовалют на ринку, то ця сума вже перевищує 17800 станом на березень 2022 базуючись на біржу https://coinmarketcap.com/ і цей феномен продовжує дивувати, однак ніхто насправді не контролює технологію блокчейн. Кожен, хто володіє знаннями про те, як працює технологія, може використовувати її для розробки власної віртуальної валюти. Також збільшення кількості криптовалют було спричинено зростанням популярності в свій час Ethereum з еко-системою основаній

на смарт-контрактах DeFi (Decentralized finance), що спонукало до створення різних типів криптовалют для вирішення вже існуючих проблем блокчейну, включаючи конфіденційність, масштабованість, швидкість транзакцій та високі витрати на газ.

1.3 Аналіз моделей прогнозування часових рядів (авторегресійні, фрактальні)

Аналіз часового ряду відноситься до визначення загальних закономірностей, які відображаються даними за певний період часу. Серед найпопулярніших методів моделювання та прогнозування часового ряду є авторегресійні моделі, найяскравішим представником серед яких виступає ARIMA (AutoRegressive Integrated Moving Average). Під час процесу моделювання ми знаходимо 3 параметри [5]. Autoregressive (AR) порядку p, який описує число значущих затримок часового ряду, порядок інтегрованості ряду d та Moving Average (MA) порядку p, що описує кількість значущих помилок прогнозу. Узагальненням цієї моделі є фрактальна модель ARFIMA (AutoRegressive Fractionally Integrated Moving Average), яка дає можливість моделювання часових рядів з довгою пам’яттю. Тобто показник d може набувати нецілих значень. Загалом такого виду узагальнення дозволяє здійснювати необхідний аналіз різних часових рядів з врахуванням тривалого шоку в часовому ряді, оскільки модель ARIMA позволяє фіксувати процеси або з короткою пам’яттю з d = 0 або з нескінченною при d = 1.

У своєму дослідженні [6], присвяченому порівнянню моделювання стаціонарних часових рядів за допомогою процесів ARMA і ARFIMA, Андерсон (1998), застосовуючи симуляції Монте Карло і порівнюючи помилки прогнозів, показує, що ігнорування довгої пам'яті, коли вона насправді має місце, призводить до серйознішого погіршення результатів, ніж її накладення за відсутності такої. Це спостереження є вкрай важливим, оскільки на практиці дослідникові ніколи не відомо, який процес у дійсності лежить в основі динаміки цін фінансових активів. На основі наведених вище міркувань можна розглядати використання процесів

ARFIMA як один з найбільш сучасних та актуальних підходів для вивчення фінансових часових рядів.

1.4 Аналіз програмних засобів

Дослідження, пов’язані з аналізом часових рядів та побудови фрактального диференціювання, були проведені за допомогою мови програмування Python версії 3.6.5 з використанням бібліотек pandas версії 1.1.3 та numpy версії 1.19.2. Також з використанням цієї мови програмування та бібліотеки pyTelegramBotAPI версії 4.8.0 було реалізовано Телеграм бот. Для прогнозування часових рядів була використана мова програмування R версії 4.1.3, разом з бібліотеками forecast версії 8.16 та arfima версії 1.8.0. Розділення на дві мови програмування для реалізації цієї роботи було вимушеним, оскільки ще поки немає в таких популярних мовах програмування як Python бібліотек пов’язаних з роботою з моделлю ARFIMA чи іншими фрактальними моделями.

Також було використано хмарні технології, а саме Google Colab та Microsoft Azure.Colab було використано разом з сховищем даних Google Drive. Він надає інтерактивне середовище програмування подібне до Jupyter Notebook, з яким одразу ж можна починати роботу. Є можливість вибору мови програмування як і Python, так і R. В моєму випадку було створене R-середовище на базі останньої версії R, а саме 4.1.3, в якому я підключався до даних розміщених на Google Drive і загалом повторював кроки пов’язані з моделюванням та прогнозуванням, які попередньо виконав на власних потужностях ПК.

Щодо Microsoft Azure, то це є публічна платформа для хмарних обчислень корпорації Microsoft. Вона надає широкий набір сервісів включаючи обчислювальні, аналітичні, мережеві та сховища даних. Це надає клієнтам достатню гнучкість у використанні бажаних технологій, однак особливою перевагою використання цієї хмарної платформи, це зручна інтеграція з іншими продуктами компанії Microsoft. Azure пропонує чотири різні форми хмарних обчислень:

- Infrastructure as a service (IaaS);
- Platform as a service (PaaS);
- Software as a service (SaaS);
- Serverless functions.

Корпорація Microsoft стягує плату за Azure на основі оплати за використання (pay-as-you-go), тобто передплатники щомісяця отримують рахунок лише за певні ресурси та послуги, якими вони користуються.

1.5 Висновки

Розглянуто загальні відомості щодо предметної області та методів прогнозування часових рядів. Обґрунтовано доцільність використання фрактальних методів, що виходить з важливості врахування довгої пам’яті при аналізі часових рядів. Надано перелік програмних засобів для реалізації алгоритмів та моделей прогнозування, а також хмарних сервісів, що дозволяють оптимізувати виконання кінцевого програмного продукту, а також забезпечити надійну синхронізацію даних.

2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Датасет для прогнозування динаміки курсів криптовалют

Датасет був взятий з платформи Kaggle [7]. Дані для його створення були отримані між 28 квітня 2013 аж до 7 липня 2021 року та взяті з біржі https://coinmarketcap.com/. Датасет має по одному csv-файлу для кожної криптовалюти. Було обрано 5 найбільш популярних криптовалют за версією https://www.bankrate.com/investing/types-of-cryptocurrency/, а саме Bitcoin, Ethereum, Binance Coin, Cardano, однак без врахування стейб-коінів, ціна яких прив'язана до доларів США, а також замість криптовалюти Ripple було обрано менш популярний Stellar, який базований на протоколі Ripple. Дані щодо кожної валюти бралися щоденно і описуються декількома характеристиками, такими як дата, відкриваюча ціна на ринку, максимальна, мінімальна, закриваюча ціна, об'єм транзакцій, а також капіталізація ринку. З рисунку 1.1 можна більш детально подивитися на метадані, які описують криптовалюту Bitcoin.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2991 entries, 0 to 2990
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   SNo        2991 non-null   int64
 1   Name       2991 non-null   object
 2   Symbol     2991 non-null   object
 3   Date       2991 non-null   object
 4   High       2991 non-null   float64
 5   Low        2991 non-null   float64
 6   Open       2991 non-null   float64
 7   Close      2991 non-null   float64
 8   Volume     2991 non-null   float64
 9   Marketcap  2991 non-null   float64
dtypes: float64(6), int64(1), object(3)
memory usage: 233.8+ KB
```

Рисунок 1.1.Метадані для датасету криптовалюти Bitcoin

Потрібно було вирішити проблему різного розміру історичних даних для обраних криптовалют, оскільки різні криптовалюти з'являлись в різний час. Було взято розмір найсучаснішої з обраних криптовалют, а саме, Cardano(ADA), історія

транзакцій якої починається з 3 жовтня 2017 року. З рисунку 1.2 можна побачити остаточну кількість даних для подальшого аналізу. Також це рішення є корисним в сенсі побудови моделі прогнозування, оскільки лише в 2017 році стався крипто-бум і попередні історичні дані слабо відображали патерни ринку сьогодення.

```
Binance Coin -> Rows: 1374, Cols: 10
Bitcoin -> Rows: 1374, Cols: 10
Cardano -> Rows: 1374, Cols: 10
Ethereum -> Rows: 1374, Cols: 10
Stellar -> Rows: 1374, Cols: 10
```

Рисунок 1.2. Остаточна кількість даних після очистки для часових рядів найпопулярніших криптовалют

2.2 Дослідження даних для найпопулярніших криптовалют

Для більш детального розуміння загальних патернів крипторинку необхідно визначити певні залежності між його елементами. Так, наприклад, було проаналізовано кореляцію Пірсона між обраними в попередньому розділі криптовалютами. Загалом кореляція Пірсона є показником лінійної залежності між двома змінними X та Y, який набуває значень від −1 до +1 включно. Він широко використовується в науці для вимірювання ступеня лінійної залежності між двома змінними. Можна побачити матрицю цих показників залежностей з рисунку 2.1 і дійти до висновку, що ринок є дуже зв’язаний, а особливо це стосується найпопулярніших криптовалют, для яких кореляція не була нижча за показник 0.86.

Рисунок 2.1. Кореляція Пірсона для часових рядів найпопулярніших криптовалют

Також цікаво зауважити, що хоч і більшість інформаційного простору пов’язаного з крипторинком займає Bitcoin, однак на поточний стан для інвесторів ця валюта не настільки є прибутковою в короткій та середній перспективі. З рисунку 2.2 можна побачити коефіцієнт повернення для обраних криптовалют, який демонструє ефективність інвестицій показуючи відношення між поточним значенням криптовалюти та початковим. В даному випадку виділяється Binance Coin, який в один момент майже в 350 разів перевищував своє початкове значення, в той час як Bitcoin навіть не входить в трійку лідерів.

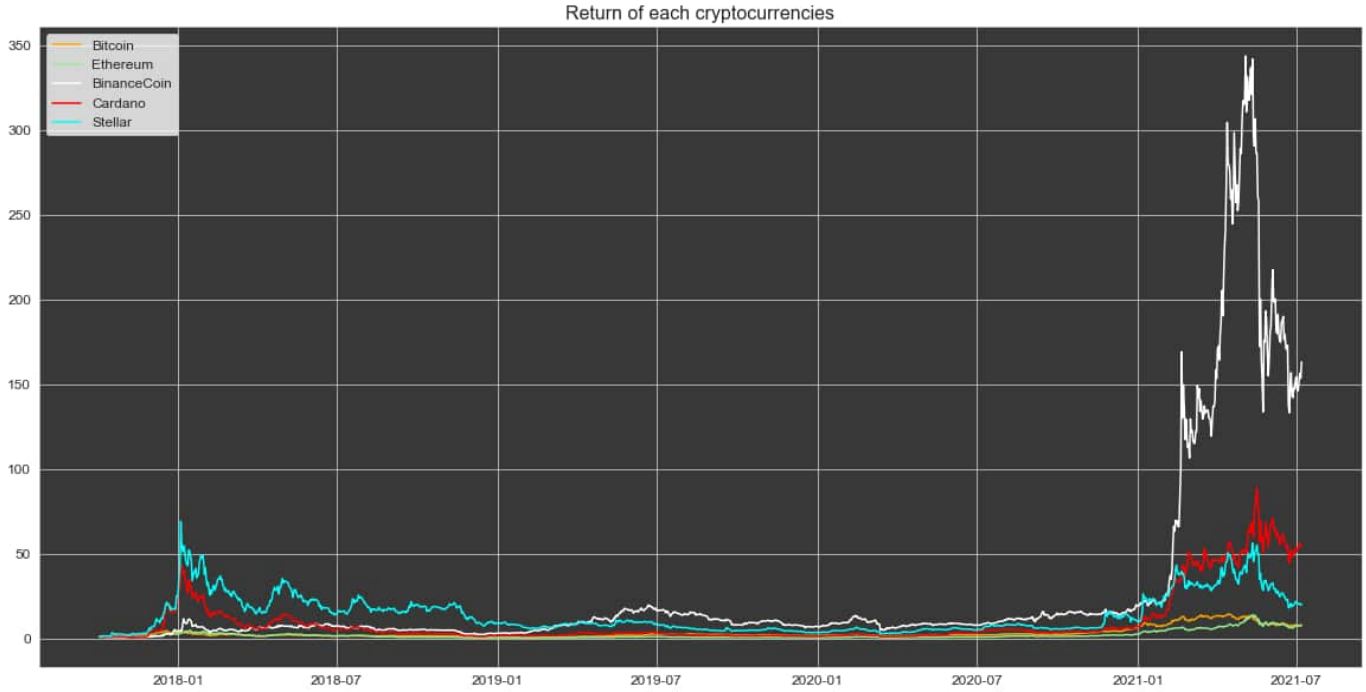


Рисунок 2.2. Коефіцієнт повернення для часових рядів найпопулярніших криптовалют

Не менш важливо розуміти стабільність обраних валют, тобто яка величина денної флуктуації. Очевидно, що для Bitcoin така величина буде найбільшою, оскільки він найдорожчий, однак відхилення в межах від -8000 до +8000 доларів США лишній раз демонструє те, що ринок досить нестабільний і не є строго регульований.

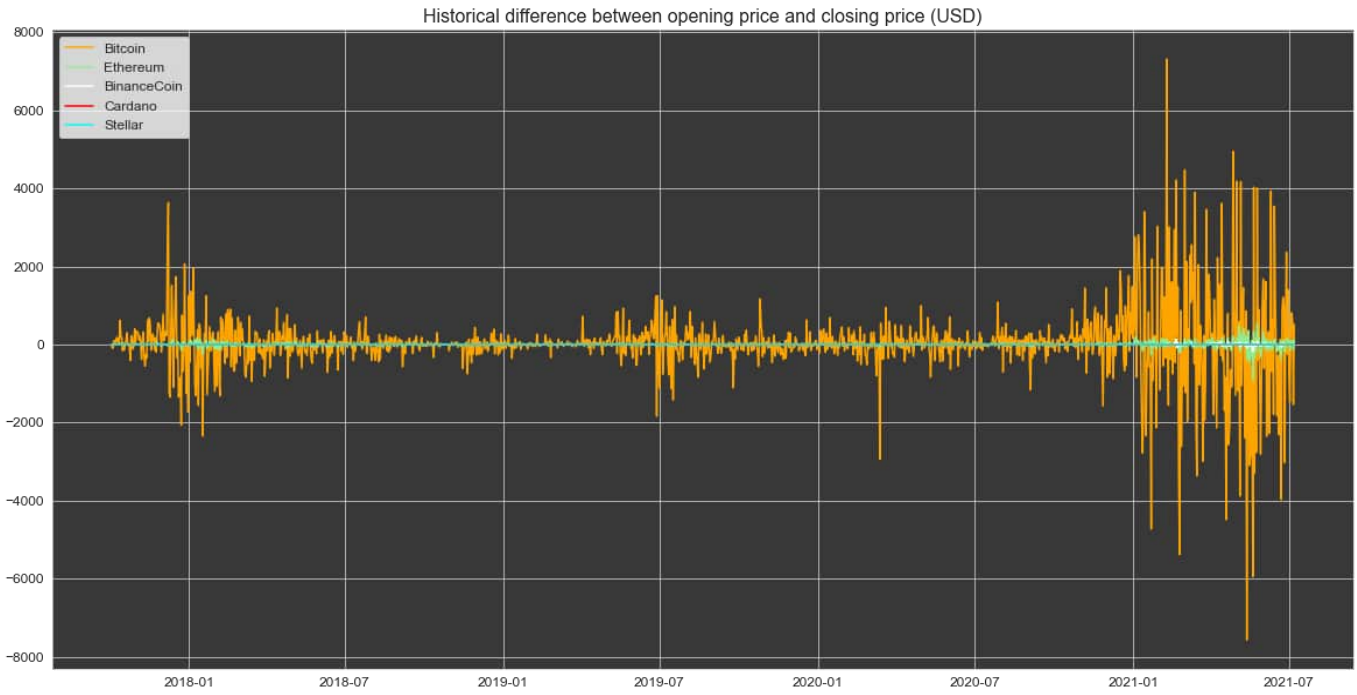


Рисунок 2.3. Флуктуація між відкриваючою та закриваючою денною ціною часових рядів найпопулярніших криптовалют

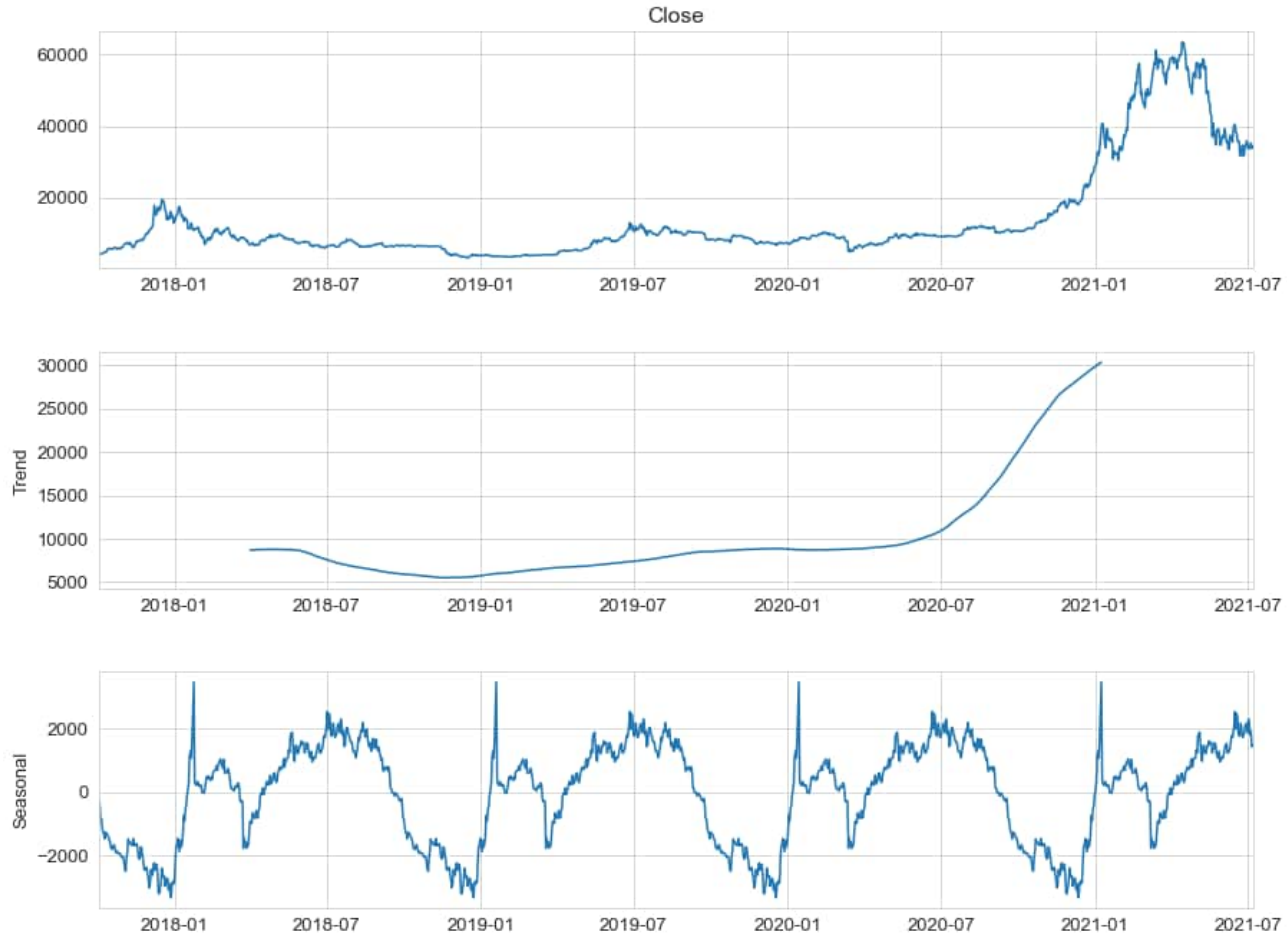
2.3 Висновки

Проведено загальні дослідження задля розуміння предметної області для 5 найбільш популярних криптовалют за версією https://www.bankrate.com/investing/types-of-cryptocurrency/, а саме Bitcoin, Ethereum, Binance Coin, Cardano, однак без врахування стейб-коінів, ціна яких прив’язана до доларів США, а також замість криптовалюти Ripple було обрано менш популярний Stellar, який базований на протоколі Ripple. Аналітика стосувалася кореляції Пірсона, коефіцієнту повернення, а також денної флуктуації для часових рядів вищезгаданих валют.

3. МАТЕМАТИЧНЕ ТА АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Дослідження наявності тренду, сезонності та білого шуму в часових рядах

Перед початком моделювання та прогнозування необхідно визначити певні властивості часових рядів, щоб розуміти доцільність використання конкретних моделей машинного навчання. Для подальшого аналізу було обрано найпопулярнішу криптовалюту, а саме Bitcoin. Так першочерговим кроком виступає декомпозиція часового ряду, яку можна побачити на рисунку 3.1, створена завдяки модулю statsmodels.api мови програмування Python. З неї можна вивести наявність тренду та чітко виражену сезонність.



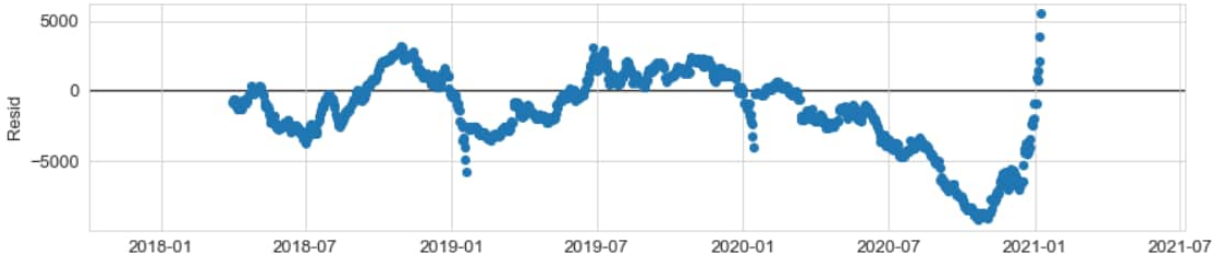


Рисунок 3.1. Декомпозиція часового ряду ціни Bitcoin

Щодо останнього графіку з рисунку 3.1, то він характеризує надлишки, тобто показує шум в нашому часовому ряді. В ідеалі він повинен мати патерни білого шуму, який характеризується нульовим середнім значенням, константною дисперсією і незалежністю змінних [8]. Тобто це фактично є випадковий набір чисел і це дозволяє при моделюванні моделей машинного навчання зафіксувати всі необхідні сигнали тренувальної вибірки, що максимізує ефективність алгоритму. Однак в нашому випадку бачимо, що хоч і середнє значення є близьке до 0, дисперсія не є стабільною і можна сміливо відкинути гіпотезу про наявність білого шуму в графіку надлишків.

3.2 Дослідження показників Херста, стаціонарності часових рядів та довгої пам’яті

Наступним етапом аналізу є знаходження показника Херста, який є одним з найбільш популярних методів оцінки пам’яті в часовому ряді. Автор цього методу Харольд Херст першим дослідив поняття довгої пам’яті коли досліджував притоки річки Ніл і оптимальні розміри резервуарів води. Загалом значення цього показника є в межах від 0 до 1. Значення експоненти Херста для часового ряду ціни Bitcoin було визначено завдяки R/S аналізу [9], код якого наведений в додатку А. В даному випадку отримали значення 0.9347, яке вказує, що часовий ряд є з довгостроковою позитивною автокореляцією, що означає, що за високим значенням у ряді, ймовірно, буде наступне інше високе значення. Тобто підтверджується наявність тренду та довгої пам'яті.

Також з рисунку 3.2 бачимо графік ACF, тобто графік автокореляцій, який показує значення кореляції між часовим рядом і його версією з відставанням при кожному значенні затримки. Чітко виражена гіпербола на графіку лишній раз підтверджує наявність довгої пам’яті. Варто зауважити, що потрібно близько 300 часових затримок, щоб повністю нівелювати ефект залежності.

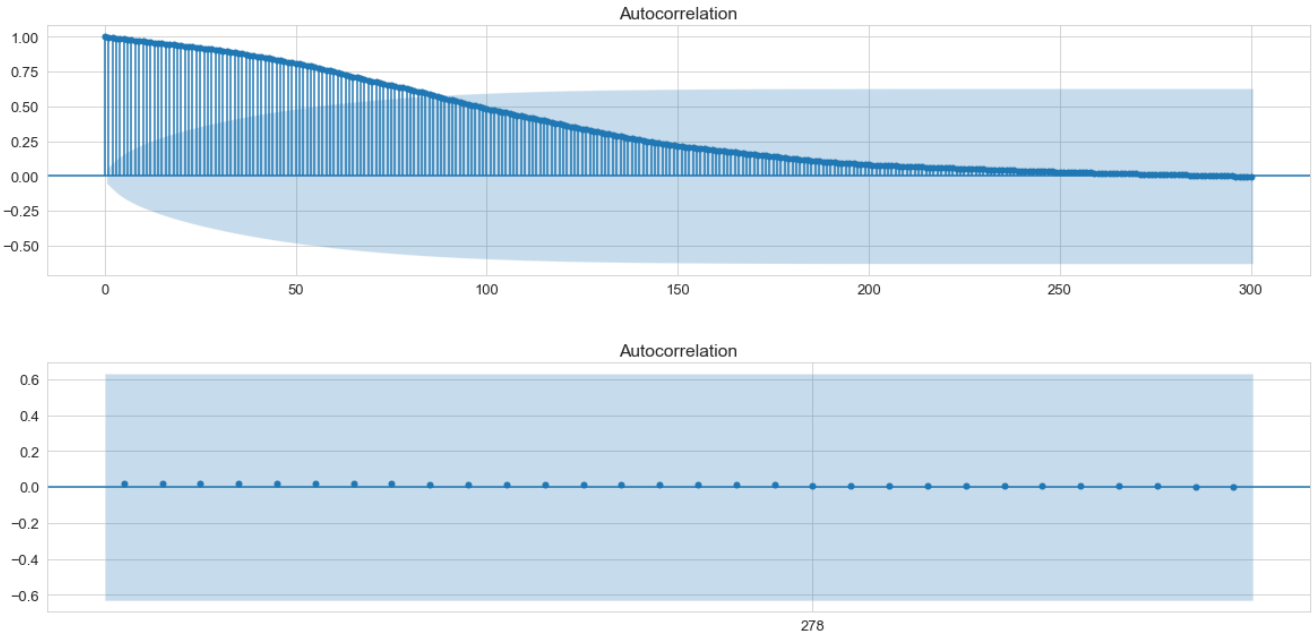


Рисунок 3.2. Функція ACF для часового ряду ціни Bitcoin

Щодо визначення стаціонарності, то існують два популярних статистичних теста ADF та KPSS [3]. Загалом кожен з них перевіряє протилежні гіпотези і результати яких є доповнювальними один до одного, тобто неможливість відкидання нульової гіпотези щодо нестаціонарності тестом ADF та відкидання цієї гіпотези щодо стаціонарності тестом KPSS служить важливим індикатором наявності одиничного кореня, тобто процесу I(1), що показує нестаціонарність часового ряду. Саме такий результат і був отриманий з використанням модуля statsmodels.tsa.stattools мови програмування Python, що показав p-значення статистики ADF 0.832468, яке є більшим за поріг 0.05, що служить підтвердженням нульової гіпотези. І p-значення статистики KPSS становить менше ніж 0.01, що є менше за поріг 0.05 і відкидає нульову гіпотезу для нашого часового ряду.

3.3 Математичний опис фрактальної моделі ARFIMA

[2] Autoregressive (AR) модель порядку p:

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \varepsilon_t \,, \tag{3.1}$$

де $\phi_1, \dots, \phi_p$ авторегресійні параметри, *c* константа і випадкова змінна ε_t це є білий шум.

Moving Average (MA) модель порядку q:

$$X_t = \mu + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t \,, \tag{3.2}$$

де $\theta_1, \dots, \theta_p$ параметри рухомого середнього, μ константа і випадкові змінні ε_t, ε_{t-1}, ... це білий шум.

Узагальненням перелічених вище моделей виступає:

$$(1 - \sum_{i=1}^{p} \phi_i B^i)(1 - B)^d (X_t - \mu) = (1 + \sum_{i=1}^{q} \theta_i B^i)\varepsilon_t \,, \tag{3.3}$$

де $(1 - B)^d$ називають оператором диференціювання. Моделі ARMA та ARIMA можуть фіксувати тільки процеси з короткою пам'яттю, оскільки параметр *d* набуває тільки цілочислових значень. Тому для того, щоб фіксувати процеси з довгою пам'яттю потрібно використати фрактальну модель ARFIMA(p,d,q), де показник d набуває вже дробових значень.

Ми також можемо розкласти оператор диференціювання використовуючи біноміальний розклад для будь-якого дійсного числа d:

$$(1 - B)^d = \sum_{k=0}^{\infty} \binom{d}{k} (-B)^k = 1 - dB + \frac{d(d-1)}{2!} B^2 - \frac{d(d-1)(d-2)}{3!} B^3 + \dots \tag{3.4}$$

[9] Також можна представити оператор диференціювання через Гамма-функцію: $(1 - B)^d = \sum_{k=0}^{\infty} \left(\frac{\Gamma(k-d)}{\Gamma(k+1)\,\Gamma(-d)}\right)(-B)^k$,(5)

де Γ(.) позначає Гамма-функцію і представляється через Γ(z) = $\int_0^\infty x^{z-1}e^{-x}\,dx$. При d=0, X_t виступає просто білим шумом, а його автокореляційна функція дорівнює 0. Коли d=1, X_t є випадковим блуканням, значення автокореляційної функції якого дорівнює 1, і його можна розглядати як білий шум після диференціації першого порядку, яка полягає в різниці попереднього значення від поточного в часовому ряді. Коли d є вже дійсним числом, $X_t = -\sum_{k=1}^{\infty}\left(\frac{\Gamma(k-d)}{\Gamma(k+1)\,\Gamma(-d)}\right)X_{t-k} + \varepsilon_t$, і, отже, на X_t впливають усі історичні дані (X_{t-1}, X_{t-2}, …).

3.4 Ідентифікація оптимального параметра фрактального диференціювання моделі ARFIMA

[10] Різноманітні методи оцінки параметра фрактального диференціювання все частіше згадуються в науковій літературі пов’язаній з аналізом часових рядів. Загалом, вони можуть класифікуватися в дві групи – параметричні та напівпараметричні. У першій групі найпопулярніші методи включають в собі функцію правдоподібності. В другій найбільш популярний, який зазвичай називають методом GPH, був запропонований авторами Geweke та Porter-Hudak. Суть напівпараметричного підходу полягає в оцінці параметрів моделі в два етапи: окремо оцінюється параметр d, а вже потім відбувається оцінка інших параметрів, а не одночасно як для методів першої групи.

Оскільки навіть модель ARFIMA з бібліотеки forecast мови програмування R з автоматичним підбором параметрів згідно документації використовує напівпараметричний підхід, то детальніше опишемо метод GPH.

[11] GPH метод починається з оцінки параметру d. Він заснований на регресії найменших квадратів в спектральній області, експлуатує зразок форми із полюсів спектральної густини при початку координат:

$$f_X(\lambda) \sim \lambda^{-2d}, \lambda \to 0$$

Щоб продемонструвати цей метод, ми можемо написати функцію спектральної щільності стаціонарної моделі $X_t, t = 1, \dots, T$ як

$$f_X(\lambda) = \left[4sin^2(\frac{\lambda}{2})\right]^d \ f_\varepsilon(\lambda). \quad (3.5)$$

де $f_\varepsilon(\lambda)$ є спектральною щільністю ε_t, припускаючи що вона скінченна і неперервна функція на проміжку $[-\pi;\ \pi]$;

Логарифм з функції спектральної щільності можна представити наступним чином

$$\log\{f_X(\lambda)\} = \log\{f_\varepsilon(0)\} - d\log\left\{4sin^2\left(\frac{\lambda}{2}\right)\right\} + \log\left\{\frac{f_\varepsilon(\lambda)}{f_\varepsilon(0)}\right\}. \quad (3.6)$$

Нехай $I_X(\lambda_j)$ є періодограмою виконаною на частотах Фур'є $\lambda_j = \frac{2\pi j}{T}, j = 1,2, \dots, m,$

T є числом досліджень та m є числом розглянутих частот Фур'є, що є кількістю ординат періодограми, що будуть використовуватися в регресії.

$$\log\{I_X(\lambda_j)\} = \log\{f_\varepsilon(0)\} - d\log\left\{4sin^2\left(\frac{\lambda_j}{2}\right)\right\} + \log\left\{\frac{f_\varepsilon(\lambda_j)}{f_\varepsilon(0)}\right\} + \log\left\{\frac{I_X(\lambda_j)}{f_X(\lambda_j)}\right\}. \quad (3.7)$$

де $\log\{f_\varepsilon(0)\}$ константа, $\log\left\{4sin^2\left(\frac{\lambda_j}{2}\right)\right\}$ екзогенна змінна та $\log\left\{\frac{I_X(\lambda_j)}{f_X(\lambda_j)}\right\}$ це помилка.

Можна відзначити, що вибір m є важливим питанням, оскільки він сильно впливає на результати оцінювання. З одного боку, m має бути достатньо мала, щоб розглядати лише частоти, близькі до нуля. З іншого боку, m має бути достатньо велика, щоб забезпечити збіжність оцінки методу найменших квадратів.

Оцінка GPH потребує два головних припущення пов'язаних з асимптотичною поведінкою рівняння (3):

H1: для низьких частот, ми припускаємо що $\log\left\{\frac{f_\varepsilon(\lambda_j)}{f_\varepsilon(0)}\right\}$ є незначною

H2: випадкові змінні $\log\left\{\frac{I_X(\lambda_j)}{f_X(\lambda_j)}\right\}$ $j = 1,2,\dots,m$ є асимптотично незалежні однаково розподілені (н.о.р) випадкові величини.

Під гіпотезами H1 та H2 ми можемо написати лінійну регресію

$$\log\{I_X(\lambda_j)\} = \alpha - d\log\left\{4sin^2\left(\frac{\lambda_j}{2}\right)\right\} + e_j, \tag{3.8}$$

де $e_j \sim$ н. о. р. $(-c, \frac{\pi^2}{6})$. Нехай Y_j = - $\log\left\{4sin^2\left(\frac{\lambda_j}{2}\right)\right\}$, GPH оцінка це є оцінка методу найменших квадратів для регресії $\log\{I_X(\lambda_j)\}$ на константах α та Y_j. Оцінка показника d, позначена як $\hat{d}_{GPH}$ визначається наступним чином

$$\hat{d}_{GPH} = \frac{\sum_{j=1}^{m}(Y_j - \bar{Y})\ \log\{I_X(\lambda_j)\}}{\sum_{j=1}^{m}(Y_j - \bar{Y})^2} \tag{3.9}$$

де $\bar{Y} = m^{-1}\sum_{j=1}^{m}(Y_j)$ та $m = g(T)$ з $\lim_{T\to\infty} g(T) = \infty$ та $\lim_{T\to\infty} g(T)/T = 0$.

Geweke та Porter-Hudak показали, що якщо $T \longrightarrow \infty$ та $|d| < \frac{1}{2}$ маємо

$$\sqrt{m}\left(\hat{d}_{GPH} - d\right) \sim N\left[0, \frac{\pi^2}{6}\left\{\sum_{j=1}^{m}(Y_j - \bar{Y})^2\right\}^{-1}\right] \tag{3.10}$$

Porter-Hudak (1990), Crato та de Lima (1994), показали, що параметр m повинен бути вибраний так, що $m = T^{\upsilon}$, $\upsilon = 0.5, 0.6, 0.7$. Згідно з припущенням нормальності для X_t, було доведено, що результуюча оцінка є узгодженою та асимптотично нормальною, так що оцінене середнє квадратичне відхилення можна використовувати для висновку.

З огляду на використання цього методу з тренувальною вибіркою часового ряду ціни криптовалюти Bitcoin, то можемо побачити з таблиці 1.1 порівняння оцінок та їхніх середніх квадратичних відхилень. В підсумку це тільки ще раз підтверджує факт нестаціонарності часового ряду і високої волатильності ринку, оскільки d ≥ 1/2, а також те що необхідно брати достатньо велику вибірку для аналізу, щоб отримати прийнятну похибку.

Розмір вибірки	Оцінка параметру d	Середнє квадратичне відхилення
1845	1.3232	0.1123
1344	1.3048	0.1120
500	1.1757	0.1951

Таблиця 3.1. Порівняння оцінок параметру d та їхньої точності для різного розміру вибірки часового ряду ціни Bitcoin

3.5 Висновки

Досліджено часовий ряд криптовалюти Bitcoin на такі характеристики як наявність довгої пам’яті, тренду, сезонності, білого шуму та стаціонарності. Довгу пам’ять було виявлено як з графіку автокореляцій, так і за допомогою значення показника Херста. Також описано математичний апарат для фрактальної моделі ARFIMA та алгоритм GPH для оцінки параметра фрактального диференціювання, що є ключовим параметром вищезгаданої моделі.

4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Програмна реалізація фрактальної моделі ARFIMA

Для більшого розуміння внутрішньої будови моделі ARFIMA я побудував модель, яка може симулювати часовий ряд з необхідними для моделі ARFIMA властивостями. В першу чергу це врахування ефектів довгої пам'яті при чому, щоб часовий ряд був стаціонарним. Стаціонарність є необхідною характеристикою, оскільки термін «авторегресивний» в ARFIMA означає, що це модель лінійної регресії, яка використовує власні затримки як предиктори при моделюванні. Моделі лінійної регресії, як відомо, найкраще працюють, коли предиктори не корелюють і незалежні один від одного. Найпростішим методом створення стаціонарного ряду є вже згадана в попередньому пункті звичайна диференціація. Однак таким чином ми можемо занадто передиференціювати наш ряд і втратити всю необхідну пам’ять, тому виникає питання: яка мінімальна величина диференціації, яка робить ціновий ряд стаціонарним і зберігає якомога більше пам'яті?

Для вирішення цієї проблеми використано фрактальне диференціювання, яке лежить в основі моделі ARFIMA. Алгоритм фрактального диференціювання включає два етапи: визначення ваг та застосування визначених ваг у добутку з нашим часовим рядом використовуючи метод фіксованого вікна. Алгоритм описав Marcos De Prado у своїй книжці [12], код якого наведено в додатку Б.

Далі необхідно тільки описати Autoregressive (AR) та Moving Average (MA) моделі, формульне представлення яких подане в пункті 3.3, а код описуючий їхню логіку наведений в додатку В. Для реалізації симуляції фрактального процесу ARFIMA початковий часовий ряд генерується випадковим чином з використанням нормального або стійкого розподілу. В подальшому включаються впливи MA коефіцієнтів, виконується фрактальне диференціювання і в підсумку враховуються AR коефіцієнти. Послідовність виконання цієї логіки, яка викликає методи з додатків Б та В, наведена на рисунку 4.1.

```
def arfima_sim(
    ar_params,
    d,
    ma_params,
    n_points,
    noise_std = 1,
    noise_alpha = 2,
    warmup = 0):
    """Generate discrete series using ARFIMA process"""
    ma_list = ma_model(
        ma_params, n_points + warmup, noise_std=noise_std, noise_alpha=noise_alpha
    )
    ma_df = pd.DataFrame(ma_list, columns=["MA Series"])
    frac_ma = fracdiff_FFD(ma_df, d)
    arfima_series = arma_model(ar_params, frac_ma)
    return arfima_series[-n_points:]
```

Рисунок 4.1. Функція симуляції процесу ARFIMA для якої необхідно вказати значення коефіцієнтів AR(p), d, MA(q) та кількість необхідних для генерування елементів часового ряду

Приклад виконання генерування часового ряду з симуляцією процесу ARFIMA поданий на рисунку 4.2.

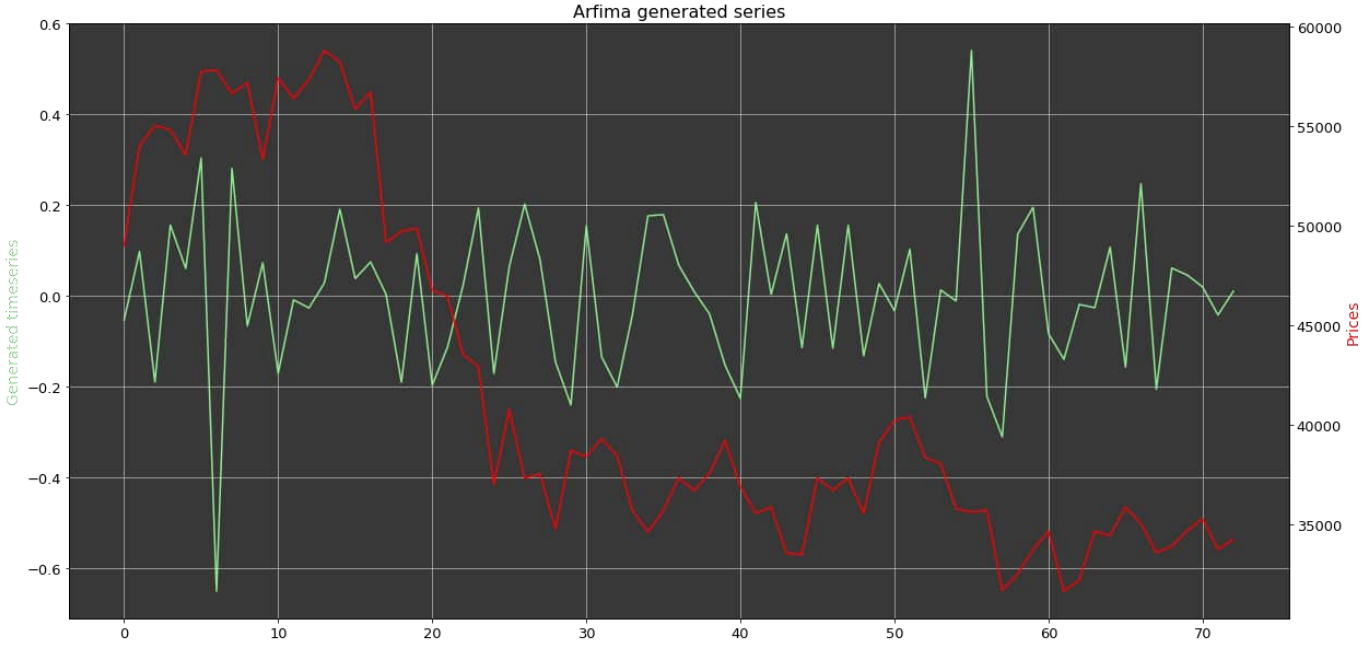


Рисунок 4.2. Симульований часовий ряд процесом ARFIMA (салатовий колір) та порівняння з вихідним часовим рядом (червоний колір)

4.2 Особливості реалізації задачі прогнозування для моделі ARFIMA

Для визначення параметрів AR(p) та MA(q) моделі ARFIMA нам потрібно отримати стаціонарний ряд, оскільки як було доведено в розділі 3 наш часовий ряд ціни Bitcoin є нестаціонарний і це було підтверджено двома статистичними тестами ADF та KPSS.

Завдяки вже згаданому в попередньому пункті фрактальному диференціюванні було підібрано мінімальний параметр інтегрованості ряду d таким чином, щоб відхилялася нульова гіпотеза тестом ADF, тобто щоб отриманий ряд був стаціонарний. В результаті можна отримати наступний графік на рисунку 4.3.

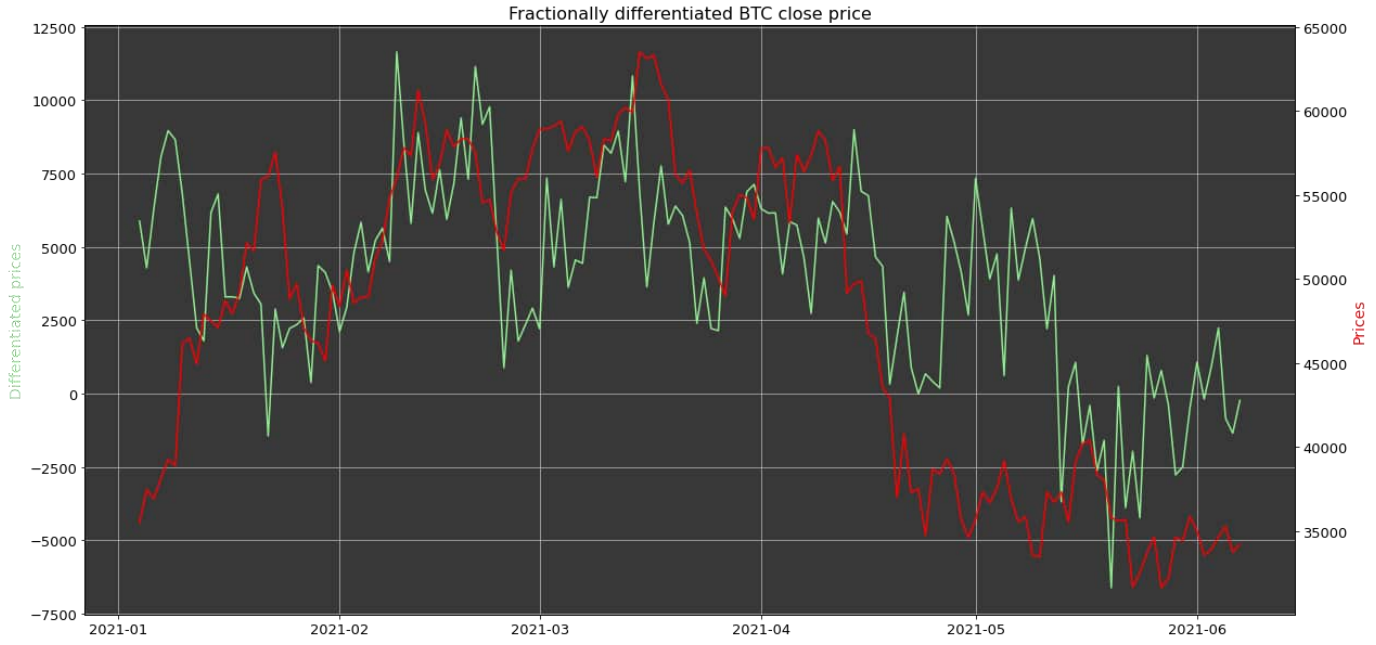


Рисунок 4.3. Диференційований часовий ряд ціни Bitcoin (салатовий колір) та порівняння з вихідним часовим рядом (червоний колір)

З рисунку 4.4 можна побачити значення функцій ACF та PACF для нашого диференційованого ряду і кількість тих значущих затримок, тобто тих, які більші за діапазон інтервалу довіри (голуба область на рисунку 4.4), буде визначати максимальне число для наших параметрів AR(p) з графіку ACF та MA(q) з графіку PACF відповідно.

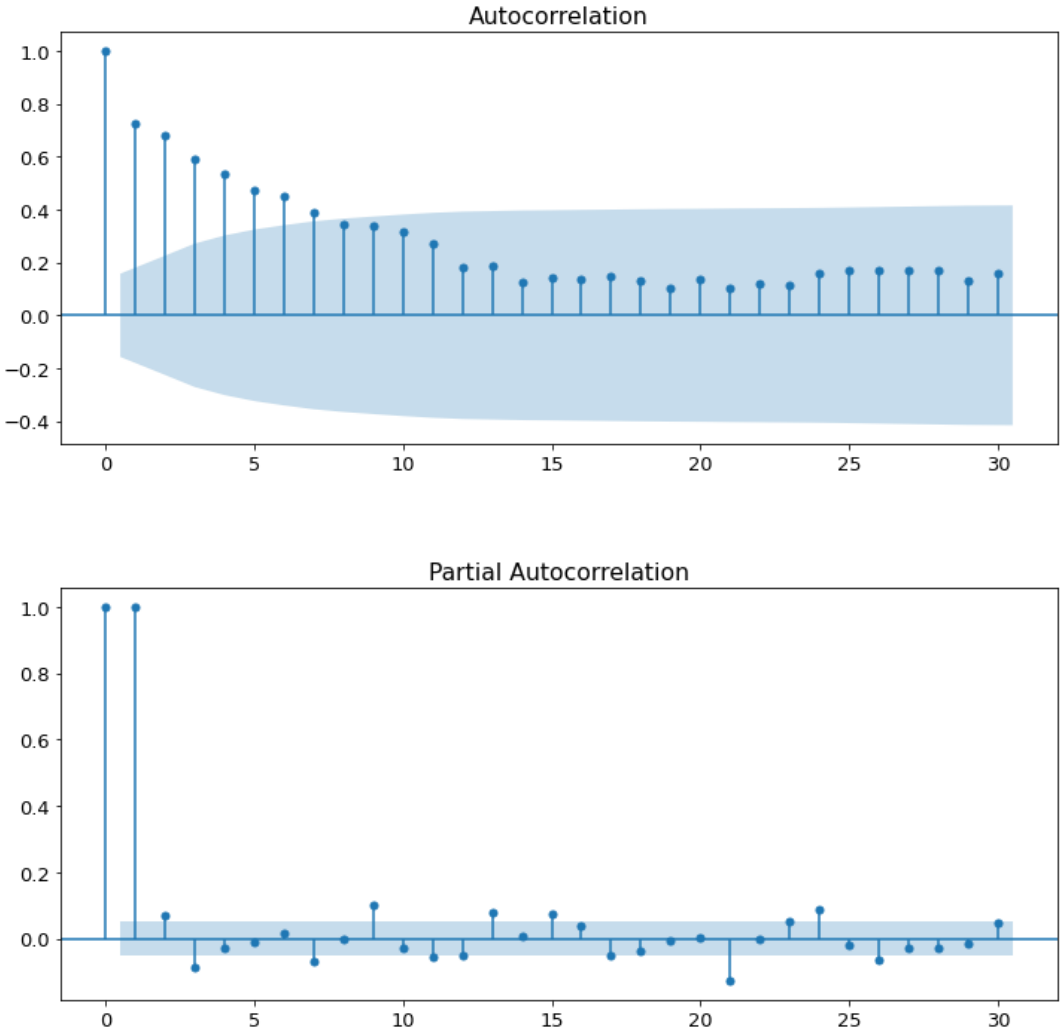


Рисунок 4.4. Функції ACF та PACF для диференційованого часового ряду ціни Bitcoin

А після визначення можливих значень порядку AR(p) та MA(q) визначався показник d інтегрованості ряду за допомогою функції fracdiff з бібліотеки fracdiff мови програмування R. Вибір оптимальної моделі визначався по оцінці RMSE з огляду на різницю між тестовими даними. Також для порівняння я виконав моделювання за допомогою моделей ARFIMA та ARIMA з автоматичним підбором параметрів. Були також проведені дослідження із зміною розмірів тестової і тренувальної вибірки для кожної з вищезгаданих моделей.

Спрогнозовані дані можна візуально представити, так, наприклад, як на рисунку 4.5 і для досить невеликої тестової вибірки результат такої візуалізації буде в достатній мірі демонстративним. Варто наголосити, що крім представлення

прогнозованих даних ще показується 95% інтервал довіри, який на рисунку 4.4 позначений голубою областю і означає інтервал, у межах якого з заданою довірчою імовірністю можна чекати значення оцінюваної випадкової величини.

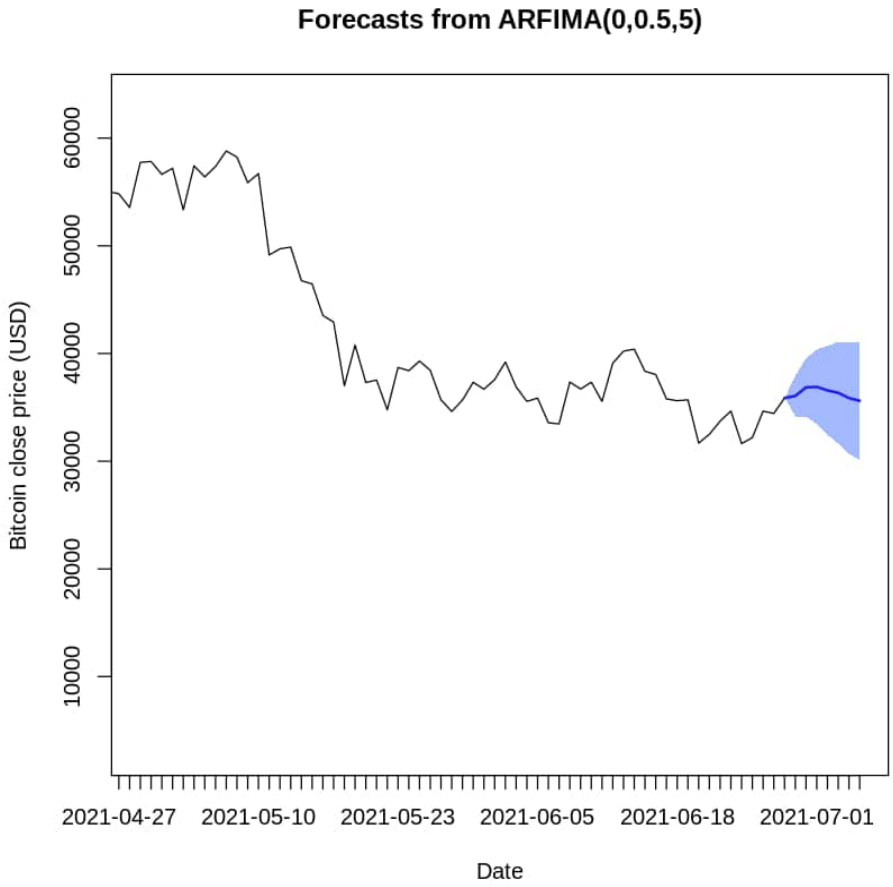


Рисунок 4.5. Тижневе прогнозування Bitcoin моделлю ARFIMA(0, 0.4994476, 5)

Загалом порівняння між вищезгаданими моделями я проводив з використанням двох метрик для тестової вибірки, а саме, RMSE та MAPE. RMSE представляє собою середнє квадратичне відхилення між прогнозованими та реальними значеннями, а MAPE же ж описує середнє абсолютне відсоткове відхилення. Результати подані на таблицях 4.1-4.3.

Модель	RMSE	MAPE (%)
(auto) ARFIMA	**2109.5**	4.81
(auto) ARIMA	2317.6	5.35
(manually fitted) ARFIMA	2140.78	**4.53**

Таблиця 4.1. Порівняння точності прогнозування тестових даних між моделями з 1344 тренувальних, 30 тестових даних

Модель	RMSE	MAPE (%)
(auto) ARFIMA	2136	5.76
(auto) ARIMA	1636.34	4.43
(manually fitted) ARFIMA	**654.14**	**1.71**

Таблиця 4.2. Порівняння точності прогнозування тестових даних між моделями з 1367 тренувальних, 7 тестових даних

Модель	RMSE	MAPE (%)
(auto) ARFIMA	2741.5	6.92
(auto) ARIMA	2308.63	5.3
(manually fitted) ARFIMA	**2205.05**	**4.67**

Таблиця 4.3. Порівняння точності прогнозування тестових даних між моделями з 500 тренувальних, 30 тестових даних

4.3 Реалізація програмного продукту у вигляді Telegram bot

Телеграм використовує власний протокол шифрування MTProto (він же Telegram API) - це API (Application Programming Interface), через який додаток Телеграм зв'язується з сервером. Telegram API повністю відкритий, тому будь-який

розробник може написати свій клієнт месенджера. Власне для написання ботів було створено Telegram Bot API, що виступає надбудовою над Telegram API. Використовуючи техніку HTTP Long Poll було реалізовано виклик до потрібного API. В результаті серверу не потрібно чекати, поки клієнт надішле запит. У Long poll сервер не закриває з'єднання після отримання запиту від клієнта.

Загальну архітектуру роботи застосунку показано на рисунку 4.6, однак в цьому підрозділі будемо розглядати роботу застосунку в межах локального сервера, а вже в наступному буде описано хмарну інтеграцію.

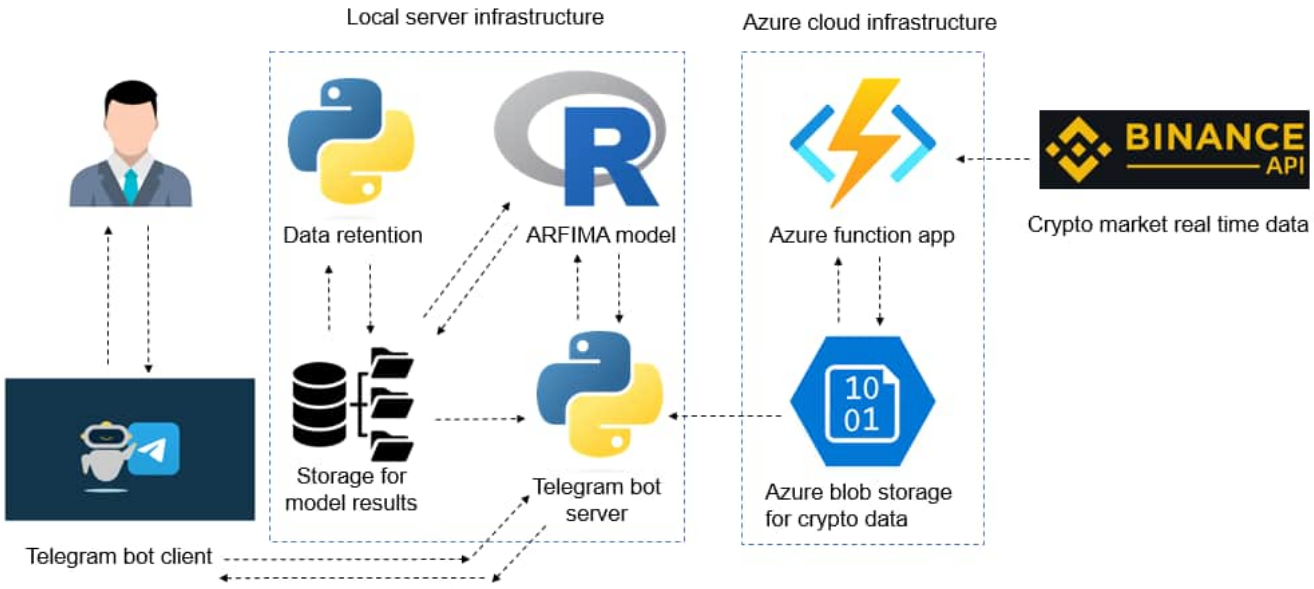


Рисунок 4.7. Базова архітектура роботи застосунку

Першим кроком роботи застосунку є обробка запиту від користувача, приклад яких можна побачити на рисунку 4.8. Ця обробка реалізована за допомогою бібліотеки для мови програмування Python pyTelegramBotAPI. Виклик методів обробників подій відбувається ітеративно і починається з декоратора бібліотеки, що дозволяє обернути базову функцію-обробник на необхідні команди для розширення її функціональності без безпосередньої зміни.

Основною логічною компонентою виступає виклик бекенд-застосунку на мові програмування R, що містить тренування моделі ARFIMA та запис результатів в сховище на локальному сервері. Виклик реалізований за допомогою Python

модуля subprocess, що дозволяє створювати нові процеси. При цьому він може підключатися до стандартних потоків введення/виводу/помилок та отримувати код повернення. За допомогою subprocess можна, наприклад, виконувати будь-які команди Linux зі скрипту. І залежно від ситуації отримувати результат чи лише перевіряти, що команда виконалася без помилок.

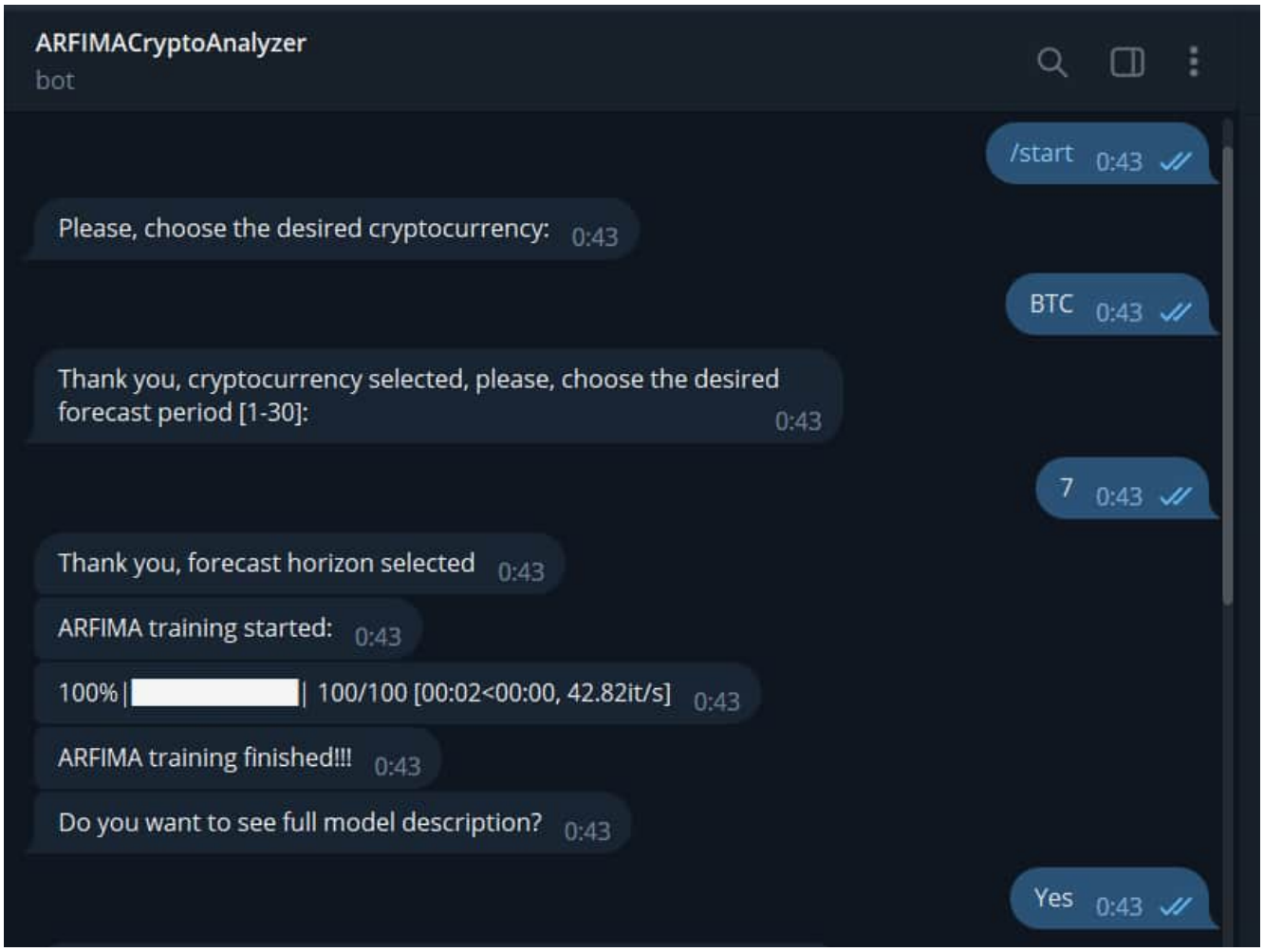


Рисунок 4.9. Приклад використання телеграм-боту

Далі відбувається процес обробки неструктурованих даних зі сховища, приклад яких можна побачити на рисунку 4.10. Оскільки працювати з телеграм ботом можуть велика кількість користувачів, то була необхідність унікальної ідентифікації файлів конкретної сесії, тому до кожного файлу дописується позначка часу, тобто послідовність символів, що показує, коли відбулася певна подія.

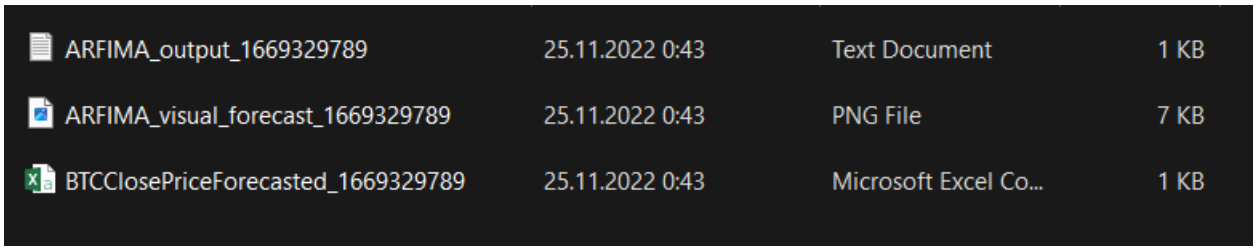


Рисунок 4.10. Приклад результуючих файлів після виконання моделі ARFIMA

Також був реалізований окремий функціонал, який забезпечує поняття “періоду зберігання” (retention period), що означає кількість часу, протягом якого зберігається інформація на сервері. Різні дані повинні мати різні терміни зберігання. Згідно з передовою практикою, дані слід зберігати лише до тих пір, поки вони корисні, що в даному випадку актуально тільки на один день, тому далі дані затираються щоб забезпечувати можливість обробки даних багатьох користувачів без необхідності вертикального масштабування серверу.

Щоб вищезгаданий функціонал відбувався постійно навіть після перезапуску сервера, який має операційну систему Windows, то необхідно було створити архітектуру з трьох файлів:

- Файл bat, який може запускатися (безпосередньо або через ярлик) автоматично системою Windows у папці запуску;
- скрипт PowerShell менеджера для виконання будь-яких кроків налаштування, таких як активація середовища, а потім запуск планувальника;
- скрипт Python планувальника, який запускає функцію.

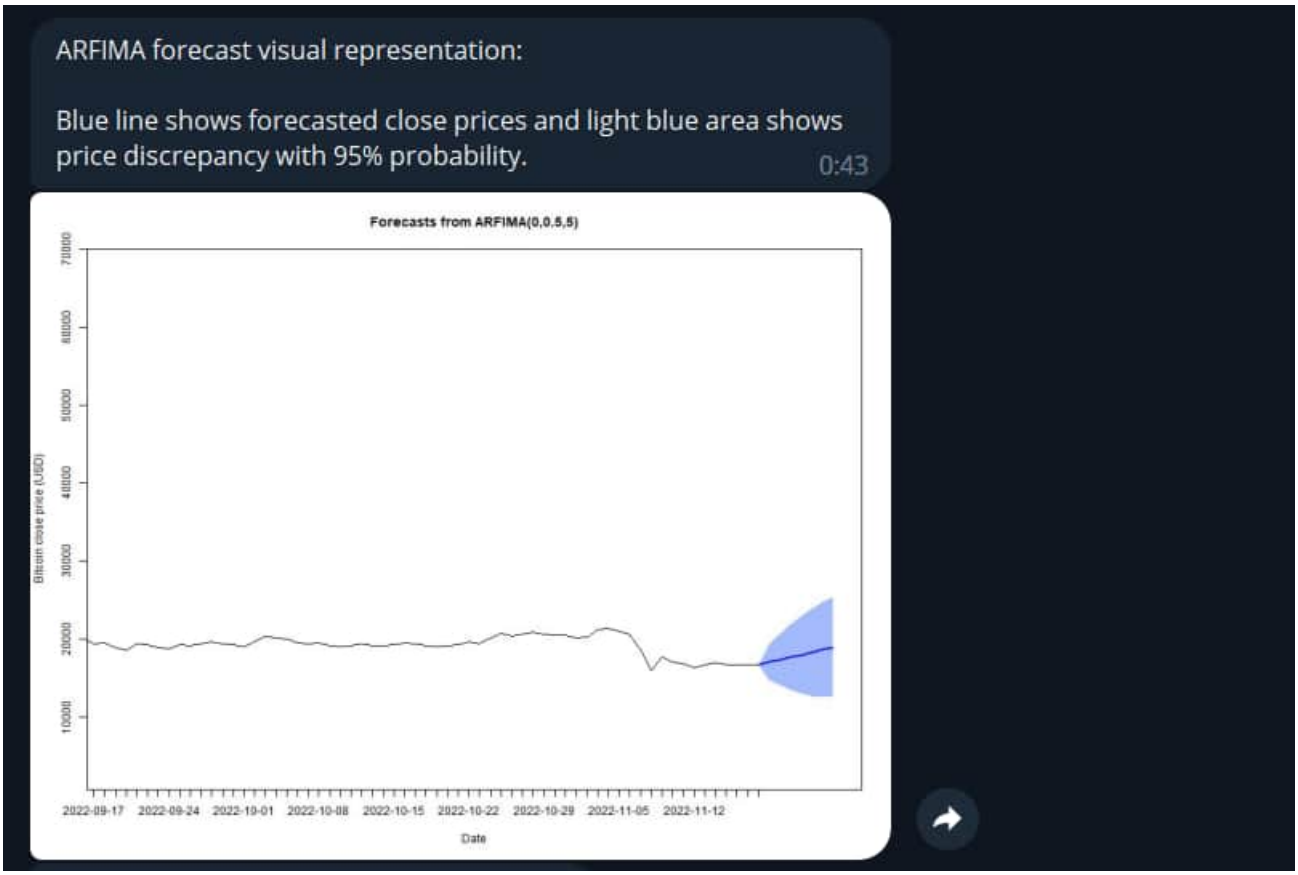
Сама ж функція виконується за допомогою Python модуля schedule, виклик якого можна побачити на рисунку 4.11. Це є свого роду простий для використанням API без будь-яких зовнішніх залежностей.

```
if __name__ == "__main__":
    schedule.every(1).days.do(job)

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

Рисунок 4.11. Виклик модуля schedule для періодичного виконання функції-очисника

Остаточні результати виконання програмного продукту можна подивитися на рисунку 4.12. Це є найбільш розгорнутий опис результатів, що включає в себе як графічне представлення, так і опис параметрів моделі, а також відповідно таблицю з прогнозованими ціновими значеннями. Однак користувач може вибрати скорочений варіант представлення результатів і отримати тільки числові прогнозовані значення ціни для необхідної йому криптовалюти.



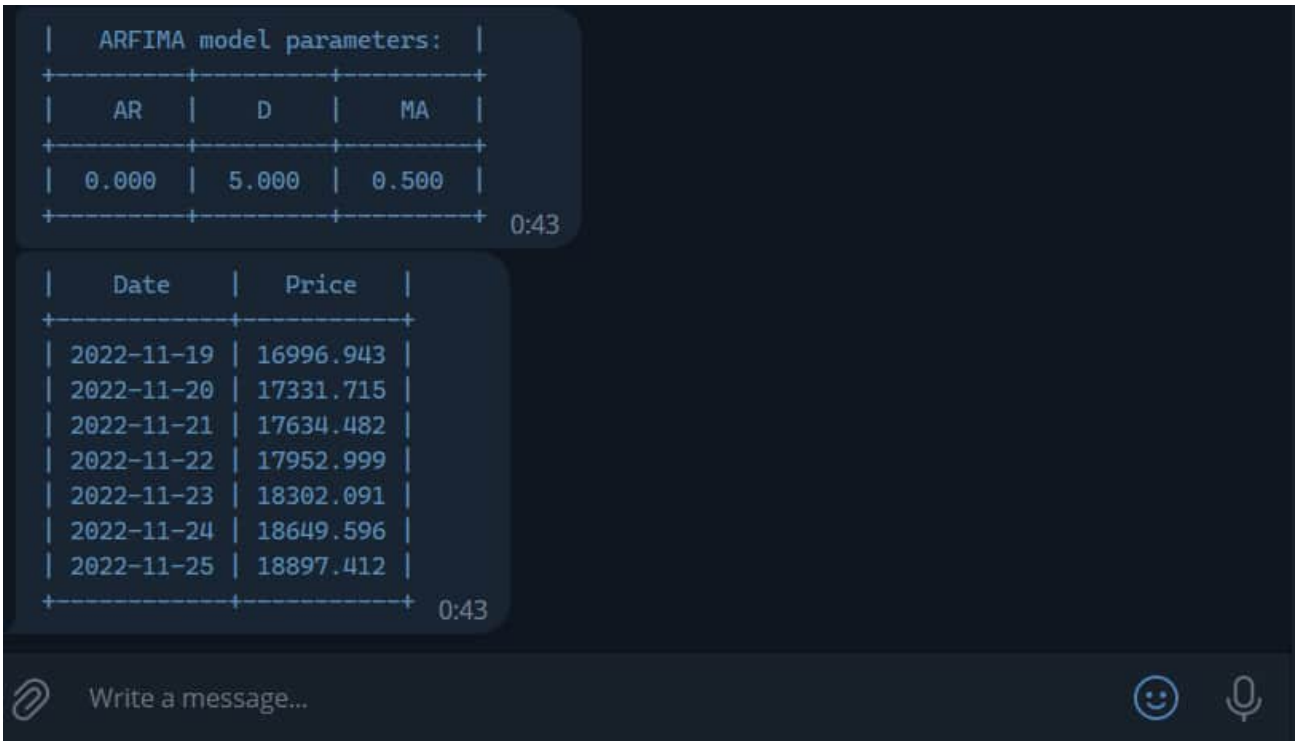


Рисунок 4.12. Приклад результатів виконання запиту від користувача

4.4 Застосування хмарних технологій

4.4.1 Google Colab

Загалом рішення щодо вибору хмарного сервісу для тренування фрактальної моделі було здійснене з огляду на ціну та простоту використання. Саме Google Colab зміг задовольнити відповідні критерії, оскільки він безкоштовний. Також він забезпечує необхідну модель надання хмарних сервісів PaaS. В цьому випадку хмарний провайдер надає доступ до всієї інфраструктури, включаючи засоби розробки і тестування. З точки зору користувача, необхідно тільки подбати про свій програмний продукт, який можна почати створювати в зручному інтерактивному режимі, як з використанням мови програмування Python, так й з R language. При цьому більшість популярних бібліотек вже буду встановлені на віртуальній машині, а більш вузькоспеціалізовані можна завантажити командою !pip install, в будь-якій вільній комірці для Python-коду та install.packages() для R-коду.

```
install.packages("googledrive")
install.packages("httpuv")
install.packages("forecast")
install.packages("tsbox")
install.packages("lubridate")
install.packages("knitr")
install.packages("pracma")
install.packages("arfima")
install.packages("Metrics")
install.packages("fracdiff")
```

Рисунок 4.13. Підключені бібліотеки для мови R для задачі моделювання та прогнозування

У випадку цієї роботи так потрібно було встановити наступні бібліотеки наведені на рисунку 4.13. Сховище Google Drive, яке забезпечує модель надання хмарних сервісів SaaS, можна використати для зберігання даних для тренування моделей, а також для записування результатів прогнозування. Щоб використати таку можливість в Colab, потрібно підключитись до свого сховища командами з рисунку 4.14. Після чого можна буде доступитись до даних диску за шляхом /~/Colab Notebooks/.

```
library("googledrive")

# authorize google drive
if (file.exists("/usr/local/lib/python3.7/dist-packages/google/colab/_ipython.py")) {
  install.packages("R.utils")
  library("R.utils")
  library("httr")
  my_check <- function() {return(TRUE)}
  reassignInPackage("is_interactive", pkgName = "httr", my_check)
  options(rlang_interactive=TRUE)
}

drive_auth(use_oob = TRUE, cache = FALSE)

drive_file = drive_get("~/Colab Notebooks/CryptoDatasets/BitcoinTransformedClose.csv")
drive_download(drive_file)
```

Рисунок 4.14. Підключення Google Drive в Colab (R language)

На жаль, не можна обрати який саме процесор буде наданий, оскільки його автоматично обирає хмарний провайдер. В даному випадку був використаний сервер з процесором Intel® Xeon® CPU @ 2.00GHz. Однак з таблиці 4.4 бачимо, що він показав трішки довший час виконання в порівнянні з власними потужностями ПК, на якому розміщений процесор Intel® Core™ i7-8550U CPU 2.00GHz.

	Google Colab	**On Premises**
Час виконання	0.4983599 с	0.454999 с

Таблиця 4.4. Порівняння часу виконання моделювання та прогнозування моделі ARFIMA (з автоматичним підбором параметрів) між хмарним сервісом та особистими потужностями ПК

4.4.2 Microsoft Azure

Також для реалізації архітектури процесу обробки даних для телеграм-боту мені потрібно повноцінна хмарна інфраструктура і одним з найбільш популярних рішень – це Microsoft Azure що є платформою хмарних обчислень та Інтернет-порталом, який дозволяє отримувати доступ до хмарних служб та ресурсів, що надаються корпорацією Microsoft. На відміну від Google Colab, Azure є платним, однак його було використано в межах підписки Azure for Students, що дозволило розгорнути необхідне хмарне середовище без використання власних коштів.

В даному випадку зручність хмарного середовища полягало в щоденному оновленні даних криптовалют з використанням API-сервісу біржі Binance. Дані зберігаються в Azure сервісі Blob storage, що використовується для зберігання величезних обсягів неструктурованих даних. Неструктуровані дані – це дані, які не відповідають певній моделі даних або визначенню, наприклад текстові або двійкові дані, в моєму випадку це csv-файли, які відіграють роль тренувальних даних для фрактальної моделі. Також варто наголосити, що сховище Azure пропонує різні рівні доступу до даних і в моєму випадку був необхідний рівень доступу hot–tier,

який оптимізований для зберігання даних, до яких часто звертаються або змінюються і відповідно має найвищі витрати на зберігання. З рисунку 4.15 можна побачити загальний вміст мого сховища, що містить поділ історичних цінових даних після 2017 року для потрібних валют.

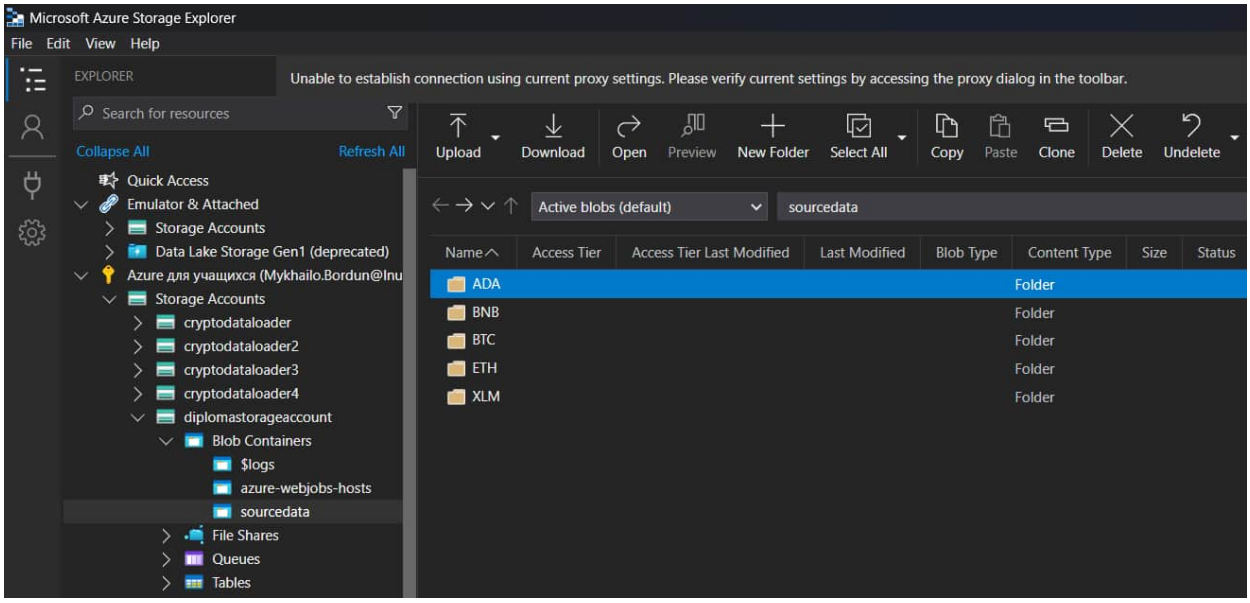


Рисунок 4.15.Вікно Azure Storage Explorer, що показує вміст Blog storage

Основна логіка функціоналу, архітектура якого наведена на рисунку 4.16, полягає в PaaS сервісі Azure Functions, що забезпечує безсерверні обчислювання, з можливістю запускати код, ініційований подіями, без необхідності явного надання або керування інфраструктурою. Серед переліку подій найбільш відповідним для реалізації технічного завдання виступає Timer Trigger, який можна виставити з потрібним графіком через Cron вираз, що задає періодичність виконання функції з точністю до секунд.

Цей вираз складається з шести полів:

{секунда} {хвилина} {година} {день} {місяць} {день тижня}

Кожне поле може мати одне з таких значень:

- специфічне значення – 0 0 0 * * * (щоденно на початку доби);

- всі значення (*) – 0 * 0 * * * (кожну хвилину, починаючи з початку доби);
- оператор проміжку (-) – 0-5 * * * * * (5 разів на хвилину, перші 5 секунд);
- множина значень (,) – 0,3,5 * * * * * (три рази на хвилину, на 0, 3 і 5 секунду);
- значення інтервалу (/) – 0 */5 * * * * (12 разів на годину –кожну п’яту хвилину).

Це дозволяє мені отримувати свіжі дані з криптобіржі на початку доби кожного дня. Оскільки кількість криптовалют може розширюватися, то рішення про перенесення оновлень даних для кожної з них в хмарне середовище допоможе знизити навантаження на локальний сервер, що покращує швидкодію програми.

Також серед значущих оптимізацій виконання оновлень даних є інкрементальне завантаження даних, а не повна, що в моєму випадку означає аналіз даних у вихідному файлі і виділення останнього часового значення серед даних для того щоб добавляти тільки дані з біржі для конкретного часового вікна. Код такого виду завантаження даних показаний в додатку Г. Однак це тестувалось лише на власному сервері, для Blob storage було реалізована схема повного завантаження, оскільки для модуля BlobClient з Python бібліотеки azure-storage-blob немає жодних рішень для оновлення поточних блоків файлу даних.

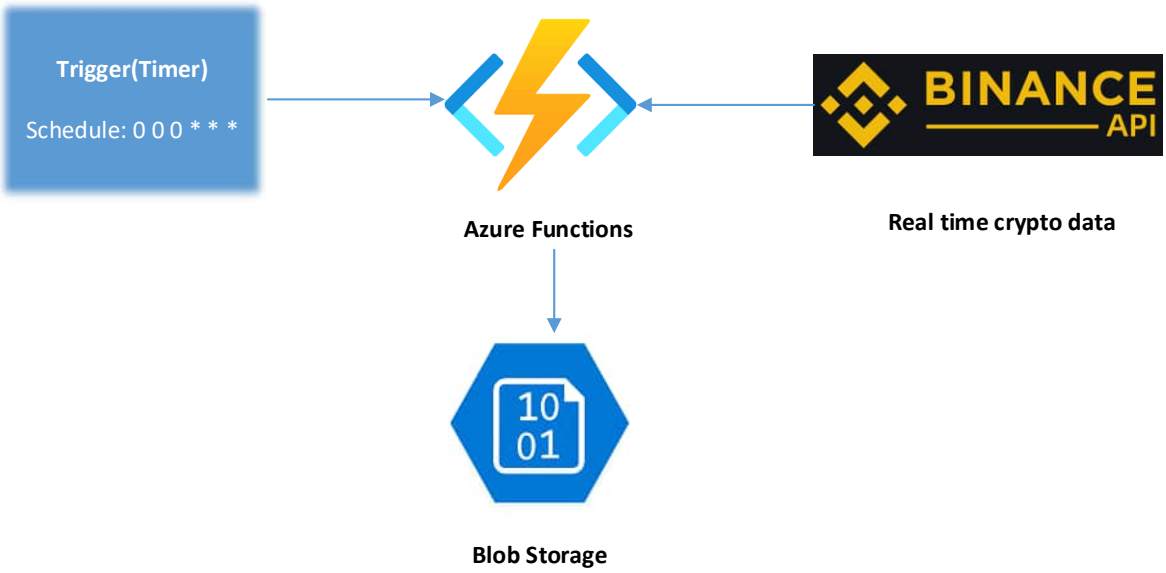


Рисунок 4.16. Архітектура хмарної обробки вхідних даних

4.5 Висновки

Для більшого розуміння фрактальної моделі ARFIMA було реалізовано симуляцію часового ряду, в основі якої лежить функція фрактального диференціювання. Цей процес і був також використаний для визначення параметрів AR(p) та MA(q) моделі ARFIMA при реалізації задачі прогнозування. Порівняно ефективність створеної фрактальної моделі з такою ж моделлю з автоматичним підбором параметрів, а також з найбільш відповідною авторегресійною моделлю на різних розмірах тренувальних та тестових даних. Виконано навчання фрактальної моделі з використанням хмарного сервісу Google Colab та порівняно час моделювання в порівнянні з локальною машиною.

Було реалізовано кінцевий програмний продукт у вигляді телеграм-боту, що включає достатньо складний процес обробки даних, оскільки вхідні дані для моделі беруться щоденно з використанням функціоналу, розміщеному на хмарній платформі Microsoft Azure. Також для результуючих даних на локальному сервері реалізовано процес автономної очистки, що затирає неактуальні дані в межах необхідного часового вікна.

ВИСНОВКИ

У даній роботі розглянуто інформаційну технологію прогнозування курсу криптовалют за наявності довготривалої пам’яті в нестаціонарних часових рядах з використанням історичних валютних даних.

Обґрунтовано вибір програмних засобів для реалізації алгоритмів та аналізу даних за допомогою мови програмування Python версії 3.6.5 з використанням бібліотек pandas версії 1.1.3 та numpy версії 1.19.2. Також з використанням цієї мови програмування та бібліотеки pyTelegramBotAPI версії 4.8.0 було реалізовано Телеграм бот. Для прогнозування часових рядів використана мова програмування R версії 4.1.3, разом з бібліотеками forecast версії 8.16 та arfima версії 1.8.0. Проведено дослідження показників Херста, стаціонарності часових рядів та довгої пам’яті.

На прикладі криптовалюти Bitcoin результати виконання прогнозування на повному наборі тренувальних даних також підтверджують доцільність використання моделі ARFIMA в порівнянні з моделлю ARIMA з огляду на різноманітні оцінки тестової похибки такі як RMSE, MAPE. Адаптовано алгоритм для вибору оптимального параметра d фрактального диференціювання моделі ARFIMA.

Загалом можна сказати, що більша кількість даних як тренувальних, так і тестових однозначно посилює перевагу фрактальних моделей, оскільки в даному випадку в нас наявний тривалий ефект шоку, тобто яскраво виражена довга пам’ять в розглянутому часовому ряді. Однак слід завжди перевіряти дані і очищувати аномальні відхилення, які спричиняють похибку в оцінці прогнозування.

Також були успішно проведені дослідження моделювання та прогнозування з використанням хмарного сервісу Google Colab, а також сховища даних Google Drive.

Було реалізоване програмне рішення для моєї інтелектуальної системи, що включає в себе взаємодію як локального сервера, так і хмарної інфраструктури Microsoft Azure з використанням обчислювального сервісу Azure Functions та сервісу зберігання даних Blob Storage, що в кінцевому результаті дає можливість користувачу з використанням телеграм-бота на Python отримувати результати прогнозування для даних з біржі в реальному часі з використанням функціоналу для тренування та виконання прогнозування моделлю ARFIMA, який вже був створений з використанням R language.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Бордун М. Інтелектуальна система прогнозування динаміки курсів криптовалют з використанням фрактального аналізу / М. Бордун // Міжнародна студентська наукова конференція з прикладної математики та комп'ютерних наук (МСНКПМКН-2022), 5-6 травня 2022 р. – Львів:2022. – С. 79- 83. Режим доступу: https://ami.lnu.edu.ua/wp-content/uploads/2022/05/ISSCAMCS-2022.pdf

[2] Bordun M. DEVELOPMENT OF SOFTWARE AND ALGORITHMIC SECURITY FOR FORECASTING THE CRYPTOCURRENCY COURSE USING FRACTAL ANALYSIS METHODS / M. Bordun, Y. Sokolovskyy, M. Levkovych // XXX International Ukrainian-Polish Conference CAD IN MACHINERY DESIGN IMPLEMENTATION AND EDUCATIONAL ISSUES, 1 - 2 December, 2022, Lviv, Ukraine. – P. 10.

[3] Ya. I. Sokolovskyy, M. I. Bordun, M. V. Levkovych. DEVELOPMENT OF SOFTWARE AND ALGORITHMIC SECURITY FOR FORECASTING THE CRYPTOCURRENCY COURSE USING FRACTAL ANALYSIS METHODS// Computer Design Systems. Theory and Practice , Vol. 4, No. 1, 2022. P.81-94.

[4] Shah V. Forecasting Market Prices using DL with Data Augmentation and Meta-learning: ARIMA still wins! / V. Shah, G. Shroff.– 2021.– Available from: https://arxiv.org/abs/2110.10233

[5] Liu K. An Evaluation of ARFIMA (Autoregressive Fractional Integral Moving Average) Programs / K. Liu, Y. Chen, X. Zhang // Axioms.– 2017.– Vol. 6.– P. 1-16.

[6] Перцовский О. Моделирование валютных рынков на основе процессов с длинной памятью: Препринт WP2/2004/03 — М.: ГУ ВШЭ.– 2003.– 52 с.

[7] Cryptocurrency Historical Prices [Electronic resource]: Kaggle, — 2021. — Available from: https://www.kaggle.com/datasets/sudalairajkumar/cryptocurrencypricehistory

[8] How to Detect Random Walk and White Noise in Time Series Forecasting [Electronic resource]: Medium, — 2021. — Available from: https://towardsdatascience.com/how-to-detect-random-walk-and-white-noise-in-time-series-forecasting-bdb5bbd4ef81

[9] Safitri D. Gold price modeling in Indonesia using ARFIMA method / D. Safitri, Mustafid, D. Ispriyanti, Sugito // IOP Conf. Series: Journal of Physics: Conf. Series 1217.– 2019.– P. 7-9.

[10] Mohamed B. Estimation of the long memory parameter in non stationary models: A Simulation Study / B. Mohamed, R. Khalfaoui.– 2011.

[11] Reisen V. Estimation of Parameters in ARFIMA Processes: A Simulation Study / V. Reisen, B. Abraham, S. Lopes.– 2006.

[12] De Prado M. L. Advances in Financial Machine Learning / M. L. de Prado // John Wiley & Sons, Inc., Hoboken, New Jersey.– 2018.– P. 75-91.

ДОДАТКИ

Додаток А. Реалізація R/S аналізу для знаходження показника Херста

```
def get_hurst_exponent_r_s(time_series):
    """Returns the Hurst Exponent of the time series using R/S analysis"""
    time_series = list(time_series)
    N = len(time_series)
    R_S_dict = []

    for k in range(10, int(np.floor(N/2))+1):
        R, S = 0, 0

        # split time_series into subsets
        subset_list = [time_series[i:i+k] for i in range(0, N, k)]

        if np.mod(N, k) > 0:
            subset_list.pop()

        # calc mean of every subset
        mean_list=[np.mean(x) for x in subset_list]

        for i in range(len(subset_list)):
            cumsum_list = pd.Series(subset_list[i] - mean_list[i]).cumsum()
            R += max(cumsum_list) - min(cumsum_list)
            S += np.std(subset_list[i])

        R_S_dict.append({"R":R/len(subset_list), "S":S/len(subset_list), "n":k})

    log_R_S, log_n = [], []

    for i in range(len(R_S_dict)):
        R_S = (R_S_dict[i]["R"]) / (R_S_dict[i]["S"])

        log_R_S.append(np.log(R_S))
        log_n.append(np.log(R_S_dict[i]["n"]))

    reg = np.polyfit(log_n, log_R_S, 1)

    return reg[0]

hurst_exp = get_hurst_exponent_r_s(close_price.values)
print(f"Hurst exponent using R/S analysis: {hurst_exp:.4f}")
```

Метод get_hurst_exponent_r_s, який приймає часовий ряд та повертає показник Херста за допомогою R/S аналізу

Додаток Б. Реалізація фрактального диференціювання

```
def get_weights(d, length, threshold=1e-5):
    """
    Computes the weights for the fractionally differentiated features up to a given threshold
    requirement for fixed-window fractional differencing.
    """
    w, k, w_curr = [1.], 1, 1

    while(k < length):
        w_curr = (-w[-1]/k*(d-k+1))

        if(abs(w_curr) <= threshold):
            break

        w.append(w_curr)
        k += 1

    #reshape from a single row to a single column so they can be applied to time-series values easier
    w = np.array(w[::-1]).reshape(-1,1)

    return w
```

Метод get_weights, який приймає показники параметру інтегрованості часового ряду, довжини часового ряду, а також порогового значення, та повертає ваги, необхідні для визначення фрактального диференціювання

```python
def fracdiff_FFD(series, d, threshold = 1e-5):
    """Computes fractionally differentiated series using fixed-width window"""
    weights = get_weights(d, len(series), threshold)
    width = len(weights) - 1
    df = {}

    for name in series.columns:
        #forward fill through unavailable prices and create a temporary series to hold values
        curr_series = series[[name]].fillna(method='ffill').dropna()
        df_temp = pd.Series(dtype='float64')

        #loop through all values that fall into range to be fractionally differentiated
        for iloc1 in range(width, curr_series.shape[0]):
            #set values for first and last time-series point to be used in current pass of fractional
                #difference

            loc0 = curr_series.index[iloc1-width]
            loc1 = curr_series.index[iloc1]

            #make sure current value is valid
            if not np.isfinite(curr_series.loc[loc1,name]):
                continue

            #dot product of weights with values from first and last indices
            df_temp.at[loc1] = np.dot(weights.T, curr_series.loc[loc0:loc1])[0,0]

        df[name] = df_temp.copy(deep=True)
    df = pd.concat(df, axis=1)

    return df
```

Метод fracdiff_FFD, який приймає часовий ряд, параметр інтегрованості часового ряду, а також порогове значення, та повертає фрактально диференційований ряд отриманий алгоритмом фіксованого вікна

Додаток В. Реалізація MA та AR моделей

```python
def ma_model(params, n_points, noise_std = 1, noise_alpha = 2):
    """
    Generate discrete series using MA process
    Coefficients used by the MA process:
        x[t] = params[1]*epsi[t-1] + params[2]*epsi[t-2] + ... + epsi[t]
    """
    ma_order = len(params)

    if noise_alpha == 2:
        noise = norm.rvs(scale=noise_std, size=(n_points + ma_order))
    else:
        noise = levy_stable.rvs(
            noise_alpha, 0, scale=noise_std, size=(n_points + ma_order)
        )

    if ma_order == 0:
        return noise

    ma_coeffs = np.append([1], params)
    ma_series = np.zeros(n_points)

    for idx in range(ma_order, n_points + ma_order):
        take_idx = np.arange(idx, idx - ma_order - 1, -1).astype(int)
        ma_series[idx - ma_order] = np.dot(ma_coeffs, noise[take_idx])

    return ma_series[ma_order:]
```

Метод ma_model, який приймає показники MA-предикторів, кількість необхідних для генерування елементів часового ряду, середнє квадратичне відхилення та альфа значення при застосуванні стійкого розподілу, та повертає згенерований ряд за моделлю MA

```python
def arma_model(params, noise):
    """
    Generate discrete series using ARMA process
    Coefficients used by the AR process:
        x[t] = params[1]*x[t-1] + params[2]*x[t-2] + ... + epsi[t]
    """
    ar_order = len(params)

    if ar_order == 0:
        return noise

    n_points = len(noise)
    arma_series = np.zeros(n_points + ar_order)

    for idx in range(ar_order, len(arma_series)):
        take_idx = np.arange(idx - 1, idx - ar_order - 1, -1).astype(int)
        arma_series[idx] = np.dot(params, arma_series[take_idx]) + noise.iloc[idx - ar_order]

    return arma_series[ar_order:]
```

Метод arma_model, який приймає показники AR-предикторів та показники шуму, які визначаються фрактальним диференціюванням часового ряду за моделлю MA, та повертає згенерований ряд за моделлю ARMA

Додаток Г. Реалізація інкрементального завантаження криптовалютних даних

```
# global vars
url = 'https://api.binance.com/api/v3/klines'
interval = '1d'

# helper function to create folder
def create_folder(folder_name: str):
    if not os.path.exists(folder_name):
        os.makedirs(folder_name)

# function to get last executed data to capture only latter data
def get_last_executed_data(file_name: str):
    last_executed_data = ""
    if os.path.isfile(file_name):
        if os.stat(file_name).st_size > 0:
            with open(file_name, "r") as f:
                last_executed_data = f.readlines()[-1].split(",")[0]
    return last_executed_data


# function to implement incremental load for crypto data
def incremental_data_load(symbol_: str, last_exec_data: int):
    start = last_exec_data
    end = int(dt.datetime.today().timestamp() * 1000)

    diff = (dt.datetime.fromtimestamp(end / 1000) - dt.datetime.fromtimestamp(
        start / 1000)).days

    repeats = diff // 1000 + 1
    final_data = pd.DataFrame()

    for i in range(repeats):
        par = {'symbol': symbol_,
               'interval': interval,
               'startTime': start,
               'endTime': end,
               'limit': 1000}

        data = pd.DataFrame(json.loads(requests.get(url, params=par).text))
        data.columns = ['datetime', 'open',
                        'high', 'low',
                        'close', 'volume',
                        'close_time', 'qav',
                        'num_trades', 'taker_base_vol',
                        'taker_quote_vol', 'ignore']

        data = data[['datetime', 'close']]
        data = data.astype(float)
        final_data = pd.concat([final_data, data])

        start = int(final_data.iloc[-1]['datetime'] + 24*3600)
        end = int(dt.datetime.today().timestamp() * 1000)

    final_data.index = [dt.datetime.fromtimestamp(x / 1000.0).date()
                        for x in final_data.datetime]
    final_data = final_data['close']
    final_data = final_data.astype(float)
    return final_data
```

```
# main function to combine all the logic
def update_crypto_data(sign: str):
    database = os.path.abspath("historical_crypto_data")
    create_folder(database)

    crypto = os.path.join(database, sign)
    create_folder(crypto)

    historical_data = os.path.join(crypto, "ClosePrice.csv")

    symbol = f'{sign}USDT'
    last_executed_data = get_last_executed_data(historical_data)
    if last_executed_data != "":
        last_executed_data = [int(i) for i in last_executed_data.split("-")]
    else:
        last_executed_data = [2017, 10, 2]

    final_data = incremental_data_load(symbol,
                                       int(dt.datetime(last_executed_data[0],
                                                       last_executed_data[1],
                                                       last_executed_data[2]+1)
                                           .timestamp() * 1000))
    final_data.index.name = 'Date'
    final_data = final_data.rename('Close')
    final_data.to_csv(historical_data, mode='a+', header=not os.path.exists(
        historical_data), sep=',', encoding='utf-8')
```

Метод update_crypto_data, який приймає символ вказаної криптовалюти та викликає методи отримання останньої дати у вихідному часовому ряді і виклику оновлення даних для потрібного часового вікна