КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра прикладної статистики

Кваліфікаційна робота бакалавра
на здобуття ступеня бакалавра
за спеціальністю 124 Системний аналіз

на тему:

ВИКОРИСТАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ПОКРАЩЕННЯ ЗАСТОСУВАННЯ СТОХАСТИЧНИХ ПРОЦЕСІВ У АНАЛІЗІ ВИСОКОЧАСТОТНИХ ФІНАНСОВИХ ДАНИХ

Виконав: студент 4-го курсу
Токарчук Данило Костянтинович _____
(підпис)

Науковий керівник:
доцент, доктор фіз.-мат. наук
Розора Ірина Василівна _____
(підпис)

Засвідчую, що в цій роботі немаєзапозичень
з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Роботу розглянуто й допущено до захисту
на засіданні кафедри прикладної статистики
«_05_» ___червня_____ 2023 р.,
протокол № 11
Завідувач кафедри
Розора І.В. _____
(підпис)

Київ – 2023

РЕФЕРАТ

Дипломна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел. Загальний обсяг роботи становить 59 сторінок, основний текст роботи викладено на 57 сторінках. Текст роботи містить 14 ілюстрацій та 3 таблиці. Використано 20 джерел. Додатків до роботи - 2.

Ключові слова: процес Орнштейна – Уленбека, вінерівський процес, часові ряди, прогнозування, стохастичний компонент, моделювання, машинне навчання, градієнтний спуск.

Об'єкт та предмет дослідження: об’єктом дослідження в роботі є процес Орнштейна – Уленбека, предметом дослідження – використання процесу Орнштейна – Уленбека при моделюванні часових рядів високочастотних фінансових даних.

Мета роботи: розробка то програмна реалізація моделі прогнозування динаміки часових рядів високочастотних фінансових даних з використанням процесу Орнштейна – Уленбека.

Методи та інструменти дослідження. Математична модель, що запропонована, є результатом синтезу тезисів, висунутих на підставі аналізу наукової літератури, що присвячена теоретичним основам вивчення ймовірнісних процесів, зокрема процесу Орнштейна – Уленбека. Для перевірки коректності побудованої моделі та можливості її практичного застосування для прогнозування часових рядів високочастотних фінансових даних побудовано комп’ютерну модель машинного навчання з використанням мови програмування Python та бібліотеки машинного навчання Scikit-learn. Тестування моделі проводилось на часовому ряді котирувань акцій сервісу потокового відео Netflix.

Результати та їх новизна, зв’язок з іншими роботами: результатом виконаної роботи є розроблена та сконструйована програмно модель прогнозування часових рядів високочастотних фінансових даних, де

стохастичний компонент моделюється з використанням процесу Орнштейна – Уленбека. Незважаючи на використання цього стохастичного процесу в моделюванні деяких типів часових рядів, у спеціалізованій літературі згідно проведеному аналізу відсутні згадки про використання стохастичного процесу Орнштейна – Уленбека в прогнозуванні часових рядів фінансових даних.

Інформація щодо впровадження: з використанням програмної реалізації моделі прогнозування проаналізовано та спрогнозовано котирування акцій Netflix, оцінені результати прогнозування за встановленими метриками та проведено порівняння результатів з результатами прогнозування поширеною моделлю ARIMA.

Рекомендації щодо використання результатів роботи: код розробленої моделі оцінювача, що базується на патерні оцінювача BaseEstimator пакету Scikit-learn, викладено на GitHub. Розроблена, сконструйована та закодована модель може бути використана аналітиками даних для прогнозування часових рядів як окремо, так й в складі більш складних ансамблевих моделей.

Сферою застосування є проекти в області Data Science, мета яких пов’язана з прогнозування часових рядів.

Робота є значимою через використанням моделі прогнозування, що певною мірою відрізняється від моделей сім’ї ARIMA, що широко використовуються в практичних задачах. Певна новизна підходу та задовільна якість отриманих результатів у порівнянні з результатами використання інших моделей прогнозування часових рядів свідчить на користь доцільності подальшого дослідження та розбудови створеної моделі. В якості напрямків подальшого дослідження можна вказати підвищення точності за рахунок декількох прогонів моделі та крос-валідації результатів, а також більш статистичний аналіз поведінки залишків часових рядів високочастотних фінансових даних.

ЗМІСТ

ВСТУП

Процес Орнштейна-Уленбека (ОУ) вперше було представлено у статті Л.С. Орнштейна та Є.Г. Уленбека 1930 [1] в якості моделі швидкостей частинок в процесі зіткнення з оточуючими їх частинками. Процес Орнштейна-Уленбека цікавий тим, що єдиним (і нетривіальним) стаціонарним гаусівським марківським процесом, що було доведено у роботі [2]. Також процес ОУ має властивість повернення до середнього. Всі ці властивості сприяли поширенню його використання у фінансах та фінансової інженерії.

У класичній роботі Васічека 1977 [3] представлено модель для оцінки миттєвої процентної ставки. Після 2000 року були описані способи використання процесу ОУ в задачах ціноутворення опціонів, оптимізації портфеля та теорії ризиків.

В даний час процес ОУ вивчений досить глибоко і для вчених цікавлять різні модифікації та узагальнення цього процесу.

В роботі буде використано процес Орнштейна-Уленбека для побудови стохастичної моделі прогнозування часового ряда. Це застосування дещо відрізняється від класичних прикладів застосування процесу, що вивчається, у фінансовій математиці, зокрема застосуванні у моделі Васічека, що згадана вище. Попри це, прогнозування часових рядів є однією з найпоширеніших задач фінансової математики, що найчастіше застосовується при створенні прогнозів вартості різноманітних активів: золота, конвенційних та криптовалют, акцій компаній, тощо.

Буде розглянуто побудову авторегресійної моделі, стохастичний компонент якої описується процесом Орнштейна-Уленбека. Практичну реалізацію моделі буде виконано з використанням мови програмування Python та патерну проектування оцінювача, що рекомендований авторами поширеної бібліотеки машинного навчання Scikit-learn. Тестування та оцінку якості моделі проведемо загальноприйнятим способом, застосувавши побудований

алгоритм до декількох різних часовим рядів й порівнявши фактичні та прогнозовані дані. На додаток порівняємо отримані з використанням побудованої моделі результати з результатами використання моделі ARIMA в її реалізації бібліотекою pmdarima. Вибір бібліотеки продиктований її частим використанням у дослідженнях та комерційній розробці.

1 ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ МОДЕЛІ

1.1 Класичний процес Орнштейна – Уленбека

Спочатку дамо визначення деяких ключових понять для розгляду класичного процесу Орнштейна – Уленбека, потім дамо власне його визначення.

Припустимо, що дано імовірнісний простір $(\Omega, \mathcal{F}, P)$ та деяку множину індексів T. Будемо називати відображення $\xi: \Omega \times T \to \mathbb{R}$ випадковим процесом (або стохастичним процесом), якщо при кожному фіксованому $t \in T$ відображення $\xi(\omega, t)$ є випадковою величиною (тобто вимірним відображенням Ω в$\mathbb{R}$). Іноді замість $\xi(t, \omega)$ пишуть $\xi_t(\omega)$ або просто ξ_t.

Множина T може мати різну природу, але, як правило, T буде являти собою відрізок $[0, T_0]$ та інтерпретуватися як час. Якщо $T = \mathbb{N}$, ми отримуємо послідовність випадкових величин.

При фіксуванні $\omega \in \Omega$ отримують траєкторію випадкового процесу. Якщо $T = [0, T_0]$, то траєкторія є функцією на відрізку $[0, T_0]$. Часто дуже плідною є така точка зору: точка $\omega \in \Omega$ відображається у функцію $t \to \xi_t(\omega)$, тобто $\xi_t(\omega)$ є випадковою функцією. При суворому підході треба уточнювати, який простір функцій використовується та якою σ −алгеброю він наділений.

Два випадкові процеси ξ_t, η_t називаються еквівалентними (стохастично еквівалентними), якщо $P(\xi_t \neq \eta_t) = 0$ для всіх $t \in T$, траєкторії еквівалентних випадкових процесів можуть суттєво відрізнятися.

Дамо визначення вінерівського процесу (броунівського руху), що є основною неперервною моделлю теорії випадкових процесів.

Випадковий процес $W_t, t \in [0, T]$ називається вінеровським процесом (броунівським рухом), якщо він має наступні властивості:

1) Випадковий вектор $(W_{t_1}, \dots, W_{t_n})$, $0 \le t_1 < t_2 < \dots < t_n < T$ має гауссівський розподіл і $W_0 = 0$

2) $\mathbb{E}W_t = 0$, $\mathbb{E}(W_t, W_s) = t \wedge s$

3) Траєкторії $t \to W_t(\omega)$ безперервна для всіх $\omega \in \Omega$.

Вінерівський процес є гауссівським процесом (тобто, процесом з гауссівськими кінцевими розподілами) з безперервними траєкторіями.

Вінерівський процес є з незалежними приростами, тобто випадкові величини

$$W_{t_1} - W_{t_0}, W_{t_2} - W_{t_1}, \dots, W_{t_n} - W_{t_{n-1}}, \quad 0 \le t_1 < t_2 < \dots < t_n < T \quad (1.1)$$

незалежні. Крім цього, витримана властивість

$$\mathbb{E}W_t = 0, \quad D(W_t - W_s) = t - s, \quad s \le t \quad (1.2)$$

Дійсним є зворотнє твердження, гаусівський процес з безперервними траєкторіями та незалежними приростами є винеровским, якщо виконано властивість (1.2) та $W_0 = 0$.

Довгий час Ейнштейн вважався піонером у фізико-математичній теорії броунівського руху, але приблизно через 50 років була перевідкрита робота Л. Башельє ”Th'eorie de la sp'eculation”, написана в 1900 р. У цій роботі Башельє фактично застосував вінерівський процес до опису цінних паперів на французькому ринку Його робота залишилася непоміченою до кінця 50-х років.

Перший суворий математичний доказ існування вінеровського процесу отримав Норберт Вінер у 1922-23 рр. Його конструкція ґрунтувалася на досить абстрактних методах (інтеграл Даніеля) та була дуже важкою. Спрощення було досягнуто пізніше. Стандартний доказ ґрунтується на теоремі Колмогорова про побудову заходу щодо кінцевих розподілів та його ж теоремі про існування безперервної модифікації процесу. Поль Леві запропонував доказ, заснований на збіжності випадкових рядів (фактично на розкладі процесу у ряд Фур'є з випадковими коефіцієнтами.

Вінерівський процес як модель руху зважених частинок у рідині не був цілком задовільним із фізичної точки зору. У 30-х роках XX століття було запропоновано іншу модель, яка враховувала ньютонівську взаємодію частинок. Поведінка частки описувалося рівнянням Ланжевена [4]

$$dv(t) = -\beta v(t)dt + dW_t \quad (1.3)$$

де $\beta > 0$ - деякий коефіцієнт, v - швидкість частинки. Формально це рівняння можна переписати у вигляді 1.4

$$ma = m\frac{dv(t)}{dt} = -m\beta v(t) + m\frac{W_t}{dt} \quad (1.4)$$

У математиці та фізиці розв'язок рівняння (1.4) отримало назву процесу Орнштейна-Уленбека.

Процесом Орнштейна-Уленбека називається рішення стохастичного диференціального рівняння

$$d\xi_t = -\xi_t dt + \sqrt{2}dW_t, \ \xi_0 = x \quad (1.5)$$

Для розв'язання рівняння (1.5) знаходять диференціал процесу $e^t\xi_t$ [5]

$$d(e^t\xi_t) = e^t\xi_t dt + e^t\left(-\xi_t dt + \sqrt{2}dW_t\right) = \sqrt{2}e^t dW_t \quad (1.6)$$

Отже

$$e^t\xi_t = x + \sqrt{2}\int_0^t e^s\, dW_s \quad (1.7)$$

й відповідно

$$\xi_t = xe^{-t} + \sqrt{2}e^{-t}\int_0^t e^s\, dW_s \quad (1.8)$$

З цього співвідношення видно, що ξ_t – процес Гауса [6]. Отже, його кінцево вимірні розподіли повністю визначаються середнім $\mathbb{E}\xi_t$ та функцією коваріації

$$K(s,t) = \mathbb{E}(\xi_t - \mathbb{E}\xi_t)(\xi_s - \mathbb{E}\xi_s) \quad (1.9)$$

звідки

$$\mathbb{E}\xi_t = xe^{-t} \quad (1.10)$$

та використовуючи ізомертрію стохастичного інтеграла [7]

$$\mathbb{E}(\xi_t - \mathbb{E}\xi_t)(\xi_s - \mathbb{E}\xi_s) = 2e^{-t-s}\,\mathbb{E}\left(\int_0^t e^u\,dW_u \int_0^s e^u\,dW_u\right)$$

$$= 2e^{-t-s}\int_0^{\min(t,s)} e^u\,dW_u = e^{-t-s}\left(e^{2\min(t,s)} - 1\right) \quad (1.11)$$

Напівгрупа Орнштейна-Уленбека $T_t f$ визначається наступним чином [8]

$$T_t f(x) = \mathbb{E} f(\xi_t) \qquad (1.12)$$

Знаючи розподіл ξ_t, отримують явне представлення T_t

$$T_t f(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} f\left(xe^{-t} + \sqrt{1-e^{-2t}}y\right) e^{-\frac{y^2}{2}}dy \qquad (1.13)$$

Напівгрупа Орнштейна-Уленбека $T_t f$ має наступні властивості

1.) (T_t — напівгрупа)

$$T_{t+s} f = T_t(T_s f) \qquad (1.14)$$

2. (Генератор T_t) $u(t,x) = \mathrm{T_t} f$ є розв'язанням параболічного рівняння у часткових похідних

$$u_t = u_{xx} - xu_x, \qquad u(0,x) = f(x) \qquad (1.15)$$

3. (Інваріантна міра) стандартний гаусівський розподіл $\gamma = \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy$ є інваріантною мірою відносно T_t

$$\int T_t f \cdot g d\gamma = \int f \cdot T_t g d\gamma \qquad (1.16)$$

4. (Ергодична властивість)

$$\lim_{t\to\infty} T_t f = \int f d\gamma \qquad (1.17)$$

Наприкінці узагальнимо поняття стохастичного диференціального рівняння. Стохастичним диференціальним рівнянням називається рівняння вигляду [9]

$$d\xi_t = \sigma(\xi_t)dW_t + \beta(\xi_t)dt, \qquad \xi_{t_0} = \eta \qquad (1.18)$$

де η – деяка випадкова величина, вимірювана відносно $\mathcal{F}_{t_0}$. Це рівняння слід розуміти як інтегральне

$$\xi_t = \eta + \int_{t_0}^{t} \sigma(\xi_s)\, dW_s + \int_{t_0}^{t} \beta(\xi_s) ds \qquad (1.19)$$

Функції σ і β носять назви коефіцієнт дифузії та коефіцієнт зносу.

1.2 Прикладне застосування у фінансовій математиці

Моделі відсоткових ставок - один з розділів сучасних фінансів, що найбільш інтенсивно розвивається. Насамперед це пов'язано з бурхливим зростанням у останні два десятиліття ринків відсоткових інструментів, насамперед - ринків всіляких різновидів похідних, що породжує потреба у розробці методів їхньої оцінки та використання у стратегіях хеджування.

Хеджування - страхування ризиків на фінансових ринках. Інвестор вдається до них, коли побоюється, що ціна купленого чи проданого активу може змінитись у небажаний для нього бік.

Джерелом процентного ризику є невизначеність, неможливість точного прогнозування майбутніх процентних ставок. Однак, не знаючи - якими точно будуть процентні ставки через місяць або через рік, можна визначити, що коливання відсоткових ставок схильні до певних закономірностей. Ці знання включають існуючі особливості організації та функціонування ринку, а також дані про минулий (історичний) його розвиток. Як мінімум, відомо, що різні ринкові відсоткові ставки (і тим самим - ціни існуючих інструментів) більш менш тісно взаємопов'язані між собою. Наявність взаємозв'язку між випадковими величинами означає, що вони перебувають під впливом загальних чинників. Вибір факторів є, як правило, першим і одним із найважливіших кроків у побудові моделі тимчасової структури. Він може бути зроблений досить довільно, коли переслідується мета простоти та прозорості моделі може ґрунтуватися на економетричному аналізі ринкових даних або на деякій економічній моделі рівноваги. Далі необхідно визначити - відповідно до яких закономірностей змінюються обрані фактори (іншими словами –

якими випадковими процесами описується їх динаміка) і як вони пов'язані з реальними ринковими цінами (відсотковими ставками). При відповіді на останнє питання зазвичай припускають, що існуючі ринкові ціни допускають можливості арбітражу.

Випадкові процеси, що описують динаміку факторів, завжди залежать від певного набору параметрів, тому необхідно відкалібрувати модель - знайти такі значення параметрів, які більш точно відповідали б ринковим цінам, що спостерігаються. На етапі статистичного аналізу моделі важливо переконатися, що модель є достатньою ступеня адекватної, тобто. задовільно описує співвідношення між реальними ринковими показниками та його динаміку. Важливою (але не завжди обов'язковою) властивістю є стійкість моделі – коли вона продовжує задовільно описувати реальність навіть за відносно суттєвої зміни ринкових умов.

Наявність моделі (у тому числі – статистичних оцінок її параметрів) дозволяє вирішувати низку найважливіших завдань:

1. Модель дає можливість оцінювати фінансові інструменти, грошові потоки за якими залежать від вибраних випадкових факторів. Оцінка у разі означає визначення такого значення ціни, що виключає арбітраж. Маючи інформацію про ліквідні інструменти, на основі моделі можна оцінювати інструменти, які менш ліквідні чи взагалі не звертаються на ринку.
2. Модель необхідна для розробки стратегій хеджування ризиків, - вибору портфелів фінансових інструментів, вартість яких не схильна (точніше - меншою мірою схильна) до впливу непередбачуваних коливань випадкових чинників Якщо деяку стратегію хеджування розроблено без явного застосування моделі - це просто означає, що вона неявно ґрунтується на деякій (можливо, не цілком адекватній) моделі.
3. Модель може використовуватись як інструмент контролю ризику - наприклад, для прогнозування наслідків ринкових для фінансового інституту загалом чи окремих позицій.

Спосіб побудови моделі структури процентних ставок у часі може бути різним. У моделях рівноваги відправною точкою є переваги економічних агентів, і прийняті на основі цих переваг рішення. Метою моделювання є визначення абсолютних цін у стані рівноваги – коли всі агенти вибрали оптимальні собі рішення. Результат залежить від вибраних припущень, які можуть бути дуже спрощують дійсність. Найбільш відомою моделлю рівноваги у сфері моделювання динаміки відсоткових ставок є модель Кокса-Інгерсолла-Росса (1985). Однак ряд інших моделей, таких як модель Мертона або модель Васічека, які не базуються явно на умовах рівноваги економічної системи, також можуть бути віднесені до цього класу, оскільки в них постулюються закономірності, відповідно до яких поводяться випадкові чинники.

У відповідності з Даффі і Каном (1996), будь-яке явне припущення про закономірності динаміки приоцентних ставок, означає, що в його основі явно чи неявно лежить модель рівноваги.

Арбітражні моделі (прикладами є моделі Хо-Лі, Халла-Уайта, та багато інших) значно меншою мірою покладаються на припущення про переваги інвесторів: як правило, досить вважати, що економічні агенти віддають перевагу більшому доходу меншому.

Існуючі ринкові ціни активів, що торгуються на ринку, сприймаються як даність, а ціни інших (наприклад, похідних) інструментів визначаються по відношенню до відомих ринкових цін так, щоб можливість отримання арбітражного прибутку була відсутня.

І в моделях рівноваги, і в арбітражних моделях відправною точкою є припущення про випадкові фактори, які впливають на ринкові ціни та закономірності їх динаміки.

Найпростіше припущення, яке може бути зроблено щодо структури відсоткових ставок у часі та цін інструментів з фіксованим доходом – залежність їх від єдиного випадкового фактору. Формули цін простих

дисконтних облігацій визначаються на підставі припущень про ймовірнісні властивості даного випадкового фактору.

Як єдиний чинник найчастіше виступає короткострокова спот-ставка - дохідність інструментів з терміном погашення один період - у разі дискретного часу, або миттєва ставка - для безперервного часу. Загалом миттєва ставка є скоріше теоретичною абстракцією, оскільки на реальному ринку важко знайти відсоткову ставку, яка цілком відповідала б цьому поняттю. Найбільш підходящими на роль миттєвої ставки є досить довгі ставки (наприклад, тижневі чи місячні, на відміну від ставок овернайт).

Навіть якщо вважати, що випадковим фактором є не короткострокова ставка, а будь-який інший параметр, модель завжди може бути перетворена таким чином, що єдиним випадковим фактором є саме короткострокова ставка. Припущення про єдиний випадковий фактор, що визначає коливання всіх цін на ринку є надзвичайно привабливим внаслідок своєї простоти: зокрема, воно дає можливість розробки щодо простих методів оцінки фінансових інструментів.

Маючи припущення про випадковий процес, що описує динаміку короткострокової ставки, відносні ціни простих дисконтних облігацій, як і ціни інших, більш складних інструментів, визначаються з принципу відсутності можливості арбітражу.

Одні з перших моделей тимчасової структури відсоткових ставок було запропоновано Робертом Мертоном (1970, 1973), та Олдрічем Васічеком (1977. Невизначеність, відповідно до даного підходу, що панує теоретично тимчасової структури відсоткові ставки, моделюється з допомогою вінеровського процесу (процесу броунівського руху).

У безперервному часі поведінка випадкового чинника (короткострокової ставки x) описується процесом Ito (стохастичним) диференціальним рівнянням)

$$dx = \mu_x(t,x)dt + \sigma_x(t,x)d\omega \qquad (1.20)$$

де x – миттєва спот-ставка тобто. доходність простої дисконтної облігації з нескінченно малим терміном погашення (позначення використане виключно для спрощення запису, насправді x – випадковий процес, тобто. випадкова змінна, що залежить від часу t і стану природи);

dt - нескінченно малий інтервал часу$

dx - приріст (зміна) величини x за час dt

$\mu_x(t,x)$ - очікуване значення приросту миттєвої ставки, наведене до річного вимірювання;

$\sigma_x(t,x)$- стандартне відхилення (також у річному вимірі) приросту миттєвої ставки, яке у фінансах прийнято називати волатильністю (мінливістю);

ω - стандартний вінерівський процес.

Вінерівський процес (не в останню чергу, внаслідок своєї простоти) є найважливішим засобом моделювання невизначеності в сучасній фінансовій математиці. Визначення стандартного вінерівського процесу було дано в підрозділі 1.1.

Перший доданок у виразі (1.20) моделює тенденцію зміни x , другий доданок - випадкові коливання. Як тенденція $\mu_x(t,x)$ так й мінливість (волатильність) $\sigma_x(t,x)$ у загальному випадку можуть бути функціями часу і значення змінної x.

Нехай t – деякий момент часу. Вважаючи, що ціна простої дисконтної облігації з терміном погашення $\tau, p(\tau)$, є функцією часу t і короткострокової процентної ставки x (тобто $p(\tau) \equiv p(t,T,x), \tau = T - t,$ у відповідності до відомої леми Іто отримаємо вираз для приросту ціни за нескінченно малий проміжок часу [11, 12]

$$\mathrm{d}p(\tau) = \mu_p(t,x)dt + \sigma_p(t,x)d\omega \qquad (1.21)$$

де

$$\mu_p = \frac{\partial p}{\partial t} + \mu_x(t,x)\frac{\partial p}{\partial x} + \frac{1}{2}\sigma_x^2(t,x)\frac{\partial^2 p}{\partial x^2} \qquad (1.22)$$

$$\sigma_p = \sigma_x(x,t)\frac{\partial p}{\partial x} \qquad (1.23)$$

Ціни простих дисконтних облігацій на конкурентному ринку мають бути такими, щоб не було можливості арбітражу. Виберемо портфель, що складається з облігацій двох видів (два різні терміни погашення - t_1 та t_2) таким чином, щоб його прибутковість за нескінченно малий проміжок часу була детермінованою величиною. Двох видів облігацій для цієї мети достатньо, оскільки існує лише один фактор ризику – миттєва спот-ставка. Вартість портфеля, що складається з однієї простої дисконтної облігації з терміном погашення t_1 і облігацій η терміном t_2 дорівнює [11, 12]

$$V = p(t_1) + \eta p(t_2) \qquad (1.24)$$

Для спрощення запису $p(t_1)$ будемо позначати як p_1, $p(t_2)$ – як p_2. Приріст вартості портфеля за нескінченно малий проміжок часу дорівнюватиме

$$dV = dp_1 + \eta dp_2 = (\mu_1 + \eta\mu_2)dt + (\sigma_1 + \eta\sigma_2)d\omega \qquad (1.25)$$

де

$$\mu_i = \frac{\partial p_i}{\partial t} + \mu_x(t,x)\frac{\partial p_i}{\partial x} + \frac{1}{2}\sigma_x^2(t,x)\frac{\partial^2 p_i}{\partial x^2}, \qquad \sigma_i = \sigma_x\frac{\partial p_i}{\partial x}, \qquad i = 1,2 \qquad (1.26)$$

Для того, щоб дохідність портфеля за час dt була детермінованою, необхідно, щоб у виразі (1.25) коефіцієнт при випадковій величині $d\omega$ дорівнював нулю, тобто

$$\frac{\partial p_1}{\partial x} + \eta\frac{\partial p_2}{\partial x} = 0 \qquad (1.27)$$

тим самим, коефіцієнт хеджування (у даному випадку - кількість облігацій терміном погашення t_2) має дорівнювати

$$\eta = -\left.\frac{\partial p_1}{\partial x}\right/\frac{\partial p_2}{\partial x} \qquad (1.28)$$

Прибутковість безризикового портфеля (приріст вартості у відсотках) час dt, якщо арбітраж неможливий, повинна дорівнювати безризиковій миттєвій ставці

$$\frac{dV}{V} = xdt \qquad (1.29)$$

або, враховуючи вміст портфеля

$$dV = x(p_1 + \eta p_2)dt \qquad (1.30)$$

Прирівнюючи вирази (1.25) та (1.30), з урахуванням того, що величина η обрана так, щоб був відсутній ризик (тобто відповідно до (1.28)), після перетворень отримаємо

$$(\mu_1 - xp_1)\Big/\frac{\partial p_1}{\partial x} = (\mu_2 - xp_2)\Big/\frac{\partial p_2}{\partial x} \qquad (1.31)$$

Розділивши останній вираз на σ_x, отримаємо, що для всіх простих дисконтних облігацій, якщо відсутня можливість арбітражу, маєвиконуватися умова

$$\frac{\mu_p - xp}{\sigma_p} = \lambda \qquad (1.32)$$

Тут λ- величина, що називається ринковою премією за ризик. Справді, ліва частина останнього рівняння є так званий коефіцієнт Шарпа для простої дисконтної облігації: чисельник - це різниця між очікуваним доходом по облігації за нескінченно малий проміжок часу і безризиковим доходом, що отримується за цей же час від інвестування суми p (ціна облігації), знаменник – волатильність облігації (всі величини у річному вимірі). У загальному випадку λ може залежати від часу та рівня короткострокової ставки $\lambda \equiv \lambda(t)$, але не залежить від характеристик конкретної облігації (зокрема терміну погашення).

Якщо λ дорівнює нулеві, це означає, що всі прості дисконтні облігації, незалежно від терміну погашення, приносять інвесторам однакову доходність, що дорівнює короткостроковій ставці $\mu_p = xp$, тобто, інвестори не отримують

премії за ризик, інвестуючи у відносно більш довгострокові інструменти. Це можливо тільки якщо інвестори мають властивість нейтральності до ризику (нейтральною до ризику називають людину для якої гарантований дохід C і випадковий дохід $\tilde{C}$, очікувана величина якого дорівнює С, є досконалими замінниками). Насправді це не так. Набагато правдоподібніше вважати, що більшість інвесторів до ризику не схильні, тобто вибираючи між відомою сумою грошей C і випадковою $\tilde{C}$, такою, що $\mathbb{E}[\tilde{C}] = C$ завжди віддають перевагу гарантованому отриманнюС. Це означає, що ризикові вкладення можуть зацікавити інвестора, тільки якщо вони приносять більший середній дохід у порівнянні з безризиковими. У розглядуваному випадку, за малий проміжок часу інвестор може отримати гарантовано x відсотків річних, або може купити облігацію з більш віддаленим терміном погашення - дохід у ній за мінімальний інтервал часу випадковий, оскільки залежить від майбутніх значень відсоткової ставки. Середня (очікувана) прибутковість (у відсотках річних) по облігації дорівнює μ_p/p і дана облігація може бути привабливою для несхильного до ризику інвестора тільки якщо $\frac{\mu_p}{p} - x > 0$), що означає $\lambda < 0$ (оскільки $\sigma_p = \frac{\partial p}{\partial x}\sigma_x < 0$ внаслідок зворотної залежності цін відсоткових ставок).

Величина λ показує скільки одиниць додаткового доходу інвестор може отримати з розрахунку на одну додаткову одиницю ризику (волатильності), над ринком вона має бути однакова всім інструментів, оскільки інакше можливий так званий міжчасовий арбітраж - за окремими облігаціями на одиницю ризику можна буде отримати більший дохід, ніж за іншими, що неминуче призведе до коригування цін, поки що для всіх облігацій не виконуватиметься (1.32).

Переписавши рівняння (1.32), підставляючи вирази для μ_p та σ_p (відповідно до (1.22), (1.23)) отримаємо наступне диференціальне рівняння у часткових похідних [11,12]

$$\frac{\partial p}{\partial t} + (\mu_x - \lambda\sigma_x)\frac{\partial p}{\partial x} + \frac{1}{2}\sigma_x^2\frac{\partial^2 p}{\partial x^2} - xp = 0 \qquad (1.33)$$

Знайти вираз для ціни простої дисконтної облігації, можна розв'язавши дане рівняння (для певного виду функцій $\mu_x(t,x)$) та $\sigma_x(t,x)$. Граничною умовою під час вирішення рівняння (7.12) є рівність ціни облігації в останній момент погашення одиниці, тобто, якщо T - момент погашення облігації, то в момент $T = t$ виконується $p(T,T) = 1$.

Моделлю Васічека називають модель тимчасової структури процентних ставок, у якій динаміка короткострокової ставки має тенденцію до рівноважного значення, тобто $\mu_x(t,x) = \alpha(\mu - x)$, де μ - довгострокове рівноважне значення короткострокової ставки (константа), α - параметр, що визначає швидкість поворотної тенденції (швидкість наближення x до рівноважного значення μ). Стандартне відхилення приросту процентної ставки в моделі Васічека є константою, яка не залежить від часу і значення x: $\sigma_x(t,x) \equiv \sigma$. Випадковий процес, що описує динаміку короткострокової ставки в моделі Васічека є процесом Орнштейна-Уленбека

$$dx = \alpha(\mu - x)dt + \sigma d\omega \qquad (1.34)$$

Диференційне рівняння у часткових похідних (1.33) запишеться як

$$\frac{\partial p}{\partial t} + [\alpha(\mu - x) - \lambda\sigma]\frac{\partial p}{\partial x} + \frac{1}{2}\sigma\frac{\partial^2 p}{\partial x^2} - xp = 0 \qquad (1.35)$$

причому ринкова премія за ризик λ у моделі Васічека також є константою. Рішенням даного рівняння (з граничною умовою $p(T,T) = 1$) є вираз для ціни простої дисконтної облігації, що погашається в момент Т (через $\tau = T - t$ років).

$$p(t,T) \equiv p(\tau) = e^{\chi(\tau)\tau} \qquad (1.36)$$

де $\chi(\tau) \equiv \chi(t,T,x)$ - прибутковість простої дисконтної облігації (спот-ставка) визначається як

$$\chi(\tau) = x\frac{\phi(\tau)}{\tau} + \left(\mu - \frac{\lambda\sigma}{\alpha} - \frac{\sigma^2}{2\alpha^2}\right)\frac{\tau - \phi(\tau)}{\tau} + \frac{(\sigma\phi(\tau))^2}{4\alpha\tau} \qquad (1.37)$$

де $\phi(\tau) = {(1 - e^{\alpha\tau})}/{\alpha}$. Позначимо через x_∞ значення $\chi(\tau)$ при $\tau \to \infty$ граничне значення довгострокової ставки

$$x_\infty = \lim_{\tau\to\infty} \chi(\tau) = \mu - \frac{\lambda\sigma}{\alpha} - \frac{\sigma^2}{2\alpha^2} \qquad (1.38)$$

тоді вираз для ставки спот моделі Васічека може бути представлений у вигляді

$$\chi(\tau) = x_\infty + (x - x_\infty)\frac{\phi(\tau)}{\tau} + \frac{(\sigma\phi(\tau))^2}{4\alpha\tau} \qquad (1.39)$$

Визначивши функцію

$$\psi(\tau) = \left(\mu - \frac{\lambda\sigma}{\alpha} - \frac{\sigma^2}{2\alpha^2}\right)\big(\tau - \phi(\tau)\big) + \frac{(\sigma\phi(\tau))^2}{4\alpha\tau} \qquad (1.40)$$

вираз (1.37) ціни для простої дисконтної облігації може бути записаний у вигляді

$$p(\tau) = \exp\big(-\phi(\tau)x - \psi(\tau)\big) \qquad (1.41)$$

1.3 Часові ряди в економіці

Типовими для економічної та фінансової практики даними є часові дані, тобто випадки, коли значення економічної змінної (або змінних у багатовимірному випадку), спостерігаються в часовому інтервалі із заданою частотою записів (кожного торгового дня, у моменти транзакцій, щомісяця, тощо). Під періодичністю запису розуміють або довжину інтервалів між конкретними спостереженнями (наприклад, календарні місяці), або регулярність спостережень (наприклад, кожен торговий день). Щодо регулярності, фінансові дані часто спостерігаються нерегулярно (дані з нерегулярним інтервалом), наприклад, ціни на акції на біржі котируються зазвичай у моменти транзакцій від відкриття до закриття торговельного дня, частота угод зазвичай нижча після відкриття, в обідню пору і в другій половині дня перед закриттям (можливий підхід у такій ситуації привласнює ціну

закриття або переважну ціну в цей день). Важливою властивістю даних часу є те, що вони впорядковані хронологічно в часі.

Термін тимчасові ряди позначає будь-яку послідовність даних $y_1, \ldots, y_n$, упорядкованих у хронологічному порядку в часі. Це могло б виправдати спрощений погляд на часові ряди як на набір чисел упорядкованих у часі (історично так було, наприклад, для астрономічних спостережень). Однак дуже важливим аспектом часових рядів є не лише їхня динаміка, а й також їхній імовірнісний характер. Щоб бути адекватним, аналіз часових рядів має застосовувати такі моделі, що ґрунтуються на стохастичних засадах і здатні генерувати тимчасові послідовності, подібні зі стохастичного погляду на залежність, що спостерігається. Такі моделі позначаються як випадкові процеси і можуть розглядатися як певні алгоритми, засновані на випадкових числах.

У літературі прийнято, що термін тимчасові ряди іноді інтерпретується як траєкторія, а іноді як сам випадковий процес. Правильний зміст слід визначати з контексту.

Загальна мета аналізу часових рядів, включно з додатками в економіці та фінансах є побудова адекватної моделі, що лежить в основі випадкового процесу.

Така модель зазвичай дає змогу:

- зрозуміти механізм (або алгоритм), який генерує спостережуваний часовий ряд;
- перевірити гіпотези апріорних очікувань і припущень у статистично надійний спосіб (наприклад, чи показують фондові ринки довгострокове зростання);
- передбачити (прогнозувати, екстраполювати) майбутній розвиток системи (наприклад, яких відсоткових ставок або курсів валют можна очікувати наступного місяця із заданою впевненістю);

– контролювати й оптимізувати динаміку системи, включно з налаштуванням параметрів, початковими умовами, саморегулюванням тощо. (наприклад, налаштування параметрів пенсійної реформи).

У будь-якому разі дані у вигляді часових рядів мають безліч специфічних особливостей. Вони допомагають аналізувати такі набори даних, але, з іншого боку, це може викликати ускладнення, які мають бути подолані відповідними процедурами і регулюваннями.

Нижче дамо короткий перелік основних складнощів та їхніх груп, пов'язаних із прогнозуванням часових рядів.

Проблеми, пов'язані з вибором точок часу спостереження.

Тимчасові ряди в дискретному часі, що переважають в економіці та фінансах, зазвичай виникають у результаті такими способами:

– вони дискретні за своєю природою (наприклад, денні міжбанківські ставки LI-BOR);
– тимчасові ряди дискретизуються в безперервний час (наприклад, котирування акцій біржі, що закриваються, призначені на кінець певних торгових днів у контексті безперервної торгівлі);
– накопичуються (агрегуються) значення за задані проміжки часу (наприклад, накопичені суми страхових виплат, виплачених за певні квартали); часто замість агрегатів отримують середні;

При цьому найчастіше під час формування часових рядів спостерігач не може обирати час, у якій проводиться вимірювання, тобто час, у який фіксується спостереження.

Таким чином, в економіці, а ще частіше у фінансах спостереження в часових рядах не є рівновіддаленими одне від одного в часі.

Проблеми, пов'язані з календарем.

Природа, безумовно, відповідальна за деякі проблеми, пов'язані з календарем. Наприклад, кількість днів в одному сонячному році не ціла, для різних географічних зон потрібні часові зрушення. Однак більша частина проблем із календарем виникає через людські умовності, наприклад:

- різна тривалість календарних місяців;
- чотири або п'ять вихідних на місяць;
- різна кількість робочих або торгових днів на місяць;
- перенесені свята (наприклад, Великдень один раз у першому кварталі і наступного разу в другому);
- зимовий і літній періоди;

Усе це відбивається на динаміці явищ, представлених часовими рядами в економіці та фінансах, і відповідно, має враховуватися в моделях. Так, наприклад, більшість моделей прогнозування часових рядів мають можливість урахування се-зонності.

Проблеми довжини часового ряду.

Довжина часового ряду - це кількість n спостережень даного часового ряду (не часовий діапазон між початком і кінцем часового ряду). Отже, наприклад, щомісячний часовий ряд за 10 років має довжину 120. Логічно, що обсяг інформації, доступної для аналізу, збільшується зі збільшенням довжини часового ряду.

Однак довжина часового ряду не є унікальною мірою інформації, що міститься в ньому. Наприклад, подвоєння довжини часового ряду за рахунок зменшення вдвічі вихідного інтервалу між сусідніми точками спостереження зазвичай не означає подвоєння інформації про цей часовий ряд. Необхідно враховувати також внутрішню структуру даного часового ряду.

Що стосується довжини часового ряду, зазвичай розумним компромісом є мінімальна практична необхідність. З одного боку, деякі методи часових рядів вимагають достатньо довгого ряду (наприклад, звичайне застосування

методології Бокса - Дженкінса не рекомендується для часових рядів коротших за 50). З іншого боку, характерні особливості довгих часових рядів зазвичай змінюються з часом, тож побудова адекватної моделі ускладняється зі збільшенням довжини часового ряду.

Так само типова проблема в довших часових рядах виникає через те, що виміри на початку даного часового ряду не обов'язково мають бути зіставними з тими, що перебувають наприкінці, наприклад, через інфляцію, зростання цін, технічний розвиток тощо. У такому разі слід скоригувати дані за допомогою відповідного індексу.

Перейдемо до розгляду методології моделювання часових рядів.

Декомпозиція часових рядів.

Реальність показує, що часові ряди економічного характеру зазвичай можна розкласти на кілька специфічних компонентів, а саме:

Компонент тренду Tr_t

Сезонна складова I_t

Циклічний компонент C_t

Залишкова (випадкова, нерегулярна) складова E_t

Ця декомпозиція пояснюється очікуваннями того, що певні її компоненти покажуть більш системні, чіткіші та більш простежуванні залежності, порівнюючи із загальним розглядом часового ряду.

Класична декомпозиція розглядає трендові, сезонні та циклічні компоненти як детерміновані функції часу, а залишковий компонент як стохастичну функцію часу (тобто як випадковий процес). Ці неспостережувані функції мають такі відмінні риси:

Тренд являє собою довгострокові зміни рівня часових рядів (наприклад, довгострокове збільшення або зменшення). Можна уявити, що трендовий компонент виникає як наслідок сил, що діють в одному напрямку. Наприклад, взаємопов'язані сили, що спричиняють зростання обсягів іпотечного кредитування, - це підвищені потреби деяких сегментів населення, зміни

заробітної плати, підвищення ринкової орендної плати, змін на ринку нерухомості, тощо. Компонент тренду носить відносний характер: зміни клімату, які економісти сприймають як довгострокові рухи, з погляду кліматологів, лише короткострокові відхилення.

Сезонна складова описує періодичні зміни часових рядів, які повторюються щороку. Ці зміни спричинені чергуванням сезонів, і вони суттєво впливають на більшість видів економічної діяльності (типові сезонні явища, наприклад, сільськогосподарське виробництво, безробіття, аварійність автомобілів, обсяги продажів, зняття депозитів тощо). Здебільшого місячні та квартальні дані типові для сезонного аналізу економічних часових рядів. Піврічні спостереження показують найнижчу частоту (позначається як частота Найквіста в спектральному аналізі часових рядів), що дає змогу статистично ідентифікувати сезонність. Сезонна структура змінюється в часі, наприклад, глобальне потепління знижує зимові падіння в будівельній індустрії. З практичної точки зору виключення сезонності зазвичай необхідне для отримання висновків з економічних часових рядів. Державні статистичні служби (в ЄС, США та інших країнах) зобов'язані публікувати тимчасові ряди важливих для народного господарства даних у двох видах: до елімінації сезонної компоненти і після. Спеціальні програмні продукти професійно справляються із сезонністю (наприклад, програмні системи X-12-ARIMA або X-13ARIMA-SEATS, які використовують у США).

Циклічна складова - найсуперечливіша складова часових рядів. Деякі автори уникають позначати цей компонент як циклічний (або навіть періодичний), і вони кажуть радше про коливання навколо тренду, де фази зростання (підйоми) чергуються з фазами спаду. Довжина окремих циклів, тобто відстань між сусідніми верхніми точками повороту (тобто локальними максимумами) або між сусідніми нижніми точками повороту (тобто локальними мінімумами) зазвичай змінюються, а також у часі може змінюватися інтенсивність окремих фаз кожного циклу. Циклічне зведення

може бути спричинене очевидними зовнішніми ефектами, але іноді причини цього важко виявити. Типовим представником цього компонента є так званий діловий цикл, який являє собою (регулярне) чергування підйомів і спадів - тривалість ділових циклів зазвичай становить від 5 до 7 років. Усунення циклічних компонент зазвичай складне як із фактичних причин (нелегко знайти й оцінити його походження), так і з розрахункових причин (характер може змінюватися в часі аналогічно до сезонної складової). Іноді сезонні та циклічні компоненти позначають разом як періодичні компоненти часових рядів.

Залишковий компонент (званий також випадковим або нерегулярним компонентом) залишається в часових рядах після виключення трендових і періодичних складових. Він формується випадковими рухами (коливаннями, флуктуаціями) часових рядів, які не мають розпізнаваної систематичної природи. Тому цей компонент не входить в описані вище систематичні компоненти.

Залишкова складова також охоплює помилки вимірювання та округлення, і помилки, допущені під час моделювання цього часового ряду. Щоб виправдати деякі статистичні процедури, які використовуються для класичної декомпозиції, зазвичай припускають, що залишкова складова - це так званий білий шум (або навіть нормально розподілений білий шум). Тут термін "білий шум" позначає послідовність $\{\varepsilon_t\}$ некорельованих випадкових величин з нульовим середнім значенням і постійною (кінцевою) дисперсією $\sigma^2 > 0$.

$$E(\varepsilon_t) = 0, \quad var(\varepsilon_t) = \sigma^2 > 0, \quad cov(\varepsilon_s, \varepsilon_t) = 0 \text{ для } s \neq t \qquad (1.42)$$

де ε_t- незалежні й однаково розподілені випадкові величини з нульовим середнім значенням і постійною дисперсією). Назва "білий шум" походить зі спектрального аналізу і відноситься до властивості постійного спектра з рівною величиною на всіх частотах (або довжинах хвиль). Величини ε_tтакож

називають інноваціями, оскільки вони відповідають непередбачуваним рухам у часових рядах.

Очевидно, що цей економічний часовий ряд можна розглядати як тренд, пов'язаний з періодичними складовими (тобто сезонними і циклічними) і білим шумом. Причому розкладання може бути як адитивним, так і мультиплікативним.

Адитивний розклад має вигляд

$$y_t = Tr_t + C_t + I_t + E_t \qquad (1.43)$$

За адитивного розкладання всі компоненти вимірюються в одиницях часового ряду y_t, тобто всі компоненти є абсолютними (а не виміряними відносними, наприклад, у відсотках від тренда).

Мультиплікативний розклад має вигляд

$$y_t = Tr_t \cdot C_t \cdot I_t \cdot E_t \qquad (1.44)$$

Легко помітити, що використання логарифмічної шкали дає змогу перетворити адитивну декомпозицію на мультиплікативну і навпаки.

1.4 Висновки до розділу 1

На базі проведеного дослідження предметної області можна зробити наступні висновки:

Процес Орнштейна – Уленбека може бути застосований в імітаційних моделях та розрахунках довгострокового доходу по акціях, найпростішою з подібних моделей, де використовується процес Орнштейна – Уленбека є модель Васічека, запропонована в 1977 році. З одного боку, мова йде про імітаційне моделювання, що неможливе без застосування обчислювальної техніки; з іншого – імітаційне моделювання не відноситься до галузі машинного навчання.

Більш загальне застосування стохастичного процесу Орнштейна – Уленбека може бути показано при моделюванні часових рядів, де стохастичний компонент описується процесом Орнштейна – Уленбека. В другому розділі буде розглянуто побудову моделі прогнозування, з використанням модифікації методології Бокса – Дженкінса, де стохастичний компонент описується процесом Орнштейна – Уленбека. При побудові моделі, яка буде моделлю навчання з вчителем, треба враховувати, що процес Орнштейна – Уленбека тяжіє до рівноваги.

2 ПРОЕКТУВАННЯ ТА ПОБУДОВА МОДЕЛІ

2.1 Методологія Бокса - Дженкінса

Визначившись з принципами представлення часових рядів, необхідно також вибрати належну модель, яка б змогла дати певний прогноз, заснований на історичних даних. Методологія побудови моделей прогнозування часових рядів була запропонована Джорджем Боксом та Гвілімом Дженкінсом. Їх концепція має фундаментальне значення в області аналізу і прогнозування часових рядів. Методологія Боксу-Дженкінса не передбачає будь-якої конкретної закономірності в історичних даних ряду, які необхідно спрогнозувати. Методологія використовує ітеративний підхід ідентифікації моделі, оцінки параметрів і діагностичної перевірки для визначення кращої можливої моделі прогнозування часового ряду.

На підставі обраного критерію, а частіше за все використовують інформаційний критерій Акайке

$$AIC(p) = n \cdot ln\left(\widehat{\sigma_e}^2/n\right) + 2p \qquad (2.1)$$

або Байесівський інформаційний критерій

$$BIC(p) = n \cdot ln\left(\widehat{\sigma_e}^2/n\right) + p + p \cdot ln(n) \qquad (2.2)$$

обирають найкращу з переглянутих ітеративно моделей.

Методологія Бокса - Дженкінса приймає основою для побудови моделей часових рядів залишкову складову (тобто компонент випадкового характеру). Ключові статистичні інструменти тут полягають у кореляційному аналізі, і, отже, ця методологія може успішно справлятися із взаємно корельованими спостереженнями.

Наприклад, одна з найпростіших моделей методології Бокса - Дженкінса - це так званий процес ковзного середнього першого порядку, що позначається

як MA(1). Такий підхід доречний для таких часових рядів, де всі спостереження є взаємно некорельованими за винятком прямих сусідів. Ця модель може мати такий вигляд

$$y_t = \varepsilon_t + 0.7\varepsilon_{t-1} \qquad (2.3)$$

де y_t - модельований часовий ряд. Інші види моделі, що застосовуються в рамках методології Бокса - Дженкінса, називаються процесами авторегресії AR і процесами ARMA.

На перший погляд може здатися, що увага, яку приділяє дана методика випадковій складовій, є надмірною, і втрачається можливість моделювання нестаціонарних часових рядів з явним трендом або сезонним характером (так звана стаціонарність часового ряду означає, що поведінка цього ряду є стійкою в певному сенсі). Однак методологія Бокса - Дженкінса здатна керувати також цими випадками за допомогою так званих інтегрованих процесів ARIMA та сезонних процесів SARIMA, де моделюються тренд або сезонні складові стохастичним способом (на відміну від детермінованого моделювання при використанні класичного підходу декомпозиції). Наприклад, у дуже простій моделі ARIMA (0, 1, 0)

$$y_t = y_{t-1} + \varepsilon_t \qquad (2.4)$$

стохастичний тренд можна охарактеризувати таким чином, що його прирощення перевищує окремі інтервали спостереження є випадковими у вигляді білого шуму (отже, це пояснює, чому процес називається випадковим блуканням. Завдяки цьому стохастичному підходу, методологія Бокса - Дженкінса є дуже гнучким моделюванням, зокрема для нестандартних часових рядів, які не піддаються контролю за допомогою класичного підходу декомпозиції.

Завдання прогнозування історично виникло під час дослідження часових рядів і спроби передбачення їхніх значень через деякий проміжок часу. У класичній задачі прогнозування навчальна вибірка являє собою набір вимірювань $X = \{x[i]\}_{i=1}^{n}$, що являють собою вектор дійсних величин $x[i] =$

$(x_1[i], \dots, x_d[i])$, які відповідають деякому моменту часу. Потрібно побудувати алгоритм (предиктор), який повернув би точну оцінку $\{\hat{x}[i]\}_{i=n+1}^{n+q}$, довірчий інтервал $\{(x_-[i],\ x_+[i])\}_{i=n+1}^{n+q}$ або апостеріорний розподіл $p(x[n+1], \dots, x[n+1]|x[1], \dots, x[n]$)прогнозу на задану глибину q. На відміну від задачі відновлення регресії, здійснюється прогноз не за ознаками, а за часом. Сформульовану задачу також називають задачею авторегресії.

Якщо повернутись до визначення, Модель, в якій розрахункові значення рівнів ряду визначаються як лінійна функція від попередніх спостережень, називають авторегресійною.

Авторегресійні моделі широко використовуються для опису стаціонарних випадкових процесів. Характерною особливістю стаціонарних часових рядів є те, що їх імовірнісні властивості рядів не змінюються в часі. Інакше кажучи, функції розподілу стаціонарних динамічних рядів не змінюються при зсуві часу.

Ідентифікація AP(p) моделі полягає у визначенні її порядку p. Однією з передумов побудови моделі цього типу є застосування їх до стаціонарного процесу. Тому в більш широкому значенні ідентифікація моделі включає також вибір способу трансформації вихідного ряду спостережень, як правило, має деяку тенденцію, в стаціонарний (або близький до нього) ряд. Один з найбільш поширених способів вирішення цієї проблеми - послідовне взяття різниць, тобто перехід від вихідного ряду до ряду перше, а потім і другий різниць.

"Чисті" авторегресійні процеси мають плавно загасаючу автокореляційну функцію (АКФ). У цьому випадку в якості порядку моделі вибирається лаг, після якого всі часткові автокореляційні функції (ЧАКФ) мають незначну величину. Однак на практиці рідко зустрічаються процеси, які легко було б ідентифікувати. Тому порядок моделі зазвичай визначається методом проб з декількох альтернатив. У число кандидатів включаються моделі, у яких порядок відповідає ЧАКФ, що перевищує стандартне

відхилення $1/N$. При обробці різницевих рядів іноді орієнтуються на АКФ, вибираючи моделі, у яких порядок відповідає максимальному значенню, за умови, що воно перевищує стандартне відхилення.

2.2 Математичний базис побудови моделі

Будемо вважати, що розглядається певний часовий ряд, залежна змінна якого має сезонну залежність та певну лінію тренду. Графічна візуалізація подібного часового ряду показана на рисунку 2.1.

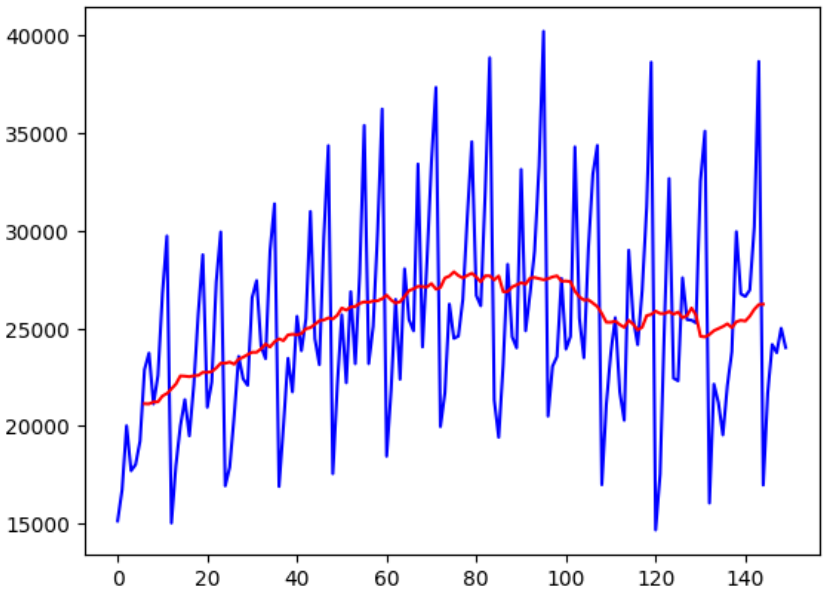


Рисунок 2.1 – Приклад часового ряду.

Червоним коліром показано лінію тренду, отриману шляхом обчислення ковзного середнього.

Згідно попереднього розділу, залежний фактор може бути представлений у вигляді композиції (1.43) або іншої подібної композиції, що є наслідком певної декомпозиції часового ряду на незалежні компоненти різної природи.

Перепишемо (1.43) у вигляді (2.5)

$$W(t) = w(t) + v(t) \qquad (2.5)$$

де $w(t)$ – імітує типову поведінку (тренд) залежного фактору. Друга складова $v(t)$ – випадкові флуктуації залежного фактору. Ці пульсації можуть бути описані процесом типу Орнштейна – Уленбека [13]

$$v(t) = \beta \cdot v(t - \Delta t) + \sigma \cdot \varphi \qquad (2.6)$$

де параметри зносу (дрейфу) $\beta = \beta(\Delta t)$ та волатильності $\sigma = \sigma(\Delta t)$ можна розрахувати різними методами з використанням статистичних даних, а випадкова величина φ має розподіл Вейбула або інший симетричний.

Коефіцієнт β визначає швидкість повернення до середнього рівня. Величина σ (волатильність) характеризує дисперсію відхилень, які мають нормальний розподіл, і може прийматися постійною для досліджуваного періоду; випадковий процес при цьому вважається стаціонарним.

Таким чином, маємо розкладення часового ряду вигляду

$$y_t = Tr_t + C_t + E_t \qquad (2.7)$$

Тобто з відсутнім сезонним компонентом. З урахуванням поставленої задачі аналізу високочастотних фінансових даних, що досить рідко мають сезонну складову, це не погіршує якість моделі прогнозування.

При наявності історичних даних, які містять послідовності значень у вигляді часового ряду, математична модель поточного стану відповідного параметра може бути представлена як дискретна модель блукання, де поточні значення лежать в певному коридорі відносно середнього.

Таким чином, модель прогнозування часових рядів, що будується, базується на (2.5) [14]

$$W(t) = w(t) + \beta \cdot v(t - \Delta t) + \sigma \cdot \varphi(t) \qquad (2.8)$$

$w(t)$ – імітує типову поведінку (тренд) залежного фактору;

β – параметр зносу (дрейфу);

σ – волатильність.

Задачею є пошук методом градієнтного спуску (стохастичного градієнтного спуску) таких параметрів (2.8), при яких відхилення розрахованих величин від відповідних спостережень є мінімальним.

Загальний план методів спуску.

1. Вибирається початкове наближення - деяка точка x^0. Доцільно використати всю наявну інформацію про поведінку цільової функції $f(x)$, щоб вибрати x^0 ближче до точки мінімуму.

2. Нехай наближення x^k до точки мінімуму знайдено і $x^k \neq x^*$. Вибирається напрямок спуску, тобто вектор $p^k \neq 0$, такий, що для всіх достатньо малих $\alpha > 0$ справедлива нерівність

$$f(x^k + \alpha p^k) < f(x^k) \qquad (2.9)$$

3. Визначають величину кроку спуску за напрямком, тобто додатне число $\alpha_k > 0$, для якого виконується нерівність

$$f(x^k + \alpha_k p^k) < f(x^k) \qquad (2.10)$$

4. За чергове наближення до точки мінімуму приймається

$$x^{k+1} = x^k + \alpha_k p^k \qquad (2.11)$$

5. Перевіряється виконання критерію закінчення ітерацій для $i = k + 1$. Якщо критерій виконується, то ітерації припиняються й вважається $x^* = x^{k+1}$. В іншому випадку ітерації продовжуються з пункту 2.

Критерієм припинення процесу обчислень на i-му кроці часто вибирають близькість градієнта до нуля

$$\left\| f'(x^i) \right\| < \varepsilon \qquad (2.12)$$

Також можуть бути використані критерії

$$\left\| x^i - x^{i-1} \right\| < \varepsilon_1 \qquad \left\| f\left(x^i\right) - f\left(x^{i-1}\right) \right\| < \varepsilon_2 \qquad (2.13)$$

або обидва критерій чи їх комбінація з (2.9)

2.3 Основні засади побудови моделей машинного навчання

Модель - це певна конструкція або об'єкт, що подумки уявляється, який створюється, щоб краще зрозуміти системи реального світу.

Моделювання - триєдиний процес побудови, вивчення та застосування моделей.

Найчастіше, модель може бути поділена на три структурні одиниці: введення інформації, інформаційний процесор і виведення очікуваних результатів.

Машинне навчання (ML) - це напрямок штучного інтелекту (ШІ), зосереджений на створенні систем, які навчаються і розвиваються на основі одержуваних ними даних. Штучний інтелект - це широкий термін, який включає в себе комп'ютерні системи, що імітують людський інтелект. Машинне навчання і ШІ часто йдуть пліч-о-пліч, і терміни іноді використовуються взаємозаміно, але, строго кажучи, це не одне й те саме. Різниця полягає в тому, що машинне навчання завжди має на увазі використання ШІ, однак ШІ не завжди має на увазі машинне навчання.

Якщо розглядати машинне навчання у застосуванні до часових рядів, мова піде про навчання з вчителем, оскільки задачею машинного навчання при роботі з часовими рядами найчастіше стає задача прогнозування динаміки часового ряду на базі спостережень попередніх періодів.

Машинне навчання з учителем (supervised learning) - це метод машинного навчання, за якого алгоритм опрацьовує розмічені дані (дані, для яких відома правильна відповідь), щоб виявити патерни та створити математичну модель, яку можна використовувати для передбачення відповіді на нових вхідних даних.

При використанні навчання з вчителем мета полягає в тому, щоб знайти оптимальну функцію відображення $f\,(x)$, яка мінімізує функцію втрат навчальної вибірки

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L\left(y^i, f\left(x^i, \theta\right)\right) \qquad (2.14)$$

де N - кількість елементів навчальної вибірки;

θ - параметр відображення $f\,(x)$;

x^i- вектор ознак і-го зразке;

y^i- відповідне і-му зразку значення залежної змінної;

L - втрата функції втрат.

$\in$ багато видів функцій втрат у навчанні з учителем, як-от квадрат евклідової відстані, кроссентропія, втрата контрасту (contrast loss), втрата шарніра (hinge loss), приріст інформації тощо. В побудові моделі для оцінки будемо використовувати стандартну метрику середньо – квадратичного відхилення (RMSE) та інформаційний критерій Акайке (2.1).

В задачі оптимізації, метою розв'язання якої, є знаходження параметрів цільової функції, будемо використовувати метод градієнтного спуску та метод стохастичного градієнтного спуску.

Таблиця 2.1 – Характеристики та порівняння методів оптимізації.

Метод градієнтного спуску – пошук оптимального значення в напрямку градієнту спуску. Збігається з лінійною швидкістю.	
Переваги	Недоліки
Рішення є глобально оптимальним, коли цільова функція опукла.	У кожному оновленні параметра градієнта необхідно розрахувати всі значення, тому вартість розрахунку висока.
Стохастичний градієнтний спуск - параметри оновлення розраховують із використанням випадково обраної міні-партії. Метод сходиться із суб-лінійною швидкістю	
Переваги	Недоліки
Час розрахунку для кожного оновлення не залежить від загальної кількості навчальних вибірок, і істотна частина часу зберігається.	Складно вибрати відповідну швидкість навчання. Використання однакової швидкості навчання за всіма параметрами не прийнятне.

	Рішення може застрягти в сідловій точці в деяких випадках.

2.4 Висновки до розділу 2

В поточному розділі розглянуто основні принципи побудови моделей часових рядів, моделей машинного навчання взагалі та на підставі дослідження першого розділу сформульовано математичну модель, що буде використано в якості базису для конструювання та реалізації програмного забезпечення прогнозування часових рядів з використанням процесу Орнштейна – Уленбека.

В якості методології побудови моделі буде використано модифіковану методологію Бокса – Дженкінса, основним рівнянням, що використовує програмне забезпечення є рівняння (2.8). Оцінка моделей буде проводитись з використанням середньо – квадратичної похибки та інформаційного критерію Акайке. Для оптимізації параметрів обрано методи з сімейства методів градієнтного спуску.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПОБУДОВАНОЇ МОДЕЛІ

3.1 Конструювання програмного забезпечення

3.1.1 Інструменти розробки, аналізу та візуалізації

Основна мова програмування – Python [15], версія 3.11. Python обраний як одна з трьох найпопулярніших мов програмування. Python - скриптова мова, яка не компілюється до запуску програми. Програмувати та використовувати Python – скрипти можна практично на всіх платформах: від iOS та Android до серверних ОС.

З урахуванням необхідності використання широкого спектру математичних обчислень, а також використання бібліотек машинного навчання, зокрема Scikit-learn [16], та реалізації окремих моделей, що використовуються для аналізу та порівняння результатів, у тому числі ARIMA моделі, Python стає для реалізації проекту практично безальтернативним.

Використовувані в програмній реалізації пакети:

NumPy – перетворення даних;

Pandas – робота з датафреймами (таблиці даних);

Sklearn [17] – бібліотеки машинного навчання, в поточній задачі використовується для зручного розрахунку необхідних метрик;

Pmdarima [18] – бібліотека реалізації моделі ARIMA, яка буде застосовуватись для порівняльного аналізу. Це статистична бібліотека, призначена для заповнення порожнечі в можливостях аналізу часових рядів Python.

Matplotlib [19] – основний пакет візуалізації даних;

Seaborn [20] – обгортка Matplotlib для зручної візуалізації даних з датафреймів Pandas;

Як середовище розробки використовуватиметься IDE PyCharm Professional з розширенням, що дозволяє працювати безпосередньо з сервером Jupiter Notebook. Це дозволяє поєднати зручність повноцінної IDE розробки з функціоналом середовища Jupiter. Більшість завдань аналізу даних та побудови математичних моделей реалізується саме з використанням середовища використання IPython.

Усі етапи побудови, вивчення та порівняння моделей, що будуть створюватись, буде виконано в оточені Jupiter.

3.1.2 Постановка задачі програмної реалізації

Задачею програмної реалізації є побудова апарату для прогнозування часових рядів високочастотних фінансових показників, засобів оцінки побудованих моделей та візуалізації результатів аналізу та прогнозування. Також до кола задач включимо оцінку побудованої моделі за обраними метриками якості та алгоритм оптимізації гіперпараметрів моделі.

Гіперпараметри - це змінні, які регулюють сам процес навчання моделі. Наприклад, частина налаштування глибокої нейронної мережі полягає в тому, щоб вирішити, скільки прихованих шарів вузлів використовувати між вхідним і вихідним шарами і скільки вузлів повинен використовувати кожен шар. Ці змінні не мають прямого відношення до навчальних даних. Це змінні конфігурації.

Оцінку якості моделі будемо вести з використанням RMSE згідно другого розділу.

3.1.3 Паттерн проектування моделі

В якості патерну реалізації моделі – предіктора використаємо єдиний базовий API поширеної бібліотеки Scikit-learn. Це дозволить отримати код, що має чітку та перевірену практикою структуру. З одного боку такий код та побудова є звичним для спеціалістів Data Science, з іншого боку, абстрактний базовий клас лінійних моделей Scikit – learn вимагає реалізації мінімуму методів.

Основними об'єктами в scikit-learn є чотири об'єкти. При цьому один клас може реалізовувати декілька з наведених нижче інтерфейсів.

Таблиця 3.1 – Базові інтерфейси Scikit - learn

Estimator	`estimator = estimator.fit(data, targets)` `estimator = estimator.fit(data)`
	Базовий об'єкт, що реалізує метод *fit()* – навчання на тренувальних даних.
Predictor	`prediction = predictor.predict(data)` `probability = predictor.predict_proba(data)`
	Для навчання з вчителем та деяких алгоритмів навчання без вчителя реалізується метод *predict()* створення передбачення з використанням навченої моделі.
Transformer	`new_data = transformer.transform(data)` `new_data = transformer.fit_transform(data)`

	Для фільтрування або модифікації даних з використанням алгоритмів навчання з вчителем та без реалізується метод *transform()* або *fit_transform()*. Останній виконує одночасне навчання моделі та перетворення даних з її використанням.
Model	`score = model.score(data)`
	Велика кількість моделей може бути оцінена за певним критерієм (метрикою), що задовольняє правилу більше – краще.

Для моделі, що буде побудовано, доцільно реалізовувати *fit(), predict()* та *score()*.

API має один головний об'єкт: оцінювач. Оцінювач — це об'єкт, який відповідає моделі на основі деяких навчальних даних і здатний виводити деякі властивості нових даних. Це може бути, наприклад, класифікатор або регресор. Усі оцінювачі реалізують метод підгонки (реалізація є обов'язковою при успадковувані від класу базової моделі):

```
estimator.fit(X, y)
```

Усі вбудовані оцінювачі (estimators) також мають метод *set_params*, який встановлює незалежні від даних параметри (перевизначаючи попередні значення параметрів, передані в *__init__)*.

Усі оцінювачі в основній кодовій базі scikit-learn мають бути успадковані від *sklearn.base.BaseEstimator*.

Інстанціювання. Це стосується створення об'єкта. Метод *__init__* об'єкта може приймати константи як аргументи, які визначають поведінку. Однак конструктор не повинен приймати фактичні навчальні дані як аргумент, оскільки це залишається для методу *fit()*.

Усі аргументи, які приймає *__init__*, мають бути ключовими аргументами зі значенням за замовчуванням. Іншими словами, користувач

створеного класу повинен мати можливість створити екземпляр оцінювача, не передаючи йому жодних аргументів. Усі аргументи мають відповідати гіперпараметрам, що описують модель або задачу оптимізації, яку намагається вирішити оцінювач. Ці початкові аргументи (або параметри) завжди запам'ятовуються оцінювачем.

Крім того, кожен аргумент ключового слова, прийнятий *__init__*, повинен відповідати атрибуту екземпляра. Певні алгоритми Scikit-learn, в тому числі GridSearchCV, покладаються на можливість пошуку аргументів за ім’ям. Правильний приклад вигляду конструктора подано нижче.

```
def __init__(self, param1=1, param2=2):
    self.param1 = param1
    self.param2 = param2
```

Навчання моделі. Метод *fit()* приймає навчальні дані як аргументи, які можуть бути одним масивом у випадку навчання без вчителя або двома масивами у випадку навчання з вчителем.

Треба зауважити, що зазвичай модель підігнана за допомогою *X* і *y*, але об’єкт не містить посилань на *X* і *y*.

Аргументи методу виглядають так, як представлено в таблиці 3.2.

Таблиця 3.2 – Аргументи методу *fit()*

X	масив даних типу NDArray (NumPy); розмір масиву n – спостережень, m – ознак
y	масив даних типу NDArray (NumPy); розмір масиву n – спостережень
kwargs	опційні параметри, значення яких є залежними від набору даних

Атрибути моделі, що визначаються під час навчання мають завжди ім’я, що закінчується символом підкреслення в кінці, наприклад, коефіцієнти деякого регресійного оцінювача зберігаються в атрибуті *coef_* після виклику *fit()*. Оцінені атрибути перевизначаються після повторного виклику *fit().*

При створюванні власного оцінювача є можливість перевірити, чи він відповідає базовим принципам побудови оцінювачів Scikit-learn. Для використовується *check_estimator()* з модулю *sklearn.utils.estimator_checks*.

Побудова оцінювача з використанням належного патерну дозволить використовувати його сумісно з іншим інструментарієм Scikit-learn, зокрема методами пошуку гіперпараметрів та всередині конвеєрів обробки даних.

Окрім рекомендацій в документації Scikit – learn, в репозиторії на GitHub розміщено відповідні рекомендаціям шаблони.

Деякі загальні функції залежать від типу переданого оцінювача. Наприклад, перехресна перевірка в *model_selection.GridSearchCV* і *model_selection.cross_val_score* за замовчуванням стратифікується, коли використовується в класифікаторі, але не в інших випадках. Подібним чином оцінювачі для усередненої точності, які використовують безперервне передбачення, повинні викликати *decision_function* для класифікаторів, але *predict* для регресорів. Ця різниця між класифікаторами та регресорами реалізована за допомогою атрибута *_estimator_type*, який приймає рядкове значення. Це має бути *"classifier"* для класифікаторів, *"regressor"* для регресорів і *"clusterer"* для методів кластеризації, щоб працювати належним чином. Успадкування від *ClassifierMixin*, *RegressorMixin* або *ClusterMixin* автоматично встановлює атрибут.

На завершення буде наведено деякі рекомендації щодо написання коду з офіційної документації Scikit-learn.

– проект *scikit-learn* намагається точно слідувати офіційним інструкціям *Python*, викладеним у PEP8, які детально описують форматування коду та відступи;

- слід використовувати підкреслення для розділення слів у назвах, що не належать до класів: *n_samples* замість *nsamples*.
- слід уникати кількох тверджень в одному рядку, надаючи перевагу поверненню рядка після оператора потоку керування (if/for).
- слід використовувати відносний імпорт для посилань у scikit-learn.
- модульні тести є винятком із попереднього правила; вони повинні використовувати абсолютний імпорт, точно так само, як і код клієнта. Як наслідок, якщо *sklearn.foo* експортує клас або функцію, реалізовану в sklearn.foo.bar.baz, тест має імпортувати їх із sklearn.foo.
- заборонене використання імпорту *. Офіційні рекомендації Python вважають його шкідливим. Це ускладнює читання коду, оскільки походження символів більше не вказується явно, але найголовніше, це запобігає використанню інструментів статичного аналізу, таких як *pyflakes*, для автоматичного пошуку помилок у *scikit-learn*.
- Для docstrings слід використовувати формат docstring numpy.

З урахуванням того, що необхідне вивчення та аналіз побудови моделі, доцільно додати до шаблону класу оцінювача не тільки метод оцінки моделі *score()*, але й метод, що відповідає за візуалізацію порівняння отриманих результатів прогнозування з тестовими даними (ground truth).

3.2 Аналіз обраної методології з використанням ПЗ

Розглянемо запропоновані в попередніх розділах підходи, застосувавши їх до конкретного часового ряду. В якості даних, що аналізуються, обираємо котирування акцій стрімінгового сервісу Netflix за останній рік.

Одним з ресурсів, що надають необхідні дані з використанням API, є Yahoo Finance. Ресурс, який стабільно функціонує на протязі довгого терміну та надає необхідні дані з визначеною користувачем детальністю: можливо

отримувати історичні дані як з інтервалом в одну хвилину так й лише за кожен день.

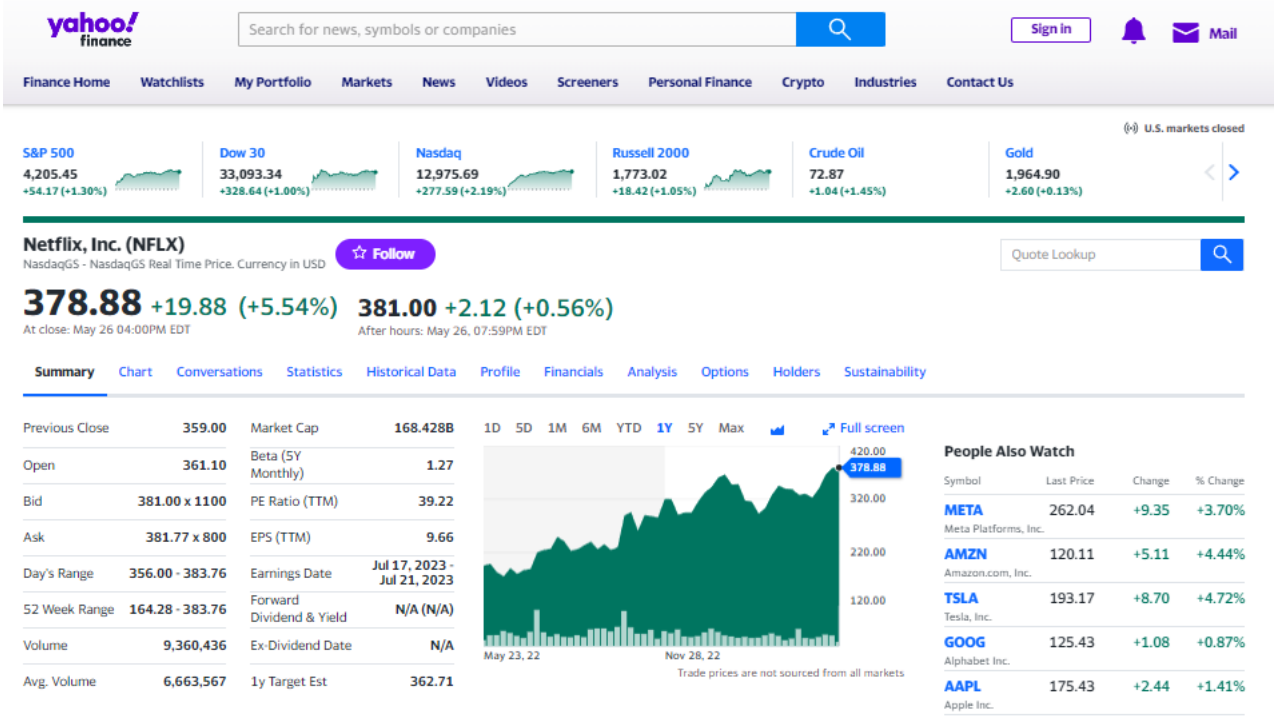


Рисунок 3.1 – сторінка NFLX на Yahoo Finance.

Найбільщ зручним способом роботи з API Yahoo Finance, з урахуванням використання Python в якості мови програмування, є бібліотека yfinance.

Бібліотека yfinance, останнє оновлення якої з'явилося 16 квітня поточного року, є високорівневою обгорткою над сервісом Yahoo Finance API, дозволяючи завантажувати необхідні дані у одну чи декілька рядків коду без безпосереднього використання низкорівневих запитів до API Yahoo та отримання результатів у вигляді числених JSON відгуків, що потребують подальшої агрегації у таблиці даних. Агрегація, звичайно, потребує додаткових операцій та тестування їх. Бібліотека, що розглядається, підтримує багатопотоковість, взаємодіє з протоколами безпеки Yahoo Finance та розповсюджується під ліцензією Apache License, version 2.0. Отже, являє собою зручний, швидкий та легальний спосіб отримання необхідних для подальшого аналізу даних з Yahoo Finance.

Відобразимо графічно динаміку котирувань акцій Netflix, одночасно створивши згладжені криві, використовуючі метод ковзного середнього з вікнами $n = 30$ та $n = 60$. Загальна довжина ряду складає 250, що є достатньою кількістю даних для аналізу та побудови прогнозу. Як було зауважено в попередніх розділах, при побудові моделей прогнозування часових рядів зазвичай необхідно 50 та більше спостережень.

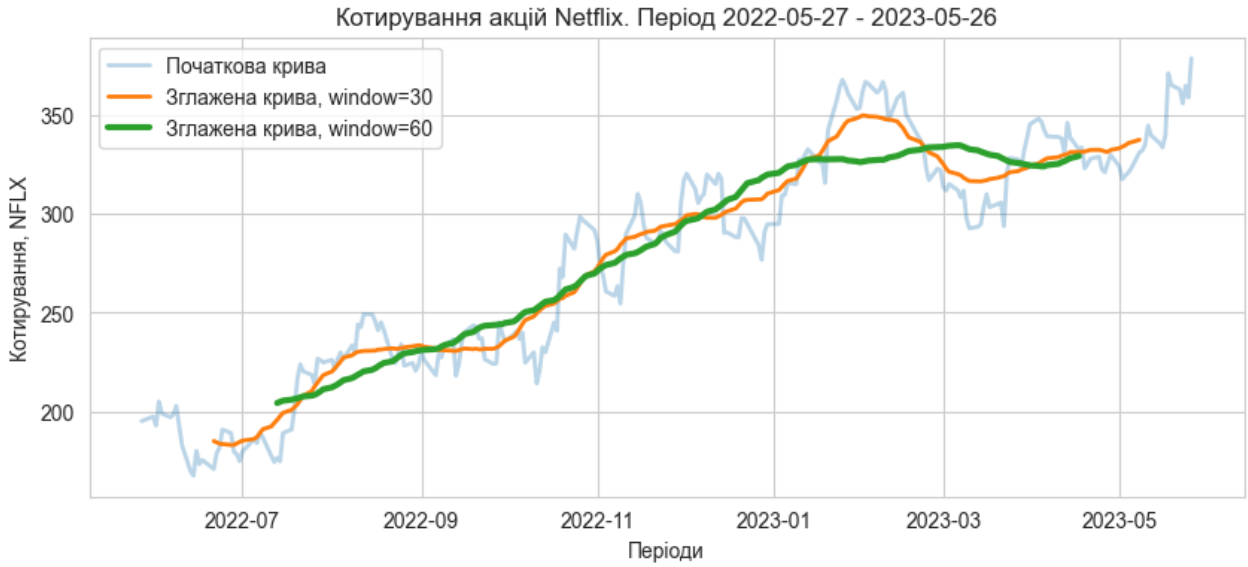


Рисунок 3.2 – Динаміка котирувань.

Розглядаючи графічне відтворення динаміки котирувань, можна зробити висновок, що початковий часовий ряд дійсно має певний тренд та певні осциляції значень навколо цього тренду.

Отже, абсолютно доречною є декомпозиція на функцію, відтворюючу тренд, та функцію, що відтворює ці осциляції на базі аналізу залишків.

Проведемо декомпозицію та видалимо тренд з даних, що розглядаються. Динаміка даних з видаленим трендом показана на рисунку 3.3.

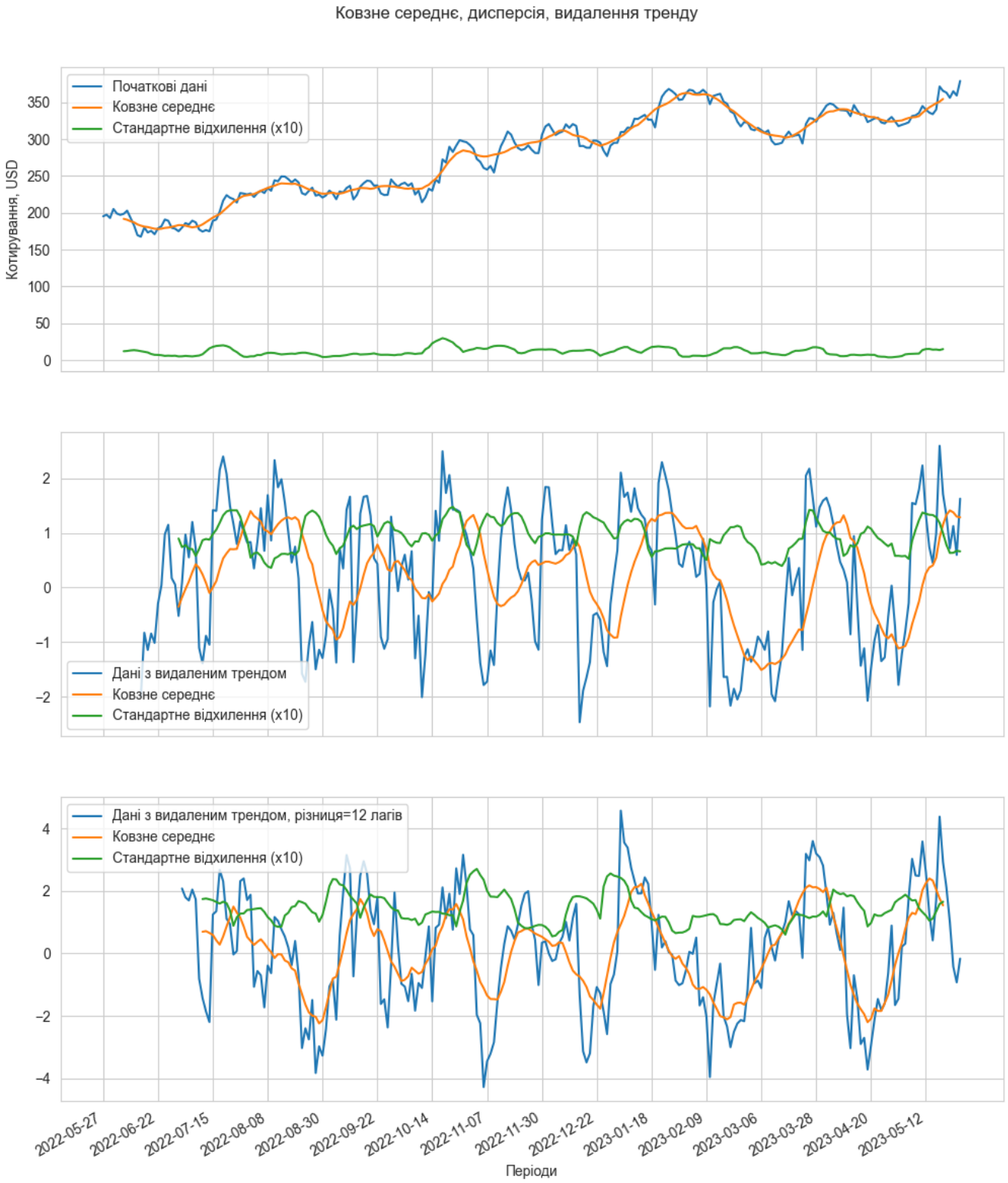


Рисунок 3.3 – Динаміка даних з видаленим трендом.

Розглянувши динаміку даних з видаленим трендом, можна зробити висновок, що відбувається осциляція відносно нульового рівня, стандартне

відхилення також здійснює певні незначні коливання відносно константного значення.

Застосуємо розширений тест Дікі – Фулера до даних, що розглядаються.

Результати застосування показано на рисунку 3.4 – самі дані не є стаціонарними, на відміну від даних з видаленим трендом.

```
Початкові дані
Значення статистики тесту = -0.940
P-значення = 0.775
Критичні значення:
        1%: -3.457 - Дані не є стаціонарними з довірчою ймовірністю 99%
        5%: -2.873 - Дані не є стаціонарними з довірчою ймовірністю 95%
        10%: -2.573 - Дані не є стаціонарними з довірчою ймовірністю 90%

Дані з видаленим трендом
Значення статистики тесту = -5.710
P-значення = 0.000
Критичні значення:
        1%: -3.458 - Дані  є стаціонарними з довірчою ймовірністю 99%
        5%: -2.874 - Дані  є стаціонарними з довірчою ймовірністю 95%
        10%: -2.573 - Дані  є стаціонарними з довірчою ймовірністю 90%

Дані з видаленим трендом, різниця=12 лагів
Значення статистики тесту = -4.699
P-значення = 0.000
Критичні значення:
        1%: -3.461 - Дані  є стаціонарними з довірчою ймовірністю 99%
        5%: -2.875 - Дані  є стаціонарними з довірчою ймовірністю 95%
        10%: -2.574 - Дані  є стаціонарними з довірчою ймовірністю 90%
```

Рисунок 3.4 – Застосування розширеного тесту Дікі – Фулера.

Використаємо Q-тест Льюнг — Бокса. Значення p-значення для різної кількості лагів. Результати застосування тесту показані на рисунку 3.5.

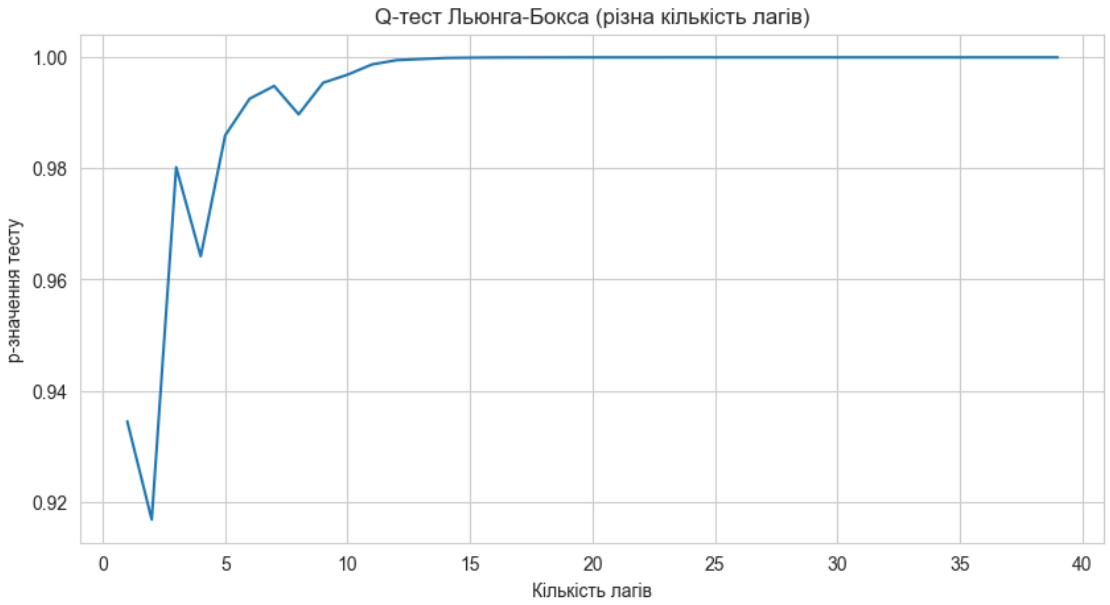


Рисунок 3.5 - P-значення тесту Льюнг — Бокса для різної кількості лагів.

Q-тест Льюнг-Бокса - статистичний критерій, призначений для знаходження автокореляції часових рядів. Замість тестування на випадковість кожного окремого коефіцієнта він перевіряє на відміну від нуля відразу кілька коефіцієнтів автокореляції.

Нульова гіпотеза: дані є випадковими (тобто є білий шум).

Альтернативна гіпотеза: дані не випадкові.

Впевнившись, що починаючи з невеликою кількості лагів, дані є випадковими відносно попередніх, за допомогою розробленого програмного коду, з’ясуємо функцію розподілу залишків.

Результати програмного визначення типу розподілу представлені на рисунку 3.6. Розподілами, до яких найбільш подібний отриманий розподіл, є в тому числі подвійний гамма – розподіл, подвійний розподіл Вейбула та нормальний зворотний розподіл Гауса. Перелічені розподіли є симетричними або наближені до симетричних.

	sumsquare_error	aic	bic	kl_div	ks_statistic	ks_pvalue
dgamma	0.018125	955.168843	-1509.101295	inf	0.042773	0.906996
dweibull	0.018495	955.822133	-1505.727521	inf	0.046680	0.843076
genhyperbolic	0.018840	956.682683	-1492.401283	inf	0.041871	0.919472
norminvgauss	0.018949	957.818799	-1496.554761	inf	0.042622	0.909155
hypsecant	0.019058	952.477493	-1505.837275	inf	0.039767	0.944791

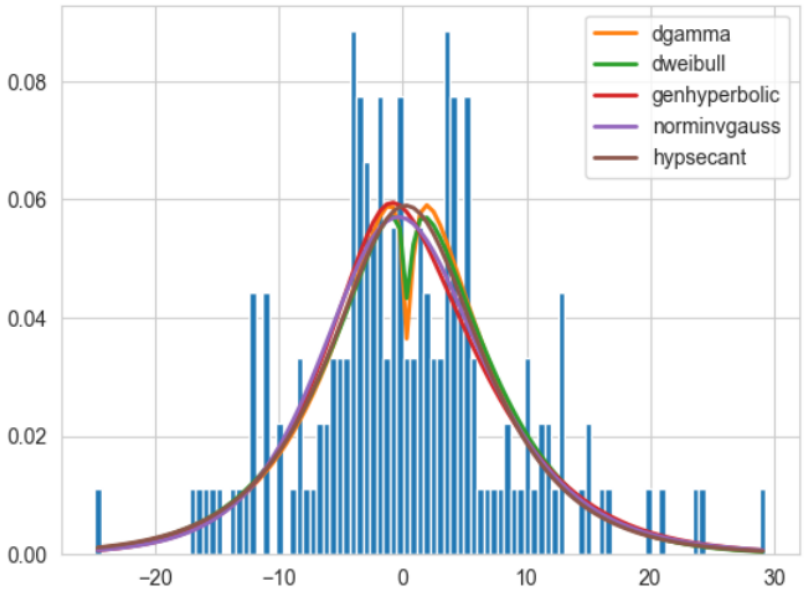


Рисунок 3.6 – Визначення типу розподілу залишків.

На підставі наведеного дослідження даних можна вважати, що для прогнозування цих даних може бути застосована математична модель (2.8), описана в попередньому розділі.

Ітеративна модель прогнозування буде побудовано аналогічно наведеному дослідженню даних. Є алгоритмом є декомпозиція на функцію тренду та функцію моделювання й визначення оптимального порядку та коефіцієнтів моделі за визначеними в розділі 2 критеріями.

3.3 Програмна реалізація розробленої моделі

Програмна реалізація моделі відповідає патерну побудови нащадка Base Estimator, що був наведений в попередніх підрозділах. Детально програмний код наведений у додатках, в поточному підрозділі зосередимось на основних моментах програмної реалізації.

Головними методами класу Predictor є конструктор класу, методи fit() та predict().

Сігнатура методу *fit()*: *fit(y: float) -> None,* де *y* – відповідні їм значення. Використовуються тільки значення *y,* оскільки розглядається випадок однакових проміжків між спостереженнями. Таке припущення не обмежує загальності розгляду, оскільки подібне припущення робиться у багатьох випадках аналізу та прогнозування часових рядів, в тому числі в моделі ARIMA. Метод *fit()* шляхом використанням приватних методів, описаних нижче, визначає параметри моделі та записує їх у відповідні параметри класу. На підставі частки тренувальних даних та параметрів класу відбувається прогнозування на певну кількість інтервалів методом *predict(),* що описаний далі.

Сігнатура методу *predict()*: *predict(n: int) -> NDArray[float],* аргументом методу є кількість значень, що повинні бути передбачені, а повертає метод NumPy масив таких значень.

Отримання оцінки моделі реалізовано шляхом визначення властивості (property) *score_* - визначається оцінка за критерієм Акайке для тренувальних даних, так реалізацією методу *score(y_true: NDArray[float], y_pred: NDArray[float]) -> float* для оцінки моделі шляхом порівняння отриманого прогнозу та фактичних даних (Ground truth).

Інша властивість *params_* -> Dict[str, float] дозволяє отримати параметри моделі, що були отримані під час навчання моделі. Методи get_params() та set_params() не реалізовані, оскільки модель не має суттєвих гіперпараметрів, які можуть встановлюватись незалежно від даних.

Для зручності аналізу ефективності моделі додано метод *show_plot(),* який дозволяє відтворити графічне порівняння прогнозованих та фактичних даних за умови, що до моменту виклику метода була проведена оцінка моделі методом *score()*. Реалізація такого методу від части є протиріччям принципам побудови Base Estimator тому, що вимагає збереження саме даних навчання та

прогнозування в змінних класу. Але з іншого боку це полегшує аналіз реалізації моделі й може бути видалено в фінальному зразку розробленої моделі прогнозування.

Інші методи, що реалізовані в основному класі Predictor відносяться до зони видимості в рамках класу, тому їх назви починаються з _.

Метод *_define_trend(y: NDArray[float]) -> Tuple[int, float, float, float, float. NDArray[float]]* виділяє тренд ітеративно з використанням для оцінки оптимального рішення метрик, що були визначені раніше. Повертає порядок моделі, параметри тренду та значення відповідних метрик, а останнім з значень, що повертаються, є масив залишків після виділення тренду.

Інший метод _define_params_on_residuals(res: NDArray[float]) -> Tuple[float, float) розраховує параметри дрейфу (зсуву) та волатильності.

3.4 Тестування та порівняння розробленої моделі

Першим кроком в тестуванні створеного програмного коду моделі, є безумовно, модульне тестування. Модульні тести, що перевіряють коректність розрахунків в моделі, наведені у відповідних додатках й не мають суттєвого інтересу з математичної точки зору чи точки зору перевірки коректності роботи побудованої моделі. Отже, в поточному розгляді не будемо зосереджуватись на них, зауваживши, що подальше тестування моделі зроблено після того, як модульне тестування підтвердило коректну поведінку створеного програмного коду.

Для перевірки коректності роботи побудованої моделі та її спроможності прогнозувати на задану кількість кроків «вперед», поділимо отриманий раніше набір даних на тренувальну та тестову частину.

Графічно результати такого поділу подані на рисунку 3.7.

Рисунок 3.7 – Поділ на тренувальні та тестові дані.

Частини графіку не з’єднані, оскільки остання точка тренувального набору та перша точка тестового набору не поєднуються.

Проведемо тренування створеної моделі та спрогнозуємо певну кількість періодів «вперед» з використанням навченої моделі.

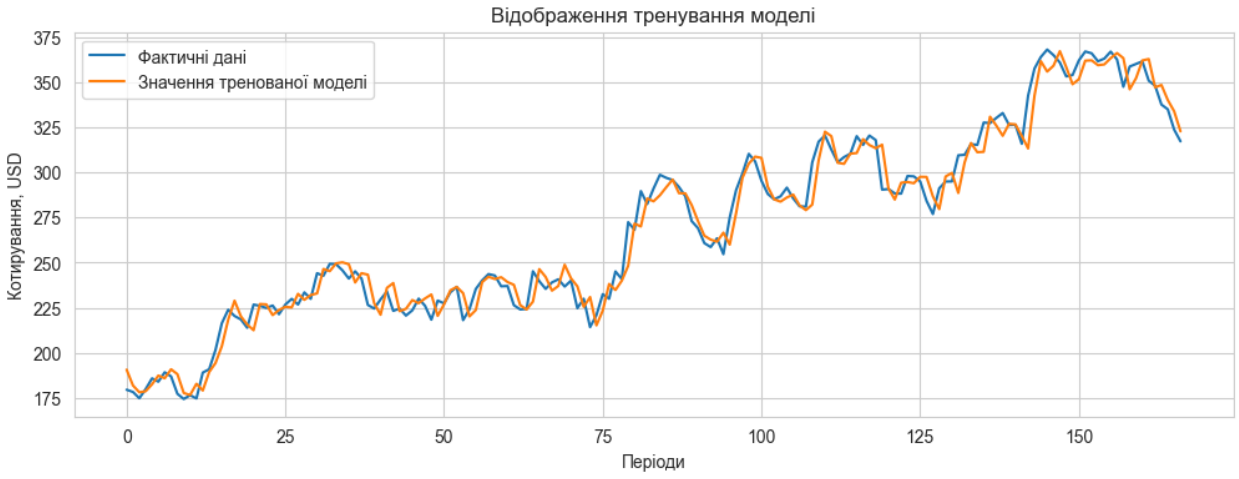


Рисунок 3.8 – Тренування моделі.

Результати виділення моделлю тренду показані на рисунку 3.8, такі результати можуть бути визнані задовільними.

Рахуємо та представляємо графічно залишки після віднімання двох перших розрахованих компонент рівняння (2.8). Графічне представлення знаходиться на рисунку 3.9.

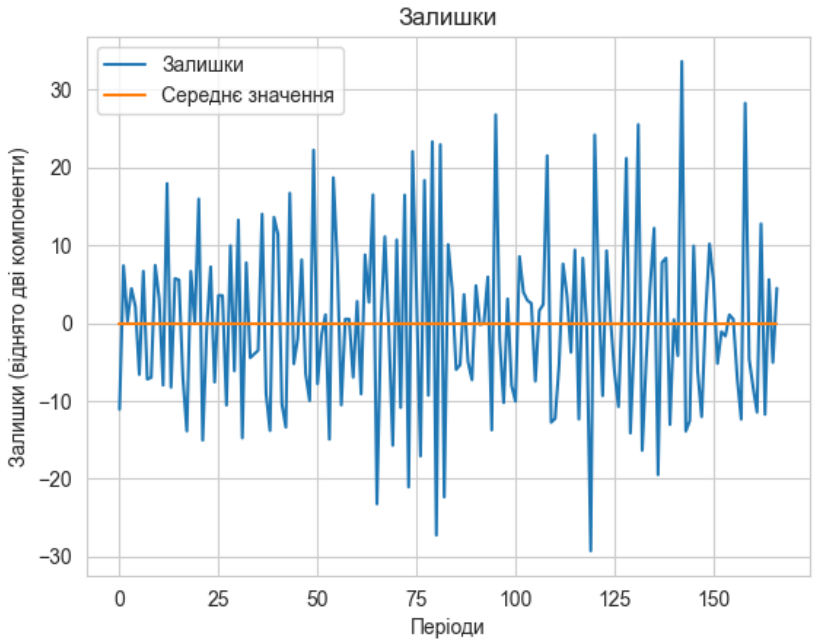


Рисунок 3.9 – Графічне відтворення залишків.

Середнє значення залишків дорівнює -0.026, отже дійсно отримана осциляція залишків відносно нульової відмітки.

Залишки описуються розподілом випадкової величини Вейбула. Так часто називають розподіл мінімального екстремального значення Вейбулла з теорії екстремальних значень (теорема Фішера-Гнеденка) Він виникає як граничний розподіл перемасштабованого мінімуму випадкових змінних.

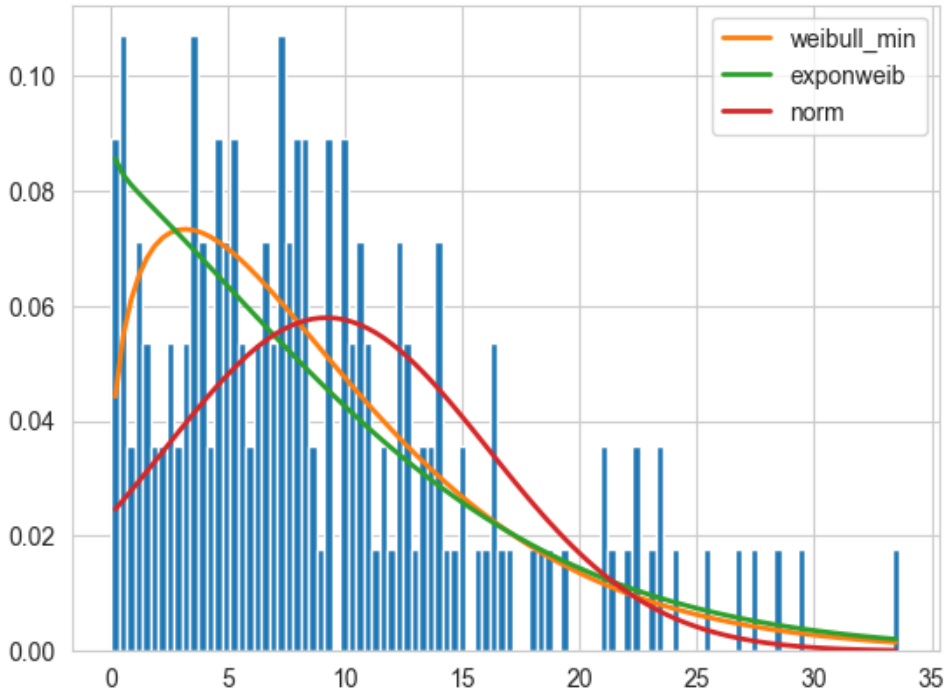


Рисунок 3.10 – Дистрибуція отриманих залишків в моделі.

На підставі вищенаведеного аналізу можна зробити висновок, що модель дійсно відповідає структурі (2.8).

З використанням натренованої моделі зробимо прогноз на 5, 10 та 64 інтервали «вперед». Остання кількість відповідає довжині тестової частини даних про котирування, що була сформована раніше.

Отримані результати прогнозування графічно показані на рисунку 3.11. Розглянувши рисунок 3.11 можна зробити висновок, що модель здатна прогнозувати тренд, але звичайно виникають труднощі з точним прогнозуванням залишків, оскільки ті є результатом випадкового блукання, як було показано вище. Отже, результат можна вважати цілком прийнятним. Із збільшенням кількості прогнозованих періодів знижується точність прогнозування.

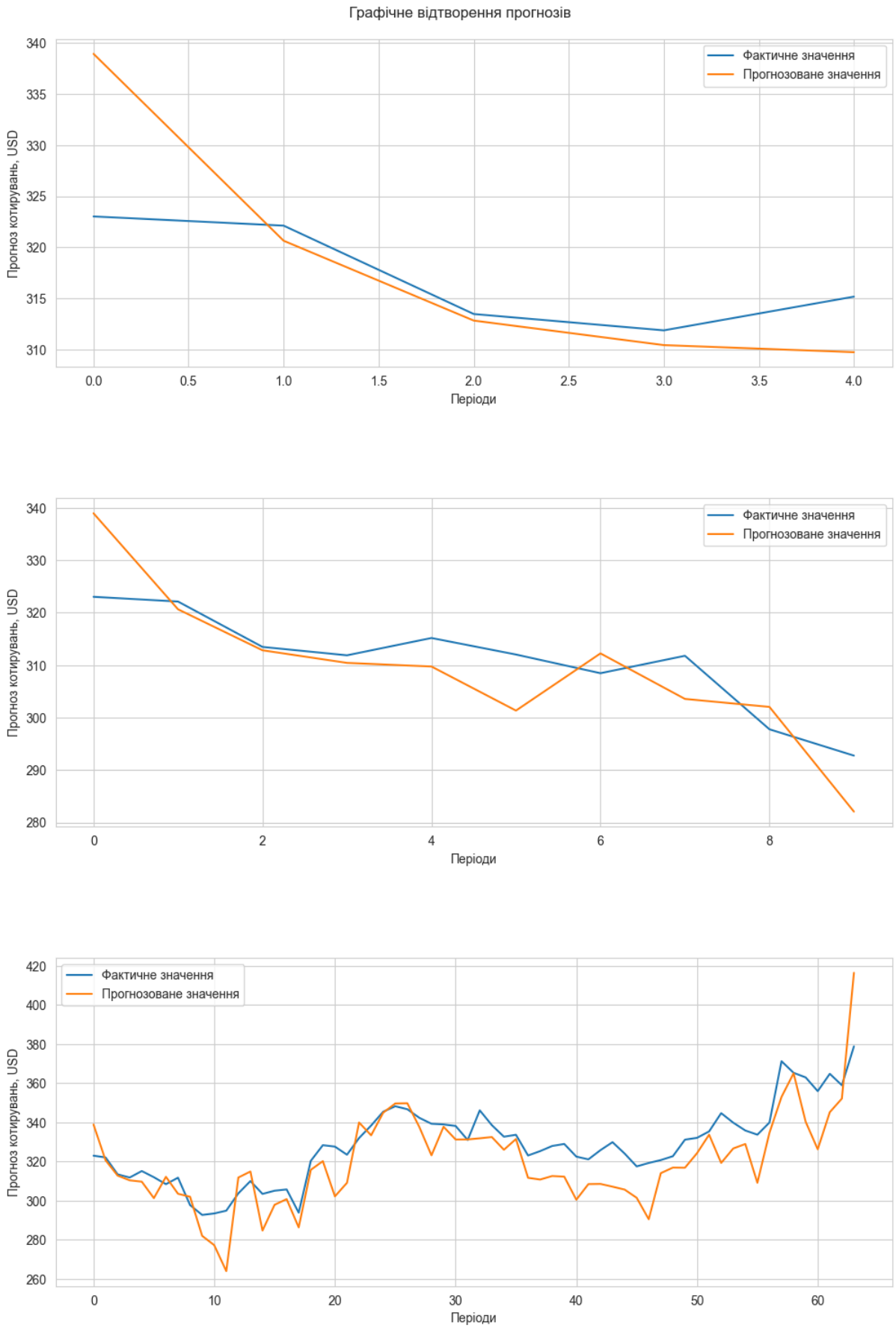


Рисунок 3.11 – Графічне відтворення результатів прогнозування.

Взагалі можна стверджувати, що прогнозування котирувань є одним з найскладніших, коли мова йде про часові ряди. Пояснюється це тим, що в наборі даних відсутні сезонні або подібні до них закономірності, а стохастичний компонент, як можна бачити, грає відносно велику роль в порівнянні з трендом. Наприклад, в задачах прогнозування споживання роль стохастичного компонента, як виявляється, суттєво менша.

Порівняємо отримані результати прогнозування з результатами, яки дозволяє отримати модель ARIMA, запропонована за методологією Бокса – Дженкінса.

Для цього, як було вказано раніше, скористаємося реалізацією моделі pmdarima, яка окрім іншого дозволяє автоматично визначати порядок моделі.

Для визначення порядку моделі (d) скористаємося одночасно (порівнявши отримані значення) розширеним тестом Дікі – Фулера та тестом KPSS (Kwiatkowski–Phillips–Schmidt–Shin). За результатами застосування тестів максимальний порядок моделі становить $d = 1$. Створюємо екземпляр auto_arima за допомогою коду, наведеного нижче

```
auto =
pm.auto_arima(scl.fit_transform(y_train.reshape(-1,
1)).ravel(), # тренувальний набір даних
d=1,                        # визначений порядок різниць
seasonal=False,             # сезонність
stepwise=False,             # використання алгоритму
Hyndman and Khandakar для визначення параметрів моделі
suppress_warnings=True,     # відключення попереджень
error_action="ignore",      # дія при виникненні помилок
max_p=15,                   # максимальне значення p при
                              налаштуваннях моделі
max_order=30,               # максимальне значення p+q+P+Q
                              при налаштуваннях моделі
```

```
trace=True,                    # виведення логу
n_jobs=6,                      # кількість потоків
n_fits=10                      # кількість ітерацій для
                                  одного набору (p, d, q)
)
```

За результатами тренування моделі найкращою визначена модель ARIMA(1, 1, 3).

Спочатку перевіряємо якість «підгонки» моделі, відобразивши графічно результати auto.fittedvalues(). Результати показано на рисунку 3.12.

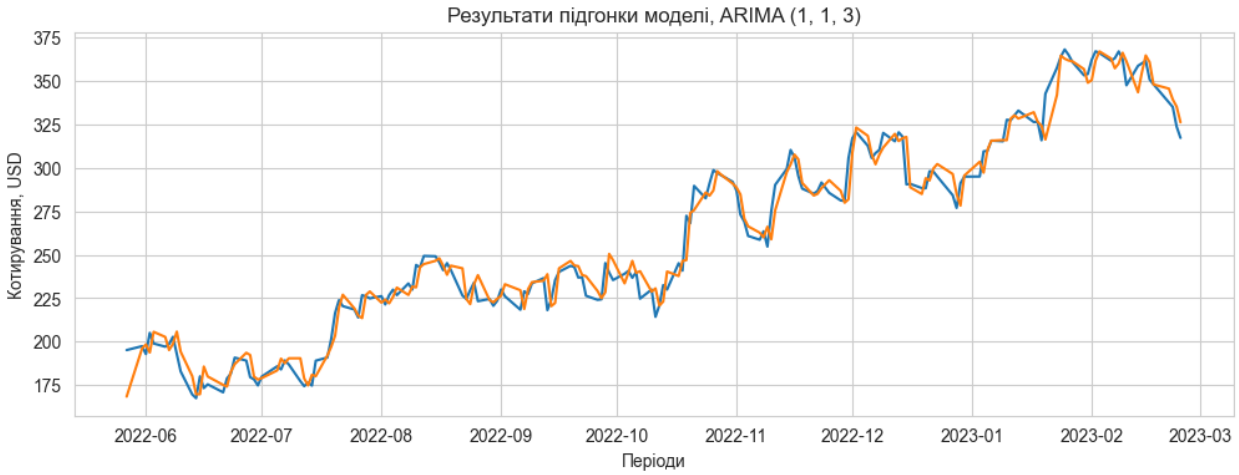


Рисунок 3.12 – Результати тренування (підгонки) моделі.

Тепер відобразимо результати прогнозування навченою моделлю вперед, кількість кроків прогнозування дорівнює виділеній частині тестових даних. Також відобразимо довірчі інтервали, які автоматично формуються предіктором.

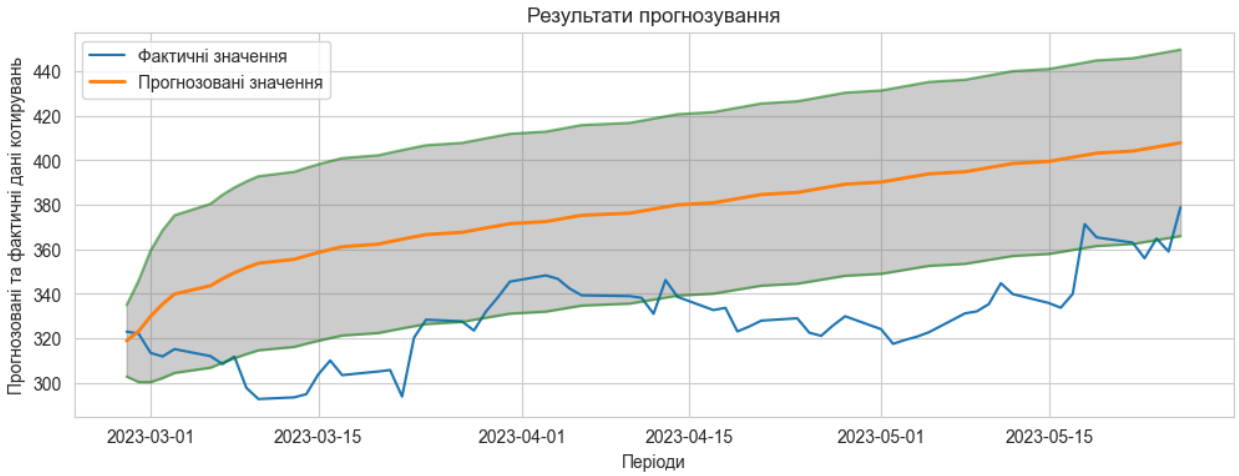


Рисунок 3.13 – Результати прогнозування ARIMA (1, 1, 3).

Як можна бачити, побудована та налаштована модель ARIMA (1,1,3) дозволяє передбачити загальний тренд, але не є досить хорошою при прогнозуванні окремих осциляцій навколо лінії тренду. Це пояснюється випадковим характером осциляцій, що не є пов’язаними з попередніми історичними даними.

ВИСНОВКИ

Задачею роботи було вивчення стохастичного процесу Орнштейна – Уленбека та його практичних застосувань у фінансовій математиці, зокрема для аналізу та прогнозування високочастотних фінансових даних.

В першому розділі роботи розглянуто теоретичні основи, зокрема вінерівські процеси, класичний процес Орнштейна – Уленбека та його типові застосування у фінансовій математиці. Одним з таких застосувань є моделі прогнозування вартості акцій Васічека. Окрім того розглянуто базові підходи до вивчення, аналізу та прогнозування часових рядів, зокрема принципи декомпозиції часових рядів та стохастичний підхід Бокса – Дженкінса до прогнозування часових рядів.

В другому розділі визначено конкретний математичний апарат та алгоритмізацію побудови моделі прогнозування часових рядів, що використовує процес Орнштейна – Уленбека для моделювання стохастичного компонента.

В третьому розділі розглядається практична реалізація моделі з використанням мови програмування Python – усі моделі, що використовують методологію Бокса – Дженкінса, потребують реалізації з використанням ПК.

Модель прогнозування часових рядів побудовано на базі рекомендацій для побудови типового оцінювача Scikit-learn. Це дозволяє з одного боку використовувати відомі спеціалістам Data Science інтерфейси, а з іншого боку, означає можливість використання побудованої моделі в більш складних ансамблевих моделях та конвеєрах Scikit-learn, а також застосовувати певний інструментарій Scikit – learn до побудованої моделі, зокрема моделі налаштування гіперпараметрів та оцінювачі моделі.

Якість побудованої моделі оцінено на тестовому датасеті з використанням метрики RMSE та інформаційного крітерія Акайке, що є часто використовуваним для такого типу задач. Також виконано графічне

порівняння з початковими даними та з результатами прогнозування моделі AutoARIMA (pmdarima).

На точність прогнозів обох моделей суттєво впливає стохастичний компонент часових рядів котирувань, що є відомим фактом. Його вплив набагато вище у порівнянні з прогнозування, наприклад, споживання певних товарів, послуг, енергоносіїв, тощо. Зокрема, деякі дослідники вважають коливання котирувань криптовалют випадковим блуканням на періодах більших за 5-10 днів (періодів).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Uhlenbeck G.E. and Ornstein L.S. On the Theory of Brownian Motion. Phys. Rev., 1930, том 36 – стор. 823-841.
2. Breiman L. Probability. SIAM, Philadelphia, USA, 1992 – 311 стор.
3. Vasicek. O. A. An equilibrium characterisation of the term structure. J. Fin. Econ., 1977, том 5 – стор. 177-188.
4. Oksendal B., Stochastic differential equations, Springer, 2000 - 332 стор.
5. Гихман И.И., Скороход А.В. Стохастические дифференциальные уравнения, Наукова думка, 1982 - 612с.
6. Oxford Dictionary of Statistics, Oxford University Press, 2002 - стор. 104
7. Kloeden, P.E. & Platen, E. Numerical Solution of Stochastic Differential Equations. Springer, Berlin. 1992 – 385 стор.
8. Бабак В.П., Марченко Б.Г., Фриз М.Є. Теорія ймовірностей, випадкові процеси та математична статистика, Техніка, 2004. – 287 стор.
9. Maruyama G. Continuous Markov processes and stochastic equations, Rendiconti del Circolo Matematico di Palermo Series 2. – том 4. Частина 1, 1955. – стор. 48.
10. Long H. Least squares estimator for discretely observed Ornstein–Uhlenbeck processes with small Lévy noises, Statistics & Probability Letters, том 79, частина 19 – стор. 2076–2085.
11. Kloeden P., Platen E. Numerical Solution of Stochastic Differential Equations, Springer-Verlag Berlin Heidelberg, 1995 – 668 стор.
12. Ross M., Gernot M., Alexander S., Mikosch T., Kreiß J.-P., Davis R.A., Andersen T.G. Ornstein–Uhlenbeck Processes and Extensions / Handbook of financial time series, Berlin: Springer Berlin, Heidelberg, 2009 – стор. 421–437.

13. Blackwell P. Ornstein–Uhlenbeck Process: Overview, 2014 - [Електронний ресурс] – Режим доступу до ресурсу: https://onlinelibrary.wiley.com/doi/10.1002/0470011815.b2a07038.
14. Kleptsyna M.L., Le Breton A. Statistical analysis of the fractional Ornstein-Uhlenbeck type process, Statistical Inference for Stochastic Processes – Berlin: Springer Berlin, Heidelberg, 2002. випуск 3, том 5 – стор. 229–248.
15. Програмування мовою Python / Васильєв О/ Богдан, 2020, 504 стор.: іл.
16. Документація Python [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/
17. Документація Scikit -learn [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/index.html
18. Документація Pmdarima [Електронний ресурс] – Режим доступу до ресурсу: http://alkaline-ml.com/pmdarima/
19. Документація Matplotlib [Електронний ресурс] – Режим доступу до ресурсу: https://matplotlib.org/stable/index.html
20. Документація Seaborn [Електронний ресурс] – Режим доступу до ресурсу: https://seaborn.pydata.org/

ДОДАТОК А. ЛІСТИНГ ПРОГРАМНОГО КОДУ.

research.ipynb

In [1]:

```
import os
import pathlib
from datetime import timedelta

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import pmdarima as pm
import seaborn as sns
import yfinance as yf
from fitter import Fitter
from pmdarima.arima import ndiffs
from sklearn.preprocessing import MinMaxScaler
from statsmodels.tsa.stattools import adfuller
import warnings

from predictor import Predictor
warnings.simplefilter('ignore')
```

Завантаження та попередній аналіз даних

In [2]:

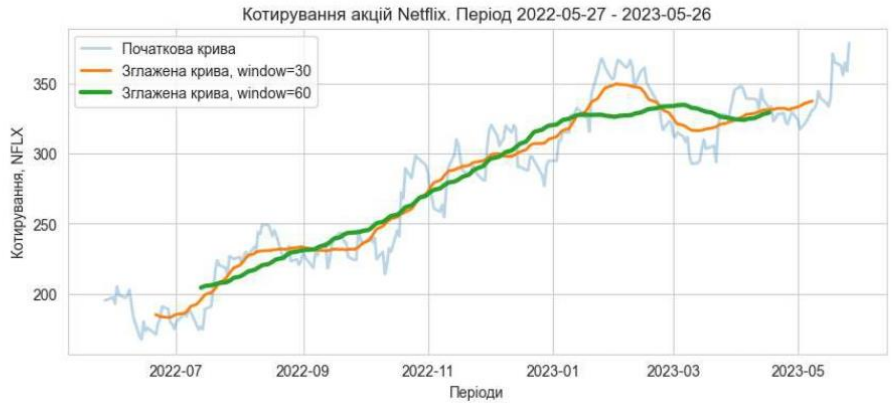
```
# завантаження даних, використовуючи Yahoo Finance API та yfinance
if not (pathlib.Path(os.getcwd()) / 'NFLX.pickle').exists():
    df = yf.download(tickers='NFLX', period = '1y', interval = '1d')
    df = df[['Close']].copy()
    df['date'] = df.index
    df.rename(columns={'Close': 'rate'}, inplace=True)
    df.reset_index(inplace=True, drop=True)
    df.date = df.date.apply(lambda x: x.date())
else:
    # читання раніше збережених даних
    df = pd.read_pickle('NFLX.pickle')

df.head(7)
```

Out[2]:

	rate	date
0	195.190002	2022-05-27
1	197.440002	2022-05-31
2	192.910004	2022-06-01
3	205.089996	2022-06-02
4	198.979996	2022-06-03
5	197.139999	2022-06-06
6	198.610001	2022-06-07

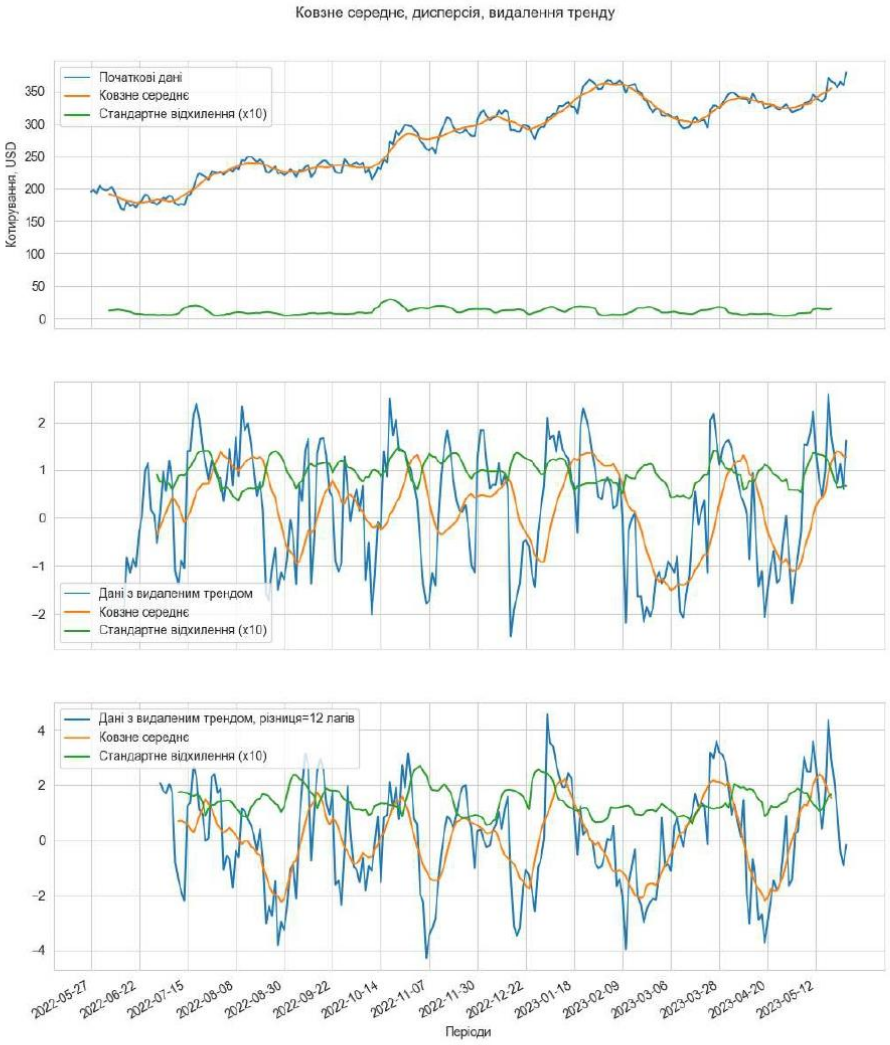
In [3]:

```
# графічна репрезентація початкових та згладжених даних
# з метою виділення тренду
WINDOW_SIZE_1, WINDOW_SIZE_2 = 30, 60
df[f'rol_rate_{WINDOW_SIZE_1}'] = df['rate'].rolling(window=WINDOW_SIZE_1,
                                                    center=True).mean()
df[f'rol_rate_{WINDOW_SIZE_2}'] = df['rate'].rolling(window=WINDOW_SIZE_2,
                                                    center=True).mean()
fig, ax = plt.subplots(figsize=(10, 4))
sns.lineplot(data=df, x='date', y='rate', ax=ax, alpha=.3, linewidth=2,
             label='Початкова крива')
sns.lineplot(data=df, x='date', y=f'rol_rate_{WINDOW_SIZE_1}',
             ax=ax, linewidth=2, label=f'Згладжена крива, window={WINDOW_SIZE_1}')
sns.lineplot(data=df, x='date', y=f'rol_rate_{WINDOW_SIZE_2}',
             ax=ax, linewidth=3, label=f'Згладжена крива, window={WINDOW_SIZE_2}')
ax.set_xlabel('Періоди')
ax.set_ylabel('Котирування, NFLX')
plt.title(f'Котирування акцій Netflix. Період {min(df.date)} - {max(df.date)}')
plt.show()
```



In [4]:

```
# видалення тренду, обчислення ковзного середнього
df['z_data'] = (df['rate'] - df['rate'].rolling(window=12).mean())
/ df['rate'].rolling(window=12).std()
df['zp_data'] = df['z_data'] - df['z_data'].shift(12)

def plot_rolling(df):
    fig, ax = plt.subplots(3,figsize=(12, 15), sharex=True)
    ax[0].plot(df.index, df['rate'], label='Початкові дані')
    ax[0].plot(df['rate'].rolling(window=12, center=True).mean(),
               label="Ковзне середнє")
    ax[0].plot(df['rate'].rolling(window=12, center=True).std(),
               label="Стандартне відхилення (x10)")
    ticks, labels = [], []
    for i in range(df.shape[0]):
        if i % 16 == 0:
            ticks.append(df.index[i])
            labels.append(df['date'][i])
    ax[0].set_xticks(ticks, labels)
    ax[0].set_ylabel('Котирування, USD')
    ax[0].legend()

    ax[1].plot(df.index, df.z_data, label="Дані з видаленим трендом")
    ax[1].plot(df.z_data.rolling(window=12).mean(),
               label="Ковзне середнє")
    ax[1].plot(df.z_data.rolling(window=12).std(),
               label="Стандартне відхилення (x10)")
    ax[1].legend()

    ax[2].plot(df.index, df.zp_data, label="Дані з видаленим трендом
               , різниця=12 лагів")
    ax[2].plot(df.zp_data.rolling(window=12, center=True).mean(),
               label="Ковзне середнє")
    ax[2].plot(df.zp_data.rolling(window=12, center=True).std(),
               label="Стандартне відхилення (x10)")
    ax[2].set_xlabel('Періоди')
    ax[2].legend()

    fig.autofmt_xdate()

plot_rolling(df)
plt.suptitle('Ковзне середнє, дисперсія, видалення тренду',
             y=0.92)
plt.show()
```


Ковзне середнє, дисперсія, видалення тренду
Початкові дані
Ковзне середнє
Стандартне відхилення (x10)
Котирування, USD
Дані з видаленим трендом
Ковзне середнє
Стандартне відхилення (x10)
Дані з видаленим трендом, різниця=12 лагів
Ковзне середнє
Стандартне відхилення (x10)
Періоди

In [5]:

```
# використання розширеного тесту Дікі - Фулера з метою визначення стаціонарності ряду,
# порядку різниць, які мають бути використані при подальшому аналізі та прогнозуванні

def show_adfuller_results(data: pd.Series, message: str) -> None:
    print(message)
    dftest = adfuller(data, autolag='AIC')
    print("Значення статистики тесту = {:.3f}".format(dftest[0]))
    print("P-значення = {:.3f}".format(dftest[1]))
    print("Критичні значення:")
    for k, v in dftest[4].items():
        print("\t{}: {:.3f} - Дані {} є стаціонарними з довірчою ймовірністю {}%".
              format(k, v, "не" if v<dftest[0] else "", 100-int(k[:-1])))

show_adfuller_results(df['rate'],
                      'Початкові дані')
show_adfuller_results(df.z_data.dropna(),
                      '\nДані з видаленим трендом')
show_adfuller_results(df.zp_data.dropna(),
                      '\nДані з видаленим трендом, різниця=12 лагів')
```

```
Початкові дані
Значення статистики тесту = -0.940
P-значення = 0.775
Критичні значення:
        1%: -3.457 - Дані не є стаціонарними з довірчою ймовірністю 99%
        5%: -2.873 - Дані не є стаціонарними з довірчою ймовірністю 95%
        10%: -2.573 - Дані не є стаціонарними з довірчою ймовірністю 90%

Дані з видаленим трендом
Значення статистики тесту = -5.710
P-значення = 0.000
Критичні значення:
        1%: -3.458 - Дані  є стаціонарними з довірчою ймовірністю 99%
        5%: -2.874 - Дані  є стаціонарними з довірчою ймовірністю 95%
        10%: -2.573 - Дані  є стаціонарними з довірчою ймовірністю 90%

Дані з видаленим трендом, різниця=12 лагів
Значення статистики тесту = -4.699
P-значення = 0.000
Критичні значення:
        1%: -3.461 - Дані  є стаціонарними з довірчою ймовірністю 99%
        5%: -2.875 - Дані  є стаціонарними з довірчою ймовірністю 95%
        10%: -2.574 - Дані  є стаціонарними з довірчою ймовірністю 90%
```

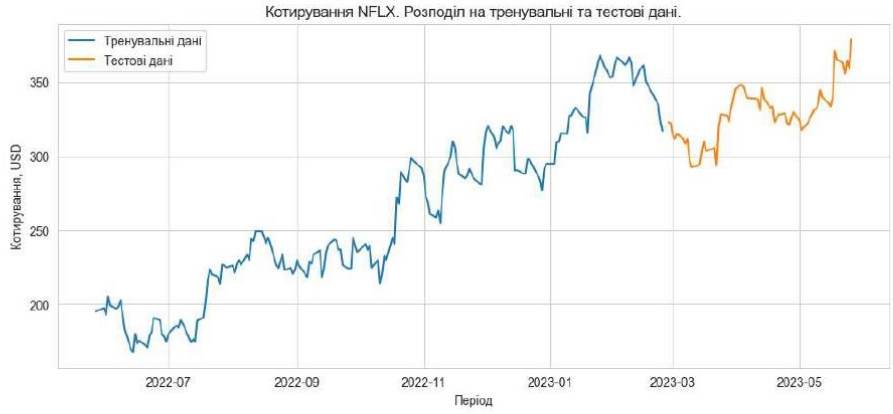
In [6]:

```
# визначення кінцевої дати тренувальної частини датасету
# з метою подальшого розподілу на тренувальну та тестову частину
train_end = max(df['date']) - timedelta(days=90)
print(f'Кінцева дата тренувального датасету {train_end}')
```

```
Кінцева дата тренувального датасету 2023-02-25
```

In [7]:

```
# поділ
df_train = df.loc[df.date <= train_end].copy()
df_test = df.loc[df.date > train_end].copy()

# візуалізація поділу
fig, ax = plt.subplots(figsize=(12, 5))
ax.plot(df_train['date'], df_train['rate'], label='Тренувальні дані')
ax.plot(df_test['date'], df_test['rate'], label='Тестові дані')
ax.legend()
ax.set_xlabel('Період')
ax.set_ylabel('Котирування, USD')
plt.title(f'Котирування NFLX. Розподіл на тренувальні та тестові дані.')
plt.show()
```



In [8]:

```
# виділення даних в масиви NumPy
x_train = df_train.date.values
x_test = df_test.date.values
y_train = df_train.rate.values
y_train_rol_first = df_train[f'rol_rate_{WINDOW_SIZE_1}'].values
y_train_rol_second = df_train[f'rol_rate_{WINDOW_SIZE_2}'].values
y_test = df_test.rate.values
y_test_rol_first = df_test[f'rol_rate_{WINDOW_SIZE_1}'].values
y_test_rol_second = df_test[f'rol_rate_{WINDOW_SIZE_2}'].values
```

In [9]:

```
# оцінка порядку різниці як максимального значення з результатів двох тестів:
# розширеного тесту Дікі - Фулера та KPSS

# тест Kwiatkowski-Phillips-Schmidt-Shin (KPSS)
kpss_diffs = ndiffs(y_train, alpha=0.05, test='kpss', max_d=6)

# augmented Dickey-Fuller (ррозширений тест Дікі-Фулера)
adf_diffs = ndiffs(y_train, alpha=0.05, test='adf', max_d=6)
n_diffs = max(adf_diffs, kpss_diffs)
print(f'Оцінка порядку різниці складає: {n_diffs}')
```

```
Оцінка порядку різниці складає: 1
```

In [10]:

```
# перетворення даних до діапазону [0; 1]
scl = MinMaxScaler()
y_train_transformed = scl.fit_transform(y_train.reshape(-1, 1)).ravel()
y_test_transformed = scl.transform(y_test.reshape(-1, 1)).ravel()
```

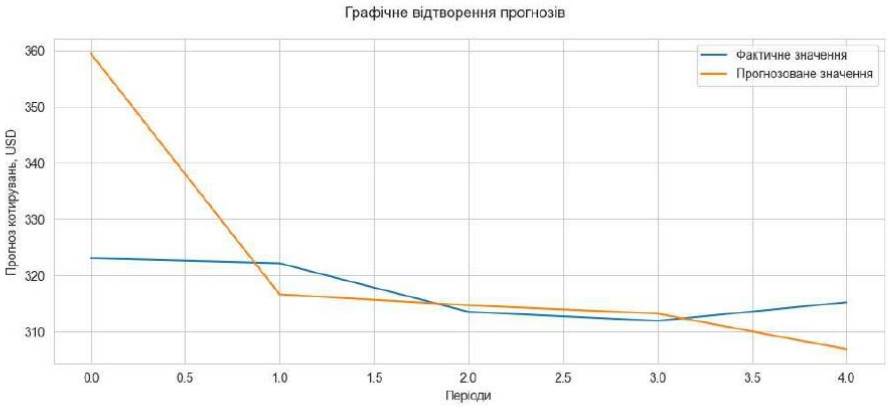
In [11]:

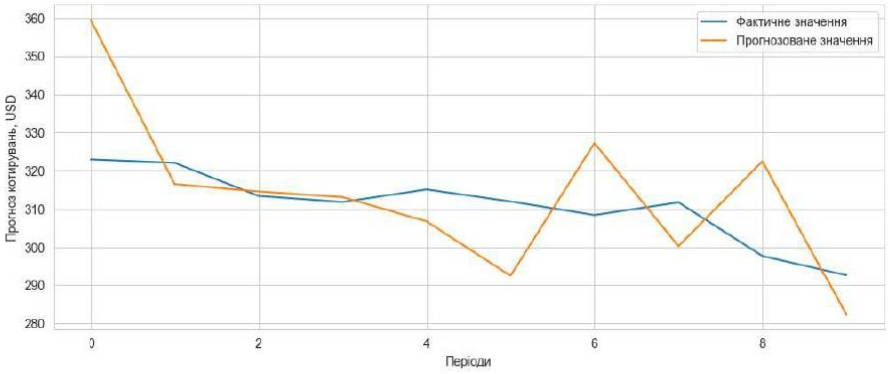
```
# тренування створеної моделі та отримання прогнозів з її використанням

# тренування моделі
pred = Predictor()
pred.fit(y_train_transformed)

# створення прогнозів на короткі терміни та на більший термін
# після отримання прогнозу відбувається зворотне перетворення
# з використанням раніше створеного екземпляра MinMaxScaler()
y_short_pred_5 = scl.inverse_transform(pred.predict(n=5).reshape(-1, 1)).ravel()
y_short_pred_10 = scl.inverse_transform(pred.predict(n=10).reshape(-1, 1)).ravel()
y_full_pred_64 = scl.inverse_transform(pred.predict(n=64).reshape(-1, 1)).ravel()
```

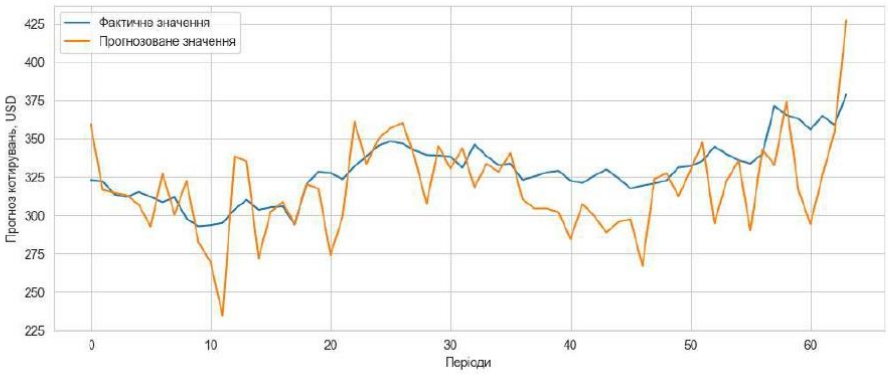
```
1: RMSE = 0.04149872179804473
2: RMSE = 0.04150346093644245
3: RMSE = 0.041414049375346186
4: RMSE = 0.04085975178201408
5: RMSE = 0.041252986521423014
6: RMSE = 0.04165128774486452
7: RMSE = 0.04143664479080143
8: RMSE = 0.04295889166246071
9: RMSE = 0.04465927739313896
10: RMSE = 0.04499728416621595
11: RMSE = 0.04437622142938924
12: RMSE = 0.04504825761113872
13: RMSE = 0.04476625550900025
14: RMSE = 0.04536893156235975
15: RMSE = 0.04573289366034226
16: RMSE = 0.04546554970069936
17: RMSE = 0.04535838481661004
18: RMSE = 0.04566538464515353
19: RMSE = 0.04538489580113604
20: RMSE = 0.04734878559644213
```

In [12]:

```
fig, ax = plt.subplots(3, 1, figsize=(12, 18))
sns.lineplot(x=list(range(5)), y=y_test[:5], ax=ax[0],
             label='Фактичне значення')
sns.lineplot(x=list(range(5)), y=y_short_pred_5, ax=ax[0],
             label='Прогнозоване значення')
ax[0].set_xlabel('Періоди')
ax[0].set_ylabel('Прогноз котирувань, USD')

sns.lineplot(x=list(range(10)), y=y_test[:10], ax=ax[1],
             label='Фактичне значення')
sns.lineplot(x=list(range(10)), y=y_short_pred_10, ax=ax[1],
             label='Прогнозоване значення')
ax[1].set_xlabel('Періоди')
ax[1].set_ylabel('Прогноз котирувань, USD')

sns.lineplot(x=list(range(len(y_test))), y=y_test, ax=ax[2],
             label='Фактичне значення')
sns.lineplot(x=list(range(len(y_test))), y=y_full_pred_64, ax=ax[2],
             label='Прогнозоване значення')
ax[2].set_xlabel('Періоди')
ax[2].set_ylabel('Прогноз котирувань, USD')
plt.suptitle('Графічне відтворення прогнозів', y=.9)
plt.subplots_adjust(hspace=.4)
plt.show()
```

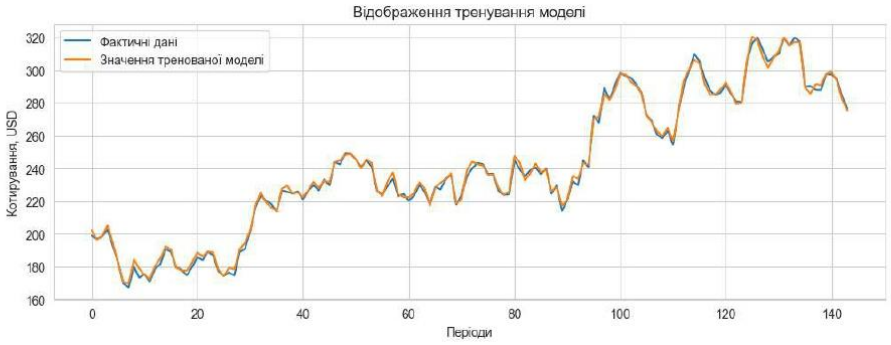

Графічне відтворення прогнозів
Фактичне значення
Прогнозоване значення
Прогноз котирувань, USD
Періоди


Фактичне значення
Прогнозоване значення
Прогноз котирувань, USD
Періоди


Фактичне значення
Прогнозоване значення
Прогноз котирувань, USD
Періоди

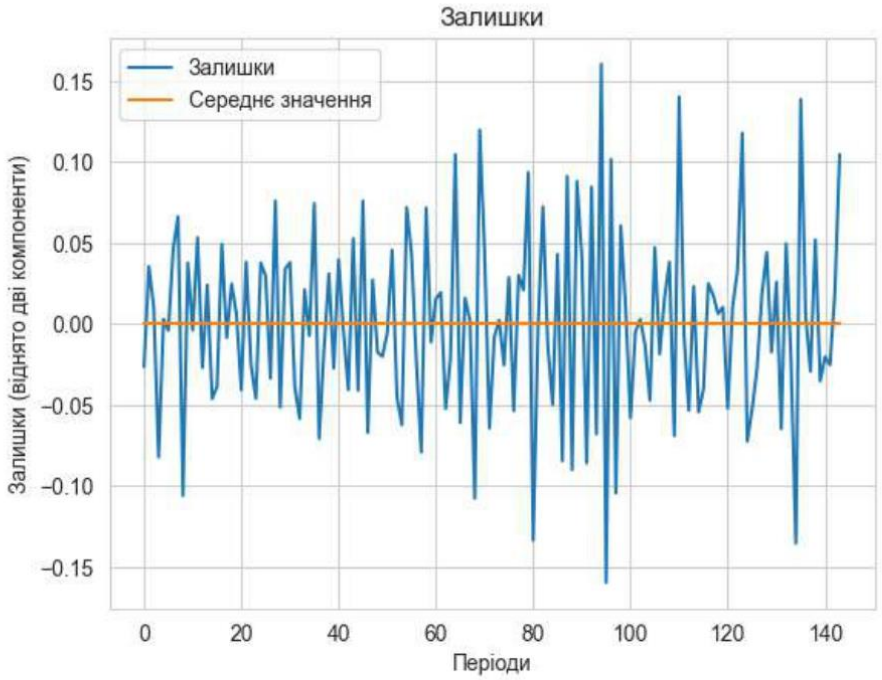
In [13]:

```
p, fitted_values = pred.fitted_values()
fitted_values = scl.inverse_transform(fitted_values.reshape(-1, 1)).ravel()

_, ax = plt.subplots(1, 1, figsize=(12, 4))
sns.lineplot(x=list(range(len(y_train[p: len(fitted_values) + p]))),
             y=y_train[p: len(fitted_values) + p],
             ax=ax, label='Фактичні дані')
sns.lineplot(x=list(range(len(fitted_values))),
             y=fitted_values,
             ax=ax, label='Значення тренованої моделі')
ax.set_title('Відображення тренування моделі')
ax.set_xlabel('Періоди')
ax.set_ylabel('Котирування, USD')
plt.show()
```



In [14]:

```
_, ax = plt.subplots()
sns.lineplot(x=list(range(pred.residuals.size)),
             y=pred.residuals, label='Залишки')
sns.lineplot(x=list(range(pred.residuals.size)),
             y=pred.residuals.mean(), label='Середнє значення')
ax.set_xlabel('Періоди')
ax.set_ylabel('Залишки (віднято дві компоненти)')
ax.set_title('Залишки')
plt.show()
```



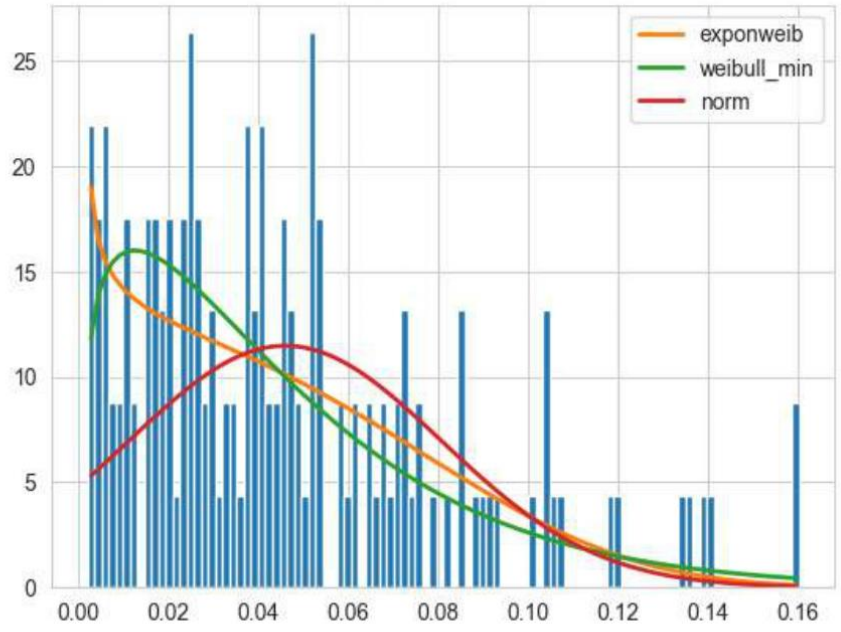
In [15]:

```
f = Fitter(np.abs(pred.residuals),
           distributions=['exponweib', 'weibull_min', 'norm'])
f.fit()
f.summary()
```

```
Fitting 3 distributions: 100%|██████████| 3/3 [00:00<00:00, 13.28it/s]
```

Out[15]:

	sumsquare_error	aic	bic	kl_div	ks_statistic	ks_pvalue
exponweib	2433.544065	-242.948285	427.009105	inf	0.069916	0.461499
weibull_min	2560.249653	-258.657019	429.348172	inf	0.063278	0.589041
norm	3389.033462	-209.283781	464.761718	inf	0.115206	0.040285



In [16]:

```
### Порівняння результатів прогнозування з прогнозом моделі ARIMA
```

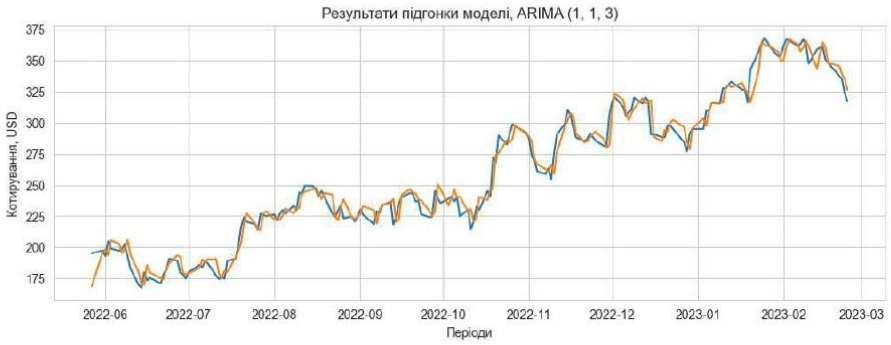
In [17]:

```
auto = pm.auto_arima(scl.fit_transform(y_train.reshape(-1, 1)).ravel(),
                                        # тренувальний набір даних
                     d=1,               # визначений порядок різниць
                     seasonal=False,    # сезонність
                     stepwise=False,    # використання алгоритму Hyndman and Khand
                                        # параметрів моделі
                     suppress_warnings=True,  # відключення попереджень
                     error_action="ignore",   # дія при виникненні помилок
                     max_p=15,          # максимальне значення p при налаштуваннях
                     max_order=30,      # максимальне значення p+q+P+Q при налашту
                     trace=True,        # виведення логу
                     n_jobs=6,          # кількість потоків
                     n_fits=10          # кількість ітерацій для одного набору (p,
                     )
```

```
Best model:  ARIMA(1,1,3)(0,0,0)[0] intercept
Total fit time: 20.893 seconds
```

In [18]:

```
_, ax = plt.subplots(1, 1, figsize=(12, 4))
sns.lineplot(x=x_train, y=y_train, ax=ax)
sns.lineplot(x=x_train, y=scl.inverse_transform(
    auto.fittedvalues().reshape(-1, 1)).ravel(), ax=ax)
ax.set_xlabel('Періоди')
ax.set_ylabel('Котирування, USD')
ax.set_title('Результати підгонки моделі, ARIMA (1, 1, 3)')
plt.show()
```



In [19]:

```
# прогнозування з використанням ARIMA на 64 кроки вперед
y_pred, conf_int = auto.predict(64, return_conf_int=True)
```

In [20]:

```
# Графічне відтворення прогнозу, зробленого з використанням отриманої моделі ARIMA
a_, ax = plt.subplots(1, 1, figsize=(12, 4))
sns.lineplot(x=x_test, y=y_test, ax=ax, label='Фактичні значення')
sns.lineplot(x=x_test, y=scl.inverse_transform(y_pred.reshape(-1, 1)).ravel(),
             ax=ax, linewidth=2,label='Прогнозовані значення')
sns.lineplot(x=x_test,
             y=scl.inverse_transform(conf_int[:, 0].reshape(-1, 1)).ravel(),
             ax=ax, alpha=.5, color='green')
c = sns.lineplot(x=x_test,
                 y=scl.inverse_transform(conf_int[:, 1].reshape(-1, 1)).ravel(),
                 ax=ax, alpha=.5, color='green')
line = c.get_lines()
plt.fill_between(line[0].get_xdata(), line[2].get_ydata(),
                 line[3].get_ydata(), color='gray', alpha=.4)
ax.set_title('Результати прогнозування')
ax.set_xlabel('Періоди')
ax.set_ylabel('Прогнозовані та фактичні дані котирувань')
plt.show()
```

predictor.py

```
from typing import Tuple, Dict

import numpy as np
import pandas as pd
from numpy.random import default_rng
from numpy.typing import NDArray
from scipy.stats import linregress
from sklearn.base import BaseEstimator
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score


class Predictor(BaseEstimator):

    def __init__(self, seed: int = -1) -> None:
        self._y_train = None
        self._y_pred = None

        self.p = 0
        self.beta = 0
        self.sigma = 0
        self._residuals = None
        self.best_train_df = None
        self.df = None

        self.generator = default_rng(seed) if seed > -1 else
default_rng()

    @property
    def fitted_values_(self) -> Tuple[int, NDArray]:
        """
        Return autoregression shift and fitted during the
train process values
        :return:
        """
        return                                          self.p,
self.best_train_df['Predicted_Values'].values

    @property
    def parameters(self) -> Dict:
        """
        Returns the parameters, which have been defined
during fit()
        :return:
        """
        return {'beta': self.beta, 'sigma': self.sigma}
```

```
    @property
    def residuals_(self) -> NDArray:
        """
        Return residuals after the trend and drift component
deduction
        :return:
        """
        return self._residuals

    def fit(self, y_train: NDArray) -> NDArray:
        """
        Fits the estimator
        :param y_train: time-series values to train on
        :return: None
        """
        self._y_train = y_train
        self._reset()

        df = pd.DataFrame(y_train, columns=['Value'])
        trained_df = []
        best_p = 1
        RMSE_best = 1e8
        for i in range(1, 21):
            df_train_shift, df_test, theta, intercept, RMSE
= self._fit_trend(i, df)
            trained_df.append(df_train_shift)
            print(f"{i}: RMSE = {RMSE}")
            if RMSE < RMSE_best:
                RMSE_best = RMSE
                best_p = i

        self.best_train_df = trained_df[best_p]
        self.p = best_p

        res = pd.DataFrame()
        res['Residuals']  =  self.best_train_df['Value']  -
self.best_train_df.Predicted_Values
        lr_results = linregress(y_train[:-1], y_train[1:])
        beta = lr_results.slope
        drifting = np.hstack(
            (np.zeros(1),
             beta * (self.best_train_df['Value'].values[: -
1] - self.best_train_df['Predicted_Values'].values[: - 1])))

        self._residuals     =     (res['Residuals']     -
drifting).values
```

```
        self.sigma = np.stdf(self._residuals)

    def predict(self, n: int = 10) -> NDArray:
        """
        Makes prediction
        :param n: number of moments to predict the values for
        :return: the NumPy array of predictions
        """
        predicted_values = []
        values_vector = self._y_train[:-self.p]
        for i in range(n):
            value = values_vector[i] * (1 + self.beta) +
self.sigma * self.generator.weibull(1)
            predicted_values.append(value)
        return np.array(values_vector[self.p:])

    def _reset(self) -> None:
        """
        Resets the fitted values
        :return: None
        """
        self.p = 0
        self.beta = 0
        self._residuals = None
        self.best_train_df = None
        self.df = None

    @staticmethod
    def score(y_true: NDArray, y_predict: NDArray) -> float:
        """
        Scores the prediction using R^2 score
        :param y_true: true values
        :param y_predict: predicted values
        :return: the r2 score value
        """
        return r2_score(y_true, y_predict)

    @staticmethod
    def _fit_trend(p, df) -> Tuple[pd.DataFrame,
pd.DataFrame, float, float, float]:
        """
        Internal methods which defines the trend of the rime
series
        :param p: the shift
        :param df: the dataframe with the values
        :return: processed dataframes and the appropriate
parameters
```

```
        """
        df_temp = df.copy()

        # Generating the lagged p terms
        for i in range(1, p + 1):
            df_temp['Shifted_values_%d' % i] =
df_temp['Value'].shift(i)

        train_size = int(0.8 * df_temp.shape[0])

        # Breaking data set into test and training
        df_train = pd.DataFrame(df_temp[0:train_size])
        df_test =
pd.DataFrame(df_temp[train_size:df.shape[0]])

        df_train_2 = df_train.dropna()
        # X contains the lagged values ,hence we skip the
first column
        X_train = df_train_2.iloc[:, 1:].values.reshape(-1,
p)
        # Y contains the value,it is the first column
        y_train = df_train_2.iloc[:, 0].values.reshape(-1,
1)

        # Running linear regression to generate the
coefficients of lagged terms
        lr = LinearRegression()
        lr.fit(X_train, y_train)

        theta = lr.coef_.T
        intercept = lr.intercept_
        df_train_2['Predicted_Values'] =
X_train.dot(lr.coef_.T) + lr.intercept_
        # df_train_2[['Value','Predicted_Values']].plot()

        X_test = df_test.iloc[:, 1:].values.reshape(-1, p)
        df_test['Predicted_Values'] = X_test.dot(lr.coef_.T)
+ lr.intercept_

        return df_train_2, df_test, theta, intercept, \
            np.sqrt(mean_squared_error(df_test['Value'],
df_test['Predicted_Values']))
```