МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра прикладної математики та моделювання складних систем

Допущено до захисту

Завідувач кафедри ПМ та МСС

_____ Коплик І.В.

«__» _____ 20__ р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»

спеціальність 113 «Прикладна математика»

освітньо-професійна програма «Прикладна математика»

тема роботи **«Моделювання впливу соціальних мереж на динаміку ринку криптовалют»**

Виконавець

студентка факультету ЕлІТ

Кальченко М.В. _____

Науковий керівник

кандидат економічних наук

Маринич Т.О._____

Суми – 2021

РЕФЕРАТ

Назва документа: кваліфікаційна робота бакалавра на тему «Моделювання впливу соціальних мереж на динаміку ринку криптовалют».

Ключові слова: МОДЕЛЮВАННЯ, РИНОК КРИПТОВАЛЮТ, ВАЛЮТНИЙ КУРС, ПРОГНОЗУВАННЯ, АНАЛІЗ ДАНИХ.

Метою роботи є розробка алгоритму прогнозування курсу криптовалют та дослідження впливу дописів у соціальних мережах на їх динаміку.

Предметом даного дослідження є пошук оптимальних високоточних моделей для прогнозування часових рядів курсу криптовалют з урахуванням впливу дописів соціальних мереж.

Об’єктом даного дослідження є щоденні дані біржових курсів криптовалюти Bitcoin та дописи у соціальній мережі Twitter за період з 1 жовтня 2019 року до 17 квітня 2021 року.

Аналітичний огляд містить відомості про механізми роботи ринку криптовалют, опис класичних моделей прогнозування часових рядів і моделей машинного навчання. Основна частина складається зі збору початкових даних, їх дослідження і аналізу, побудови моделі для прогнозування курсу криптовалют. У кінці роботи описані отримані висновки та порівняні результати прогнозування.

У додатках містяться коди програм, які оброблювали, аналізували, моделювали і прогнозували дані.

Кількість сторінок: 62.

Кількість рисунків: 23.

Кількість таблиць: 2.

Кількість формул: 22.

Кількість використаних джерел: 35.

Кількість додатків: 6.

ЗМІСТ

ВСТУП

Світова економіка розвивається відповідно до вимог часу. Багато сфер нашого життя переходять у електронний формат. Так само і економічні процеси існують у онлайн-просторі.

З 2009 року починає свій розвиток перша мережа криптовалют Біткоїн –повністю децентралізована система електронної готівки, що не вимагає довіри третім сторонам. Система створена Сатоші Накамото.

Криптовалюти набувають все більшої популярності щодня. Наразі створено безліч різних електронних валют на основі Біткоїну. Вони відрізняються від фіатних систем грошового обігу більшою доступністю, анонімністю, високим рівнем захисту і низькими комісіями за транзакції. Криптовалюта вважається гарним інструментом для довгострокових інвестицій, спекуляцій на біржах та для обміну на товари або послуги.

Людям, які тільки починають цікавитись ринком криптовалют або тим, хто добре знайомий з цим напрямком важливо знати, від чого залежить електронна валюта, як вплинути на курс.

Криптовалюту можна назвати цінним ресурсом, тому закони запиту і пропозиції для неї такі самі, як, наприклад, для золота. Чим більше нових інвесторів мають бажання придбати певну криптовалюту, тим вища ціна. Також помічено, що на курс електронних грошей впливають політичні події. Зменшення довіри населення до фіатних грошей за рахунок певних політичних подій підштовхує до більшої зацікавленості криптовалютами. ЗМІ мають незаперечний вплив на усі сфери життя людей. Це стосується і ринку криптовалют. Чим більше висвітлюється ситуація на ринку, розповідається про електронні гроші і систему їх функціонування, тим більший інтерес населення до них, що може залучити в галузь нових людей. Суспільство виражає своє ставлення до криптовалюти у соціальних мережах, що теж має свій вплив на курс. Відомі випадки, коли популярні користувачі

соціальних мереж публікували дописи, пов’язані з їх ставленням до криптовалют і на ринку електронних грошей відразу відбувалися зміни курсу. Але вплив соціальних мереж та їх відомих користувачів на криптовалютний ринок наразі недостатньо досліджений, а саме зв’язок між постами популярних крипто-експертів та лідерів думок, опублікованих у соціальних мережах, та зміною курсу окремих криптовалют.

Метою роботи є розробка алгоритму прогнозування курсу криптовалют та дослідження впливу дописів у соціальних мережах на їх динаміку.

Предметом даного дослідження є пошук оптимальних високоточних моделей для прогнозування часових рядів курсу криптовалют з урахуванням впливу дописів соціальних мереж.

Об’єктом даного дослідження є щоденні дані біржових курсів криптовалюти Bitcoin та дописи у соціальній мережі Twitter за період з 1 жовтня 2019 року до 17 квітня 2021 року.

Емпіричними даними є дані з жовтня 2019 року по квітень 2021 року, які відображають курс найпопулярніших криптовалют згідно міжнародного сервісу для відслідковування стану ринку криптовалют coinmarketcap.com. У якості чинників, які впливають на зміни курсу криптовалют, використані дописи користувачів соціальної мережі Twitter, які були отримані в результаті запиту на права користуванням API Twitter.

Програмна реалізація здійснена у інтегрованому середовищі Python Notebook та використане програмне забезпечення EViews.

Відповідно до мети даної роботи, було досліджено такі задачі:

1. Аналіз доменної галузі (принципи функціонування ринку криптовалют; фактори впливу на курс; огляд відомих досліджень).
2. Пошук джерел для отримання початкових даних дослідження.

3. Огляд теоретичних відомостей щодо аналізу часових рядів, методів моделювання і прогнозування.

4. Підготовка даних для дослідження.

5. Аналіз даних, моделювання і прогнозування.

6. Опис результатів дослідження, формулювання висновків.

1. ОГЛЯД ОБЛАСТІ ДОСЛІДЖЕННЯ

1.1 Механізми роботи ринку криптовалют

Принципи роботи ринку криптовалют мають певні відмінні риси від фіатного ринку тобто ринку класичних валют, які законодавчо прийняті урядом. Криптовалюти це різновид альтернативних валют, які є децентралізованими, тобто незалежними від будь-якої держави чи організації. Учасники ринку можуть придбати або продати свої накопичення напряму, без участі сторонніх організацій [1]. Такі транзакції називаються блокчейном – люнцюжок блоків транзакцій, де кожен блок ланцюга є захищеним. Електронні криптовалютні гаманці прив’язані до блокчейну, а тому теж є захищеними. Такий зв'язок може надавати гарантію про те, чи дійсно відбулася певна транзакція.

Рисунок 1.1 – Принцип роботи блокчейну

Отримати криптовалюту можна не лише за рахунок обміну на фіатні гроші, а і здобути їх. Процес здобуття криптовалюти за рахунок обчислювальних потужностей свого обладнання називається майнінгом. Обладнання використовується для складних математичних обчислень, що у свою чергу використовуються для підтримки роботи мережі криптовалюти. Зазвичай для одного обчислення об'єднується декілька користувачів. Після успішного виконання завдання нагорода розподіляється між усіма учасниками, в залежності від наданої потужності. Усі свої накопичення користувачі зберігають у гаманцях, які знаходяться в рамках онлайн-сервісів або на власних носіях, фактично у вигляді файлу. Кожен гаманець захищений власним криптографічним ключем.

Криптовалюти активно використовуються в якості альтернативного методу оплати багатьма організаціями і ресурсами. Наприклад, Microsoft, WordPress, Reddit, Subway, Namecheap, Expedia, Newegg, Steam, Wikipedia, Zynga, Whole Foods, Bloomberg, Suntimes, Shopify. Криптовалюти можна обмінювати між собою в залежності від курсу на ринку або на фіатні гроші.

Зацікавленість криптовалютами можна відслідкувати за допомогою Google Trendsна рис. 1.2 [2]. Google Trends збирає статистику запитів користувачів у Google. Запити сортуються за темами і кожен користувач може переглянути статистику запитів за певний період у будь-якій доступній країні або по всьому світу. Даний графік сформовано за рахунок порівнянь запитів користувачів в усьому світі з 2016 до 2021 року.

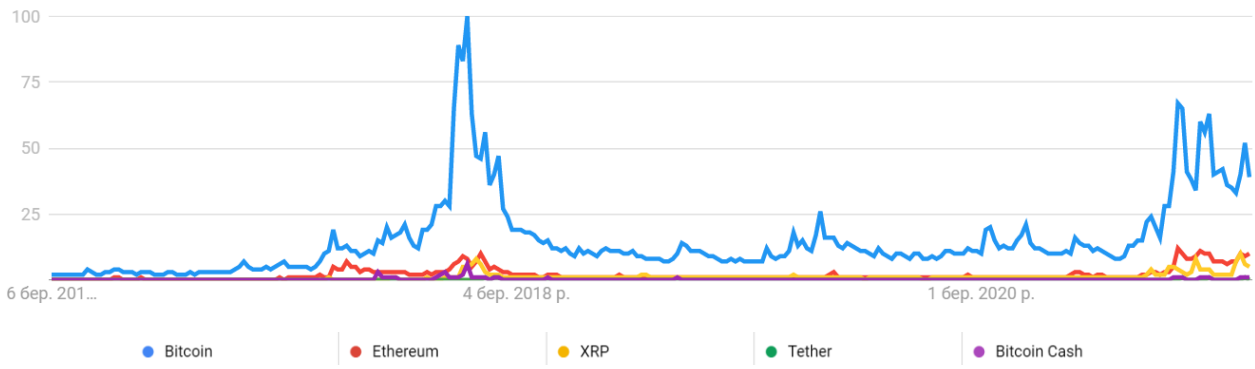


Рисунок 1.2 – Частота запитів у Google з 2016 до 2021 року

Цифри показують популярність пошукового терміна відносно найвищої точки на графіку для певного регіону та періоду часу. 100 – це пік популярності терміна. 50 означає, що популярність терміна вдвічі менша. 0 означає, що було замало даних про цей термін.

Наразі відомо більш ніж 1000 різних криптовалютних токенів (валют), які базуються на технології блокчейну, але найпопулярнішим залишається біткоїн [3].

На рис. 1.3 представлений графік капіталізації найпопулярніших токенів у період 2013-2021 років [4]. Графік ілюструє відсоток сумарної кількості кожної окремої криптовалюти на ринку від загальної кількості усіх фіатних грошей у світі.

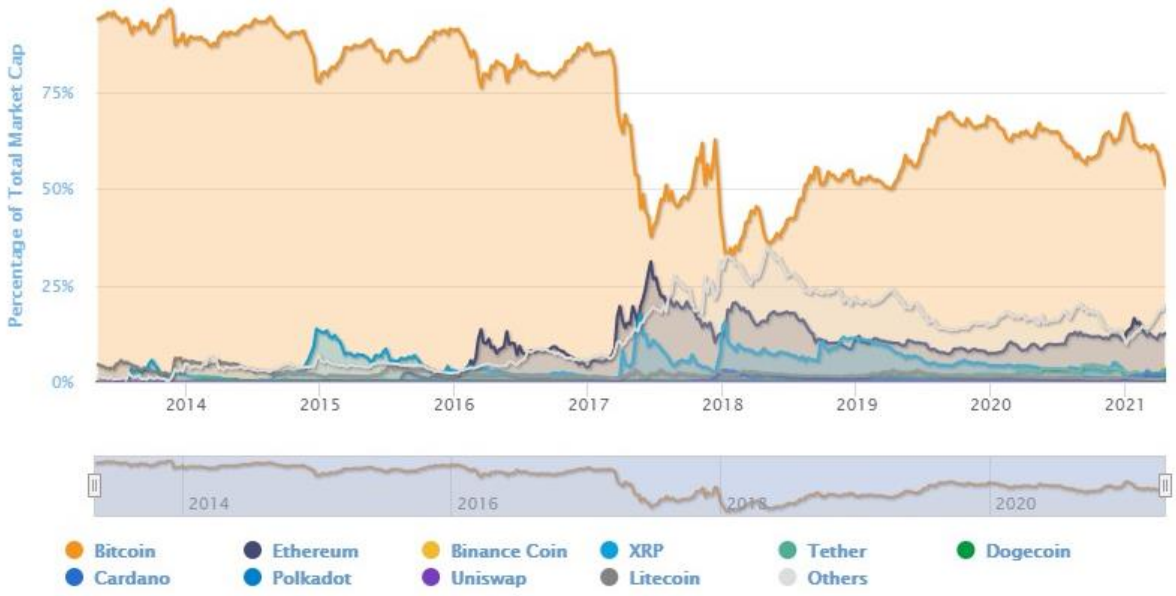


Рисунок 1.3 – Капіталізація найпопулярніших токенів з 2013 до 2021 року

1.2 Відомі дослідження ринку

На ринок криптовалют сильний вплив має суспільна думка і висловлювання популярних осіб в інтернеті. Ідея дослідити вплив соціальних

мереж на економіку, а саме електронний ринок валют зацікавила вчених Стенфордського університету. Стюарт Коліанні, Стефані Розалес і Майкл Сігнаротті вирішили протестувати гіпотезу, згідно з якою твіти можуть корелювати не тільки зі зміною індексів фондового ринку, але і з падінням або зниженням ціни біткоїна [5]. З'ясувалося, що точність передбачення динаміки ціни по годинах у такому випадку становить 59%.

Шведськими вченими Евітою Стенквіст і Джейкобом Ленне було визначено, що ключовим фактором аналізу є релевантність дописів. Їх тестова модель показала високу точність в передбаченні поведінки ціни біткоіна в залежності від настроїв повідомлень в Twitter – 83% прогнозів виявилися вірними [6].

Дослідження були проведені і вченими з Єльського університету, де на основі 2.27 мільйона твітів, в яких згадувався біткоін, з 79% ймовірністю вдалося точно передбачати поведінку ринку криптовалют. У Національному бюро економічних досліджень США стверджують, що за зростанням кількості повідомлень в тематиці біткоїни в Twitter слідує підвищення курсу в середньому на 2,5% [7].

Підвищення ціни токена відповідає закону попиту і пропозиції. Закон попиту: між величиною попиту і ціною існує обернено пропорційна залежність, тобто підвищення ціни знижує величину попиту, а зниження ціни підвищує величину попиту (рис. 1.4). Пропозиція характеризує можливість і бажання продавця пропонувати свої товари для реалізації на ринку за певними цінами.

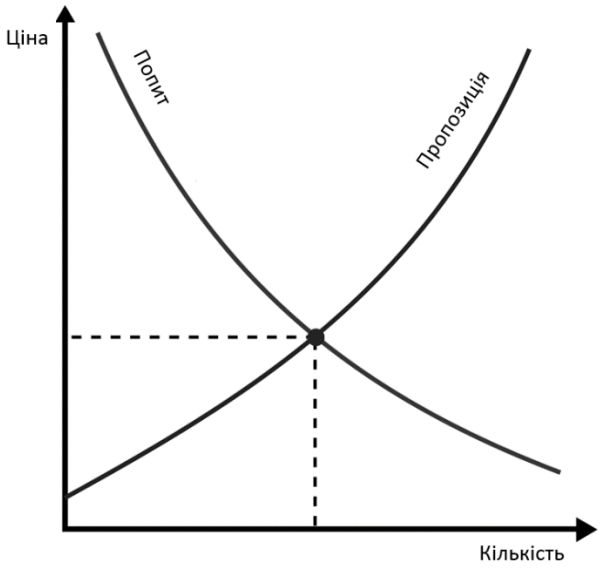


Рисунок 1.4 – Графічне зображення рівноваги за законом попиту і пропозиції

Рівноважна ціна можлива, коли максимальна ціна, за якої можуть собі дозволити придбати його покупці, збігається з ціною, мінімально прийнятною для продавців, що означатиме встановлення на ринку єдиної стійкої рівноважної ціни [8]. Але така ситуація є ідеальною і не відбувається у реальних ринкових відносинах, тому що існує вплив багатьох факторів.

Існує думка, що після підвищення ціни на біткоїн зменшується ціна на альткоїни (інші криптовалюти). Це відбувається через підвищення зацікавленості біткоїном і бажання отримати прибуток, поки ціна не зросла занадто сильно, при все ще стабільному курсі альткоїнів [9].

Загалом дослідження вчених стверджують, що дописи у соціальних мережах мають вплив на зміну курсу криптовалют. Чим більше релевантних дописів, тим більший відсоток правдивості прогнозу. Окрім того, варто звертати увагу і на те, як змінюється курс у порівнянні з іншими криптовалютами.

2. ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО МОДЕЛЮВАННЯ ЧАСОВИХ РЯДІВ

2.1 Визначення і аналіз часових рядів

Часовим рядом називають послідовність значень статистичного показника (ознаки), впорядковану у хронологічному порядку. Також використовуються такі терміни як "ряд динаміки", "динамічний ряд". Окремі спостереження часового ряду називають рівнями або елементами. Кожний рівень ряду відповідає певному моменту часу. Рівні ряду можуть набувати як детермінованих, так і випадкових значень [10].

Особливістю прогнозування часових рядів є те, що аналізуються лише дані спостережень без додаткової інформації, без аналізу впливу зовнішніх сил. Метою статистичного аналізу часових рядів є побудова математичної моделі ряду, за допомогою якої можна пояснити поведінку ряду і здійснити прогноз на майбутні періоди [11].

Існують класи часових рядів, розподілені за різними факторами. На рис. 2.1 представлена класифікація часових рядів за цими факторами.

Рисунок 2.1 – Класифікація часових рядів

При дослідженні часового ряду виділяють декілька складових:

- T – тренд, плавно змінювана компонента, що описує чистий вплив довготривалих факторів, тобто тривалу тенденцію зміни ознаки

(наприклад, зростання населення, економічний розвиток, зміна структури споживання тощо);

- S – сезонність, що відображує повторюваність економічних процесів протягом не дуже тривалого періоду (року, іноді місяця, тижня і т. д., наприклад, обсяг продажів товарів або перевезень пасажирів в різні пори року);
- C – циклічність, що відображає повторюваність економічних процесів протягом тривалих періодів. Включає підйоми і спади, що проходять 4 фази: пік, рецесія, депресія та підйом.
- E – випадкова компонента, що відображає вплив випадкових факторів, які не піддаються обліку та реєстрації.

Розкладення або декомпозиція часового ряду відбувається на основі тренду, сезонності, циклічності і випадковості. Існують такі варіанти моделей часових рядів [12]:

- Адитивна:

$$x_t = T + S + C + E, \qquad \text{де } t = 1, 2, \dots n \tag{2.1}$$

- Мультиплікативна:

$$x_t = T \times S \times C \times E, \qquad \text{де } t = 1, 2, \dots n \tag{2.2}$$

- Змішана:

$$x_t = T \times S + E, \qquad \text{де } t = 1, 2, \dots n \tag{2.3}$$

Часові ряди поділяються на стаціонарні і нестаціонарні. Стаціонарний часовий ряд відрізняється тим, що не змінює свої ймовірнісні властивості з часом. Визначити тип ряду можна кількома способами:

1. Побудова графіку. На графіку можна прослідкувати тренд чи сезонність. Розмах значень з часом може або наростати, або спадати, що вказує на мінливість середнього значення і дисперсії.

2. Побудова корелограми (графік, на якому відображені значення автокореляції з послідовними лагами). При побудові значення ACF (вибіркова функція автокореляції), значення мають тенденцію швидко зменшуватися до нуля для стаціонарних часових рядів, тоді як для нестаціонарних даних зменшення відбуватиметься повільніше. Графік PACF (часткова функція автокореляції) зменшується після перших значень для стаціонарних часових рядів, а для нестаціонарних рядів значення не виходять за межі довірчого інтервалу [13].

   Коефіцієнт автокореляції r_k визначається за такою формулою:

$$r_k = \frac{\sum_{t=k+1}^{T}(y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T}(y_t - \bar{y})^2}, \quad (2.4)$$

   де T – довжина часового ряду; k = 1, 2, … n.

   Якщо максимальним виявився коефіцієнт автокореляції першого порядку, то часовий ряд включає лише тренд. Якщо максимальним виявився коефіцієнт автокореляції порядку k, то ряд включає циклічні коливання з періодичністю k моментів часу. Якщо жоден із коефіцієнтів не є значимим (близькі до 0), можна зробити висновок, що ряд не має тенденцію і циклічні коливання або ряд має нелінійну тенденцію, для виявлення якої проводять додатковий аналіз [14].

3. Статистичне тестування. Це більш точний підхід до визначення типу стаціонарності. Часовий ряд вважається нестаціонарним (інтегрованим), якщо він містить одиничні корені. Існують тести на наявність одиничних коренів, за допомогою яких можна визначити тип стаціонарності часового ряду. У процесі тестування на наявність одиничних коренів використовується нульова гіпотеза. Нульова гіпотеза є загальним припущенням при статистичному тестуванні про те, що не існує ніякого зв'язку між двома вимірюваними явищами, або

не існує ніякого зв'язку між групами. Нульова гіпотеза є бездоказовим припущенням, здійсненим до проведення дослідження. Поняття нульової гіпотези передбачає існування альтернативної гіпотези. Для перевірки нульової гіпотези використовують тест Філіпса-Перрона та тест Дікі-Фулера.

Загалом, підхід до тестування модульного кореня неявно передбачає, що тимчасові ряди, що підлягають тестуванню $[y_t]_{t=1}^{T}$, можна записати, як:

$$y_t = D_t + z_t + \varepsilon_t \qquad (2.5)$$

де D_t є детермінованою складовою (тенденція, сезонна складова тощо); z_t є стохастичною складовою; ε_t – це стаціонарний процес помилок.

Завдання тесту – визначити, чи містить стохастичний компонент одиничний корінь чи він нерухомий. До модульних кореневих тестів належать:

- тест Дікі-Фуллера (ADF-тест)
- тест Філіпса-Перрона
- ADF-GLS тест
- KPSS тест
- тест Зівота і Ендрюса

Також існують напівпараметричні одиничні кореневі тести (наприклад, тест відношення дисперсії) і непараметричні (наприклад, критерій знаків, критерій знакових рангів Уілкоксона, U-критерій Манна-Уїтні) [15].

Методи статистичного моделювання передбачають або вимагають, щоб часові ряди були стаціонарними для отримання більш точних результатів. Якщо часовий ряд не представляє стаціонарний процес другого порядку, його необхідно привести до стаціонарного процесу. Це робиться за допомогою таких дій як взяття кінцевих різниць, логарифмування, розрахунку темпів приросту тощо.

Важливими характеристиками стаціонарного ряду є дисперсія і математичне очікування. Сталі значення цих величин у кожній точці спостереження вказують на те, що ймовірнісні властивості ряду не змінюються з часом.

(дисперсія – квадрат середньоквадратичного відхилення)

$$\sigma^2 = \frac{\sum(x - \bar{x})^2}{n}, \tag{2.6}$$

де $\bar{x}$ – середнє значення.

(мат.очікування)

$$M(X) = \sum_{k=1}^{n} p_k x_k, \tag{2.7}$$

де x_k – значення з вибірки; p_k – ймовірність отримання x_k.

Методи аналізу часових рядів, за допомогою яких можна пояснити поведінку ряду і здійснити прогноз на майбутні періоди [16]:

1. Побудова графіку і опис даних.
2. Кореляційний аналіз, що допомагає виявити залежності і лаги.
3. Спектральний аналіз дозволяє відслідкувати періодичність.
4. Фільтрація для виключення аномалій і сезонних коливань.
5. Моделювання обраним методом.
6. Прогнозування на основі обраної моделі.

2.2 Класичні моделі

Модель - це спрощене уявлення про реальний об'єкт, процес чи явище. Головним призначенням моделі є необхідність продемонструвати суттєві властивості об'єктів, процесів чи явищ з метою вивчення, аналізу, передбачення.

Процес моделювання включає такі етапи:

1. Визначення мети та задачі моделювання.
2. Опис змінних і структури дослідження.
3. Вивчення теоретичної бази, на основі якої відбуватиметься дослідження.
4. Опис процесу дослідження за допомогою формальної моделі (схеми, формули, теореми тощо).
5. Побудова моделі.
6. Перевірка працездатності моделі (зміни у вхідних даних).
7. Перевірка адекватності моделі.
8. Опис результатів моделювання і їх оцінка.
9. Формування висновків.

Важливим етапом є вибір правильної моделі для отримання реалістичних результатів. Краще обрати не одну, а декілька моделей і порівняти результати. До класичних моделей часових рядів належать

- Регресійні

Створення регресійної моделі являє собою ітераційний процес, спрямований на вираження змін і залежностей у системі випадкових величин. Кореляційне рівняння має такий вигляд:

$$y_t = f(x_t, b) + \varepsilon_t, \tag{2.8}$$

де b – параметри моделі; ε – випадкова помилка моделі

Найпростішою моделлю регресії є лінійна регресія. Рівняння для часових рядів має такий вигляд:

$$\Delta y_t = b_0 + \sum_{j=1}^{p} b_j \Delta x_{tj} + \sum_{i=1}^{k} c_i S_i + \varepsilon_t, \tag{2.9}$$

де b_j, c_i– оцінки параметрів регресії; Δx_t – стаціонарні фактори моделі (регресори); S – сезонність; $\Delta x_t = x_t - x_{t-1}$; $\Delta y_t = y_t - y_{t-1}$; t = 1, 2, … – часова складова.

Перш ніж починати аналіз необхідно оцінити невідомі параметри регресії. Це можна зробити за допомогою методу найменших квадратів [17]:

$$b_0 = \frac{n\sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2} \quad (2.10)$$

$$b_k = \frac{\sum_{i=1}^{n} y_i - a\sum_{i=1}^{n} x_i}{n}, \ \mathrm{k} = \overline{1,\mathrm{n}}, \quad (2.11)$$

За умови існування нелінійної регресії для аналізу необхідно перетворити її на лінійну за допомогою заміни змінних.

- Моделі ARIMA

Модель ARIMA – (AutoregRessive Integrated Moving Average) - один з найбільш поширених методів аналізу і прогнозування часових рядів. ARIMA використовує три основні параметри (p, d, q), які виражаються цілими числами. Разом ці три параметри враховують сезонність, тенденцію і шум в наборах даних: p – порядок авторегресії (AR), який дозволяє додати попередні значення часового ряду; d – порядок інтегрування (I) – він додає в модель поняття різниці часових рядів (визначає кількість минулих часових точок, які потрібно відняти з поточного значення); q – порядок ковзного середнього (MA), який дозволяє встановити похибку моделі як лінійну комбінацію значень помилок, що спостерігалися раніше [18]. Формальне визначення моделі має такий вигляд:

$$\Delta^d y_k = c + \sum_{i=1}^{p} a_i \Delta^d y_{k-i} + \sum_{j=1}^{q} b_j \varepsilon_{k-j} + \varepsilon_k, \quad (2.12)$$

де ε_k – стаціонарний часовий ряд; c, a_i, b_j – параметри моделі; Δ^d – оператор різниці часового ряду порядку d.

- Авторегресійна модель

Авторегресійні моделі використовуються для опису стаціонарних випадкових процесів. Особливістю стаціонарних часових рядів є те, що їх імовірнісні властивості рядів не змінюються в часі. Рівняння авторегресійної моделі має такий вигляд:

$$y_k = a_0 + a_1 y_{k-1} + a_2 y_{k-2} + \ldots + a_n y_{k-n} + \varepsilon, \quad (2.13)$$

де $y_{k-\text{i}}$ – значення у певних точках часового ряду; a_{i} – параметри; ε – випадкова помилка.

- Модель ковзного середнього

Суть цього методу складається в заміні фактичних значень показника їхніми усередненими величинами, що мають значно меншу варіацію, чим вихідні рівні ряду. Для парної і непарної кількості точок часового ряду змінна обчислюється з певними відмінностями. Рівняння моделі ковзного середнього для непарної кількості точок має такий вигляд:

$$y_k = (ay_{k-1} + ay_{k-2} + ay_{k-3})/n, \quad (2.14)$$

де $y_{k-\text{i}}$ – значення у трьох останніх точках часового ряду; n – загальна кількість точок часового ряду до шуканої; a – ваговий коефіцієнт, що має бути рівним 1.

Якщо, за умовою, кількість точок парна, то необхідно обчислити для $n-1$ та $n+1$. А потім повторити дії з отриманими результатами цих двох обчислень.

- Моделі експоненціального згладжування
  - Модель Брауна

Ідея методу полягає в тому, що прогнозне значення визначається через попереднє спрогнозоване значення, але скориговане на величину відхилення факту від прогнозу. Рівняння має такий вигляд:

$$y_k = \hat{y}_{k-1} + a(y_{k-1} + \hat{y}_{k-1}), \quad (2.15)$$

де y_{k-1} – відоме значення попередньої точки; $\hat{y}_{k-1}$ – передбачене значення попередньої точки; a – ваговий коефіцієнт, що має бути рівним від 0 до 1 при необхідності виявити певну тенденцію або від 0 до 2 для короткострокового прогнозу.

- Модель Хольта

Дана модель використовується для прогнозування часових рядів, коли є тенденція до зростання або падіння значень часового ряду. А також для

рядів, коли дані є не за повний цикл, і сезонність виділити неможливо. Рівняння моделі Хольта має вигляд:

$\hat{y}_k = a_{k-d} + db_{k-d}$, при тому, що

$$a_{k-d} = \alpha y_{k-d} + (1-\alpha)(a_{k-d-1} - b_{k-d-1}) \text{ і} \quad (2.16)$$

$$b_{k-d} = \beta(a_{k-d} - \alpha_{k-d-1}) + (1-\beta)b_{k-d-1},$$

де $\alpha, \beta -$ коефіцієнти чутливості моделі, які можуть бути рівними від 0 до 1. Використовується за умови існування тренду; d – крок [19].

- BATS і TBATS

Назва є абревіатурою для тригонометричного, перетворення Боакса-Кокса, помилок ARIMAX, трендових і сезонних компонентів. Ці два методи прогнозування використовуються у ситуації часових рядів з поступово змінною сезонністю. Моделювання кожної сезонності відбувається шляхом моделювання тригонометричного представлення на основі рядів Фур'є. Однією з основних переваг цього підходу є те, що він вимагає тільки 2 початкових стани, незалежно від тривалості періоду. TBATS відрізняється від BATS лише можливістю моделювати нецілі значення довжини періодів.

Транформація Боакса-Кокса має вигляд:

$$y_t^{(\omega)} = \begin{cases} \dfrac{y_t^{\omega} - 1}{\omega}, & \omega \neq 0 \\ \log y_t, & \omega = 0 \end{cases} \quad (2.17)$$

Спостереження моделюються як ряд:

$$y_t^{(\omega)} = l_{t-1} + \varphi b_{t-1} + \sum_{i=1}^{M} s_{t-m_i}^{(i)} + d_t, \quad (2.18)$$

де $b_t = (1-\varphi)b + \varphi b_{t-1} + \beta d_t$ – відображає глобальний тренд;

$l_t = l_{t-1} + \varphi b_{t-1} + \alpha d_t$ – відображає локальний тренд;

$d_t = \sum_{i=1}^{p} \varphi_i d_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$ – помилки у формі ARIMAX;

Кожен із M сезонних періодів складається з $s_t^{(i)} = \sum_{j=1}^{k_i} s_{j,t}^{(i)}$,

де кожен доданок моделюється за допомогою ряду Фур’є [20]:

$$s_{j,t}^{(i)} = s_{j,t-1}^{(i)} \cos \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \sin \lambda_j^{(i)} + \Gamma_1^{(i)} d_t$$
$$s_{j,t}^{(i)} = -s_{j,t-1}^{(i)} \sin \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \cos \lambda_j^{(i)} + \Gamma_2^{(i)} d_t \quad (2.19)$$

Кращим рішенням для моделювання процесу зміни курсу і прогнозування буде обрати кілька моделей і порівняти результати за кожною з них.

2.3 Аналіз часових рядів методами машинного навчання

Сутність машинного моделювання системи полягає в проведенні на обчислювальній машині експерименту з моделлю, яка являє собою деякий програмний комплекс, що описує формально або алгоритмічно поведінку елементів системи в процесі її функціонування, тобто в їх взаємодії один з одним і зовнішнім середовищем.

Під комп'ютерною моделлю розуміють умовний образ об'єкта або деякої системи об'єктів (або процесів), описаний за допомогою взаємопов'язаних комп'ютерних таблиць, блок-схем, діаграм, графіків, малюнків, анімаційних фрагментів, гіпертекстів, що відображають структуру і взаємозв'язки між елементами об'єкту. Комп'ютерні моделі такого виду називаються структурнофункціональними моделями. Також комп'ютерною моделлю називається окрема програма, сукупність програм, програмний комплекс, що дозволяє за допомогою послідовності обчислень і графічного відображення їх результатів відтворювати (імітувати) процеси функціонування об'єкта, системи об'єктів за умови впливу на об'єкт різних випадкових факторів. Такі моделі називаються імітаційними моделями.

Комп'ютерне моделювання – це метод розв'язання задачі аналізу або синтезу складної системи на основі використання її комп'ютерної моделі. Суть комп'ютерного моделювання полягає в отриманні кількісних і якісних результатів за наявною моделлю. Кількісні висновки в основному носять

характер прогнозу деяких майбутніх або пояснення минулих значень змінних характеризують систему.

Основні вимоги до комп'ютерних моделей:

1. Повнота моделі, точність і достовірність, адекватність.

2. Гнучкість моделі для відтворення різних ситуацій, алгоритмів і параметрів системи.

3. Структура моделі повинна бути блоковою, тобто допускати можливість заміни, додавання і виключення деяких частин без переробки всієї моделі.

4. Програмні і технічні засоби повинні забезпечувати ефективну (по швидкодії і пам'яті) машинну реалізацію моделі.

5. Має бути реалізовано заплановані машинні експерименти з моделлю системи з використанням аналітико-імітаційного підходу [21].

Після того як визначено завдання, підготовані дані, критерії оцінки і характеристики, можна починати моделювати. Моделювання ділиться на три частини: вибір моделі, поліпшення моделі, порівняння її з іншими [22].

Автоматизувати прогнозування можна за рахунок машинного навчання. На рис. 2.2 вказано, які існують види машинного навчання.

Рисунок 2.2 – Види машинного навчання

Існують такі моделі машинного навчання для прогнозування часових рядів [23]:

- Еластична мережа

Модель регресії з двома регуляризаторами l_1, l_2. Еластична мережа корисна, коли є кілька функцій, які корелюють один з одним. Цільова функція мінімізації має такий вигляд [24]:

$$\min_w \frac{1}{2n_{\text{samples}}} ||Xw - y||_2^2 + \alpha\rho||w||_1 + \frac{\alpha(1-\rho)}{2}||w||_2^2$$

- Спрощений алгоритм GBM

Модель є одним із видів дерев прийняття рішень. Адгоритм будує пряму поетапну адитивну модель, реалізуючи градієнтний спуск у функціональному просторі [25]:

$$f(x) = f^M(x) = \sum_{m=0}^{M} f_m(x) = f_0(x) + \sum_{m=1}^{M} \eta\rho_m\varphi_m(x)$$

$$f_0(x) = \theta = arg\ min_\theta \sum_{i=1}^{n} L(y_i, \theta)$$

- Алгоритм " до найближчих сусідів"

Регресія на основі сусідів може бути використана в тих випадках, коли мітки даних є безперервними, а не дискретними змінними. Мітка, присвоєна точці запиту, обчислюється на основі середнього значення міток її найближчих сусідів [26].

- Лассо LARS

Алгоритм аналогічний прямій покрокової регресії, але замість включення ознак на кожному кроці розрахункові коефіцієнти збільшуються в напрямку, рівному кореляції кожного з них із залишком. Замість того щоб давати векторний результат, рішення

LARS складається з кривої, що позначає рішення для кожного значення функції LARS [27].

- Стохастичний градієнтний спуск (SGD)

Ітераційний метод для оптимізації цільової функції з відповідними властивостями гладкості. Його можна розцінювати як стохастичну апроксимацію оптимізації методом градієнтного спуску, оскільки він замінює реальний градієнт [28]:

$$E(w,b) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w)$$

- Випадковий ліс

У випадкових лісах кожне дерево в ансамблі будується з вибірки з заміною. Крім того, при розбитті кожного вузла під час побудови дерева краще розбиття буде знайдено або з усіх вхідних об'єктів, або з випадкової підмножини. Метою цих двох джерел випадковості є зменшення дисперсії оцінки [29].

- Xgboost

Екстремальне градієнтне підсилювання належить до класу дерев. Це програмна бібліотека з відкритим кодом, яка пропонує систему градієнтного підсилювання [30].

- Auto-ARIMA

Auto-ARIMA працює шляхом проведення диференціальних тестів (наприклад, Квятковського–Філліпса–Шмідта–Шина, доповненого Діккі-Фуллера або Філліпса–Перона) для визначення порядку диференціювання, а потім підгонки моделей в межах певних діапазонів [31].

- Prophet

Модель Prophet працює найкращим чином з часовими рядами, яким притаманні сильні сезонні ефекти, при наявності даних за кілька

минулих сезонів. Характеризується високою точністю і швидкістю, надійністю при наявності викидів, відсутніх даних і істотних змін у часових рядах [32].

3. ДАНІ ДЛЯ АНАЛІЗУ

3.1 Опис даних

У даній роботі проводиться дослідження таких даних як курси найпопулярніших криптовалют і вплив дописів у соціальній мережі на них..

Список криптовалют було складено за даними міжнародного сервісу coinmarketcap.com станом на 20 квітня [33]. При визначені найпопулярніших токенів беруться до уваги такі показники як ціна у доларах, ринкова капіталізація, обсяги продажу за останні 24 години, кількість грошей, які знаходяться «на руках». Визначені показники порівнюються і визначаються лідери на ринку. В результаті визначено такі токени: Bitcoin, Ethereum, Binance Coin.

Отримання даних відбувалося за допомогою API – інтерфейсу прикладного програмування, який визначає взаємодію між декількома програмними додатками. Він визначає види викликів або запитів, які можуть бути зроблені, як їх зробити, формати даних, які слід використовувати, угоди, яких слід дотримуватися. API надає доступ до необхідних даних а результаті запитів.

Історія курсу даних криптовалют була отримана з відкритого джерела API на сайті coingecko.com [34]. Запит на отримання даних написано мовою програмування Python. У програмному коді, представленому у додатку А.1, підключено наступні модулі:

- PrettyTable – для виводу таблиці із даними історії курсу криптовалют;
- CoinGeckoAPI – для посилання запиту до coingecko.com на отримання щоденних даних історії курсу криптовалют;

- json – для збереження отриманих даних до json-файлу;
- datetime – для перетворення отриманих дат із UNIX-формату до формату дати та часу у вигляді «рік-місяць-день».

Був визначений масив coinsIds із ID кожного токену, що був обраний вище за допомогою coinmarketcap.com. ID токенів були взяті із coingecko.com.

Проміжок часу для отримання щоденних даних був обраний з 1 жовтня 2019 року до 17 квітня 2021 року, а валюта цін – USD (єдина доступна валюта для отримання даних з coingecko.com на даний момент).

Після отримання даних за обраний проміжок часу за допомогою модуля CoinGeckoAPI, дані заносяться у таблицю PrettyTable та до масиву, який далі заноситься до json-файлу “coinsData.json”, а таблиця виводиться користувачу у консольному вигляді. Приклад виводу даних користувачу та до json-файлу показаний на рис. 3.1 та 3.2.

bitcoin	2021-04-09	58065.644024815316
bitcoin	2021-04-10	58152.993262141834
bitcoin	2021-04-11	59979.39281571831
bitcoin	2021-04-12	59988.02095852983
bitcoin	2021-04-13	59911.020594847316
bitcoin	2021-04-14	63576.676041048275
bitcoin	2021-04-15	62807.12323259299
bitcoin	2021-04-16	63179.772446084695
bitcoin	2021-04-17	61497.299569441195
ethereum	2019-10-02	180.63244082835516
ethereum	2019-10-03	180.57893058773016
ethereum	2019-10-04	176.85852555342686
ethereum	2019-10-05	175.86315365350248
ethereum	2019-10-06	176.2772565733896
ethereum	2019-10-07	170.44710628243018
ethereum	2019-10-08	180.14424760253544
ethereum	2019-10-09	180.78979326727546
ethereum	2019-10-10	193.33993781390976

Рисунок 3.1 – Приклад виводу даних історії курсу криптовалют користувачу

Рисунок 3.2 – Приклад збереження даних історії курсу криптовалют у форматі JSON

Для дослідження соціальної думки було обрано таку соціальну мережу як Twitter. Ця мережа відмінна тим, що користувачі мають вмістити основну думки допису в обмежену кількість символів, що значно полегшує аналіз дописів і пошук ключових слів. Визначення тональності дописів і пошук ключових слів відбуватиметься за допомогою машинного навчання.

Доступ до дописів користувачів було отримано також за допомогою API:

1. Були отримані ключі доступу до Twitter API, за допомогою якого є можливість отримувати останні 3200 публічних твітів кожного окремого користувача. Ключі доступу були отримані через відповідну заявку на доступ, із описом даного дипломного проекту та його цілей.
2. Було встановлено Python-бібліотеку tweepy. Для цього необхідно відкрити Anaconda Prompt та виконати команду: pip install tweepy.
3. Для роботи з Python-файлами та задля виконання коду був використаний Jupyter Notebook. Напочатку була створена папка

проекту, а в ній - папка "tweepy_results" для збереження отриманих даних (твітів окремих людей у форматі json-файлів) та файл "credentials_file1.py", у якому зберігаються значення ключів для доступу до Twitter API, а саме:
ACCESS_TOKEN_KEY
ACCESS_TOKEN_SECRET
CONSUMER_KEY
CONSUMER_SECRET

4. У головному файлі програми, у якому будуть отримані твіти користувача Twitter, потрібно імпортувати наступні Python-модулі: print_function, tweepy, json, sys, а також змінні із файлу credentials_file1

5. Створюємо об'єкт auth для доступу до Twitter API, використовуючи модуль tweepy та наявні ключі доступу.

6. За допомогою об'єкту auth створюється об'єкт api, через який будуть посилатися запити на твіти користувача.

7. Створюємо метод get_tweets, який приймає об'єкт api та ім'я (нікнейм) користувача Twitter. У цьому методі реалізовуємо відправку запитів на твіти переданого користувача у межах циклу while, враховуючи обмеження Twitter'у на максимум 200 твітів за один запит. Таким чином отримуємо останні 3200 твітів користувача Twitter (3200 твітів - це також є обмеженням Twitter API).

8. Отримавши останні 3200 твітів користувача, записуємо їх у окремий json-файл, та надалі змінюємо нікнейм на іншого користувача, і виконуємо код знову. Таким чином можливо пройти всіх потрібних користувачів Twitter та отримати їх останні 3200 твітів.

Для аналізу було обрано акаунти осіб, які є відомими і впливовими на ринку криптовалюти.

Запит на отримання даних написано мовою програмування Python. У додатку А.2 представлений код отримання даних твітів обраних Twitter-аккаунтів.

Надалі усі твіти зі збережених json-файлів кожного окремого Twitter-аккаунту були оброблені та збережені в один json-файл за допомогою коду, представленого у додатку А.3. Це було зроблено задля зручності подальшої роботи із твітами з аналізу настрою всіх твітів та отримання цих твітів із одного консолідованого місця, замість отримання твітів із окремих десятків json-файлів. Таким чином, у програмному коді додатку А.3 розрізнені твіти різних Twitter-акаунтів були зчитані із окремих json-файлів, дані з твітів були оброблені та додані до великого масиву даних всіх твітів, розробленому у простій для подальшого зчитування формі із зрозумілими назвами полів, та даний великий масив всіх твітів був записаний до єдиного json-файлу.

3.2 Підготовка даних

Дані для аналізу отримані за допомогою мови програмування Python (код у Додатку А). Опис змінних у таблицях 3.1 та 3.2.

Таблиця 3.1 – Змінні та позначення програмного коду отримання історії курсу криптовалют

Змінна	Опис змінної
bitcoin	Ідентифікатор криптовалюти Біткоін, який використовується для отримання історії курсу Біткоіна з API сайту https://www.coingecko.com/
coinsIds	Масив ідентифікаторів криптовалют, описаних вище
cg	Об’єкт для роботи із API сайту https://www.coingecko.com/
dataCurrency	Валюта («usd»), у якій будуть отримані ціни з API сайту https://www.coingecko.com/
dataFrom	Дата, яка буде передана до API-запиту та з якої буде починатися історія курсу криптовалют. Передається

	у UNIX-форматі
dataTo	Дата, яка буде передана до API-запиту та якою буде закінчуватися історія курсу криптовалют. Передається у UNIX-форматі
x	Об’єкт таблиці, до якої будуть записані отримані дані історії курсу криптовалют та яка буде виведена користувачу
x.field_names	Назви столбців таблиці ("Coin name", "Date", "Price (in USD)")
data	Масив для збереження отриманих даних історії курсу криптовалют у JSON-файл
coinId	Об’єкт-ітератор масиву ідентифікаторів криптовалют coinsId
response	Об’єкт відповіді API-запиту до сайту https://www.coingecko.com/, який містить у собі дані історії курсу криптовалют, а саме – масив об’єктів, у яких знаходяться ідентифікатор криптовалюти, дата та ціна криптовалюти у доларах США, дійсна на цю дату. Ці об’єкти містять у собі щоденну інформацію (1 об’єкт = 1 день) із ціною криптовалюти, починаючи з дати змінної dataFrom та закінчуючи датою змінної dataTo
price	Об’єкт-ітератор масиву, що знаходиться у змінній response
unixTimestamp	Значення дати об’єкту price у UNIX-форматі, записане у формі рядка (string)
correctUnixTimestamp	Відредаговане та правильне значення дати об’єкту price у UNIX-форматі, записане у формі рядка (string)
ts	Правильне значення дати об’єкту price у UNIX-форматі, записане у формі цілого числа (int)
resultDate	Правильне значення дати об’єкту price у наступному форматі дати: YYYY-MM-DD
coinPriceDetails	Об’єкт із полями «CoinName», «Date» і «Price», у якому записується отрмана із API-запиту ціна криптовалюти за конкретну дату. Даний об’єкт додається до масиву data, та надалі весь масив зберігається до JSON-файлу

Таблиця 3.2 – Змінні та позначення програмного коду отримання твітів

ACCESS_TOKEN_KEY	Ключ доступу, взятий із Twitter API для використання у API-запитах на отримання

	твітів окремого користувача
ACCESS_TOKEN_SECRET	Секретний код доступу, взятий із Twitter API для використання у API-запитах на отримання твітів окремого користувача
CONSUMER_KEY	Ключ доступу користувача, взятий із Twitter API для використання у API-запитах на отримання твітів окремого користувача
CONSUMER_SECRET	Секретний код доступу користувача, взятий із Twitter API для використання у API-запитах на отримання твітів окремого користувача
auth	Об’єкт аутентифікації (доступу) до запитів Twitter API, створений за допомогою програмного модулю tweepy
get_tweets (api=None, screen_name=None)	Функція отримання твітів окремого користувача за допомогою запиту до Twitter API, використовуючи програмний модуль tweepy
api	Об’єкт API програмного модулю tweepy, при створенні якого був переданий об’єкт доступу до Twitter API (auth), та який використовується для запитів до Twitter API
screen_name	Ідентифікатор Twitter-акаунту користувача, твіти якого необхідно отримати. Приклад: elonmusk
timeline	Об’єкт відповіді на API-запит для отримання твітів користувача (максимум – 200 твітів за один запит). Містить у собі масив об’єктів із даними твітів
earliest_tweet	Ідентифікатор найбільш раннього твіта із останніх 200 отриманих твітів. Використовується для отримання наступних 200 твітів та ітерації у циклі отримання твітів
tweets	Об’єкт відповіді на API-запит для отримання наступних 200 твітів користувача у циклі
new_earliest	Ідентифікатор нового найбільш раннього твіта із останніх 200 отриманих твітів у циклі.
tweet	Об’єкт-ітератор із даними твіта з масиву твітів timeline

4. ПРАКТИЧНА РЕАЛІЗАЦІЯ

4.1 Аналіз дописів з соціальної мережі

Фільтрація дописів:

У додатку А.4 представлений програмний код, який відповідає за фільтрацію твітів за словами, що відносяться до крипто-сфери, а також за назвами обраних криптовалют. У даному програмному коді була проведена робота за наступними кроками:

1. Встановлено Python-бібліотеку nltk, яка містить в собі багато корисних програмних модулів, деякі з яких надалі будуть використані для фільтрації тексту твітів та його аналізу за допомогою наївного Баєсівського класифікатору, за попереднього навчання класифікатору на тестовій вибірці даних.
2. Розроблений клас PreProcessTweets для розбивання тексту твітів на окремі масиви слів.
3. Раніше збережені твіти у форматі JSON були зчитані та оброблені за допомогою розробленого класу. Таким чином, масив слів із тексту кожного твіта був записаний до нового окремого поля твіта «Tweet words», і збережений до окремого файлу у форматі JSON.
4. Був оголошений масив слів, що відносяться до крипто-сфери та містять назви обраних криптовалют.
5. За допомогою даного масиву слів твіти були відфільтровані: так, до нового результуючого файлу у форматі JSON були відібрані лише ті твіти, масиви слів яких містили в собі хоча б одне слово із оголошеного масиву слів для фільтрації. Так, із близько 70 000 твітів було відібрано близько 15 000 таких, що безпосередньо стосуються крипто-сфери та обраних криптовалют.

Аналіз дописів:

Перед переходом до аналізу настрою твітів та визначення, яким є кожен твіт – позитивним, негативним чи нейтральним, необхідно було підготувати тестовий набір даних (твітів), які вже мають відповідні позначки на кожному з твітів: «positive», «negative» або «neutral». Цей тестовий набір даних необхідний для машинного навчання нашого наївного Баєсівського класифікатору правильно визначати настрій твіта, маючи досвід попередньо оброблених твітів із вже визначеними для них настроями. Тестовий набір із твітами (з різними настроями) про компанії «Apple», «Google» та «Microsoft» має назву twitter_corpus і знаходиться у вільному доступі в Інтернеті. Він має близько 1500 твітів і був оброблений програмним кодом, описаним вище, задля розбиття тексту твітів на окремі масиви слів, які необхідні для занесення до програмного модулю nltk для машинного навчання наївного Баєсівського класифікатору. У додатку А.5 наведений програмний код, у якому були проведені наступні дії:

1. Оголошені функції buildVocabulary та extract_features, які відповідно слугують для збирання всіх слів з текстів твітів тестового набору даних до одного словника та визначення, чи входять слова твітів нашого основного набору до даного словника.

2. Для початку роботи із програмним модулем nltk для аналізу настрою тексту та з наївним Баєсівським класифікатором, тестовий (тренувальний) набір даних був зчитаний із файлу у форматі JSON. Цей набір був оброблений функцією buildVocabulary. Таким чином, на основі слів з тексту тестового набору даних був побудований словник слів для аналізу настроїв твітів нашого основного набору даних.

3. За допомогою програмних модулів nltk.classify і nltk.NaiveBayesClassifier був створений об’єкт наївного Баєсівського класифікатора. Вбудована у програмний модуль nltk.NaiveBayesClassifier функція train провела машинне навчання

створеного об’єкту класифікатора на основі тренувального набору даних trainingFeatures.

4. Надалі був відкритий файл із раніше відфільтрованими твітами, які поміж полів твіта вже містять поле «Tweet words» із масивом слів тексту твіта.

5. Після зчитування твітів із файлу, для кожного твіта була використана вбудована функція «classify» нашого об’єкту наївного Баєсівського класифікатору NBayesClassifier. За допомогою даної функції та написаної раніше у програмному коді функції extract_features, кожному твіту була присвоєна позначка настрою даного твіта («positive», «negative» чи «neutral»). Так, при обробці кожного твіта нашого основного набору даних, наш наївний Баєсівський класифікатор опирався на створений словник і на значення обробленого раніше тестового (тренувального) набору даних twitter_corpus. Маючи близько 1500 твітів із визначеними настроями, класифікатор надалі самостійно визначав настрій кожного твіта нашого основного набору даних, що складається із приблизно 15 000 твітів.

6. Визначені настрої були записані до нового поля «Sentiment label» кожного твіта, та оновлені дані із цим полем були збережені до нового файлу у форматі JSON.

На рис. 4.1 – 4.3 можна розглянути частину результату роботи програми.

```
Author: Pomp 🌪
Tweet text: "@NotoriousPtG @RaoulGMI @SantiagoAuFund Of course. These guys really don’t get it and believe bitcoiners are goi
ng… https://t.co/ZSAoEBqEFz"
Sentiment label: neutral
-----------------------------------------------
Author: Pomp 🌪
Tweet text: "The bitcoin memes have done more for marketing the greatest store of value asset than any corporate marketing te
am could ever dream of."
Sentiment label: positive
-----------------------------------------------
Author: Pomp 🌪
Tweet text: "@RaoulGMI @SantiagoAuFund Completely false.

The biggest advocates of Bitcoin are all meme lords.

Saylor, Musk, Ch… https://t.co/SjIzxR86wk"
Sentiment label: negative
```

Рисунок 4.1 – Результат фільтрації дописів

```
Author: Pomp 🌩
Tweet text: "@NEVERWAL Yes, variable rates. Bitcoin rate depends on how many Bitcoin you deposit too. They are pretty transpa
ren… https://t.co/rWkMJjFzI3"
Sentiment label: neutral
------------------------------------------------
Author: Pomp 🌩
Tweet text: "@moneyball @sqcrypto You guys have been ahead of the curve on so much, including LN and Bitcoin dev grants :)"
Sentiment label: positive
------------------------------------------------
Author: Pomp 🌩
Tweet text: "💥 2021 is gonna be fun!

I'm launching a (sold out) cohort-based course on Bitcoin/crypto in 2 weeks to help people… https://t.co/jEMxRF6OQ4"
Sentiment label: neutral
------------------------------------------------
Author: Pomp 🌩
Tweet text: "TRANSLATION: UBS will announce they bought Bitcoin in a few weeks. https://t.co/ghLBM0y0dC"
Sentiment label: neutral
```

Рисунок 4.2 – Результат фільтрації дописів

```
Tweet text: "RT @RookieXBT: Crypto a scam?

Crypto just helped me pay off my parents mortgage and then some, as well as get my pops his dream car.

From…"
Sentiment label: positive
------------------------------------------------
Author: Beastlorion
Tweet text: "What's the best exchange for selling crypto for cash?"
Sentiment label: neutral
------------------------------------------------
Author: Beastlorion
Tweet text: "RT @WSBChairman: If you don't own bitcoin, you're short bitcoin."
Sentiment label: neutral
------------------------------------------------
Author: Beastlorion
Tweet text: "I wonder if crypto will crash with the stonk market this time around. probably, right? But it's so dang bullish.
Hard to say for sure."
Sentiment label: negative
```

Рисунок 4.3 – Результат фільтрації дописів

4.2 Аналіз часового ряду

Часовий ряд побудовано на основі історії курсу криптовалюти Bitcoin за період з 10 лютого 2019 року до 17 квітня 2021 року.

Побудуємо графік, який показує зміну курсу (рис. 4.4). На графіку бачимо позитивний тренд. Тож, гістограма розподілу, яка додатково представлена ліворуч, має один пік. До того ж, варто зазначити, що графік не перетинає значення 0.

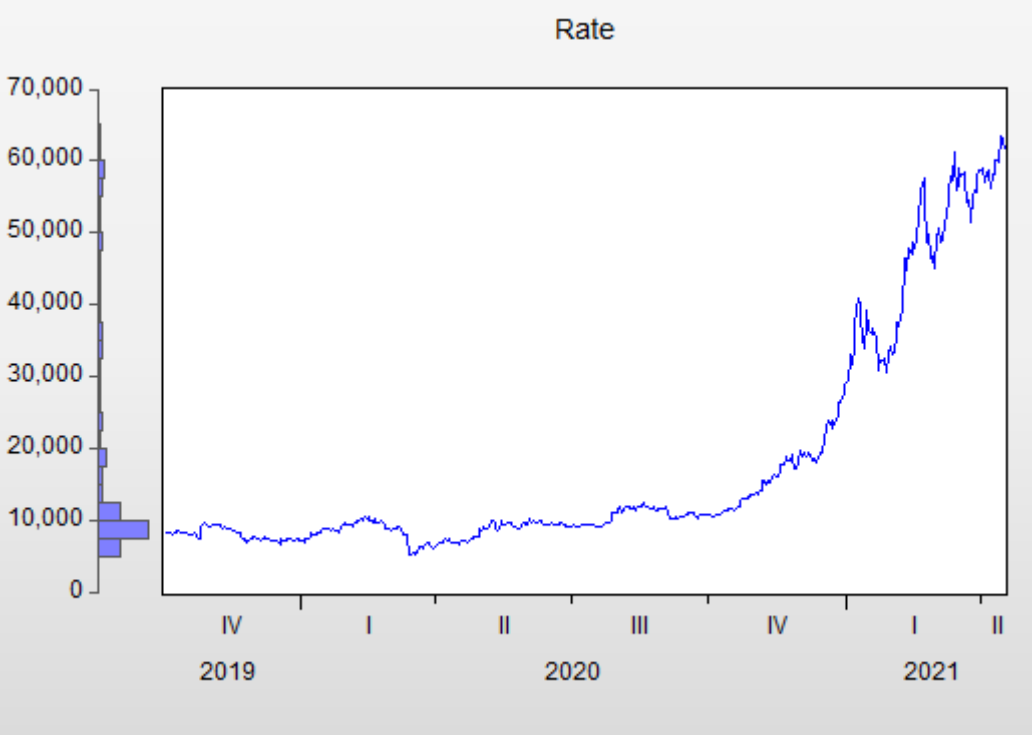


Рисунок 4.4 – Графік зміни курсу криптовалюти Bitcoin

За візуальним аналізом можна зробити висновок, що ряд не є стаціонарним. Для підтвердження виконаємо тест Дікі-Фулера. Автоматично можна оптимізувати кількість лагів у базовому рівнянні Дікі–Фулер тесту, що становить 18. Було обрано оптимізацію кількості лагів за інформаційним критерієм Шварца.

Перший крок – це тестування стаціонарності у первісному часовому ряді. Протестуємо дані в рівнях. Результат даного тестування показаний на рис. 4.5.

Augmented Dickey-Fuller Unit Root Test on RATE

Null Hypothesis: RATE has a unit root
Exogenous: Constant, Linear Trend
Lag Length: 0 (Automatic - based on SIC, maxlag=18)

		t-Statistic	Prob.*
Augmented Dickey-Fuller test statistic		-0.215342	0.9926
Test critical values:	1% level	-3.974439	
	5% level	-3.417821	
	10% level	-3.131355	

*MacKinnon (1996) one-sided p-values.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(RATE)
Method: Least Squares
Date: 05/23/21 Time: 12:03
Sample (adjusted): 10/03/2019 4/17/2021
Included observations: 563 after adjustments

Variable	Coefficient	Std. Error	t-Statistic	Prob.
RATE(-1)	-0.000874	0.004060	-0.215342	0.8296
C	-95.75374	80.52636	-1.189098	0.2349
@TREND("10/02/2019")	0.728016	0.385981	1.886144	0.0598

R-squared	0.013150	Mean dependent var	94.37086
Adjusted R-squared	0.009625	S.D. dependent var	944.2249
S.E. of regression	939.6697	Akaike info criterion	16.53425
Sum squared resid	4.94E+08	Schwarz criterion	16.55734
Log likelihood	-4651.391	Hannan-Quinn criter.	16.54326
F-statistic	3.730979	Durbin-Watson stat	2.073488
Prob(F-statistic)	0.024567		

Рисунок 4.5 – Результати тестування стаціонарності ряду

Основний результат Дікі–Фулер тесту – отримання розрахункового значення та критичних значень τ-статистики МакКінона. Як можна побачити, абсолютна величина розрахункового значення τ-статистики МакКінона (-0.215342) менша за абсолютні величини критичного значення при 1 %, 5 % та 10 % рівнях значущості. Крім того, p-value (імовірність) тесту дорівнює 0,992 (99,2 %, тобто p-value >10 %). Отже, ми не можемо відкинути нульову гіпотезу щодо наявності одиничного кореня в часовому ряді.

Таким чином, ми не відкидаємо нульову гіпотезу про наявність одиничного кореня. Це означає, що ряд в рівнях нестаціонарний. Автоматичний підрахунок виявив, що ряд стає стаціонарним у других різницях, тобто має стаціонарність 2-го порядку (розподіл вибірки

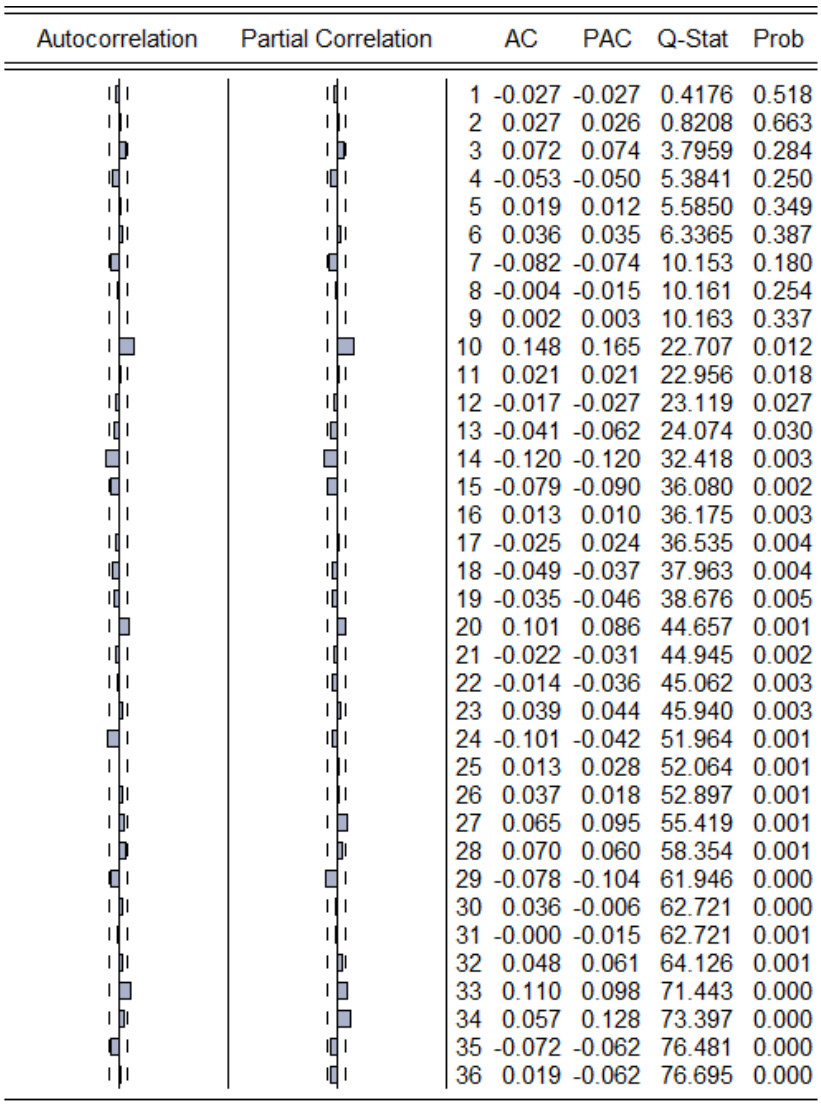
випадкового процесу рівний розподілу вибірки, зсунутої у часі для усіх значень другого порядку).

p-value (імовірність) тесту дорівнює 0,000 (0 %, тобто p-value <10 %). Це означає, що можна відкинути нульову гіпотезу щодо наявності одиничного кореня (нестаціонарності) в ряді других різниць із мінімальною імовірністю помилитися (майже в 0 % випадків зі 100 %).

4.3 Побудова ARIMA і ARIMAX моделей

Таким чином, ряд у других різницях є стаціонарним, а ряд у рівнях має порядок інтеграції 2. Отже, ARIMA модель будуватимемо для ряду в других різницях.

Зазвичай курс валют не залежить від сезонної складової. Для того, щоб перевірити цей факт на даному часовому ряді, необхідно проаналізувати корелограму часового ряду (показана на рис. 4.6).



Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	-0.027	-0.027	0.4176	0.518
		2	0.027	0.026	0.8208	0.663
		3	0.072	0.074	3.7959	0.284
		4	-0.053	-0.050	5.3841	0.250
		5	0.019	0.012	5.5850	0.349
		6	0.036	0.035	6.3365	0.387
		7	-0.082	-0.074	10.153	0.180
		8	-0.004	-0.015	10.161	0.254
		9	0.002	0.003	10.163	0.337
		10	0.148	0.165	22.707	0.012
		11	0.021	0.021	22.956	0.018
		12	-0.017	-0.027	23.119	0.027
		13	-0.041	-0.062	24.074	0.030
		14	-0.120	-0.120	32.418	0.003
		15	-0.079	-0.090	36.080	0.002
		16	0.013	0.010	36.175	0.003
		17	-0.025	0.024	36.535	0.004
		18	-0.049	-0.037	37.963	0.004
		19	-0.035	-0.046	38.676	0.005
		20	0.101	0.086	44.657	0.001
		21	-0.022	-0.031	44.945	0.002
		22	-0.014	-0.036	45.062	0.003
		23	0.039	0.044	45.940	0.003
		24	-0.101	-0.042	51.964	0.001
		25	0.013	0.028	52.064	0.001
		26	0.037	0.018	52.897	0.001
		27	0.065	0.095	55.419	0.001
		28	0.070	0.060	58.354	0.001
		29	-0.078	-0.104	61.946	0.000
		30	0.036	-0.006	62.721	0.000
		31	-0.000	-0.015	62.721	0.001
		32	0.048	0.061	64.126	0.001
		33	0.110	0.098	71.443	0.000
		34	0.057	0.128	73.397	0.000
		35	-0.072	-0.062	76.481	0.000
		36	0.019	-0.062	76.695	0.000

Рисунок 4.6 – Корелограма часового ряду

За наявності сезонності в аналізованому часовому ряді значення ACF/PACF для 18-го лагу мали би бути статистично значущими (статистично значимо відрізнятись від нуля). Вони є незначущими (стовпчики на 18-му лазі не заходять за пунктирну лінію), тобто значення ACF/PACF з лагом 18 не виходять за межі інтервалу довіри, відповідно, статистично незначно відрізняються від нуля. Значить ряд не має сезонну складову.

Аналіз поведінки ACF/PACF свідчить про змішаний характер процесу. Отже, для виявлення порядків AR- та MA-складової необхідно застосовувати спеціальні процедури. За допомогою процедури Хенона–Рісанена спочатку звичайним методом найменших квадратів оцінюється AR-складова моделі. При цьому оптимальними лагами для включення в модель вважають такі, за

яких досягається мінімальне значення Акайк-інформаційного критерію (AIC). Після визначення оптимальної AR-складової необхідно утворити ряд залишків цієї моделі для наступного використання при визначені оптимального порядку MA-складової моделі. На цьому етапі до визначеної AR-складової поступово додаються MA(1), MA(2) ... MA(q)-складові і розраховуються значення Шварц критерію. Модель, яка має найменше значення Шварц критерію, необхідна для подальшого аналізу. Після того, як визначено попередню специфікацію моделі, її потрібно переоцінити нелінійним методом найменших квадратів (NLS). Результати даної оцінки наведені на рис. 4.7.

Dependent Variable: D(RATE,2)
Method: ARMA Maximum Likelihood (BFGS)
Date: 06/01/21 Time: 15:57
Sample: 10/04/2019 4/17/2021
Included observations: 562
Convergence achieved after 87 iterations
Coefficient covariance computed using outer product of gradients

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	0.661943	0.194690	3.399982	0.0007
AR(1)	-0.570229	0.010229	-55.74613	0.0000
AR(2)	-0.970331	0.009575	-101.3377	0.0000
MA(1)	-0.462104	1.623066	-0.284711	0.7760
MA(2)	0.462088	1.549261	0.298263	0.7656
MA(3)	-0.999984	6.707085	-0.149094	0.8815
SIGMASQ	838046.9	1228369.	0.682244	0.4954

R-squared	0.541025	Mean dependent var	-2.964593
Adjusted R-squared	0.536063	S.D. dependent var	1352.466
S.E. of regression	921.2041	Akaike info criterion	16.52220
Sum squared resid	4.71E+08	Schwarz criterion	16.57615
Log likelihood	-4635.739	Hannan-Quinn criter.	16.54327
F-statistic	109.0359	Durbin-Watson stat	1.983015
Prob(F-statistic)	0.000000		

Inverted AR Roots	-.29+.94i	-.29-.94i	
Inverted MA Roots	1.00	-.27-.96i	-.27+.96i

Рисунок 4.7 – Результати оцінювання нелінійним методом найменших квадратів

Модель у форматі коду програми має такий вигляд dlog(rate,2) c ar(1) ar(2) ma(1) ma(2) ma(3). Математично модель виглядає як

$$\Delta^2 rate_t = 0.662 - 0.57\,\Delta^2 rate_{t-1} - 0.97\,\Delta^2 rate_{t-2} - 0.4621\,\varepsilon_{t-1} + 0.462\,\varepsilon_{t-2} - 0.999\,\varepsilon_{t-3} + \varepsilon_t$$

Модель ARIMAX будується аналогічно, але включає фіктивну змінну – дописи у Твіттері. Фіктивна змінна була визначена як 0 або 1. Увесь період дослідження був розбитий на інтервали. На кожному часовому інтервалі обраховано середню кількість публікацій за темою із вибірки. Якщо кількість публікацій виявлялася меншою за середнє значення, то фіктивна змінна була 0 і навпаки, якщо кількість публікацій більша за середнє значення, то фіктивна змінна – 1.

Dependent Variable: DLOG(RATE,2)
Method: ARMA Maximum Likelihood (OPG - BHHH)
Date: 06/01/21 Time: 15:50
Sample: 10/04/2019 4/17/2021
Included observations: 562
Convergence achieved after 22 iterations
Coefficient covariance computed using outer product of gradients

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	0.000184	0.000111	1.656056	0.0983
FICT	-0.000328	0.000224	-1.460818	0.1446
AR(1)	-0.806892	0.124776	-6.466725	0.0000
MA(1)	-0.264954	20.24664	-0.013086	0.9896
MA(2)	-0.735046	78.90237	-0.009316	0.9926
SIGMASQ	0.001484	0.010221	0.145155	0.8846

R-squared	0.545235	Mean dependent var	-4.45E-05
Adjusted R-squared	0.541146	S.D. dependent var	0.057170
S.E. of regression	0.038726	Akaike info criterion	-3.642518
Sum squared resid	0.833838	Schwarz criterion	-3.596274
Log likelihood	1029.548	Hannan-Quinn criter.	-3.624464
F-statistic	133.3221	Durbin-Watson stat	2.000848
Prob(F-statistic)	0.000000		

Inverted AR Roots	-.81	
Inverted MA Roots	1.00	-.74

Рисунок 4.8 – Результати оцінювання моделі з фіктивною змінною нелінійним методом найменших квадратів

Модель у форматі коду програми має такий вигляд dlog(rate,2) c fict ar(1) ma(1) ma(2) . Математично модель виглядає як

$$\Delta^2 rate_t = 0.0001 - 0.0003 - 0.807\ \Delta^2 rate_{t-1} - 0.265\ \varepsilon_{t-1} - 0.735\ \varepsilon_{t-2} + \varepsilon_t$$

4.4 Прогнозування на основі ARIMA/ARIMAX моделей

Після побудови ARIMA/ARIMAX моделей отримаємо результати прогнозних розрахунків, значення основних критеріїв прогнозної якості, а також графічне відображення прогнозних значень.

Для оцінювання прогнозної якості моделей необхідно проаналізувати значення розрахованих критеріїв прогнозної якості. Одним із найпоширеніших є критерій середньої абсолютної процентної похибки (MAPE – Mean Absolute Percent Error), який показує середню абсолютну похибку прогнозу в відсотках. Для моделі ARIMA значення MAPE = 3,817%, для моделі ARIMAX значення MAPE = 3,821%, що свідчить про високу прогнозну якість обох моделей. Значення MAPE трохи відрізняються із-за впливу фіктивної змінної. Результати оцінювання прогнозної якості моделей наведені на рис. 4.9-4.10.

Forecast: RATEF	
Actual: RATE	
Forecast sample: 4/17/2021 5/18/2021	
Included observations: 32	
Root Mean Squared Error	2600.327
Mean Absolute Error	2029.081
Mean Abs. Percent Error	3.817292
Theil Inequality Coefficient	0.023664
Bias Proportion	0.194632
Variance Proportion	0.003058
Covariance Proportion	0.802310
Theil U2 Coefficient	1.071433
Symmetric MAPE	3.745284

Рисунок 4.9 – Результати оцінювання прогнозної якості моделі ARIMA

Forecast: RATEF_FICT	
Actual: RATE	
Forecast sample: 4/18/2021 5/18/2021	
Included observations: 31	
Root Mean Squared Error	2611.041
Mean Absolute Error	2022.339
Mean Abs. Percent Error	3.821303
Theil Inequality Coefficient	0.023876
Bias Proportion	0.182719
Variance Proportion	0.006816
Covariance Proportion	0.810464
Theil U2 Coefficient	1.063740
Symmetric MAPE	3.748157

Рисунок 4.10 – Результати оцінювання прогнозної якості моделі ARIMAX

На основі моделей ARIMA/ARIMAX можна спрогнозувати курс з 95% довіри. Прогноз будується на місяць вперед від зазначеної дати. Графік даного прогнозування представлений на рис. 4.11.

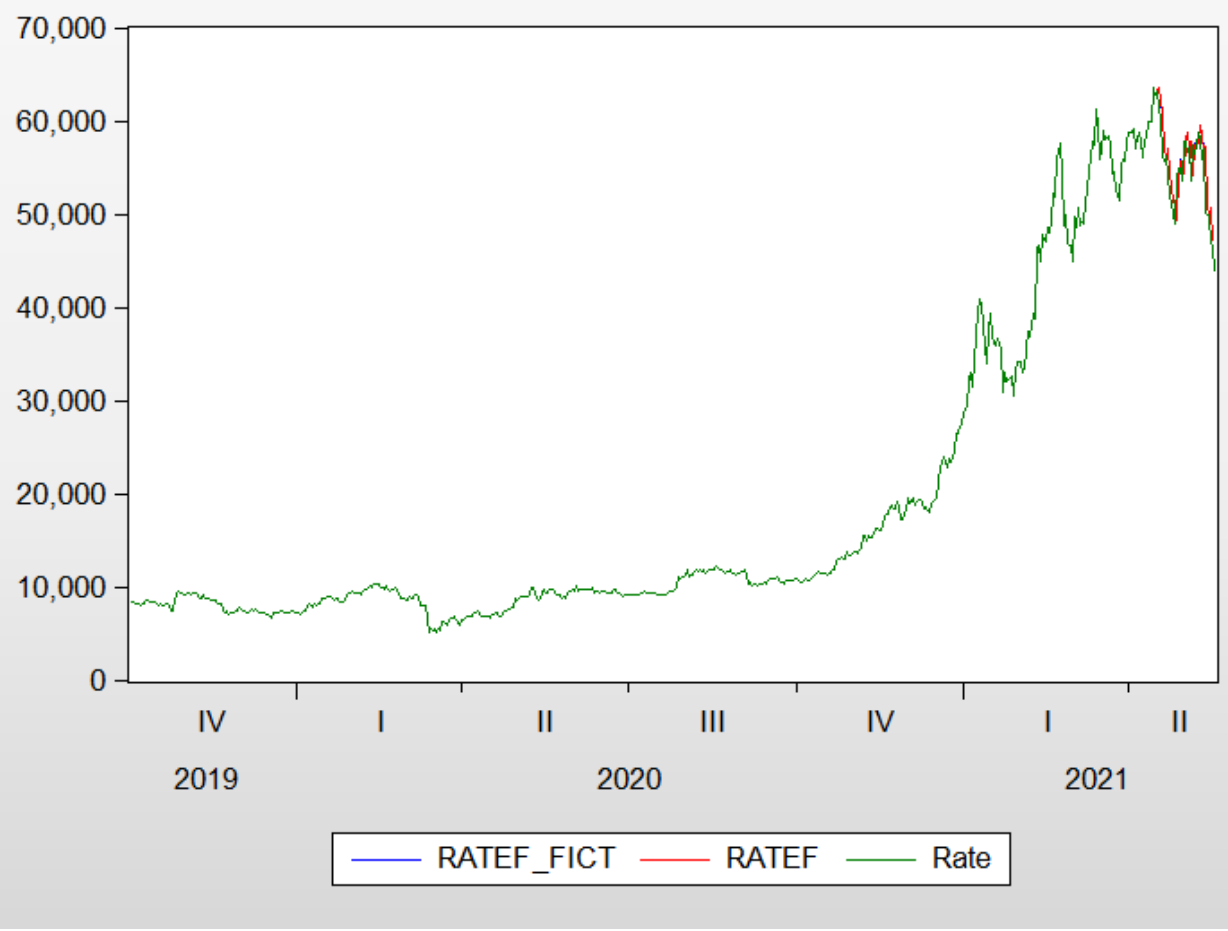


Рисунок 4.11 – Результати прогнозування

Синя лінія визначає прогноз з фіктивною змінною, а червона – без неї. Зеленою лінією позначено курс за увесь період дослідження, у тому числі той, що буд на період прогнозу. Можна вважати, що прогноз виявися достатньо точним.

4.5 Моделювання методом Prophet

Одним із методів машинного навчання створений у вигляді бібліотеки прогнозування Facebook Prophet. Це адитивна регресійна модель, яка може виявляти точки зміни для моделювання часових рядів. Також було застосовано додатковий зовнішній регресор – дописи у соціальній мережі (1 або 0). Код програми знаходиться у Додатку А.

Важливим кроком є встановлення бібліотеки fbprophet, яка включає алгоритми прогнозування. Далі необхідно підключити бібліотеки для роботи з даними і їх візуалізації, внести початкові дані, а також додати функцію, яка обраховуватиме модель і значення прогнозу. Модель часового ряду зображена на рис. 4.12.

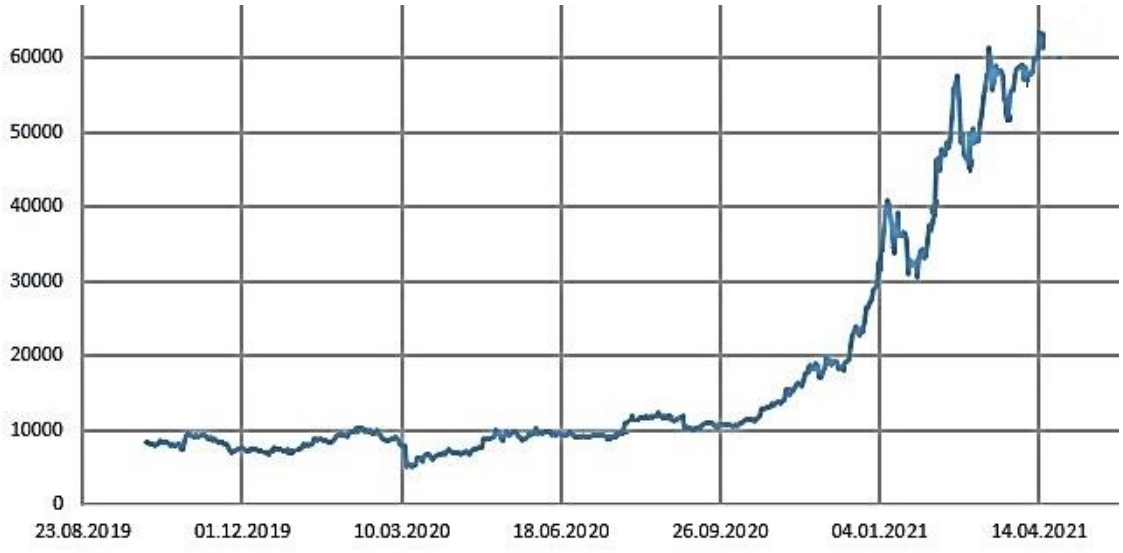


Рисунок 4.12 – Модель часового ряду

4.6 Прогнозування методом Prophet

Даний прогноз було побудовано на основі лагових значень зміних, а також застосовано побудову ряду Фур'є. Результати отримані на більш короткий період, ніж під час попереднього прогнозування, а саме на 6 місяців. Спроби прогнозувати на більш довгий період викликали помилку роботи програми. Результати прогнозування зображено на рис. 4.13.

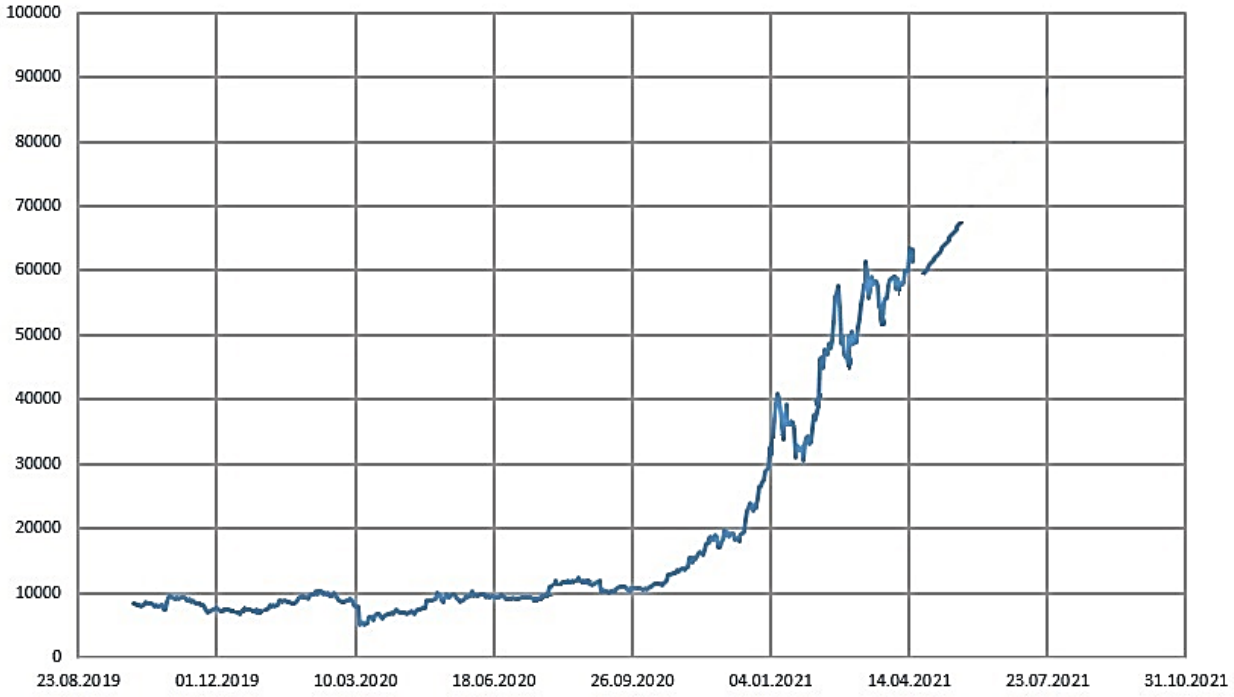


Рисунок 4.13 – Результати прогнозування

4.7 Порівняння методів моделювання і прогнозування

Тепер можна оцінити прогноз і порівняти результати Prophet з класичним методом прогнозування ARIMAX. Тож, MAPE ARIMAX складає 3,821%, а MAPE Prophet – 4,16%. Підвищити точність прогнозування можна за допомогою скорочення періоду прогнозу і додатковим факторам впливу.

Зміщена пропорція (bias proportion), що вказує на зв'язок між подіями і причинами, для Prophet складає 0,127, а для ARIMAX – 0,183.

На достовірність моделі вказує показник R^2, який для моделі Prophet складає 0,512, а для моделі ARIMAX – 0,545. Даний критерій вказує на достовірність прогнозу.

Загалом, обидва методи мають достатньо високу точність моделювання, але під час прогнозування модель ARIMAX виявилася більш точною.

4.8 Виявлення залежності курсу від дописів

Для порівняння візьмемо два графіки. Перший графік (рис 4.14) зображує зміну курсу криптовалют, а другий (рис 4.15) – динаміку відсортованих публікацій за той самий період.

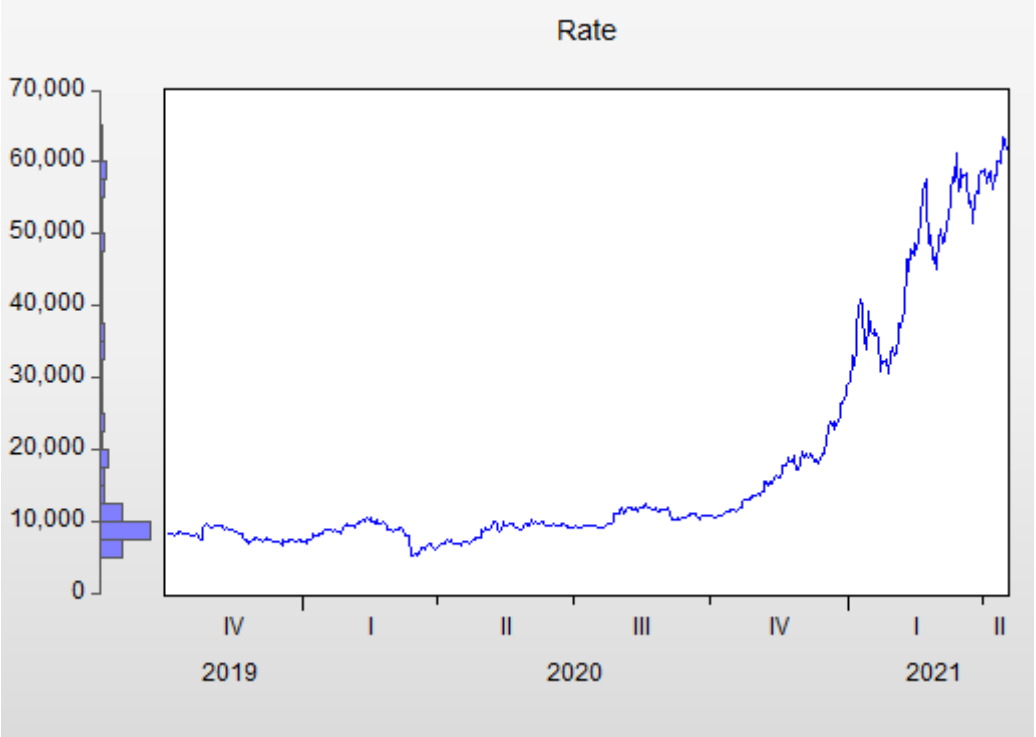


Рисунок 4.14 – Графік зміни курсу криптовалют

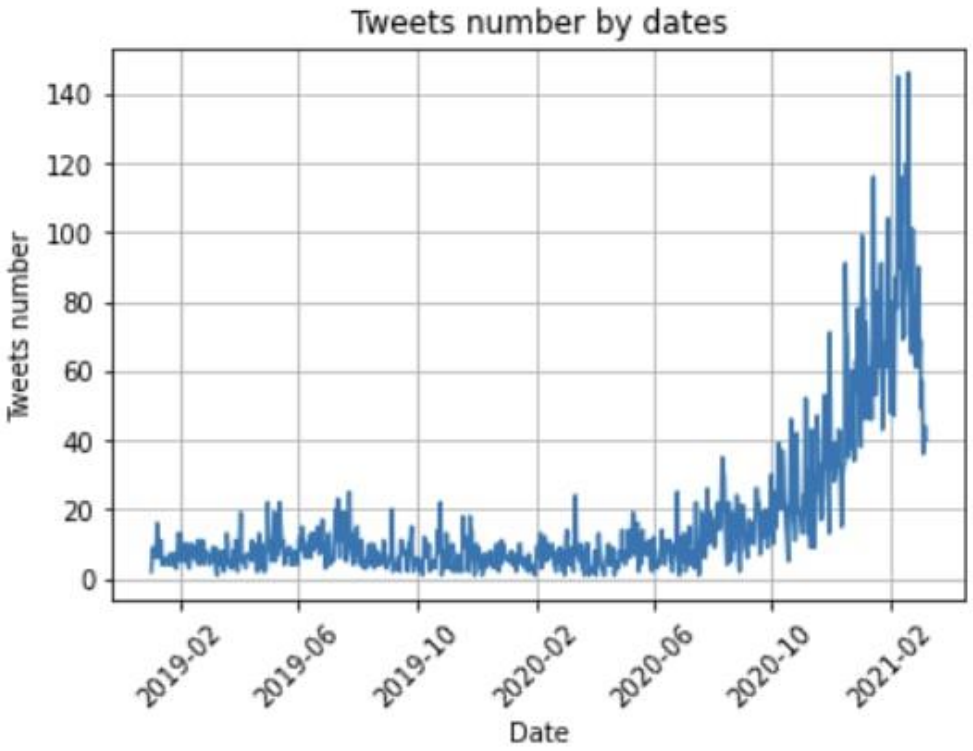


Рисунок 4.15 – Динаміка публікацій у соціальній мережі

Розглянувши графіки появи дописів у Твітері і динаміки курсу криптовалюти Біткоін, можна побачити зв'язок, але не можна стверджувати, що існує абсолютна залежність між дописами у соціальних мережах і зміною курсу криптовалют. На зміну курсу впливає багато факторів, які не враховано у даному дослідженні, а вибірка дописів недостатньо велика. Досліджено, що прогноз на короткий період з фіктивною змінною може бути достатньо точним, але прогнозувати на більш довгий період з високою точністю неможливо, використовуючи дані методи і вибіркові дані.

Також досліджено, що за період різкого підйому курсу (жовтень – листопад 2020 року) більшість дописів мають негативний «настрій».

ВИСНОВКИ

В ході виконання дипломної роботи було досліджено принципи роботи ринку криптовалют, а також, що впливає на зміни курсу різних токенів. Проаналізовано дослідження вчених щодо впливу соціальних мереж на курс криптовалют.

Визначено способи отримання початкових даних, а саме API. Знайдено джерела даних і отримано історію курсу найпопулярніших криптовалют та історію публікацій впливових осіб на ринку криптовалют у соціальній мережі Twitter.

Було розглянуто способи аналізу даних різного виду, математичного і машинного моделювання часових рядів з метою подальшого прогнозування.

Розроблено алгоритм подальшого моделювання та прогнозування даних і проведено прогнозування з подальшим описом результатів. Проведено сортування дописів з Твітеру.

При порівнянні прогнозної точності моделей (ARIMAX та Prophet) було виявлено, що модель ARIMAX у даному випадку має вищу точність.

Розглянувши графіки появи дописів у Твітері і динаміки курсу криптовалюти Біткоін, можна побачити зв'язок, але не можна стверджувати, що існує абсолютна залежність між дописами у соціальних мережах і зміною курсу криптовалют. На зміну курсу впливає багато факторів, які не враховано у даному дослідженні, а вибірка дописів недостатньо велика. Досліджено, що прогноз на короткий період з фіктивною змінною може бути достатньо точним, але прогнозувати на більш довгий період з високою точністю неможливо, використовуючи дані методи і вибіркові дані.

СПИСОК ДЖЕРЕЛ

1. Хажиахметова Е. Ш. Криптовалюта - деньги XXI века // Новая наука: от идеи к результату. — Агентство международных исследований, 2016. — № 11—2. — С. 177—179.
2. Google Trends [Електронний ресурс] / Режим доступу до ресурсу: https://trends.google.com.ua/trends/?geo=UA – Назва з титул. екрану.
3. Доступно о том, как работают криптовалюты, 2017 [Електронний ресурс] / Режим доступу до ресурсу: https://habr.com/ru/company/jincor/blog/407949/ – Назва з титул. екрану.
4. Global Cryptocurrency Charts [Електронний ресурс] / Режим доступу до ресурсу: https://coinmarketcap.com/charts/ – Назва з титул. екрану.
5. Colianni S., Rosales S., Signorotti M. Algorithmic Trading of Cryptocurrency Based on Twitter Sentiment Analysis [Електронний ресурс] / Режим доступу до ресурсу: http://cs229.stanford.edu/proj2015/029_report.pdf – Назва з титул. екрану.
6. Stenqvist E., Lönnö J. Predicting Bitcoin price fluctuation with Twitter sentiment analysis // School of Computer Science and Communication, 2017.
7. Liu Y., Tsyvinski A. Risks and returns of cryptocurrency – Working Paper 24877 // National bureau of economic research, 2018.
8. Горошко М. Ф., Кулішов В. В. Мікроекономіка: Навчальний посібник — К.: Ельга, 2003. ISBN 966-521-177-3
9. Кондратюк А. Фундаментальный и технический анализ криптовалют: связь, сходства и различия, 2018 [Електронний ресурс] / Режим доступу до ресурсу: https://forklog.com/fundamentalnyj-i-tehnicheskij-analiz-kriptovalyut-svyaz-shodstva-i-razlichiya/ – Назва з титул. екрану.
10. Леонт’єва Ю.Ю. Прогнозування: Конспект лекцій (для студентів 4 курсу заочної форми навчання напряму підготовки 6.030504 – «Економіка підприємства» і слухачів другої вищої освіти) / Леонт’єва Ю.Ю. Харк. нац. акад. міськ. госп-ва. – Х.: ХНАМГ, 2008.- 79 с. – с. 48.

11. Ставицький А.В. Теорія часових рядів [Електронний ресурс] / Режим доступу до ресурсу: http://andriystav.cc.ua/Downloads/AppliedEco/02_Time_Series.pdf – Назва з титул. екрану.

12. Кизбикенов, К. О. Прогнозирование и временные ряды [Электронный ресурс] : учебное пособие / К. О. Кизбикенов. – Барнаул : АлтГПУ, 2017. – с. 9-10

13. Процессы «единичного корня». Тесты «единичного корня»: ADF, PP, KPSS. [Електронний ресурс] / Режим доступу до ресурсу: https://bsu.by/upload/page/546923.pdf – Назва з титул. екрану.

14. Эконометрика : учебник / И. И. Елисеева; под ред. И. И. Елисеевой. – М. : Финансы и статистика, 2002. – 344 с.

15. Palachy S. Detecting stationarity in time series data, 2019 [Електронний ресурс] / Режим доступу до ресурсу: https://towardsdatascience.com/detecting-stationarity-in-time-series-data-d29e0a21e638 – Назва з титул. екрану.

16. Минзов А.С. Эконометрика. – М.: Из-во МФА, 2001. С. 54.

17. Кольцов С. Н. Регрессионный анализ [Електронний ресурс] / Режим доступу до ресурсу: https://www.hse.ru/data/2014/08/29/1313619461/лекция%205.pdf – Назва з титул. екрану.

18. Stationarity and differencing [Електронний ресурс] / Режим доступу до ресурсу: https://otexts.com/fpp2/stationarity.html – Назва з титул. екрану.

19. Кольцов С. Н. Методы и модели анализа прогнозирования экономических моделей, 2014 [Електронний ресурс] / Режим доступу до ресурсу: https://linis.hse.ru/data/2014/09/04/1316346389/лекция%205.pdf – Назва з титул. екрану.

20. Skorupa G. Forecasting Time Series with Multiple Seasonalities using TBATS in Python, 2019 [Електронний ресурс] / Режим доступу до ресурсу: https://medium.com/intive-developers/forecasting-time-series-with-multiple-seasonalities-using-tbats-in-python-398a00ac0e8a – Назва з титул. екрану.

21. Компьютерное моделирование, расчет и проектирование наносистем, 2016 [Електронний ресурс] / Режим доступу до ресурсу: https://www.ncfu.ru/export/uploads/imported-from-dle/op/doclinks2017/Metod_KompMod_280302_2017.pdf – Назва з титул. екрану.

22. A 6 Step Field Guide for Building Machine Learning Projects [Електронний ресурс] / Режим доступу до ресурсу: https://towardsdatascience.com/a-6-step-field-guide-for-building-machine-learning-projects-6e4554f6e3a1 – Назва з титул. екрану.

23. Configure automated ML experiments in Python – Supported models [Електронний ресурс] / Режим доступу до ресурсу: https://docs.microsoft.com/uk-ua/azure/machine-learning/how-to-configure-auto-train#supported-models – Назва з титул. екрану.

24. Linear Models – Elastic-Net [Електронний ресурс] / Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/linear_model.html#elastic-net – Назва з титул. екрану.

25. Li J. Boosting algorithm: GBM, 2017 [Електронний ресурс] / Режим доступу до ресурсу: https://towardsdatascience.com/boosting-algorithm-gbm-97737c63daa3 – Назва з титул. екрану.

26. Nearest Neighbors – Nearest Neighbors Regression [Електронний ресурс] / Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-regression – Назва з титул. екрану.

27. Linear Models – LARS Lasso [Електронний ресурс] / Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/linear_model.html#lars-lasso – Назва з титул. екрану.

28. Stochastic Gradient Descent – Regression [Електронний ресурс] / Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/sgd.html#regression – Назва з титул. екрану.

29. Ensemble methods – Random Forests [Електронний ресурс] / Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/ensemble.html#random-forests – Назва з титул. екрану.

30. XGBoost Parameters [Електронний ресурс] / Режим доступу до ресурсу: https://xgboost.readthedocs.io/en/latest/parameter.html – Назва з титул. екрану.

31. Pmdarima API reference – pmdarima.arima.auto_arima [Електронний ресурс] / Режим доступу до ресурсу: https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html#pmdarima.arima.auto_arima – Назва з титул. екрану.

32. Auto-train a time-series forecast model [Електронний ресурс] / Режим доступу до ресурсу: https://docs.microsoft.com/uk-ua/azure/machine-learning/how-to-auto-train-forecast – Назва з титул. екрану.

33. CoinMarketCap – Today's Cryptocurrency Prices by Market Cap [Електронний ресурс] / Режим доступу до ресурсу: https://coinmarketcap.com/ – Назва з титул. екрану.

34. CoinGecko – Cryptocurrency Prices by Market Cap [Електронний ресурс] / Режим доступу до ресурсу: https://coingecko.com/en – Назва з титул. екрану.

35. Лук’яненко І. Г., Жук В. М. Аналіз часових рядів. Частина перша: Побудова ARIMA, ARCH/GARCH моделей з використанням пакета E.Views 6.0. Практичний посібник для роботи в комп’ютерному класі / І. Г. Лук’яненко, В. М. Жук. – К. : НаУКМА ; Аграр Медіа Груп, 2013. – 187 с.

Додаток А

Програмні модулі

А.1 Програмний код отримання історії курсу криптовалют

```
import requests
import json
from prettytable import PrettyTable
from pycoingecko import CoinGeckoAPI
from datetime import datetime

coinsIds = ["bitcoin", "ethereum", "binancecoin", "ripple", "tether",
            "dogecoin", "cardano", "polkadot", "litecoin", "bitcoin-cash"]

cg = CoinGeckoAPI()

dataCurrency = "usd"
dataFrom = "1569952535"
dataTo = "1618670526"

x = PrettyTable()
x.field_names = ["Coin name", "Date", "Price (in USD)"]

data = []

for coinId in coinsIds:
    response = cg.get_coin_market_chart_range_by_id(coinId, dataCurrency, dataFrom, dataTo)

    for price in response['prices']:
        unixTimestamp = str(price[0])
        correctUnixTimestamp = unixTimestamp[:-3]
        ts = int(correctUnixTimestamp)
        resultDate = datetime.utcfromtimestamp(ts).strftime('%Y-%m-%d')
        x.add_row([coinId, resultDate, price[1]])

        coinPriceDetails = {
            'CoinName': coinId,
            'Date': resultDate,
            'Price': price[1]
        }

        data.append(coinPriceDetails)
```

```
with open('coinsData.json', 'w') as outfile:
    json.dump(data, outfile)

print(x)
```

A.2 Програмний код отримання твітів обраних Twitter-аккаунтів

```
from __future__ import print_function

import tweepy
import json
import sys
from credentials_file1 import ACCESS_TOKEN_KEY, ACCESS_TOKEN_SECRET, CONSUMER_KEY, CONSUMER_SECRET

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN_KEY, ACCESS_TOKEN_SECRET)

def get_tweets(api=None, screen_name=None):
    timeline = api.user_timeline(screen_name=screen_name, count=200)
    earliest_tweet = min(timeline, key=lambda x: x.id).id
    print("getting tweets before:", earliest_tweet)

    while True:
        tweets = api.user_timeline(
            screen_name=screen_name, max_id=earliest_tweet, count=200)

        if not tweets:
            print("no tweets anymore, bye")
            break

        new_earliest = min(tweets, key=lambda x: x.id).id

        if new_earliest == earliest_tweet:
            print("new_earliest = earliest_tweet, bye!")
            break
        else:
            earliest_tweet = new_earliest
            print("getting tweets before:", earliest_tweet)
            timeline += tweets

    return timeline


if __name__ == "__main__":
    api = tweepy.API(auth)
```

```
    screen_name = "Beastlyorion"
    print("Getting tweets for user:", screen_name)
    timeline = get_tweets(api=api, screen_name=screen_name)

    with open('tweepy_results/traiders/' + screen_name + '_timeline.json', 'w+') as f:
        for tweet in timeline:
            f.write(json.dumps(tweet._json))
            f.write('\n')
```

А.3 Програмний код обробки всіх твітів та збереження їх до json-файлу

```
import json
from os import walk

import dash
from dash.dependencies import Input, Output
import dash_table
import dash_core_components as dcc
import dash_html_components as html
import pandas as pd

jsonsPath = 'tweepy_results/Liders_of_thoughts'
_, _, filenames = next(walk(jsonsPath))

tweets = []

for filename in filenames:
    for file in open(jsonsPath + '/' + filename, 'r'):
        tweets.append(json.loads(file))

data = []

for tweet in tweets:
    tweetUser = tweet['user']

    isRetweet = "No"
    sourceUserId = "-"
    sourceUserNickname = "-"
    sourceUserAccountName = "-"

    retweetKeyErrors = 0

    try:
        if tweet['text'].startswith("RT "):
            isRetweet = "Yes"
```

```
                sourceUser = tweet['retweeted_status']['user']
                sourceUserId = sourceUser['id_str']
                sourceUserNickname = sourceUser['screen_name']
                sourceUserAccountName = sourceUser['name']
            except KeyError:
                retweetKeyErrors+=1

            hashtags = ""

            for hashtag in tweet['entities']['hashtags']:
                hashtags = hashtags + "\n" + hashtag['text']

            mediaIds = ""
            mediaTypes = ""
            mediaUrls = ""
            mediaTitles = ""
            mediaDescriptions = ""

            mediaKeyErrors = 0

            try:
                for media_file in tweet['extended_entities']['media']:
                    mediaIds = mediaIds + "\n" + media_file['id_str']
                    mediaTypes = mediaTypes + "\n" + media_file['type']
                    mediaUrls = mediaUrls + "\n" + media_file['expanded_url']
                    mediaTitles = mediaTitles + "\n" + media_file['additional_media_info']['title']
                    mediaDescriptions         =         mediaDescriptions         +         "\n"         +
media_file['additional_media_info']['description']
            except KeyError:
                mediaKeyErrors+=1

            taggedPersonsIds = ""
            taggedPersonsNicknames = ""
            taggedPersonsAccountNames = ""

            for mention in tweet['entities']['user_mentions']:
                taggedPersonsIds = taggedPersonsIds + "\n" + mention['id_str']
                taggedPersonsNicknames = taggedPersonsNicknames + "\n" + mention['screen_name']
                taggedPersonsAccountNames = taggedPersonsAccountNames + "\n" + mention['name']

            tweetDetails = {
                'User Id': tweetUser['id_str'],
                'User nickname': tweetUser['screen_name'],
                'User account name': tweetUser['name'],
                'Tweet Id': tweet['id_str'],
                'Tweet text': tweet['text'],
                'Tweet date': tweet['created_at'],
                'Tweet likes': tweet['favorite_count'],
                'Retweetes': tweet['retweet_count'],
                'Tweet language': tweet['lang'],
```

```
            'Is Retweet': isRetweet,
            'Source user Id': sourceUserId,
            'Source user nickname': sourceUserNickname,
            'Source user account name': sourceUserAccountName,
            'Hashtags': hashtags,
            'Media Ids': mediaIds,
            'Media types': mediaTypes,
            'Media URLs': mediaUrls,
            'Media titles': mediaTitles,
            'Media descriptions': mediaDescriptions,
            'Tagged person Ids': taggedPersonsIds,
            'Tagged person nickname': taggedPersonsNicknames,
            'Tagged person account name': taggedPersonsAccountNames
        }

        data.append(tweetDetails)

    with open('data.txt', 'w') as outfile:
        json.dump(data, outfile)
```

A.4 Програмний код фільтрації твітів

```
    import re
    import json
    import nltk
    from os import walk
    from nltk.tokenize import word_tokenize
    from string import punctuation
    from nltk.corpus import stopwords

    # Клас для розбивання тексту твітів на окремі масиви слів
    class PreProcessTweets:
        def __init__(self):
            self._stopwords = set(stopwords.words('english') + list(punctuation) + ['AT_USER','URL'])

        # Функція для обробки списку твітів і занесення масиву слів твіта до його окремого поля "Tweet
words"
        def processTweets(self, list_of_tweets):
            for tweet in list_of_tweets:
                tweet['Tweet words'] = self._processTweet(tweet['Tweet text'])
            return list_of_tweets

        # Функція для розбивання тексту твіта на масив слів, які можна буде далі аналізувати
        def _processTweet(self, tweet):
```

```
                tweet = tweet.lower() # конвертуємо текст до нижнього реєстру
                tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))', 'URL', tweet) # видаляємо посилання із
тексту
                tweet = re.sub('@[^\s]+', 'AT_USER', tweet) # видаляємо нікнейми користувачів Твітера із
тексту
                tweet = re.sub(r'#([^\s]+)', r'\1', tweet) # видаляємо символ # із хештегів
                tweet = word_tokenize(tweet) # видаляємо повторювані символи у словах (конвертуємо
"hellooooooooo" у "hello")
                return [word for word in tweet if word not in self._stopwords]

        preprocessedTweets = []

        # Створюємо об'єкт нашого класу обробки твітів
        tweetProcessor = PreProcessTweets()

        # Відкриваємо файл із нашими твітами у форматі JSON та конвертуємо до масиву твітів
        with open('data.txt') as json_file:
            data = json.load(json_file)

            # Оброблюємо твіти за допомогою нашого класу (розбиваємо текст твіта на масив слів та
            # записуємо його до поля "Tweet words" твіта)
            preprocessedTweets = tweetProcessor.processTweets(data)

        # Записуємо оновлені твіти із масивами слів їх текстів до нового файлу у форматі JSON
        with open('data_with_words.txt', 'w') as outfile:
            json.dump(preprocessedTweets, outfile)

        # Оголошуємо масив слів для фільтрації твітів
        filterWords = ["crypto", "binance", "altcoin", "bitcoin", "btc", "ethereum", "binancecoin"]

        filteredTweets = []

        # Відкриваємо файл із твітами у форматі JSON та конвертуємо до масиву твітів
        with open('data_with_words.txt') as json_file:
            data = json.load(json_file)

            # Визначаємо, чи містить кожен твіт будь-яке слово із масиву filterWords,
            # і якщо так - записуємо до нового масиву відфільтрованих твітів
            for tweet in data:
                filterWordExistsInTweet = False

                for tweetWord in tweet['Tweet words']:
                    for filterWord in filterWords:
                        if filterWord in tweetWord:
                            filterWordExistsInTweet = True
                            break
                    if filterWordExistsInTweet == True:
                        break

                if filterWordExistsInTweet == True:
```

```
            filteredTweets.append(tweet)

    # Сформований масив відфільтрованих твітів записуємо до нового файлу у форматі JSON
    with open('data_filtered_tweets.txt', 'w') as outfile:
        json.dump(filteredTweets, outfile)
```

А.5 Програмний код аналізу твітів на визначення їх настроїв

```
    import nltk
    import json

    def buildVocabulary(preprocessedTrainingData):
        all_words = []

        for (tweet, sentiment) in preprocessedTrainingData:
            all_words.extend(tweet["Tweet words"])

        wordlist = nltk.FreqDist(all_words)
        word_features = wordlist.keys()

        return word_features

    def extract_features(tweet):
        tweet_words = set(tweet["Tweet words"])
        features = {}
        for word in word_features:
            features['contains(%s)' % word] = (word in tweet_words)
        return features

    trainingData = []

    with open('twitter_corpus/training_dataset_with_words.txt') as json_file:
        trainingData = json.load(json_file)

    preprocessedTrainingData = []

    for tweet in trainingData:
        preprocessedTrainingData.append((tweet, tweet["Sentiment label"]))

    word_features = buildVocabulary(preprocessedTrainingData)

    trainingFeatures = nltk.classify.apply_features(extract_features, preprocessedTrainingData,
labeled=True)
```

```
NBayesClassifier = nltk.NaiveBayesClassifier.train(trainingFeatures)

filteredTweetsData = []

with open('data_filtered_tweets.txt') as json_file2:
    filteredTweetsData = json.load(json_file2)

resultTweetsWithSentiments = []

for tweet in filteredTweetsData:
    tweet["Sentiment label"] = NBayesClassifier.classify(extract_features(tweet))
    resultTweetsWithSentiments.append(tweet)

# Сформований масив проаналізованих твітів із визначеними настроями (sentiment) записуємо до нового файлу у форматі JSON
with open('tweets_results.txt', 'w') as outfile:
    json.dump(resultTweetsWithSentiments, outfile)
```

A.6 Програмний код прогнозування за допомогою Prophet

```
!pip install fbprophet

import pandas as pd

bitc_df = pd.read_csv('bitc_df')
bitc_df['Date'] = pd.to_datetime(bitc_df.Date)
bitc_df = bitc_df ['Date']

aggr_bitc_df = bitc_df.groupby('Date').count()
aggr_bitc_df.columns = ['Price']

aggr_bitc_df = aggr_bitc_df.resample('D').apply(sum)

from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
from plotly import graph_objs as go

init_notebook_mode(connected = True)

def plotly_df(df, title = ''):
data = []
```

```
for column in df.columns:
trace = go.Scatter(
x = df.index,
y = df[column],
mode = 'lines',
name = column
)
data.append(trace)

fig = dict(data = data)
iplot(fig, show_link = False)

plotly_df(aggr_bitc_df.resample('W').apply(sum))

from fbprophet import Prophet
predictions = 180

df = aggr_bitc_df.reset_index()
df.columns = ['ds', 'y']

bitc_df = df[:-predictions]

m = Prophet()
m.fit(bitc_df)
future = m.make_future_dataframe(periods = predictions)
forecast = m.predict(future)

m.plot(forecast)

import numpy as np
cmp_df['e'] = cmp_df['y'] - cmp_df['yhat']
cmp_df['p'] = 100*cmp_df['e']/cmp_df['y']
print 'MAPE', np.mean(abs(cmp_df[-predictions:]['p']))
print 'MAE', np.mean(abs(cmp_df[-predictions:]['e']))
```