МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Розроблення програмного забезпечення для автоматизації управління особистими фінансовими процесами

Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки

Виконав: студент групи МгЗІТ1-23 Валерій МИКИТЕНКО
Науковий керівник: к.т.н., доц. Тетяна ДЕМКІВСЬКА
Рецензент: д.т.н., проф. Віктор Чупринка

Київ 2024

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет / Інститут	Мехатроніки та комп’ютерних технологій
Кафедра	Комп’ютерних наук
Рівень вищої освіти	Другий (магістерський)
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
_____ Наталія ЧУПРИНКА
(підпис)
«_____» _____ 2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Микитенко Валерію Анатолійовичу

1. Тема кваліфікаційної роботи: Розроблення програмного забезпечення для автоматизації управління особистими фінансовими процесами

Науковий керівник роботи Демківська Тетяна Іванівна

затверджені наказом КНУТД від “ 03 ” 09 2024 року № 188-уч.

2. Вихідні дані до кваліфікаційної роботи Розробки кафедри комп’ютерних наук, рекомендована література.

3. Зміст кваліфікаційної роботи (перелік питань, які потрібно опрацювати) 1) Аналіз часових рядів, 2) Розроблення алгоритмічного забезпечення, 3) Розроблення програмного забезпечення

4. Дата видачі завдання 08.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	05.10.2024	
2	Розділ 1 (Аналіз часових рядів)	05.10.2024	
3	Розділ 2 (Розроблення алгоритмічного забезпечення)	05.10.2024	
4	Розділ 3 (Розроблення програмного забезпечення)	25.10.2024	
5	Висновки	25.10.2024	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	30.10.2024	
7	Подача кваліфікацйної роботи науковому керівнику для відгуку (за 14 днів до захисту)	4.11.2024	
8	Подача кваліфікаційної роботи для рецензування (за 12 днів до захисту)	20.11.2024	
9	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	23.11.2024	
10	Подання кваліфікаційної роботи завідувачу кафедри (за 7 днів до захисту)	28.11.2024	

Студент _____ Валерій Микитенко
(підпис) (Власне ім'я та ПРІЗВИЩЕ)

Науковий керівник _____ Тетяна ДЕМКІВСЬКА
(підпис) (Власне ім'я та ПРІЗВИЩЕ)

АНОТАЦІЯ

Микитенко Валерій Анатолійович. Кваліфікаційна робота за спеціальністю 122 – «Комп’ютерні науки». – Київський національний університет технологій та дизайну, Київ, 2024 рік.

У кваліфікаційній роботі розроблено алгоритм та програмне забезпечення для прогнозування особистих фінансів користувачів. Алгоритм базується на аналізі часових рядів із використанням адитивної та мультиплікативної моделей. Реалізовано процес вирівнювання даних за допомогою методу ковзної середньої, розрахунок трендових і сезонних компонент, а також побудову прогнозу для майбутніх періодів.

У ході роботи було створено програму, яка виконує автоматизований аналіз даних та генерує прогнози у вигляді графіків і таблиць. Програма включає функціонал для візуалізації результатів (графіки адитивної та мультиплікативної моделей) і експорту даних у формат Excel.

Розроблене програмне забезпечення використовує мову програмування Python та бібліотеки Pandas, NumPy, Matplotlib. Воно забезпечує зручність інтеграції у системи, орієнтовані на фінансовий менеджмент, і може бути використане у Telegram-боті для прогнозування особистих доходів і витрат.

Ключові слова: часові ряди, метод найменших квадратів, модель ковзного середнього, тренд, сезонність, прогнозування, Python, фінансовий аналіз.

ABSTRACT

Mykytenko Valerii Anatoliiovych. Qualification thesis for the specialty 122 – "Computer Science". – Kyiv National University of Technologies and Design, Kyiv, 2024.

This qualification thesis presents the development of an algorithm and software for forecasting users' personal finances. The algorithm is based on time series analysis using additive and multiplicative models. The process includes data smoothing through the moving average method, calculation of trend and seasonal components, and forecasting for future periods.

During the research, a program was developed to perform automated data analysis and generate forecasts in the form of graphs and tables. The program features functionality for visualizing results (graphs for additive and multiplicative models) and exporting data to Excel format.

The developed software utilizes the Python programming language and libraries such as Pandas, NumPy, and Matplotlib. It ensures seamless integration into systems focused on financial management and can be implemented in a Telegram bot for forecasting personal income and expenses.

Keywords: time series, least squares method, moving average model, trend, seasonality, forecasting, Python, financial analysis.

ЗМІСТ

ВСТУП 7
РОЗДІЛ 1. АНАЛІЗ ЧАСОВИХ РЯДІВ 8
1.1. Ключові поняття часових рядів 8
1.2. Методи дослідження часових рядів 14
1.3. Основні підходи до прогнозування фінансових ризиків 17
1.4. Висновки до першого розділу 21
РОЗДІЛ 2. РОЗРОБЛЕННЯ АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ 22
2.1. Адитивні і мультиплікативні моделі для аналізу часових рядів 22
2.2. Розроблення алгоритму 24
2.3. Реалізація алгоритму з реальними даними 27
2.4. Висновки до другого розділу 37
РОЗДІЛ 3. ВИБІР СЕРЕДОВИЩА ТА СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ. 38
3.1. Вибір платформи, середовища та мови програмування для реалізації додатку 38
3.2. Програмна реалізація роботи з чат-ботом 47
3.3. Налаштування аналітичної частини чат-бота 56
3.4. Висновки до третього розділу 61
ВИСНОВКИ 63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 64

ВСТУП

Актуальність роботи. Проблема ефективного управління особистими фінансами є важливою для кожного користувача в умовах сучасного економічного середовища. Відсутність інструментів для зручного прогнозування доходів і витрат, які враховують індивідуальні особливості, створює потребу в автоматизованих рішеннях. Розроблення системи, яка використовує сучасні методи аналізу часових рядів для створення точних прогнозів, є актуальною задачею.

Мета дослідження. Розробити алгоритм і програмне забезпечення для автоматизованого прогнозування особистих фінансів користувачів, що базується на адитивних і мультиплікативних моделях аналізу часових рядів.

Об'єкт дослідження. Процес аналізу часових рядів для фінансового прогнозування.

Предмет дослідження. Часові рядіи, побудова моделей для прогнозування особистих фінансових процесів.

Методи дослідження. Дослідження базується на аналізі часових рядів, моделювання, метод Крамера для визначення тренду та прогнозуванні на основі адитивних і мультиплікативних моделей.

Елементи наукової новизни. Запропоновано комплексний підхід до прогнозування особистих фінансів користувачів, який поєднує методи аналізу часових рядів із можливістю інтеграції в Telegram-бот. Реалізовано автоматизоване створення фінансових прогнозів з урахуванням сезонних і трендових компонент.

Практична значущість роботи. Результати дослідження дозволяють користувачам отримувати прогноз прибутків у зручному форматі. Програмне забезпечення може використовуватись для персонального фінансового менеджменту, сприяючи кращому плануванню бюджету. Інтеграція з Telegram-ботом забезпечує доступність та простоту у використанні.

Апробація результатів роботи. Результати дослідження були протестовані і перевірені на отриманих даних користувача.

РОЗДІЛ 1. АНАЛІЗ ЧАСОВИХ РЯДІВ

1.1. Ключові поняття часових рядів

Часовий ряд є одним із ключових інструментів у дослідженнях даних, які залежать від часу. *Простими словами, це впорядкована послідовність даних, зібраних через однакові проміжки часу.* Наприклад, уявімо, що щодня ви записуєте температуру повітря. Кожен такий запис відповідає певному дню, а їхня послідовність утворює часовий ряд. Аналогічно, якщо ми збираємо щомісячні дані про рівень безробіття в країні або щорічні показники економічного зростання, ці дані теж формують часові ряди. Їхня структура дозволяє виявляти закономірності та тенденції, прогнозувати майбутні значення або оцінювати вплив певних факторів на поведінку системи.

Часові ряди є важливими у багатьох галузях. В *економіці* вони допомагають аналізувати фінансові ринки, у *медицині* — відстежувати зміни стану пацієнтів, в *екології* — оцінювати зміни клімату, а в *інженерії* — контролювати процеси виробництва. Наприклад, серія даних про зміну ціни акцій упродовж року дозволяє трейдерам оцінювати ризики та приймати зважені рішення при роботі з цінними активами. А в екологічних дослідженнях часовий ряд може представляти щоденні вимірювання концентрації CO_2 в атмосфері, що допомагає виявляти тренди глобального потепління.

Часові ряди **не є однорідними** за своєю природою. Вони можуть значно відрізнятися залежно від характеристик даних, які ми аналізуємо. Одним із ключових аспектів їх вивчення є розподіл часових рядів на *стаціонарні та нестаціонарні.*

Стаціонарні часові ряди мають властивість *стабільності* у своїх статистичних характеристиках. Наприклад, якщо ми спостерігаємо за щоденною температурою протягом кількох місяців і виявляємо, що середня температура стабільно тримається на рівні 15°C, це вказує на стаціонарність. Так само, якщо коливання температури відносно середнього значення залишаються стабільними, ми можемо говорити про сталу дисперсію.

Дисперсія є важливим параметром у дослідженні стаціонарних рядів, адже вона дозволяє оцінити, наскільки дані відхиляються від середнього значення.

Наприклад, якщо дані про добовий обсяг продажів показують незначні коливання навколо постійного середнього, то такий ряд є стаціонарним. Приклад дисперсії наведено на рис. 1.1.

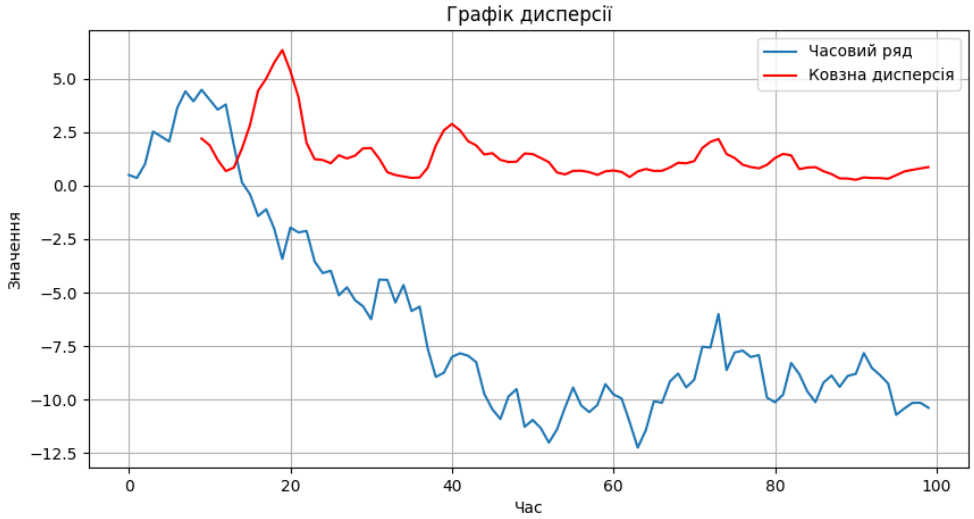


Рисунок. 1.1 – Дисперсія

Важливо зазначити, що стаціонарність є ключовою умовою для застосування багатьох методів математичного аналізу. Наприклад, моделі **ARIMA (авторегресійні інтегровані моделі ковзного середнього)** розроблені з припущенням, що часовий ряд є стаціонарним. Якщо ряд не відповідає цій умові, його потрібно перетворити, наприклад, за допомогою обчислення різниць між значеннями або логарифмічного перетворення.

На відміну від стаціонарних, **нестаціонарні часові ряди** характеризуються *змінністю* своїх властивостей у часі. Це означає, що середнє значення, дисперсія або автокореляція *змінюються залежно від періоду спостереження*. Наприклад, ціна акцій певної компанії може різко зростати під час економічного буму і знижуватися під час кризи. У такому випадку середнє значення цін змінюється, що вказує на нестаціонарність.

Змінність дисперсії також є ознакою нестаціонарності. Якщо в один період часу ціни на акції сильно коливаються, а в інший період залишаються майже незмінними, це свідчить про зміну ступеня волатильності. Аналогічно, автокореляція може змінюватися з часом: у певні моменти ціна акцій може сильно залежати від попередньої ціни, а в інші — цей зв’язок може слабшати або взагалі зникати.

Аналіз нестаціонарних рядів часто ускладнюється через необхідність їхньої попередньої трансформації. Наприклад, для прогнозування трендів або циклічних змін необхідно усунути випадкові коливання або визначити довгострокову тенденцію.

Тренд є важливим компонентом часових рядів, який відображає довгострокову тенденцію зміни значень. Уявіть, що ми аналізуємо дані про річні доходи компанії за останнє десятиліття. Якщо доходи постійно зростають, це вказує на висхідний тренд. Графік такого ряду буде виглядати як крива або пряма лінія, що піднімається вгору.

Інший приклад тренду можна знайти в екологічних дослідженнях. Наприклад, зменшення кількості льодовиків упродовж останніх п’ятдесяти років є прикладом низхідного тренду. Аналіз трендів є важливим, оскільки він дозволяє не лише описати загальні зміни в даних, але й спрогнозувати майбутні значення.

Існує багато **методів для виявлення тренду**, серед яких виділяються *візуальний аналіз, ковзне середнє, лінійна та поліноміальна регресія*. Візуалізація є найпростішим способом виявлення тренду: достатньо побудувати графік часових рядів і оцінити загальний напрямок. Наприклад, у продажах велосипедів за останнє десятиліття графік може показати загальне зростання, якщо тренд є висхідним.

Ковзне середнє дозволяє згладжувати короткострокові коливання, що може бути корисним у виявленні довгострокових тенденцій. Уявімо, що ми аналізуємо щоденні продажі у супермаркеті. Коливання можуть залежати від сезонності, вихідних або святкових днів. Використовуючи ковзне середнє, ми можемо побачити загальну тенденцію зростання або зменшення продажів без впливу короткострокових змін.

Методи регресії, *зокрема лінійна регресія*, допомагають кількісно оцінити тренд. Наприклад, якщо у нас є дані про середньорічну температуру за останні 50 років, ми можемо побудувати трендову лінію, яка покаже загальне зростання температури, пов’язане з глобальним потеплінням.

Часові ряди є потужним інструментом для аналізу даних, що змінюються в часі. Розуміння їхньої структури та властивостей дозволяє робити точні

прогнози, виявляти закономірності та розкривати вплив різних факторів. Аналіз трендів є ключовим етапом у цьому процесі, адже він допомагає не лише пояснити зміни в минулому, а й передбачити майбутнє.

Сезонність – це систематичні повторювані зміни у часовому ряді, які зумовлені певними природними, соціальними або економічними процесами. Такі зміни завжди пов’язані з конкретними часовими проміжками: вони можуть спостерігатися щодня, щомісяця, щокварталу або щороку. Завдяки сезонності можна пояснити закономірності поведінки системи, які виникають через зміну погоди, святкові дні, соціальні традиції чи бізнес-практики.

Наприклад, якщо ви коли-небудь звертали увагу на зростання продажів подарункових наборів у грудні, це явище демонструє щорічну сезонність, зумовлену підготовкою до зимових свят. Інший приклад — підвищення попиту на кондиціонери влітку, що пов’язане з підвищенням температури.

Сезонність має регулярний характер, що робить її легко передбачуваною. Ця характеристика дозволяє враховувати її у прогнозах і ефективніше планувати ресурси. Наприклад, фермери враховують сезонність погоди для планування посівних робіт, а авіакомпанії коригують кількість рейсів відповідно до сезонних піків попиту. Приклад сезонності ми можемо побачити на рис. 1.2.

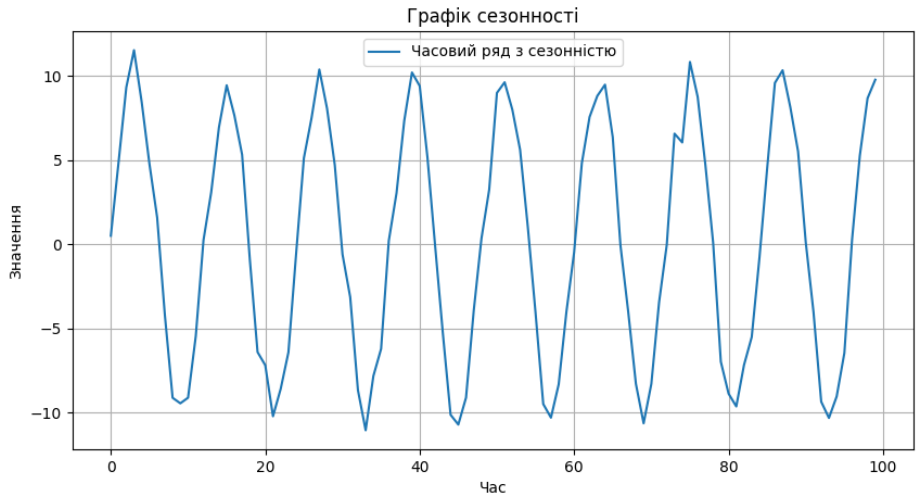


Рисунок. 1.2 – Графік сезонності

Сезонність не просто визначає характер змін у даних, але й дає змогу виділити їх окремо від інших складових часового ряду. Ці повторювані закономірності можна розділити на кілька типів залежно від частоти:

- Щоденна сезонність: Наприклад, збільшення активності у соціальних мережах у вечірній час доби.
- Місячна сезонність: Як правило, вона проявляється у фінансах — наприклад, виплати зарплат в останній тиждень кожного місяця можуть спричиняти підвищення споживчих витрат.
- Щорічна сезонність: Це найбільш поширений тип, як у випадку зі зростанням продажів ялинкових прикрас у грудні.

Для виявлення сезонності використовуються різноманітні підходи, які дозволяють краще зрозуміти її природу та вплив:

1. Графічний аналіз: Один із найпростіших і водночас найефективніших методів. Наприклад, графік продажів морозива за кілька років дозволить чітко побачити пік продажів улітку.
2. Автокореляція: Цей метод визначає, наскільки спостереження у часовому ряді взаємопов’язані через певні інтервали часу. Якщо автокореляція висока через однакові інтервали (наприклад, через 12 місяців), це свідчить про наявність сезонності.
3. Декомпозиція: Поділ часового ряду на трендову, сезонну та залишкову складові. Метод STL, наприклад, дозволяє візуалізувати сезонність окремо, що допомагає аналізувати її без впливу інших факторів.
4. Спектральний аналіз: Цей підхід дозволяє ідентифікувати частоти, які мають найбільший вплив на дані. Наприклад, аналіз енергоспоживання може виявити щоденну сезонність, пов’язану з піковими годинами використання електрики.

Розглянемо дані про щомісячні продажі морозива у місті за 5 років. Побудова графіка допомагає візуально визначити, що найбільші продажі спостерігаються влітку, тоді як узимку вони суттєво знижуються. Це явище повторюється щороку, демонструючи щорічну сезонність.

Щоб більш точно вивчити сезонність, можна використати метод декомпозиції. Наприклад, застосувавши STL, ми отримуємо три компоненти:

- Тренд, який показує загальну тенденцію до зростання продажів.

- Сезонну складову, що виявляє повторювані піки влітку та спади взимку.
- Залишкову складову, яка містить непередбачувані коливання, зумовлені випадковими подіями.

Циклічність, на відміну від сезонності, не має регулярного графіка повторення. Вона зазвичай виникає внаслідок змін у макроекономічних умовах, таких як періоди економічного підйому чи спаду. Наприклад, зростання економіки сприяє збільшенню доходів і попиту на товари, тоді як під час кризи ситуація може змінитися на протилежну.

Циклічність часто має довготривалий характер і залежить від зовнішніх факторів. Наприклад, будівельна галузь може переживати підйом у періоди економічного зростання, коли збільшується кількість нових проєктів, але значно сповільнюватися під час рецесії через зниження інвестицій. Приклад циклічності ми можемо побачити на рис. 1.3.

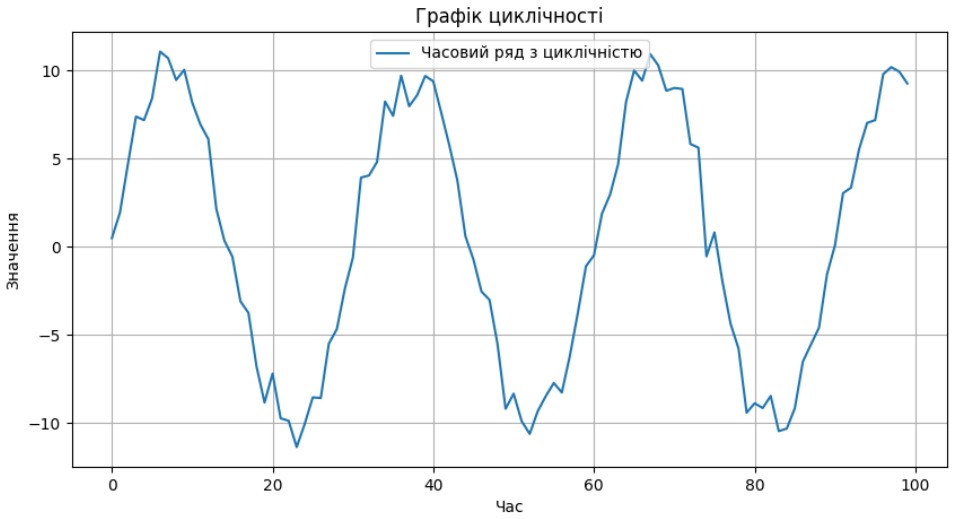


Рисунок. 1.3 – Графік циклічності

Ключова відмінність між сезонністю та циклічністю полягає у регулярності змін. Сезонність завжди повторюється через фіксовані інтервали часу, тоді як циклічність залежить від економічних умов і може мати непередбачувану тривалість.

Наприклад, при продажі велосипедів різко зростають кожного літа, це сезонність або якщо протягом кількох років спостерігається чергування періодів

активного зростання і спаду продажів через зміну економічних умов, це циклічність.

Аналіз сезонності та циклічності допомагає краще зрозуміти закономірності у даних і приймати обґрунтовані рішення. Наприклад, бізнес може адаптувати свої стратегії, враховуючи очікувані сезонні зміни, а довгострокове планування з урахуванням циклічності дозволяє підготуватися до можливих економічних ризиків.

Таким чином, сезонність і циклічність є важливими аспектами аналізу часових рядів. Розуміння цих закономірностей дозволяє не лише пояснити змінність даних, але й створювати прогнози, що враховують природні, соціальні та економічні фактори.

1.2. Методи дослідження часових рядів

Аналіз часових рядів є ключовим аспектом у багатьох галузях, включаючи економіку, фінанси, інженерію та наукові дослідження. Для дослідження часових рядів використовують різні методи, що дозволяють розкрити їхні особливості, структуру та створювати прогнози. Серед основних методів виділяють кореляційний, спектральний аналіз та вивчення автокореляційної функції. Кожен з цих підходів має свої унікальні характеристики та сфери застосування.

Кореляційний аналіз є одним із фундаментальних інструментів аналізу часових рядів, що дозволяє визначати взаємозв'язки між двома рядами або значеннями одного ряду на різних лагах. Одним із основних інструментів цього підходу є автокореляційна функція (ACF), яка аналізує зв'язок між значеннями в ряді на різних відстанях. Наприклад, для оцінки впливу температури вчора на сьогоднішні показники використовують ACF. Високі значення автокореляції на певних лагах можуть свідчити про сильну залежність, тренд або наявність сезонних закономірностей.

Частковий автокореляційний аналіз (PACF) розглядає прямий вплив одного значення на інше, виключаючи ефекти проміжних значень. Наприклад, щоб оцінити, як ціна акцій через два дні залежить від сьогоднішньої, ігноруючи вплив завтрашньої, застосовують PACF. Цей метод широко використовується для побудови моделей, таких як ARIMA.

Для ілюстрації розглянемо аналіз веб-відвідувань. Якщо ACF демонструє високі значення на лагах, що відповідають одному тижню, це може свідчити про тижневі сезонні цикли. PACF допоможе визначити, наскільки поточна відвідуваність залежить від попередніх днів, що корисно для прогнозування.

Спектральний аналіз дозволяє вивчати частотні складові часових рядів. Завдяки цьому методу можна визначити частоти, на яких спостерігаються найбільші коливання, що важливо для виявлення циклічних компонентів, які не завжди очевидні при візуальному аналізі. Спектральний аналіз використовується для дослідження періодичності, наприклад, щорічних циклів в економіці або сезонних змін у продажах.

Прикладом може бути аналіз даних про споживання електроенергії. Використовуючи спектральний аналіз, можна встановити наявність денних і тижневих циклів, які зумовлені промисловим споживанням та побутовими потребами.

ACF також корисна для перевірки стаціонарності часових рядів. Стаціонарні ряди характеризуються стабільними середніми значеннями та дисперсією з часом, що є важливою умовою для багатьох моделей. Швидке зниження ACF до нуля свідчить про стаціонарність, тоді як висока автокореляція на великих лагах може вказувати на тренд або сезонність, які потрібно враховувати додатково.

Для практичного застосування цих методів можна розглянути дані про щомісячні продажі автомобілів. Початковий аналіз графіка дозволяє виявити тренди та сезонні коливання. Подальше використання ACF та PACF допомагає зрозуміти залежності між значеннями та періодичність даних. Спектральний аналіз підтверджує ці результати та допомагає виявити основні частоти змін, що відповідають сезонним патернам. У результаті такі підходи дають змогу створити прогнозну модель, яка враховує виявлені закономірності та забезпечує точні прогнози.

Спектральний аналіз є одним із ключових методів вивчення часових рядів, що дозволяє досліджувати їх частотні компоненти. Цей підхід допомагає виявляти періодичні й циклічні закономірності, які не завжди видно при

простому візуальному аналізі даних. Метод широко застосовується у фінансах, економіці, метеорології, інженерії та інших галузях. Його основна ідея полягає в тому, що будь-який часовий ряд можна розкласти на суму синусоїд різних частот, амплітуд і фаз, тобто представити у вигляді спектра частот. Кожна частота відповідає певному періодичному компоненту в даних.

Основним інструментом цього методу є перетворення Фур'є, яке дає змогу перейти від часової області до частотної. Для дискретних рядів використовують дискретне перетворення Фур'є (DFT) або його оптимізований варіант — швидке перетворення Фур'є (FFT). Ці алгоритми дозволяють розрахувати спектр частот, що ілюструє інтенсивність коливань на кожній частоті.

Застосування спектрального аналізу починається з підготовки даних, що включає нормалізацію та видалення трендів, які можуть спотворювати результати. Потім за допомогою DFT або FFT отримують спектр частот, який показує, на яких частотах спостерігаються найбільші амплітуди. Наприклад, аналізуючи щомісячні продажі автомобілів за кілька років, спектральний аналіз може виявити річні цикли, спричинені сезонними коливаннями попиту.

Метод також дозволяє ідентифікувати основні частоти і оцінити їх амплітуду, що важливо для розуміння природи періодичних компонентів у даних. Наприклад, у фінансових дослідженнях спектральний аналіз виявляє щотижневі чи щомісячні цикли в зміні цін акцій, а в економіці — циклічність макроекономічних показників, таких як ВВП або рівень безробіття. У метеорології метод допомагає виявляти сезонні зміни температури чи опадів, а в інженерії — аналізувати вібрації та шуми в механічних системах для діагностики обладнання.

Серед переваг спектрального аналізу виділяють можливість виявлення прихованих періодичних компонентів, які важко побачити іншим шляхом, а також надання точних кількісних оцінок амплітуди та частоти коливань. Проте існують і обмеження. Зокрема, метод краще працює для стаціонарних рядів і менш ефективний для аналізу нестаціонарних даних. Крім того, визначення фази коливань може бути ускладненим.

У комплексі з іншими методами, такими як аналіз кореляційної функції та автокореляційна функція (ACF), спектральний аналіз забезпечує потужний інструментарій для вивчення часових рядів. Наприклад, ACF допомагає перевірити стаціонарність ряду, виявити тренди чи сезонність, що може підготувати дані для спектрального аналізу. У той час як PACF (часткова автокореляційна функція) уточнює зв'язки між значеннями, спектральний аналіз надає глибше розуміння їх частотної структури.

Таким чином, спектральний аналіз є невід'ємною частиною сучасного аналізу часових рядів. Завдяки можливостям виявлення циклічних патернів, оцінки характеристик коливань і створення прогнозів, цей метод має критичне значення в багатьох галузях. Його застосування забезпечує обґрунтоване прийняття рішень на основі прихованих закономірностей даних.

1.3. Основні підходи до прогнозування фінансових ризиків

Прогнозування ризиків, що виникають у фінансовій сфері, є ключовим завданням, оскільки точні прогнози допомагають фінансовим установам ухвалювати обґрунтовані рішення в управлінні активами, кредитною політикою та інвестиціями. Одним із найбільш поширених методів прогнозування ризиків є використання авторегресійних моделей.

Серед різних моделей для прогнозування ризиків, логістична регресія демонструє високий рівень стабільності, здатна підтримувати точність прогнозів протягом тривалого часу і не потребує постійних коригувань. Однак є ще одна значуща проблема: ризик, пов'язаний з клієнтами, які користуються кредитами з грейсовим періодом або достроково погашають кредити. У таких випадках клієнт може не отримати очікуваного доходу від ресурсів, що були залучені під певний відсоток, що може призвести до фінансових збитків.

Запропонована модель дозволяє ефективно сегментувати клієнтів і зменшити ризики, що можуть суттєво вплинути на прибутковість. Це дасть можливість більш точно контролювати дохідність таких продуктів, враховуючи їхні особливості кредитної політики. Метою роботи є Розроблення моделі для контролю за дохідністю револьверних карток з грейсовим періодом. Револьверні кредити — це кредити, що автоматично поновлюються, що робить їх

популярними на ринку позичкових капіталів. Грейсовий період надає пільгу по сплаті відсотків, зазвичай знижуючи їх на певний час.

Для стаціонарних рядів застосовують моделі AR, MA, ARMA, які будуть докладно розглянуті в подальшому. Існує безліч методів для прогнозування ризиків для нестаціонарних рядів, серед яких можна виділити такі моделі, як ARIMA, GARCH, нейронні мережі, регресійний аналіз, а також машинне навчання, байєсові моделі, економетричні підходи, сценарні моделі та стрес-тестування.

Модель ARIMA (авторегресійна інтегрована модель ковзного середнього) є однією з найбільш ефективних для прогнозування часових рядів. Вона поєднує в собі три ключові компоненти: авторегресію (AR), інтеграцію (I) та ковзне середнє (MA), що дозволяє моделювати як стаціонарні, так і нестаціонарні ряди.

Компонента авторегресії (AR) використовує попередні значення ряду для передбачення поточного. Наприклад, у моделі AR(1) значення поточного періоду залежить від попереднього значення, а у AR(2) — від двох попередніх значень, і так далі. Авторегресійна частина моделі виглядає наступним чином:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t$$

де y_t - поточне значення ряду, c - константа, $\phi_1, \phi_2, \ldots, \phi_p$ - коефіцієнти авторегресії, ϵ_t - випадкова помилка.

Компонента ковзного середнього (MA) передбачає поточне значення як комбінацію попередніх помилок прогнозу. Наприклад, у моделі MA(1) поточне значення залежить від попередньої помилки, у MA(2) — від двох попередніх помилок, і так далі. Формула для цієї частини моделі:

$$y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

де μ - середнє значення ряду, $\theta_1, \theta_2, \ldots \theta_q$ - коефіцієнти ковзного середнього, ϵ_t - випадкова помилка.

Інтеграція (I) необхідна для перетворення нестаціонарного ряду в стаціонарний, зазвичай через віднімання попередніх значень. Наприклад, для моделі I(1) використовуються різниці першого порядку між значеннями, а для I(2) — різниці другого порядку. Формула для інтеграції:

$y_t = \mu y'_t = y_t - y_{t-1}$

де y'_t - значення після диференціювання.

Загальна форма моделі ARIMA поєднує всі три компоненти і виглядає наступним чином: ARIMA(p, d, q), де p — порядок авторегресії, d — порядок інтеграції, q — порядок ковзного середнього. Зазначена форма моделі ARIMA:

$y'_t = c + \phi_1 y'_{t-1} + \phi_2 y'_{t-2} + \cdots + \phi_p y'_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + .. + \theta_q \epsilon_{t-q}$

Модель ARIMA є потужним інструментом для аналізу та прогнозування часових рядів, оскільки вона дозволяє враховувати як короткострокові, так і довгострокові тренди. Наприклад, банківські установи можуть використовувати модель ARIMA для прогнозування рівня дефолтів за кредитами на основі щомісячних даних.

Модель ARIMA є потужним інструментом для прогнозування часового ряду і може бути ефективно використана для прогнозування ризиків, враховуючи як короткострокові залежності, так і довгострокові тренди.

Нейронні мережі активно застосовуються для прогнозування часових рядів, особливо коли дані мають складну нелінійну структуру. Ці моделі здатні навчатися на історичних даних, виявляти приховані закономірності та патерни, що робить їх цінним інструментом у прогнозуванні фінансових показників і ризиків. Вони дозволяють моделювати залежності між численними змінними, такими як макроекономічні показники, фінансові характеристики позичальників або дані про історичні дефолти. Завдяки своїй гнучкості нейронні мережі можуть адаптуватися до змін у даних, забезпечуючи точніші прогнози. Наприклад, для аналізу кредитних ризиків мережі можуть виявляти нелінійні взаємозв’язки між різними змінними, автоматично навчаючись на вхідних даних.

Регресійний аналіз є традиційним підходом, який використовується для оцінки залежності між залежною змінною (наприклад, рівнем ризику) та незалежними змінними (макроекономічними чи фінансовими показниками). Лінійна регресія допомагає визначити основні тенденції, а нелінійні методи дозволяють враховувати більш складні взаємозв’язки. Наприклад, регресійний аналіз можна використовувати для оцінки впливу рівня безробіття на ймовірність дефолту позичальників.

Моделі машинного навчання, зокрема дерева рішень, методи ансамблю (Random Forest, Gradient Boosting), підтримувальні векторні машини (SVM) та інші, відзначаються здатністю працювати з великими обсягами даних і виявляти складні нелінійні залежності. Вони корисні для виявлення аномалій у великих масивах транзакційних даних, що може вказувати на можливі випадки шахрайства. Такі підходи також застосовуються для класифікації ризиків або прогнозування ймовірності дефолтів.

Байєсові моделі надають можливість враховувати невизначеність у прогнозах і використовувати попередню інформацію разом із новими даними. Це робить їх корисними для динамічного оновлення оцінок ризиків у міру появи нової інформації. Наприклад, байєсові моделі можуть враховувати зміни макроекономічних умов, впливаючи на поточні оцінки кредитного ризику.

Економетричні моделі, такі як векторна авторегресія (VAR), дозволяють аналізувати взаємозв'язки між кількома часовими рядами. Це дає змогу враховувати вплив макроекономічних змінних, таких як інфляція, ВВП або процентні ставки, на рівень ризиків. Такі моделі є цінними для розуміння взаємодії економічних факторів і прогнозування їхнього впливу на фінансову стабільність.

Сценарний аналіз використовується для оцінки впливу різних гіпотетичних сценаріїв на фінансові ризики. Цей підхід дозволяє моделювати наслідки можливих економічних або ринкових змін. Наприклад, сценарний аналіз може оцінювати вплив економічної рецесії або різкого зростання процентних ставок на кредитні ризики.

Стрес-тестування спрямоване на оцінку стійкості до екстремальних сценаріїв. Воно допомагає виявляти потенційно вразливі місця у фінансових системах. Наприклад, тестування може показати, як банк витримує різкі зміни валютних курсів або падіння фондового ринку.

Кожен із цих методів має свої переваги та недоліки, залежно від особливостей даних та конкретних цілей аналізу. Комбінація кількох підходів дозволяє отримати більш точні та надійні результати, сприяючи підвищенню

ефективності управління ризиками. Це забезпечує більш обґрунтоване прийняття рішень і допомагає зберігати стабільність у мінливих умовах.

1.4. Висновки до першого розділу

У розділі проведено аналіз предметної області дослідження, що стосується часових рядів. Визначено ключові поняття часових рядів, їхню структуру та складові компоненти, включаючи трендові, сезонні та випадкові елементи.

Розглянуто різницю між стаціонарними та нестаціонарними часовими рядами. Розглянуто їхні математичні та статистичні властивості, методи перетворення нестаціонарних рядів у стаціонарні та їхнє застосування для аналізу і прогнозування.

Проаналізовано основні методи дослідження часових рядів, такі як кореляційний аналіз, автокореляційна функція (ACF) та часткова автокореляція (PACF), що дозволяють виявляти залежності між даними, сезонні патерни та оцінювати придатність для прогнозування.

Приділено увагу методам прогнозування, серед яких ARIMA, спектральний аналіз і нейронні мережі. Описано переваги й недоліки кожного підходу в контексті практичного використання, зокрема для моделювання складних залежностей та довгострокового прогнозування.

Отже, розділ закладає теоретичну основу для створення ефективних алгоритмів прогнозування та автоматизації обробки фінансових даних, що є ключовим у вирішенні завдань дослідження.

РОЗДІЛ 2. РОЗРОБЛЕННЯ АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Адитивні і мультиплікативні моделі для аналізу часових рядів

Адитивні та мультиплікативні моделі є фундаментальними інструментами для аналізу часових рядів, що дозволяють виділяти ключові компоненти — тренд, сезонність та залишкові коливання. Ці моделі широко застосовуються в економіці, фінансах, екології та інших галузях, де важливо розуміти структуру і динаміку змін у часі.

Адитивна модель описує часовий ряд як суму трьох основних компонентів: тренду Tt, сезонності St та залишків Et:

$$Yt = Tt + St + Et,$$

де Yt — значення часового ряду в момент часу t. Такий підхід передбачає, що вплив сезонних коливань є сталим і не залежить від рівня тренду.

Мультиплікативна модель, на відміну від адитивної, враховує взаємозалежність компонентів і описується формулою:

$$Yt = Tt * St * Et,$$

Тут сезонні коливання масштабуються відповідно до рівня тренду, що робить цю модель більш ефективною для часових рядів із змінною амплітудою.

Для спрощення роботи з мультиплікативною моделлю часто застосовують логарифмування, перетворюючи її на адитивну форму:

$$\log(Yt) = \log(Tt) + \log(St) + \log(Et).$$

Це дозволяє використовувати ті ж методи аналізу, що й для адитивної моделі.

Декомпозиція часового ряду передбачає розділення часового ряду на три складові. Перший етап — виділення тренду Tt, який відображає довгострокову тенденцію. Для цього часто використовують метод ковзного середнього або поліноміальну регресію. Наприклад, для рядка даних із сильним трендом ковзне середнє дозволяє згладити короткострокові коливання та виявити загальну динаміку.

Сезонність St визначається як середнє відхилення даних від тренду для кожного періоду. Якщо дані мають сезонний цикл, наприклад, щоквартальний,

обчислюється середнє значення для кожного квартала з урахуванням попередніх років. Сезонна компонента дозволяє врахувати регулярні повторювані коливання.

Залишкова компонента Et обчислюється як різниця між спостереженнями та сумою тренду і сезонності (для адитивної моделі) або як відношення цих значень (для мультиплікативної).

Нормалізація даних є важливим кроком, особливо для мультиплікативних моделей. Вона дозволяє зменшити варіацію між величинами, що спрощує побудову прогнозів. Стандартизація (віднімання середнього значення і ділення на стандартне відхилення) або мінімакс нормалізація (перетворення даних у діапазон від 0 до 1) забезпечують коректну роботу моделей.

Після декомпозиції будується **прогноз**. Для адитивної моделі він визначається як:

$$\hat{Y}t = \hat{T}t + \hat{S}t.$$

У мультиплікативній моделі прогноз обчислюється як:

$$\hat{Y}t = \hat{T}t * \hat{S}t.$$

Ці формули дозволяють врахувати вплив тренду та сезонності на майбутні значення ряду.

Оцінка точності прогнозу оцінюється за допомогою таких метрик, як середня абсолютна похибка (MAE), середньоквадратична похибка (MSE) та середнє абсолютне відсоткове відхилення (MAPE). Наприклад, MAE обчислюється за формулою:

$$MAE = \frac{1}{n}\sum_{i-1}^{n} |Yi - \hat{Y}i|,$$

де n — кількість спостережень, Yi — фактичне значення, а $\hat{Y}i$ — прогнозоване значення.

Адитивні моделі прості у використанні і добре працюють із рядами, де сезонні коливання мають стабільну амплітуду. Мультиплікативні моделі є більш гнучкими і підходять для даних із варіативною сезонністю, однак вимагають попередньої трансформації даних.

Адитивні та мультиплікативні моделі ефективно використовуються в аналізі фінансових даних для прогнозування доходів і витрат. Наприклад, у

роботі застосовано адитивну модель для аналізу стабільних даних і мультиплікативну модель для рядів із варіативною сезонністю, що дозволяє враховувати як довгострокові тренди, так і короткострокові коливання.

2.2. Розроблення алгоритму

Метою даного дослідження є Розроблення алгоритму прогнозування особистих доходів користувача з використанням адитивних і мультиплікативних моделей часових рядів. Алгоритм реалізовано з урахуванням специфіки фінансових даних, таких як доходи та витрати, а також динаміки їх змін у часі. Для аналізу було обрано підхід, що дозволяє враховувати основні компоненти часового ряду: довгостроковий тренд, сезонність і випадкові коливання.

1. Попередня обробка даних та обчислення основних показників.

На першому етапі дані про доходи та витрати аналізуються для формування ключового показника — прибутку. Прибуток обчислюється як різниця між щоквартальними доходами та витратами. Потім дані агрегуються за кварталами, щоб зменшити вплив короткострокових коливань і отримати основу для аналізу сезонності.

Цей етап дозволяє створити єдину основу для подальшої роботи з часовим рядом, зберігаючи при цьому важливі деталі.

2. Згладжування даних методом ковзної середньої.

Метод ковзної середньої використовується для вирівнювання короткострокових коливань, які можуть заважати виявленню довгострокових трендів. Ковзне середнє обчислюється як середнє значення прибутку за фіксовану кількість попередніх кварталів.

Цей підхід дозволяє виявити загальну тенденцію зміни прибутку, яка буде використана на наступних етапах для визначення тренду та сезонності.

3. Розрахунок сезонної компоненти

Важливим аспектом аналізу є врахування сезонності — регулярних коливань прибутку, пов'язаних, наприклад, зі святами, сезонними витратами або доходами. Сезонність визначається двома способами, залежно від типу моделі:

- Адитивна модель: Сезонна компонента обчислюється як різниця між фактичним прибутком і його згладженим значенням.

- Мультиплікативна модель: Сезонна компонента визначається як відношення фактичного прибутку до згладженого.

Окремо підраховується середнє значення сезонної компоненти, яке буде використане для прогнозування майбутніх значень.

4. Моделювання тренду.

Для визначення довгострокової тенденції використовується метод найменших квадратів. Основна ідея методу полягає в тому, щоб побудувати рівняння прямої, яка найкраще відображає зміни прибутку з плином часу. Рівняння тренду має вигляд:

$$T = a_0 + a_1 t$$

де T значення тренду в момент часу t; t — момент часу; a_0 — початковий рівень тренду, знайдений за результатами методу найменших квадратів; a_1 — коефіцієнт нахилу, що визначає швидкість зміни тренду;

Рівні тренду T розраховуються для кожного моменту часу t шляхом підстановки значень t у рівняння тренду. Ці значення дозволяють оцінити довгострокову тенденцію зміни ряду, усунувши вплив випадкових коливань або сезонності.

Аналітичне вирівнювання виконується для коректного аналізу структури ряду та подальшого прогнозування його значень.

5. Прогнозування майбутніх значень.

На основі тренду та середньої сезонної компоненти здійснюється прогнозування майбутніх значень прибутку. Залежно від вибраної моделі прогноз розраховується за такими формулами:

- Ft=St+Tt для адитивної моделі;
- Ft=St*Tt для мультиплікативної моделі;

Для кожного з п'яти наступних періодів обчислюються значення тренду, а також додається вплив сезонності. Це дозволяє врахувати як довгострокову тенденцію, так і регулярні циклічні зміни.

Оцінка точності прогнозу

Точність прогнозу перевіряється за допомогою таких метрик:

1. Середнє абсолютне відхилення (CAO): $CAO = \frac{\sum|e|}{N}$

2. Середнє відносне відхилення (СВВ): $СООП = \frac{\sum\frac{|e|}{y_t}\cdot 100\%}{N}$

Цей алгоритм дозволяє користувачеві Telegram-бота прогнозувати майбутні доходи, враховуючи сезонність, тренди та випадкові коливання.

6. Візуалізація результатів.

Результати аналізу представлені у вигляді графіків:

- Для адитивної моделі: фактичний прибуток, ковзне середнє, тренд і прогноз.
- Для мультиплікативної моделі: аналогічні компоненти, але з урахуванням мультиплікативного підходу до сезонності.

Графіки дозволяють наочно порівняти реальні дані, тренд і прогноз, що є ключовим для інтерпретації результатів.

7. Перевірка адекватності моделі.

Для оцінки моделі використовуються два основних критерії.

Коефіцієнт детермінації (R^2) — показує, наскільки добре модель пояснює варіацію даних.

$$R^2 = 1 - \frac{\sum_{t=1}^{n} e_t^2}{\sum_{t=1}^{n}(y_t - \bar{y})^2} \quad ,$$

де e – це похибка, випадкова величина, що дорівнює різниці фактичного рівня часового ряду y_t та суми сезонної компоненти S_t і тренда T_t.

Критерій Дарбіна-Уотсона (DW) — перевіряє наявність автокореляції похибок.

$$DW = \frac{\sum_{t=2}^{T}(e_t - e_{t-1})^2}{\sum_{t=1}^{T} e_t^2},$$

де e – це випадкова величина (похибка), яка дорівнює різниці фактичного рівня часового ряду y_t та суми сезонної компоненти S_t і тренда T_t.

Значення R^2 та DW, що відповідають встановленим межам, свідчать про адекватність моделі.

8. Результати та застосування.

Розроблений алгоритм дозволяє враховувати як загальні тенденції прибутку, так і сезонні особливості. Отримані прогнози можуть бути використані

для планування витрат, оцінки майбутніх доходів та оптимізації фінансових рішень.

Такий підхід демонструє високу ефективність у ситуаціях, коли необхідно врахувати змінність у часі та сезонність.

2.3. Реалізація алгоритму з реальними даними

2.3.1. Реалізація алгоритму для адитивної моделі

Для тестування алгоритму було взято дані зі звіту фінансових операцій працівника підприємства, яке спеціалізується на будуванні сонячних станцій. Розрахунки будуть йти по прибуткам, тобто різниці доходів і витрат.

Таблиця з доходами і витратами користувача:

Квартал	**Доходи**	**Витрати**
1	15629	9529
2	15154	9195
3	16783	9731
4	16736	9847
5	17759	10456
6	17887	10164
7	18963	11003
8	19387	11278
9	19177	11143
10	20485	11815
11	20411	11544
12	21427	12200

Таблиця з прибутками, для майбутніх розрахунків має вигляд:

Кварталів	1	2	3	4	5	6	7	8	9	10	11	12
Прибуток	6100	5959	7051	6888	7303	7723	7960	8109	8034	8670	8867	9227

Процес побудови адитивної моделі включає такі етапи:

1. Згладжування вхідного часового ряду за допомогою методу ковзної середньої на основі 4 кварталів.

Квартал	Прибуток	КС	ЦКС	ОСК
1	6100			
2	5959			
3	7051	6499,5	6649,875	401,125
4	6888	6800,25	7020,75	-132,75
5	7303	7241,25	7354,875	-51,875
6	7723	7468,5	7621,125	101,875
7	7960	7773,75	7865,125	94,875
8	8109	7956,5	8074,875	34,125
9	8034	8193,25	8306,625	-272,625
10	8670	8420	8559,75	110,25
11	8867	8699,5		
12	9227			

Рисунок. 2.1 – Розрахунок значень методом ковзної середньої на чотири квартали для адитивної моделі

Стовпець ковзної середньої обчислюється як сума значень прибутків за чотири квартали, поділена на кількість кварталів, наприклад:

6499,5=(6100+5959+7051+6888)/4;

Центровану ковзну середню (ЦКС) визначають як середнє арифметичне двох суміжних значень ковзної середньої:

6649,875=(6499,5+6800,25)/2;

Оцінка сезонної компоненти (ОСК) обчислюється як різниця між фактичним значенням ряду (прибутків) і ЦКС:

401.125=7051−6649.875

2. Розрахунок сезонної компоненти.

Квартали		1	2	3	4		
Оцінка сезонності варіації		0	0	401.125	-132.75		
		-51.875	101.875	94.875	34.125		
		-272.63	110.25	0	0		
Середнє		-108.17	70.70833333	165.33333	-32.875		
Скорегована сезонна компонента		-131.92	46.9583	141.5833	-56.6250	0	Сумма значень сезонної компоненти має бути 0!
Сумма середніх значень			Корегуючий коефіцієнт				
95			23.75				

Рисунок. 2.2 – Розрахунок скорегованої сезонної варіації для адитивної моделі

Сезонну варіацію (ОСК) беруть із попередньої таблиці та розподіляють рівномірно між 4 кварталами. Середнє значення сезонної варіації визначається для кожного кварталу:

−108.167=(−51.875+(−272.624))/2

Коригуючий коефіцієнт визначають як відношення суми середніх сезонних варіацій до кількості кварталів:

23.75 = 95 / 4

Скореговану сезонну варіацію (Si) обчислюють як різницю між середнім значенням та коригуючим коефіцієнтом:

−131.917=(− 108.167 − 23.75)

Сума СКВ у адитивній моделі завжди дорівнює нулю.

3.Розрахунок параметрів тренду методом найменших квадратів.

t	y	t^2	y^2	ty
1	6231.92	1	38836785.34	6231.9167
2	5912.04	4	34952236.67	11824.083
3	6909.42	9	47740038.67	20728.25
4	6944.63	16	48227816.39	27778.5
5	7434.9167	25	55277985.84	37174.583
6	7676.0417	36	58921615.67	46056.25
7	7818.4167	49	61127639.17	54728.917
8	8165.6250	64	66677431.64	65325
9	8165.9167	81	66682195.01	73493.25
10	8623.0417	100	74356847.59	86230.417
11	8725.4167	121	76132896.01	95979.583
12	9283.6250	144	86185693.14	111403.5
78	91891	650	715119181.1	636954.25
СУММА				

Рисунок. 2.3 – Таблиця квадратів

Значення у розраховується різницею прибутків зі скоригованою сезонною компонентою (Si):

6231.92 = 6100 − (− 131.917),

Система рівнянь для визначення коефіцієнтів:

$$\begin{cases} 12x_1 + 78x_2 = 91891 \\ 78x_1 + 650x_2 = 636954.25 \end{cases}$$

Метод Крамера:

$$\Delta = \begin{vmatrix} 12 & 78 \\ 78 & 650 \end{vmatrix} = 12 \cdot 650 - 78 \cdot 78 = 7800 - 6084 = 1716$$

$$\Delta_1 = \begin{vmatrix} 91891 & 78 \\ 636954.25 & 650 \end{vmatrix} = 91891 \cdot 650 - 636954.25 \cdot 78 = 59729150 - 49682431.5 = 10046718.5$$

$$\Delta_2 = \begin{vmatrix} 12 & 91891 \\ 78 & 636954.25 \end{vmatrix} = 12 \cdot 636954.25 - 78 \cdot 91891 = 7643451 - 7167498 = 475953$$

Рисунок. 2.4 – Обчислення визначників за методом Крамера

$$x_1 = \frac{\Delta_1}{\Delta} = \frac{10046718.5}{1716} = 5854.731$$

$$x_2 = \frac{\Delta_2}{\Delta} = \frac{475953}{1716} = 277.362$$

Рівняння лінії тренду:

T=5854.731+ 277.362*t.

4. Аналітичне згладжування ряду з використанням тренду.

t	Yt	Si	Yt-Si	T	T+Si	e	|e|	e^2
1	6100	-131.92	6231.916667	6132.093	6000.18	32.093	32.093	1029.96
2	5959	46.9583	5912.041667	6409.455	6456.41	450.455	450.455	202910
3	7051	141.5833	6909.416667	6686.817	6828.4	-364.18	364.183	132629
4	6888	-56.6250	6944.625	6964.179	6907.55	76.179	76.179	5803.24
5	7303	-131.92	7434.916667	7241.541	7109.62	-61.459	61.459	3777.21
6	7723	46.9583	7676.041667	7518.903	7565.86	-204.1	204.097	41655.6
7	7960	141.5833	7818.416667	7796.265	7937.85	-163.74	163.735	26809.2
8	8109	-56.6250	8165.625	8073.627	8017	-35.373	35.373	1251.25
9	8034	-131.92	8165.916667	8350.989	8219.07	316.989	316.989	100482
10	8670	46.9583	8623.041667	8628.351	8675.31	-41.649	41.649	1734.64
11	8867	141.5833	8725.416667	8905.713	9047.3	38.713	38.713	1498.7
12	9227	-56.6250	9283.625	9183.075	9126.45	-43.925	43.925	1929.41

Рисунок. 2.5 – Адитивна модель зі значеннями тренду (T) і похибок (e)

Тренд для кожного моменту часу обчислюється підстановкою номера періоду в рівняння:

6132.093 =5854.731+ 277.362*1

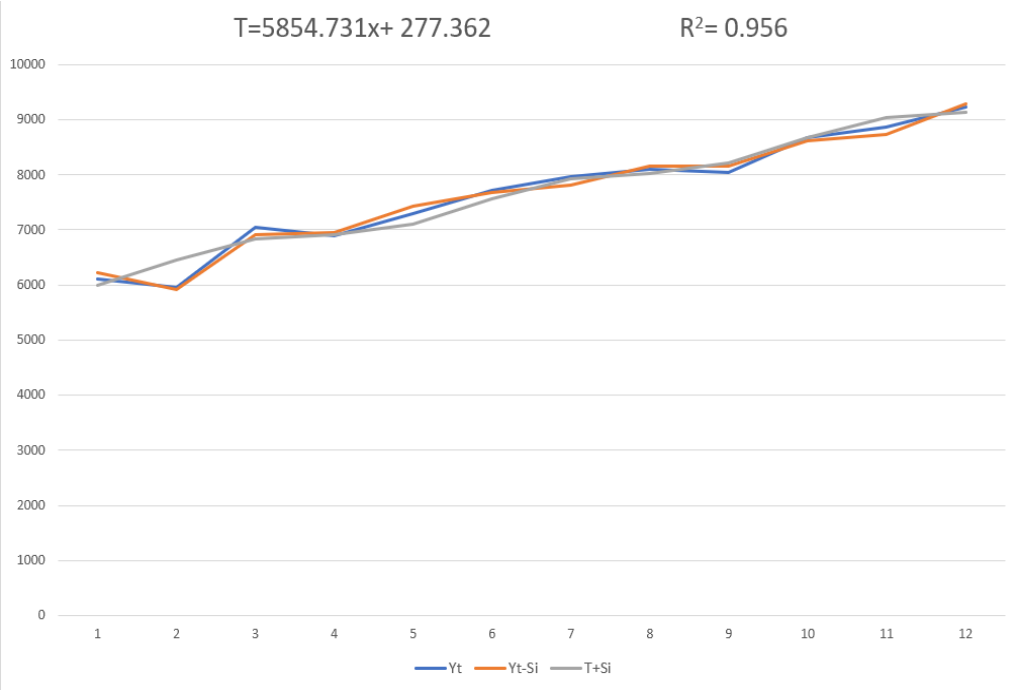


Рисунок. 2.6 – Графік адитивної моделі з трендом

5. Перевірка моделі на адекватність

Згідно з формулою коефіцієнта детермінації з підрозділу 2.1, $R^2 = 0.956$ (або 95.6%), що свідчить про те, що модель пояснює 95.6% варіації у залежній змінній.

Критерій Дарбіна-Уотсона дорівнює DW=2,60 , що свідчить про адекватність моделі.

6. Прогнозування.

Прогноз для наступних 2 кварталів визначають як суму значень скорегованої сезонної варіації (Si) та тренду (T):

F13=Si13+T13= −131.916 + 9460.437 = 9328.52;

F14=Si14+T14= 46.9583 + 9737.799 = 9784.757;

Точність прогнозу перевіряють за середньою абсолютною похибкою (САО= 145.67) та середньою відносною похибкою (СООП= 2.05%)

Значення СООП, менше 5% , тобто це вказує на високу точність прогнозу.

2.3.2. Реалізація алгоритму для мультиплікативної моделі

Для побудови мультиплікативної моделі було використано той самий експериментальний часовий ряд, що і для адитивної моделі. Аналіз однакових вихідних даних дозволяє чітко побачити відмінності між цими двома підходами.

Таблиця з прибутками, для майбутніх розрахунків має вигляд:

Кварталів	1	2	3	4	5	6	7	8	9	10	11	12
Прибуток	6100	5959	7051	6888	7303	7723	7960	8109	8034	8670	8867	9227

Процес побудови мультиплікативної моделі майже ідентичний адитивній, однак основні відмінності полягають у специфіці розрахунків. Основні етапи побудови мультиплікативної моделі включають:

1. Згладжування вхідного часового ряду за допомогою методу ковзної середньої на основі 4 кварталів.

Квартал	Прибуток	МКС	ЦКС	ОСК
1	6100			
2	5959			
3	7051	6499,5	6649,88	1,060320683
4	6888	6800,25	7020,75	0,981091764
5	7303	7241,25	7354,88	0,992946855
6	7723	7468,5	7621,13	1,013367449
7	7960	7773,75	7865,13	1,012062745
8	8109	7956,5	8074,88	1,004226072
9	8034	8193,25	8306,63	0,967179811
10	8670	8420	8559,75	1,012880049
11	8867	8699,5		
12	9227			

Рисунок. 2.7 – Розрахунок значень методом ковзної середньої на чотири квартали для мультиплікативної моделі

Стовпець ковзної середньої обчислюється як сума значень прибутків за чотири квартали, поділена на кількість кварталів, наприклад:

6499,5=(6100+5959+7051+6888)/4;

Центровану ковзну середню (ЦКС) визначають як середнє арифметичне двох суміжних значень ковзної середньої:

6649,875=(6499,5+6800,25)/2;

Оцінка сезонної компоненти (ОСК) обчислюється діленням між фактичним значенням ряду (прибутків) і ЦКС:

1,06032=7051 / 6649.875

2. Розрахунок сезонної компоненти.

Квартали	1	2	3	4		
Оцінка сезонності варіації	0	0	1.06032068	0.98109		
	0.99295	1.013367449	1.01206275	1.00423		
	0.96718	1.012880049	0	0		
Середнє	0.65338	0.675415833	0.69079448	0.66177		
Скорегована сезонна компонента	**0.97469**	**1.0075726**	**1.03051417**	**0.98722**	4.000	Сумма значень сезонної компоненти має бути 4!
Сумма середніх значень		Корегуючий коефіцієнт				
2.681358476		**1.491781139**				

Рисунок. 2.8 – Розрахунок скорегованої сезонної варіації для мультиплікативної моделі

Сезонну варіацію (ОСК) беруть із попередньої таблиці та розподіляють рівномірно між 4 кварталами, так само як і для адитивної моделі. Середнє значення сезонної варіації визначається для кожного кварталу:

−108.167=(−51.875+(−272.624))/2

Коригуючий коефіцієнт для мультиплікативної моделі знаходять діленням кількості кварталів на ССЗ (сума середніх значень):

1,49178 = 4 / 2,68136

Скореговану сезонну варіацію (Si) обчислюють як різницю між середнім значенням та коригуючим коефіцієнтом:

0.97460 = 0,65338 * 1,49178

Сума СКВ у мультиплікативній моделі завжди дорівнює кількості кварталів, у нашому випадку це чотири.

3.Розрахунок параметрів тренду методом найменших квадратів.

t	y	t^2	y^2	ty
1	6258.38	1	39167304.27	6258.379
2	5914.21	4	34977927.52	11828.428
3	6842.22	9	46815912.6	20526.646
4	6977.17	16	48680886.07	27908.676
5	7302.03	25	53319573.58	36510.127
6	7664.96	36	58751555.88	45989.738
7	7724.30	49	59664801.45	54070.096
8	8213.98	64	67469392.29	65711.803
9	8242.59	81	67940332.28	74183.333
10	8668.99	100	75151429.71	86689.924
11	8604.44	121	74036432.06	94648.868
12	9346.45	144	87356100.4	112157.383
78	91759.71	650	713331648.11	636483.401
СУММА				

Рисунок. 2.9 – Таблиця квадратів

Значення у розраховується різницею прибутків зі скоригованою сезонною компонентою (Si). Однак, на відміну від адитивної моделі, ці значення обчислюються шляхом ділення фактичного обсягу продажу на скориговану сезонну компоненту:

6258,38= 6100 / 0.97460

Система рівнянь для визначення коефіцієнтів виглядає так:

$$\begin{cases} 12x_1 + 78x_2 = 91759.71 \\ 78x_1 + 650x_2 = 636483.401 \end{cases}$$

Метод Крамера:

$$\Delta = \begin{vmatrix} 12 & 78 \\ 78 & 650 \end{vmatrix} = 12 \cdot 650 - 78 \cdot 78 = 7800 - 6084 = 1716$$

$$\Delta_1 = \begin{vmatrix} 91759.71 & 78 \\ 636483.401 & 650 \end{vmatrix} = 91759.71 \cdot 650 - 636483.401 \cdot 78 = 59643811.5 - 49645705.278 =$$

$$= 9998106.222$$

$$\Delta_2 = \begin{vmatrix} 12 & 91759.71 \\ 78 & 636483.401 \end{vmatrix} = 12 \cdot 636483.401 - 78 \cdot 91759.71 = 7637800.812 - 7157257.38 = 480543.432$$

Рисунок. 2.10 – Обчислення визначників за методом Крамера

$$x_1 = \frac{\Delta_1}{\Delta} = \frac{9998106.222.}{1716} = 5826.4022$$

$$x_2 = \frac{\Delta_2}{\Delta} = \frac{480543.432}{1716} = 280.0370$$

Рівняння лінії тренду:

T=5826.4022 + 280.0370 * t.

4. Аналітичне вирівнювання ряду з використанням тренду.

t	Yt	Si	Yt/Si	T	TxSi	e	|e|	e^2
1	6100	0.97469	6258.378725	6106.4392	5951.91	6.4392	6.4392	41.463
2	5959	1.0076	5914.214024	6386.4762	6434.84	427.476	427.476	182736
3	7051	1.0305	6842.215474	6666.5132	6869.94	-384.49	384.487	147830
4	6888	0.9872	6977.168915	6946.5502	6857.77	58.5502	58.5502	3428.1
5	7303	0.97469	7492.613087	7226.5872	7043.71	-76.413	76.4128	5838.9
6	7723	1.0076	7664.956352	7506.6242	7563.47	-216.38	216.376	46818
7	7960	1.0305	7724.299415	7786.6612	8024.26	-173.34	173.339	30046
8	8109	0.9872	8213.975425	8066.6982	7963.6	-42.302	42.3018	1789.4
9	8034	0.97469	8242.59257	8346.7352	8135.51	312.735	312.735	97803
10	8670	1.0076	8604.838997	8626.7722	8692.1	-43.228	43.2278	1868.6
11	8867	1.0305	8604.442577	8906.8092	9178.59	39.8092	39.8092	1584.8
12	9227	0.9872	9346.448545	9186.8462	9069.44	-40.154	40.1538	1612.3

Рисунок. 2.11 – Мультиплікативна модель зі значеннями тренду (T) і похибок (e)

Тренд для кожного моменту часу обчислюється підстановкою номера періоду в рівняння:

6132.093 =5854.731+ 277.362*1

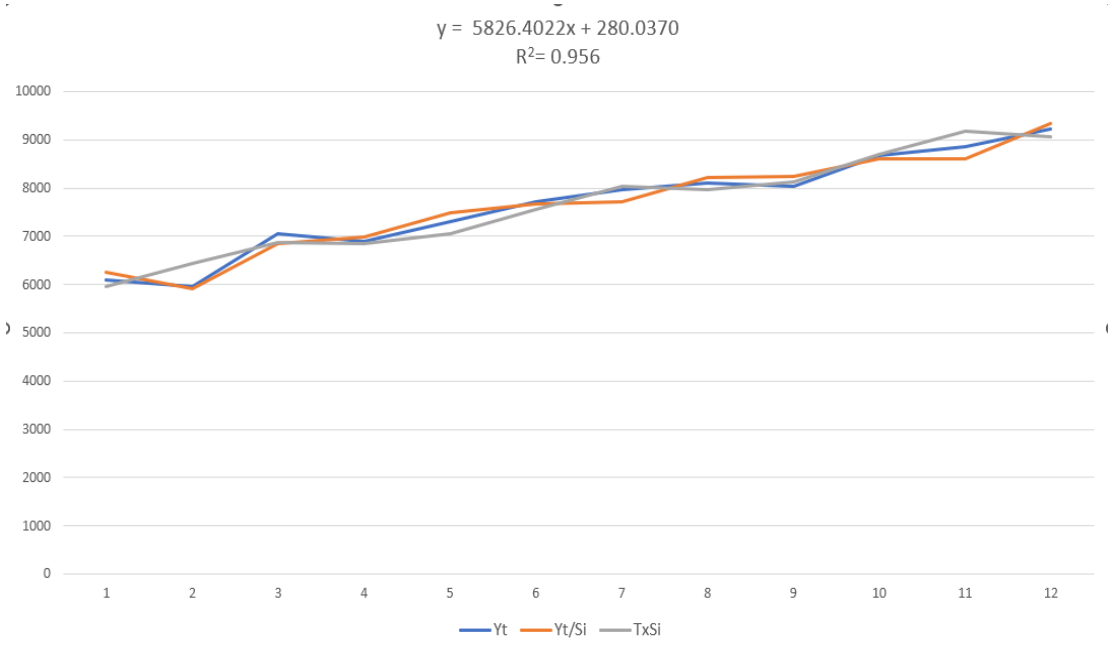


Рисунок. 2.12 – Графік мультиплікативної моделі з трендом

5. Перевірка моделі на адекватність

Згідно з формулою коефіцієнта детермінації з підрозділу 2.1, R^2= 0.956 (або 95.6%), що свідчить про те, що модель пояснює 95.6% варіації у залежній змінній.

Критерій Дарбіна-Уотсона дорівнює DW=2,599.

Модель можна вважати адекватною за високим R^2, та задовільним значенням DW.

6. Прогнозування.

Прогноз виконується на наступні два квартали, щоб забезпечити максимальну точність. У мультиплікативній моделі значення прогнозу для майбутніх періодів обчислюється як добуток значень скоригованої сезонної компоненти.

F13=Si13*T13= 0.974693 * 9466.8832 = 9227.308;

F14=Si14*T14= 1.0076 * 9746.9202 = 9820.73;

Точність прогнозу перевіряють за середньою абсолютною похибкою (САО= 152.40) та середньою відносною похибкою (СООП= 2.14%)

Значення СООП, менше 5%, тобто це вказує на високу точність прогнозу.

2.4. Висновки до другого розділу

Було розроблено алгоритм аналізу та прогнозування часових рядів, який враховує основні компоненти: тренд, сезонність і випадкові коливання. Алгоритм дозволяє інтегрувати сучасні підходи до аналізу часових рядів у фінансових додатках, забезпечуючи гнучкість і точність прогнозування.

На основі виділених компонентів побудовано прогноз майбутніх значень, що враховує як довгострокові тенденції,. Алгоритм протестовано на реальних фінансових даних, що дало змогу оцінити його ефективність та виявити сильні сторони кожної з моделей.

Порівняння результатів розрахунків показало, що адитивна модель забезпечує більшу точність у обраному діапазоні даних та у випадках зі стабільними сезонними коливаннями. Тоді як мультиплікативна модель виявилась менш точною, бо є більш ефективною для рядів із амплітудою сезонних змін, що варіюється залежно від тренду.

Крім того, було проведено оцінку адекватності моделей за допомогою метрик, таких як коефіцієнт детермінації та критерій Дарбіна-Уотсона. Обидві моделі показали високий рівень відповідності реальним даним, підтверджуючи ефективність розробленого підходу.

Таким чином, результати розробки алгоритму та його тестування підтвердили, що він може бути ефективно використаний для прогнозування фінансових показників із урахуванням сезонності та довгострокових трендів.

РОЗДІЛ 3. ВИБІР СЕРЕДОВИЩА ТА СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

3.1. Вибір платформи, середовища та мови програмування для реалізації додатку

3.1.1. Мова програмування Python

Python — це високорівнева мова програмування загального призначення, яка вирізняється простотою та читабельністю синтаксису. Вона широко використовується у веб-розробці, науці про дані, розробці додатків, штучному інтелекті, машинному навчанні, автоматизації та багатьох інших сферах.

Python підтримує кілька парадигм програмування, включаючи об'єктно-орієнтоване, функціональне та процедурне програмування, що робить його універсальним інструментом для розробників.

Основні можливості Python:

- Простота синтаксису, що сприяє швидкому освоєнню і зменшенню кількості помилок у коді.
- Велика кількість бібліотек та фреймворків, таких як Flask та Django для веб-розробки, NumPy, Pandas та Matplotlib для аналізу даних, TensorFlow та PyTorch для машинного навчання.
- Кросплатформеність: Python працює на більшості операційних систем, включаючи Windows, macOS, Linux.
- Розвинена екосистема інструментів для автоматизації та DevOps, таких як Fabric і Ansible.

Основна область застосування Python:

1. Веб-Розроблення : Використовуючи фреймворки, такі як Flask або Django, розробники можуть створювати масштабовані серверні додатки з API для інтеграції з іншими сервісами.
2. Аналіз даних та наука про дані: Завдяки таким бібліотекам, як Pandas та NumPy, Python є стандартом у сфері обробки та аналізу даних.
3. Штучний інтелект та машинне навчання: Бібліотеки TensorFlow, Keras, PyTorch дозволяють будувати та навчати моделі для прогнозів, класифікації або роботи з великими даними.

4. Автоматизація та скрипти: Python активно використовується для написання скриптів для автоматизації задач, наприклад, парсингу даних з веб-сайтів або управління системами.

5. Інтернет речей (IoT): Python часто використовується для програмування IoT-пристроїв, таких як Raspberry Pi, завдяки бібліотекам для роботи з апаратним забезпеченням.

Python був створений Гвідо ван Россумом у 1991 році як мова, орієнтована на читабельність і зручність використання. За роки існування Python здобув популярність завдяки своїй простоті, ефективності та універсальності.

Приклад реалізації простого HTTP-сервера за допомогою стандартної бібліотеки Python:

```
from http.server import BaseHTTPRequestHandler, HTTPServer
class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        self.send_response(200)
        self.send_header("Content-type", "text/html")
        self.end_headers()
        self.wfile.write(b"Hello, World!")
host = "localhost"
port = 8080
server = HTTPServer((host, port), SimpleHTTPRequestHandler)
print(f"Сервер запущено на {host}:{port}")
server.serve_forever()
```

Переваги Python:

1. Швидке прототипування завдяки простому синтаксису.

2. Величезна спільнота розробників і велика кількість доступної документації.

3. Різноманітні бібліотеки та фреймворки для будь-яких задач.

Python є ідеальним вибором для реалізації серверної частини Telegram-бота завдяки своїй простоті, широким можливостям інтеграції та підтримці популярних бібліотек, таких як python-telegram-bot для роботи з Telegram API.

3.1.2. Telegram Bot API

Telegram Bot API — це HTTP-інтерфейс для роботи з ботами в Telegram. Боти — це спеціальні облікові записи, створені для автоматичного отримання, оброблення та надсилання повідомлень. API забезпечує простий спосіб інтеграції функціональності Telegram у сторонні застосунки та серверні системи.

Telegram Bot API підтримує два основні підходи до отримання оновлень від сервера:

1. Long polling передбачає, що клієнтський застосунок регулярно опитує сервер Telegram, запитуючи інформацію про нові оновлення. У такому випадку Клієнт надсилає запит на сервер Telegram на певний інтервал часу. Якщо протягом цього часу з’являються нові повідомлення чи інші події, сервер надсилає відповідь із даними. Інтервал за замовчуванням для таких запитів становить 100 мілісекунд.

Цей спосіб підходить для простих застосунків, які не потребують миттєвої реакції на події.

2. Webhook — більш просунутий спосіб, який передбачає, що сервер Telegram самостійно надсилає нові оновлення до серверу вашого застосунку. У якому ви вказуєте URL вашого сервера, який має приймати вхідні оновлення. Як тільки з’являється нове повідомлення або подія, Telegram надсилає HTTP-запит на цей URL.

Цей метод забезпечує миттєву реакцію на оновлення і зменшує кількість запитів до серверів Telegram, проте вимагає наявності власного серверу з доступом до інтернету.

Оновлення зберігаються на сервері Telegram до 24 годин, після чого видаляються, якщо не були оброблені; відповіді Telegram містять об’єкти типу **Update**, серіалізовані у формат JSON для зручності обробки, а запити до API здійснюються через HTTPS за форматом: https://api.telegram.org/bot<token>/НАЗВА_МЕТОДУ.

Token — це унікальний ключ доступу, який отримується через спеціального бота **@BotFather**. Початкове повідомлення **@BotFather** можна побачити на рисунку 3.1. Він ідентифікує вашого бота і дозволяє взаємодіяти з API.

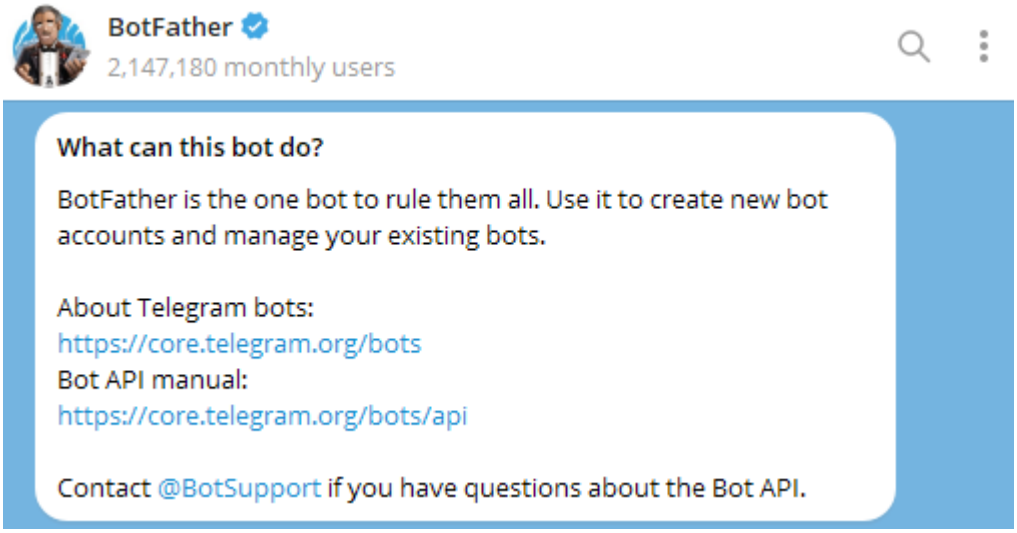


Рисунок 3.1. – Знайомлення з **BotFather**

Основні методи Telegram Bot API

1. **getUpdates** — дозволяє отримувати оновлення за допомогою технології long polling.
2. **setWebhook** — прив’язує до бота URL вашого сервера, де знаходиться програма для оброблення оновлень.
3. **sendMessage** — надсилає текстове повідомлення до чату користувача.
4. **sendLocation** — надсилає географічні координати користувачу.
5. **getFile** — завантажує файл із сервера Telegram за його унікальним ідентифікатором.

API підтримує як **POST**, так і **GET** запити, а для передачі параметрів використовується один із наступних форматів:

1. Передача параметрів у URL.
2. Формат **application/x-www-form-urlencoded**.
3. Формат **application/json** (використовується для передачі даних, але не для завантаження файлів).

4. Формат **multipart/form-data** (використовується для завантаження файлів).

Принцип взаємодії чат-бота, серверів і користувача у Telegram Bot API наведений на рисунку 3.2.

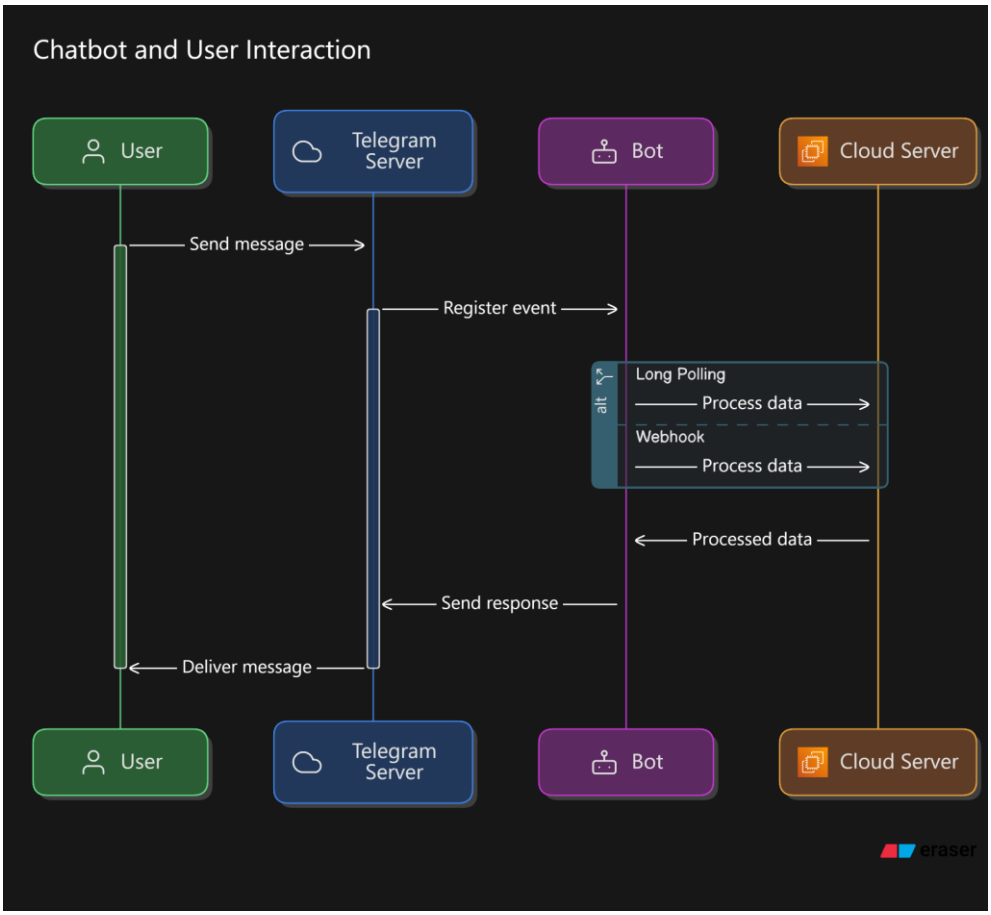


Рисунок 3.2. – Принцип взаємодії чат-бота, серверів і користувача у Telegram Bot API

3.1.3. Heroku

Heroku — це хмарна PaaS-платформа (Platform as a Service), яка підтримує низку мов програмування і належить компанії Salesforce із 2010 року. Заснована у червні 2007 року Джеймсом Лінденбаумом, Адамом Віггінсом і Оріоном Генрі, Heroku стала однією з перших хмарних платформ. Спочатку вона підтримувала лише Ruby, але з часом список мов розширився і тепер включає **Java, Node.js, Scala, Clojure, Python, Go та PHP**. Сервера платформи працюють на базі операційних систем **Debian** або **Ubuntu**.

Після придбання Heroku корпорацією Salesforce у грудні 2010 року, компанія розширила свої можливості, зокрема, у 2011 році додала підтримку Node.js і Clojure. Того ж року Юкіхіро Мацумото, творець Ruby, приєднався до

команди як головний інженер. Heroku підтримує різноманітні СУБД, такі як **PostgreSQL**, **MongoDB**, **Redis**, **Cloudant** і **Membase**, що забезпечує гнучкість для розробників.

Додатки, створені на Heroku, отримують унікальні доменні імена у форматі ім'я_додатка.herokuapp.com і працюють на ізольованих віртуальних процесах, які називаються **dynos**. Ці процеси розподіляються по спеціальній віртуальній мережі, що складається з кількох серверів, забезпечуючи масштабованість і надійність. Крім того, платформа має власну систему контролю версій, що спрощує управління кодом і розгортання додатків.

3.1.4. Середовище розроблення PyCharm

PyCharm — це інтегрована середовище розробки (IDE) для мови програмування Python, розроблена компанією JetBrains. Платформа доступна для операційних систем **Windows, macOS і Linux**, що робить її універсальним інструментом для розробників, інтерфейс якого можна побачити на рисунку 3.2.

PyCharm підтримує всі основні аспекти розробки на Python, включаючи:

- автоматичне завершення коду;
- підсвічування синтаксису;
- інтеграцію з системами контролю версій (Git, Mercurial, SVN);
- вбудований відладник;
- підтримку роботи з базами даних;
- можливості рефакторингу коду;
- інструменти для тестування та розробки вебзастосунків із використанням фреймворків, таких як Django, Flask і FastAPI.

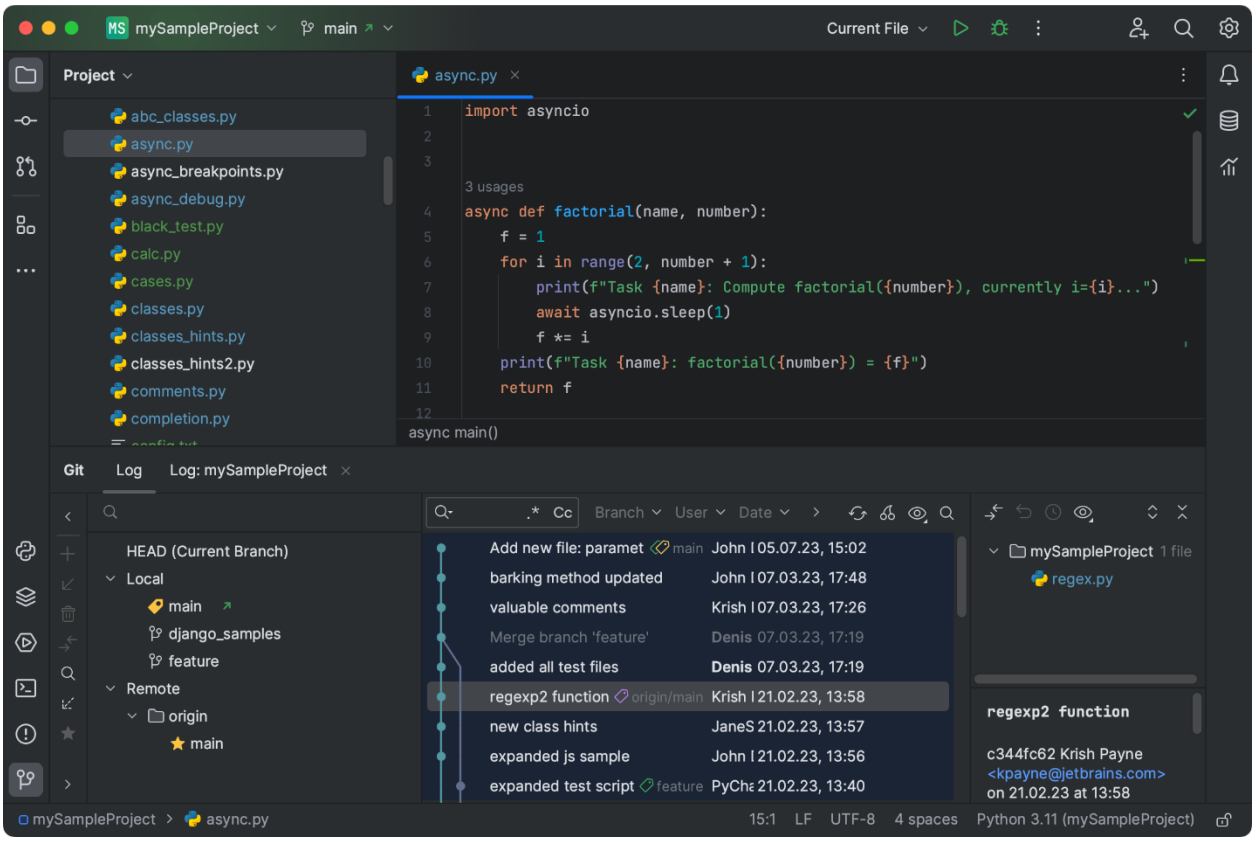


Рисунок 3.3. – Користувацький інтерфейс **PyCharm**

Серед підтримуваних мов та технологій, окрім Python, є JavaScript, HTML, CSS та SQL, що дозволяє працювати над повноцінними проєктами.

Основні можливості PyCharm:

1. **Вбудований відладник** — дозволяє здійснювати покроковий аналіз коду, встановлювати точки зупинки й аналізувати змінні.
2. **Система управління проєктами** — забезпечує легку навігацію між файлами та модульний підхід до структури проєктів.
3. **Розширення функціональності** — через плагіни та інтеграції можна додати підтримку нових інструментів і технологій.
4. **IntelliJ-платформа** — основою для PyCharm є IntelliJ IDEA, що гарантує стабільність та масштабованість середовища.

PyCharm випускається у двох версіях:

- **Professional Edition** — платна, включає всі функції для веброзробки та роботи з базами даних.
- **Community Edition** — безкоштовна, але з обмеженим функціоналом, орієнтована на базову розробку на Python.

Перший реліз PyCharm відбувся у **2010 році**, і з того часу IDE активно оновлюється, додаючи нові функції для спрощення процесу розробки. JetBrains також забезпечує якісну технічну підтримку і спільноту користувачів, яка створює додаткові розширення та ділиться практичними рекомендаціями.

Інтерфейс PyCharm включає панель для навігації між файлами, інтегрований термінал, вкладки для редагування коду та інструменти управління проєктом, що дозволяють ефективно працювати навіть з великими кодовими базами.

3.2. Реєстрація чат-бота і його початкові налаштування

Реєстрація та налаштування чат-бота через BotFather

Першим етапом розробки чат-бота в Telegram є його реєстрація через спеціального чат-бота **BotFather**. Цей бот є основним інструментом для створення і управління ботами.

Приклад реєстрації чат бота наведений на рисунку 3.4.

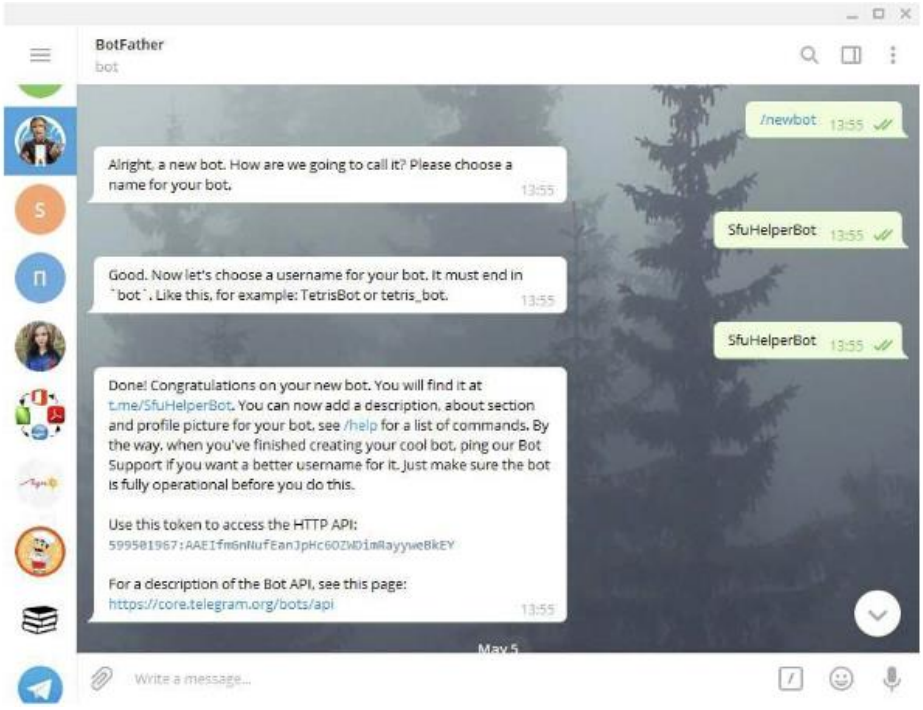


Рисунок 3.4. – Реєстрація чат бота

Етапи реєстрації:

1. Відправлення команди **/newbot** до BotFather.
2. Введення назви майбутнього чат-бота (ім’я має закінчуватися на **"Bot"** або **"_bot"**).

3. Успішна реєстрація завершується отриманням **токена** (унікального набору символів), який використовується для доступу до **HTTP API Telegram Bot**, і URL-адреси для підключення до чат-бота.

Після створення чат-бота його параметри можна змінювати або доповнювати за допомогою спеціальних команд BotFather (таблиця 3.1).

Таблиця 3.1

Команда	**Опис**
/setname	**Змінює існуюче ім’я бота.**
/setdescription	**Додає текст, що відображатиметься під час першого запуску бота.**
/setabouttext	**Додає текст у поле "Про бота".**
/setuserpic	**Додає або змінює зображення (аватар) бота.**
/setcommands	**Створює список доступних команд для користувачів.**
/deletebot	**Видаляє бота.**

Додаткові команди налаштування:

Для роботи з додатковими параметрами доступні команди з таблиці 3.2.

Таблиця 3.2

Команда	**Опис**
/token	Повертає токен для обраного бота.
/revoke	Анулює існуючий токен доступу.
/setinline	Вмикає або вимикає можливість виклику бота з інших чатів.
/setinlinegeo	Дозволяє передавати геолокацію бота з іншого чату.
/setinlinefeedback	Отримує інформацію про вибрані користувачами команди.
/setjoingroup	Визначає, чи можна додавати бота до групових чатів.
/setprivacy	Вмикає режим конфіденційності, що забезпечує обробку даних кожного користувача окремо.

Після отримання токена і виконання базових налаштувань у BotFather можна переходити до розробки програмної частини бота. Це включає використання токена для підключення до **Telegram Bot API** і реалізації

функціональності бота на обраній мові програмування (наприклад, Python, Node.js тощо).

Такий підхід забезпечує ефективний старт для створення функціонального і налаштованого чат-бота.

3.2. Програмна реалізація роботи з чат-ботом

У цьому розділі ми розглянемо етапи програмної реалізації роботи з Telegram-ботом, який забезпечує автоматизацію процесів управління фінансами. Основне завдання чат-бота — надати користувачам зручний інтерфейс для завантаження, аналізу та управління фінансовими даними.

Реалізація починається з інтеграції необхідних бібліотек, які забезпечують роботу з Telegram API, управління даними, а також аналіз і візуалізацію фінансових даних. Основними інструментами, що використовуються, є бібліотека **Aiogram** для взаємодії з Telegram API та **Pandas** для обробки даних.

Для зручності підтримки коду функції аналізу даних виділені у файл analysis.py, який містить окрему реалізацію алгоритмів обробки та прогнозування. Нижче наведено початкову частину коду, яка відповідає за підключення необхідних бібліотек та підготовку до подальшої роботи.

Для забезпечення роботи бота необхідно створити його в Telegram за допомогою BotFather і отримати унікальний токен доступу, який використовується для ідентифікації бота в системі Telegram API. Токен зберігається у змінній API_TOKEN.

```
API_TOKEN = "7612743717:AAFysM0ZZT9T0qi4cchfDt9eimXStbHcsvs"
```

Цей токен необхідно передати при ініціалізації об’єкта бота, щоб він міг взаємодіяти з Telegram API.

Для запуску бота та його взаємодії з користувачами ми створюємо асинхронну функцію main(). Вона є головною точкою входу до програми, в якій налаштовується бот і диспетчер подій, а також створюється інтерактивна клавіатура для зручного доступу до основних команд.

```
async def main():
    bot = Bot(token=API_TOKEN)
    dp = Dispatcher()
```

Створення клавіатури. Для зручності взаємодії користувачів з ботом використовується ReplyKeyboardMarkup, яка формує інтерактивну панель з кнопками, остаточний вигляд клавіатури користувача можна побачити на рисунку 3.5:

- Analyse: аналіз даних.
- Upload: завантаження файлу з даними.
- Summary: коротка інформація про фінансовий стан.
- View Data: перегляд наявних даних.
- Clear Data: очищення буфера даних.

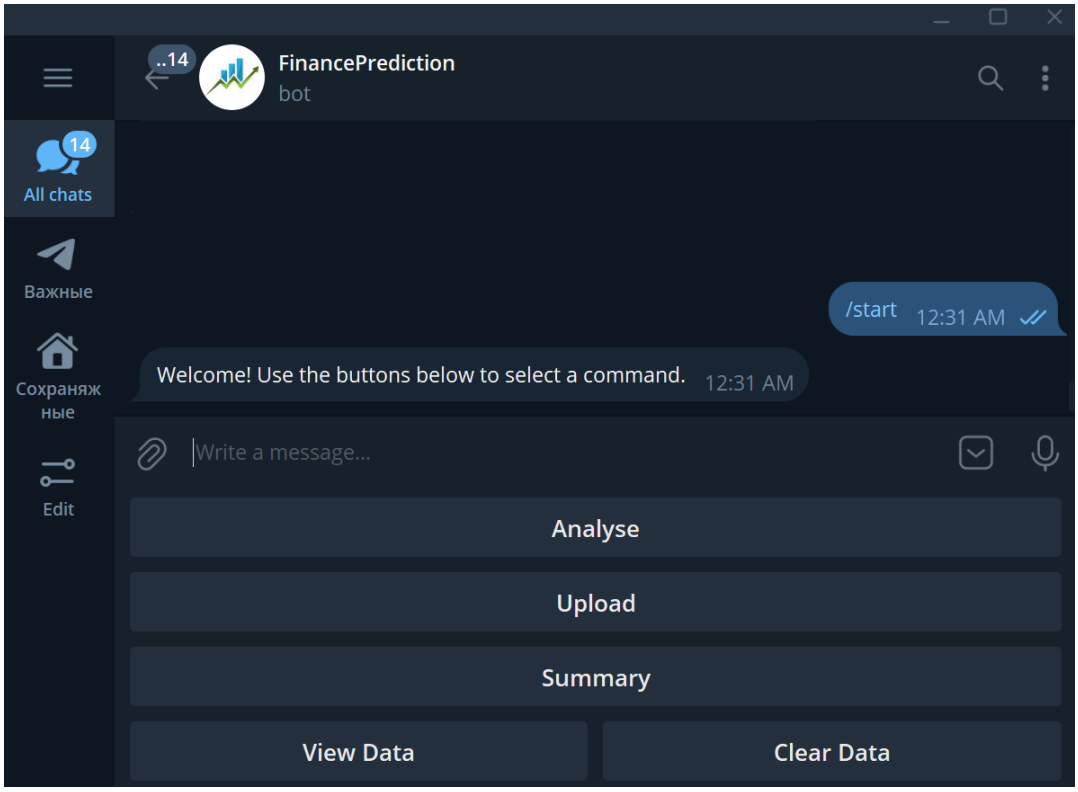


Рисунок. 3.5. – Вигляд користувацького інтерфейсу

```
      keyboard = ReplyKeyboardMarkup(
              keyboard=[
                  [KeyboardButton(text="Analyse")],
                  [KeyboardButton(text="Upload")],
                  [KeyboardButton(text="Summary")],
                  [KeyboardButton(text="View     Data"),     KeyboardButton(text="Clear
Data")],
              ],
              resize_keyboard=True,
          )
```

Функція main() забезпечує основу для побудови логіки роботи бота, зокрема додавання обробників для кожної з команд і запуск бота в режимі **polling**.

Наступний фрагмент коду визначає асинхронну функцію start_command, яка обробляє команду /start в Telegram-боті. Коли користувач надсилає цю команду, бот відповідає повідомленням із привітанням та клавіатурою, що містить кнопки для вибору команд. Клавіатура створюється заздалегідь і передається через параметр reply_markup.

```
@dp.message(Command("start"))
async def start_command(message: Message):
    await message.answer(
        "Welcome! Use the buttons below to select a command.",
        reply_markup=keyboard,
    )
```

Функція upload_file відповідає за обробку команди /upload. На початку вона використовує глобальну змінну data_buffer для збереження даних. Якщо користувач не прикріпив файл або не вніс дістатньо даних для аналізу, бот просить завантажити CSV із зазначеними колонками або надати дані у повідомленнях.

```
@dp.message(Command("upload"))
async def upload_file(message: Message):
    global data_buffer

    print("Upload command received")

    if not message.document:
        await message.answer("Please upload a CSV file with the following columns: Date, Income, Expenses.")
        return
```

Функція отримує file_id і file_name, завантажує файл у тимчасовий шлях (temp_file_path) за допомогою Telegram API, а потім перевіряє формат файлу. Якщо формат не .csv, піднімається помилка.

```
    file_id = message.document.file_id
    file_name = message.document.file_name
    temp_file_path = f"./{file_name}"

    try:
```

```
                print(f"Processing file: {file_name} (ID: {file_id})")
                file = await bot.get_file(file_id)
                print(f"File info: {file}")  # Лог даних про файл
                await bot.download_file(file.file_path, destination=temp_file_path)
                print(f"File downloaded to: {temp_file_path}")  # Лог шляху до файлу
                await message.answer("File received. Processing...")
                if not file_name.endswith(".csv"):
                    raise ValueError("The file is not in CSV format.")
```

Дані читаються через pandas і перевіряються на наявність необхідних колонок та відсутність порожніх або некоректних значень у колонках Date, Income, Expenses. Якщо дані валідні, вони додаються до глобального буфера data_buffer.

```
                data = pd.read_csv(temp_file_path)
                print(f"File loaded into DataFrame: {data.head()}")

                if data.empty:
                    raise ValueError("The uploaded CSV file is empty.")

                required_columns = {"Date", "Income", "Expenses"}
                if not required_columns.issubset(data.columns):
                    raise ValueError(f"Missing required columns: {',
'.join(required_columns)}")

                data["Date"] = pd.to_datetime(data["Date"], errors="coerce")
                if data["Date"].isna().any():
                    raise ValueError("Invalid date values detected.")
                if data[["Income", "Expenses"]].isna().any().any():
                    raise ValueError("Income or Expenses columns contain invalid
values.")
```

Якщо data_buffer вже має дані, нові додаються до нього із видаленням дублікатів, і все сортується за датою. Оновлений буфер зберігається у файл DATA_FILE.

```
                if not data_buffer.empty:
                    data_buffer = pd.concat([data_buffer, data],
ignore_index=True).drop_duplicates()
                else:
```

```
                data_buffer = data

            data_buffer.sort_values(by="Date", inplace=True)

            data_buffer.to_csv(DATA_FILE, index=False)
            await message.answer("File successfully uploaded and processed!")

        except Exception as e:
            print(f"Error during upload: {e}")  # Лог для дебагу
            await message.answer(f"File upload error: {e}")
        finally:
```

Після завершення процесу, незалежно від того, виникла помилка чи ні, функція видаляє тимчасовий файл для уникнення накопичення непотрібних даних.

```
            if os.path.exists(temp_file_path):
                os.remove(temp_file_path)
                print(f"Temporary file {temp_file_path} removed")
```

Функція view_data використовується для обробки команди /view_data і дозволяє користувачу переглядати завантажені фінансові дані в агрегованій формі. На початку функція перевіряє, чи є дані у глобальному буфері data_buffer. Якщо буфер порожній, бот надсилає повідомлення із проханням завантажити файл із даними, оскільки немає інформації для відображення. У випадку, якщо дані доступні, функція обробляє їх: колонка Date встановлюється як індекс для забезпечення роботи з часовими даними, потім дані групуються за кварталами за допомогою методу resample("ME"), який обчислює суми для колонок Income і Expenses. Далі додається нова колонка Profit, що обчислюється як різниця між доходами та витратами.

Отримана таблиця конвертується у текстовий формат за допомогою методу to_string() і надсилається користувачеві як текстове повідомлення в Telegram. Це дозволяє користувачам швидко ознайомитися з фінансовими показниками у зручній формі. Якщо під час обробки даних трапляється помилка, наприклад, через некоректний формат даних, бот інформує користувача про проблему, надіславши повідомлення з описом помилки. Таким чином, функція забезпечує гнучкий механізм перегляду агрегованих даних і обробки можливих винятків.

Реалізацію виводу таблиці з поточними даними у буфері можна побачити на рисунку 3.6

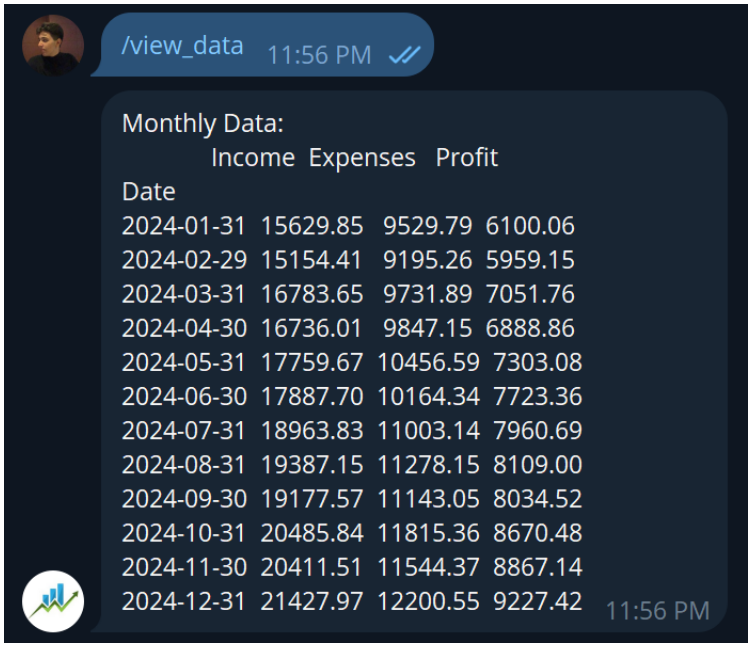


Рисунок 3.6. – Повідомлення бота з поточними даними у буфері.

```
@dp.message(Command("view_data"))
    async def view_data(message: Message):
        global data_buffer

        if data_buffer.empty:
            await message.answer("No data available. Please upload a file first.")
            return

        try:
            monthly_data = (
                data_buffer.set_index("Date")
                .resample("ME")
                .sum()[["Income", "Expenses"]]
                .assign(Profit=lambda x: x["Income"] - x["Expenses"])
            )
            table_message = monthly_data.to_string()
            await message.answer(f"Monthly Data:\n{table_message}")
        except Exception as e:
            await message.answer(f"Error viewing data: {e}")
```

Функція clear_data реалізує обробку команди /clear_data і відповідає за очищення збережених даних. Вона починається з оголошення глобального буфера data_buffer, який очищується шляхом створення нового порожнього

DataFrame. Потім перевіряється, чи існує файл DATA_FILE, де зберігалися раніше завантажені дані. Якщо файл існує, він видаляється з файлової системи.

Після успішного очищення даних бот надсилає повідомлення користувачу, що всі дані було видалено. Це забезпечує простий механізм для скидання стану програми, дозволяючи користувачам працювати із "чистого аркуша".

```
@dp.message(Command("clear_data"))
async def clear_data(message: Message):
    global data_buffer

    data_buffer = pd.DataFrame()  # Очищуємо буфер
    if os.path.exists(DATA_FILE):
        os.remove(DATA_FILE)  # Видаляємо файл, якщо він існує

    await message.answer("All data has been cleared.")
@dp.message(Command("clear_data"))
async def clear_data(message: Message):
    global data_buffer

    data_buffer = pd.DataFrame()  # Очищуємо буфер
    if os.path.exists(DATA_FILE):
        os.remove(DATA_FILE)  # Видаляємо файл, якщо він існує

    await message.answer("All data has been cleared.")
```

Функція show_summary обробляє команду /summary і надає користувачеві загальні фінансові підсумки на основі даних, завантажених у data_buffer. Спочатку функція перевіряє, чи буфер не порожній. Якщо дані відсутні, бот надсилає повідомлення із запитом завантажити файл.

Якщо дані доступні, виконується обчислення основних показників: загального доходу (total_income), загальних витрат (total_expenses) і прибутку (total_profit), який є різницею між доходами та витратами. Результати формуються у текстове повідомлення у зручному форматі, що включає всі три значення. Це повідомлення надсилається користувачеві.

У випадку, якщо під час обчислення виникає помилка (наприклад, через некоректні дані), функція обробляє виняток і повідомляє користувача про проблему. Це забезпечує надійність роботи функції та інформативність для користувача. Реалізація функції summary можна побачити на рисунку 3.7.

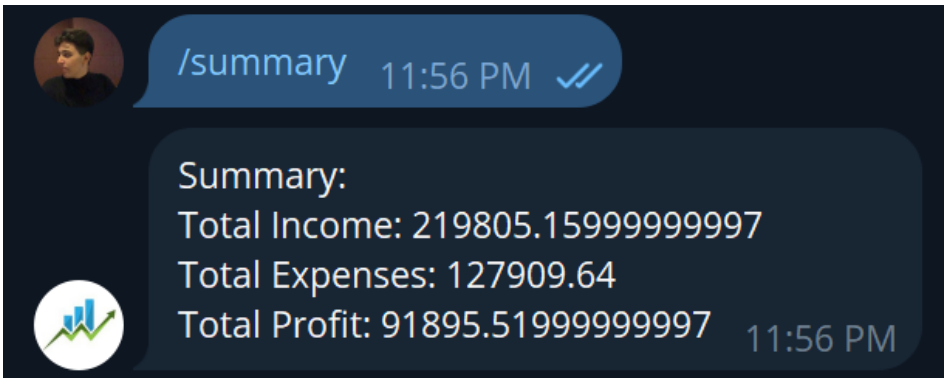


Рисунок 3.7. – Повідомлення бота з результатами функції summary.

```
@dp.message(Command("summary"))
async def show_summary(message: Message):
    global data_buffer

    if data_buffer.empty:
        await message.answer("No data available. Please upload a file first.")
        return

    try:
        total_income = data_buffer["Income"].sum()
        total_expenses = data_buffer["Expenses"].sum()
        total_profit = total_income - total_expenses

        summary = (
            f"Summary:\n"
            f"Total Income: {total_income}\n"
            f"Total Expenses: {total_expenses}\n"
            f"Total Profit: {total_profit}\n"
        )
        await message.answer(summary)
    except Exception as e:
        await message.answer(f"Summary error: {e}")
```

Функція analyze_data реалізує команду /analyze для аналізу фінансових даних, завантажених користувачем.

Обробка команди та перевірка даних.

```
@dp.message(Command("analyze"))
async def analyze_data(message: Message):
    global data_buffer

    if data_buffer.empty:
        await message.answer("No data to analyze. Please upload a file first.")
        return
```

Функція перевіряє, чи існують дані в глобальному буфері data_buffer. Якщо буфер порожній, бот надсилає повідомлення про відсутність даних і завершує виконання.

Збереження даних та запуск аналізу.

```
    try:
        data_buffer.to_csv(DATA_FILE, index=False)
        graphs, excel_path, forecast_data = perform_analysis(DATA_FILE)
```

Функція зберігає вміст буфера у файл DATA_FILE у форматі CSV, який використовується для аналізу. Після цього викликається функція perform_analysis, яка:

- Аналізує фінансові дані.
- Створює графіки та прогноз.
- Зберігає результати в Excel-файл.

```
        for graph_path in graphs:
            photo = FSInputFile(graph_path)
            await bot.send_photo(
                chat_id=message.chat.id,
                photo=photo,
                caption=f"Graph: {os.path.basename(graph_path)}",
            )

        excel_document = FSInputFile(excel_path)
        await bot.send_document(
            chat_id=message.chat.id,
            document=excel_document,
            caption="Forecast results in Excel.",
        )
```

Відправка результатів користувачеві.

Цей блок коду відправляє результати аналізу користувачеві, що кожен графік у списку graphs надсилається як фото із зазначенням його назви і Excel-файл із прогнозами надсилається як документ.

```
        for graph_path in graphs:
            photo = FSInputFile(graph_path)
            await bot.send_photo(
                chat_id=message.chat.id,
                photo=photo,
                caption=f"Graph: {os.path.basename(graph_path)}",
```

```
            )

        excel_document = FSInputFile(excel_path)
        await bot.send_document(
            chat_id=message.chat.id,
            document=excel_document,
            caption="Forecast results in Excel.",
        )
```

Обробка помилок

```
  except FileNotFoundError:
        await message.answer("Analysis error: Data file not found.")
    except ValueError as e:
        await message.answer(f"Data issue: {e}")
    except Exception as e:
        await message.answer(f"Unexpected error during analysis: {e}")
```

Передбачено **три рівні обробки помилок:**

- Відсутність файлу для аналізу.
- Помилки у форматі чи даних файлу.
- Непередбачені виключення, які надсилаються як текстові повідомлення.

Запуск опитування бота.

```
    try:
        await dp.start_polling(bot)
    finally:
        await bot.session.close()3.4.
```

Завершальна частина забезпечує запуск процесу обробки повідомлень Telegram-бота. Після зупинки роботи сесія бота автоматично закривається.

3.3. Налаштування аналітичної частини чат-бота

Далі розглянемо функціонал і структуру коду, призначеного для аналізу фінансових даних. Код базується на використанні популярних бібліотек Python для роботи з даними та візуалізації. Ми детально проаналізуємо, як реалізуються завантаження, обробка та аналіз даних, створення графіків і прогнозів, а також збереження результатів у зручному для користувача форматі. Це дозволить зрозуміти основні принципи побудови подібних аналітичних рішень.

Налаштування бекенду для побудови графіків

```
matplotlib.use("Agg")
```

Цей рядок вказує бібліотеці matplotlib використовувати бекаенд Agg.

Agg (Anti-Grain Geometry) — це рендеринг-бекаенд, який дозволяє створювати графіки без відображення їх на екрані. Це корисно у сценаріях, коли код запускається на сервері або середовищі без графічного інтерфейсу (наприклад, у веб-додатках). У цьому випадку графіки зберігаються безпосередньо у файли (наприклад, PNG).

Перевірка наявності файлу.

```
if not os.path.exists(file_path):
    raise FileNotFoundError(f"The file '{file_path}' does not exist.")
```

Перевіряє, чи існує файл за заданим шляхом file_path. Якщо файл відсутній, генерується помилка FileNotFoundError.

Завантаження та валідація даних.

```
data = pd.read_csv(file_path, parse_dates=["Date"])
required_columns = {"Date", "Income", "Expenses"}
if not required_columns.issubset(data.columns):
    raise ValueError(f"Missing required columns: {', '.join(required_columns)}")
```

Дані завантажуються з файлу CSV. Стовпець Date парсується як дата. Якщо у файлі відсутні обов'язкові колонки (Date, Income, Expenses), генерується помилка ValueError.

Агрегація та підготовка даних.

```
data["Profit"] = data["Income"] - data["Expenses"]
data = data.set_index("Date")
monthly_data = data.resample("ME").sum()
```

Розраховується стовпець Profit (різниця між доходами та витратами). Дані агрегуються за кварталами (resample("ME")).

Розрахунок тренду.

```
X = np.arange(1, len(monthly_data) + 1)
Y = monthly_data["Profit"]
a1, a0 = np.polyfit(X, Y, 1)
monthly_data["Trend"] = a0 + a1 * X
```

Розраховується лінійний тренд для Profit за допомогою методу найменших квадратів (np.polyfit).

Прогнозування.

```
forecast_periods = 5
```

```
    forecast_index = pd.date_range(start=monthly_data.index[-1], periods=forecast_periods + 1, freq="ME")[1:]
    forecast_trend = a0 + a1 * np.arange(len(monthly_data) + 1, len(monthly_data) + forecast_periods + 1)
    forecast_add = forecast_trend + monthly_data["Seasonality_Add"].mean()
    forecast_mult = forecast_trend * monthly_data["Seasonality_Mult"].mean()
```

Виконується прогнозування на 5 періодів вперед, враховуючи тренд, адитивну та мультиплікативну сезонність. Повідомлення бота з результатами прогнозування можна побачити на рис 3.8.

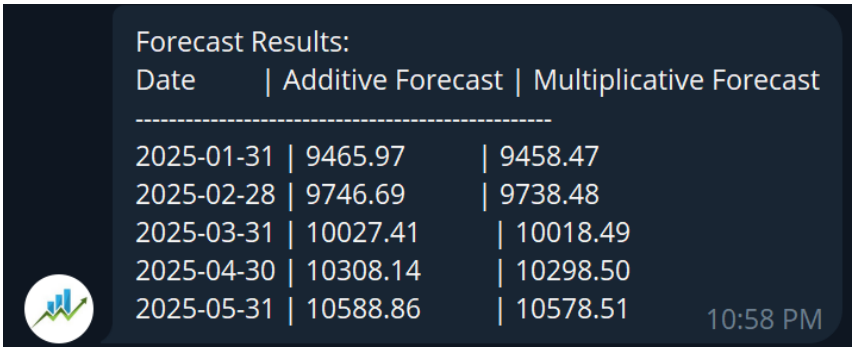


Рисунок. 3.8. – Результат прогнозування у чаті користувача

Створення графіків.

- Адитивна модель:

```
    plt.figure(figsize=(14, 7))
    plt.plot(monthly_data.index, monthly_data["Profit"], label="Actual Profit", color="blue")
    plt.plot(monthly_data.index, monthly_data["MA_Profit"], label="Moving Average Profit", color="green")
    plt.plot(monthly_data.index, monthly_data["Trend"], label="Trend (Additive)", linestyle="--", color="orange")
    plt.plot(forecast_index, forecast_add, label="Forecast (Additive)", linestyle=":", color="red")
    plt.title("Profit Analysis: Additive Model")
    plt.savefig(additive_graph_path)
```

Згенеровану функцію з адитивною моделлю можна побачити на рисунку 3.9.

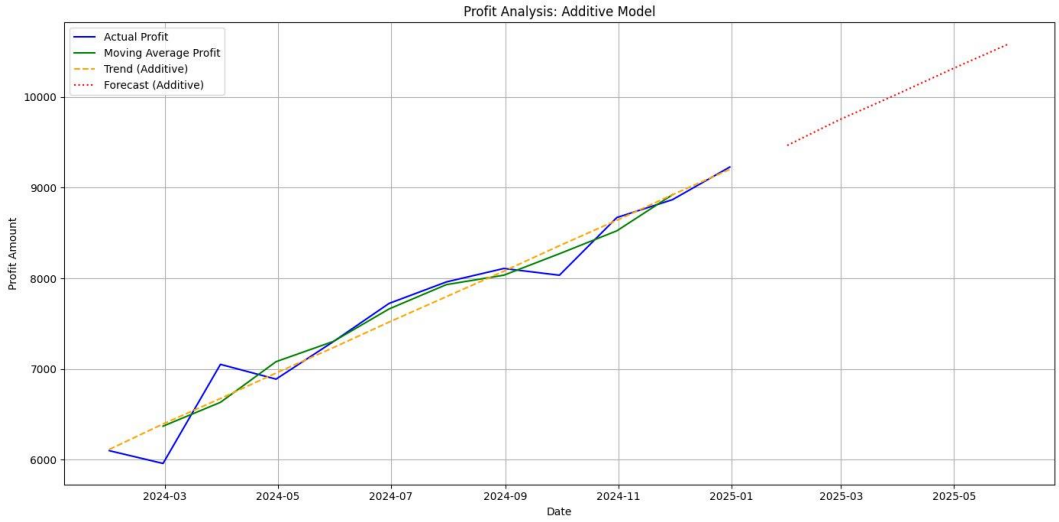


Рисунок 3.9. – Графік функції з адитивною моделлю

- Мультиплікативна модель:

```
plt.figure(figsize=(14, 7))
plt.plot(monthly_data.index, monthly_data["Profit"], label="Actual Profit", color="blue")
plt.plot(monthly_data.index, monthly_data["MA_Profit"], label="Moving Average Profit", color="green")
plt.plot(monthly_data.index, monthly_data["Trend"], label="Trend (Multiplicative)", linestyle="--", color="orange")
plt.plot(forecast_index, forecast_mult, label="Forecast (Multiplicative)", linestyle=":", color="red")
plt.title("Profit Analysis: Multiplicative Model")
plt.savefig(multiplicative_graph_path)
```

Згенеровану функцію з мультиплікативною моделлю можна побачити на рисунку 3.10.

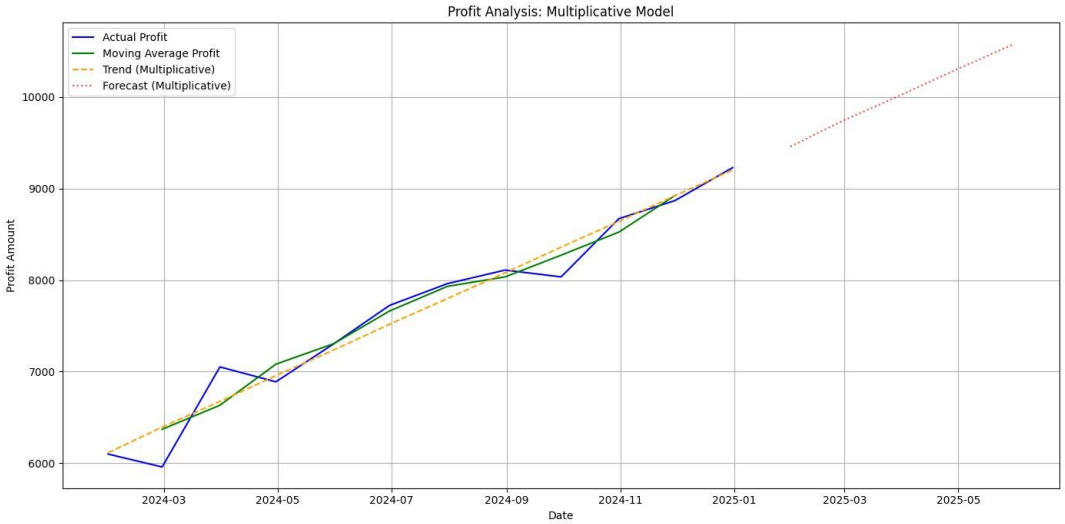


Рисунок 3.10. – Графік функції з мультиплікативною моделлю

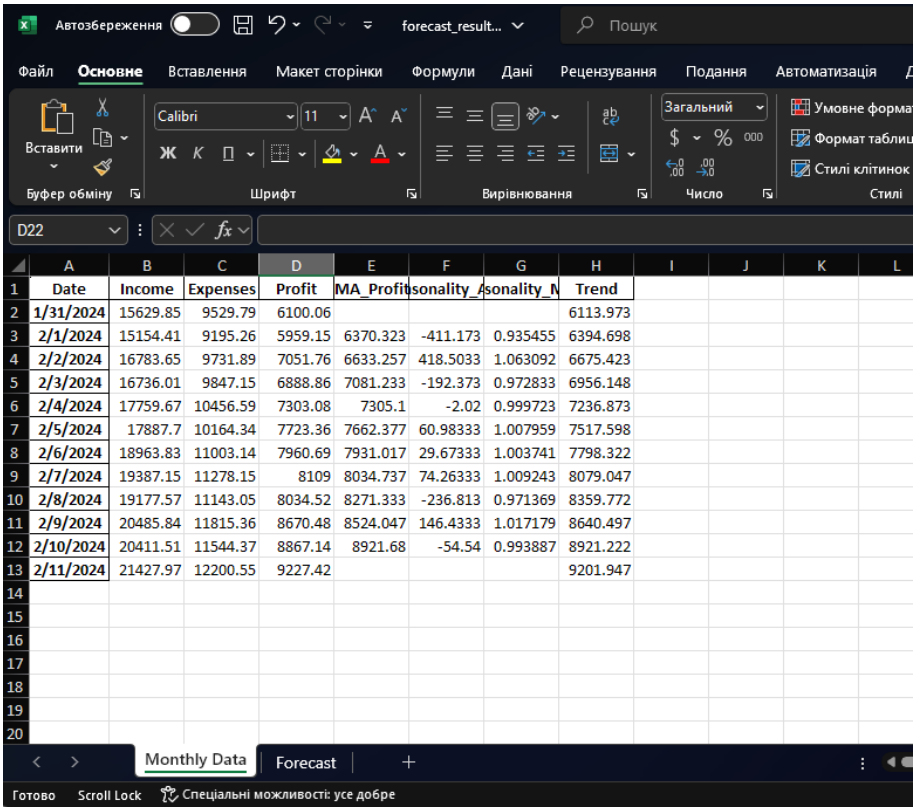
Графіки зберігаються як зображення для адитивної та мультиплікативної моделей.

Збереження результатів у Excel.

```
with pd.ExcelWriter(excel_output_path, engine="xlsxwriter") as writer:
    monthly_data.to_excel(writer, sheet_name="Monthly Data")
    forecast_df = pd.DataFrame({
        "Forecast Date": forecast_index,
        "Forecast Additive": forecast_add,
        "Forecast Multiplicative": forecast_mult
    })
    forecast_df.to_excel(writer, sheet_name="Forecast")
```

Результати зберігаються у файл Excel, де створюються два аркуші:

- Monthly Data з розрахунками рис 3.11



	A	B	C	D	E	F	G	H
1	Date	Income	Expenses	Profit	MA_Profit	sonality_A	sonality_M	Trend
2	1/31/2024	15629.85	9529.79	6100.06				6113.973
3	2/1/2024	15154.41	9195.26	5959.15	6370.323	-411.173	0.935455	6394.698
4	2/2/2024	16783.65	9731.89	7051.76	6633.257	418.5033	1.063092	6675.423
5	2/3/2024	16736.01	9847.15	6888.86	7081.233	-192.373	0.972833	6956.148
6	2/4/2024	17759.67	10456.59	7303.08	7305.1	-2.02	0.999723	7236.873
7	2/5/2024	17887.7	10164.34	7723.36	7662.377	60.98333	1.007959	7517.598
8	2/6/2024	18963.83	11003.14	7960.69	7931.017	29.67333	1.003741	7798.322
9	2/7/2024	19387.15	11278.15	8109	8034.737	74.26333	1.009243	8079.047
10	2/8/2024	19177.57	11143.05	8034.52	8271.333	-236.813	0.971369	8359.772
11	2/9/2024	20485.84	11815.36	8670.48	8524.047	146.4333	1.017179	8640.497
12	2/10/2024	20411.51	11544.37	8867.14	8921.68	-54.54	0.993887	8921.222
13	2/11/2024	21427.97	12200.55	9227.42				9201.947
14								
15								
16								
17								
18								
19								
20								

Рисунок 3.11. – Аркуш надісланого файлу у форматі Excel з агрегацією по кварталах

- Forecast з даними прогнозу рис. 3.12.

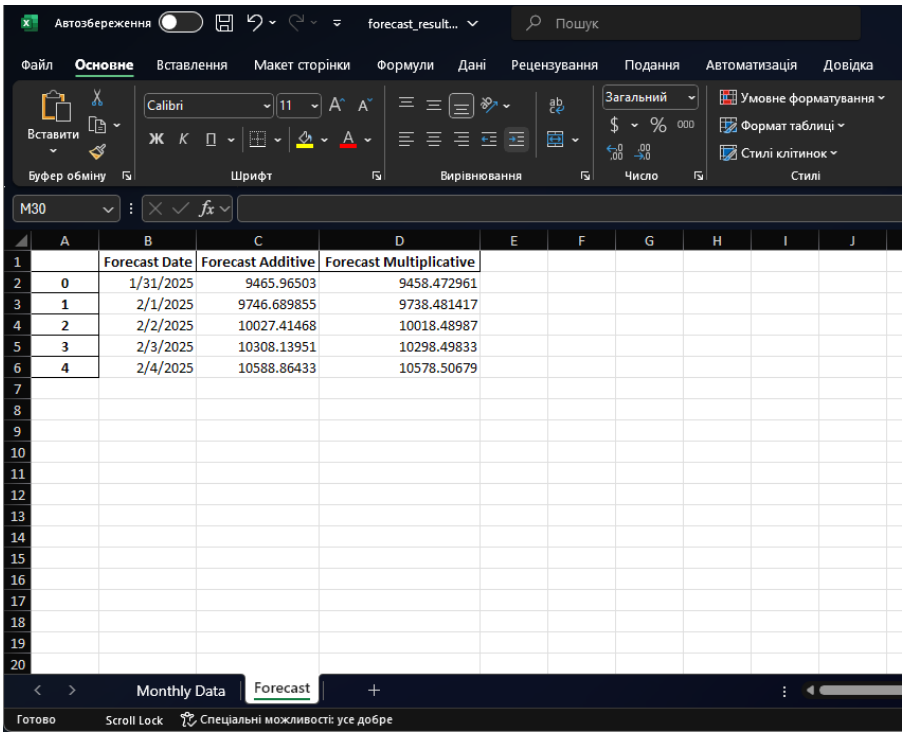


Рисунок 3.12 – Аркуш надісланого файлу
у форматі Excel з даними прогнозу

3.4. Висновки до третього розділу

Було розглянуто реалізацію програмного забезпечення для фінансового Telegram-бота, починаючи з вибору інструментів і закінчуючи впровадженням функціоналу. Основною мовою програмування обрано Python завдяки його простоті, широким можливостям інтеграції та багатій екосистемі бібліотек. Для роботи з Telegram Bot API використано бібліотеку Aiogram, яка забезпечує асинхронну обробку подій.

Розроблений Telegram-бот підтримує функції завантаження даних, їхнього аналізу, формування прогнозів, візуалізації результатів у вигляді графіків та збереження підсумків у файл Excel.

У рамках реалізації аналітична частина базується на обчисленні трендів і сезонності за адитивною та мультиплікативною моделями, що дозволяє точно прогнозувати зміни прибутків. Усі результати виводяться у форматі, зручному для користувачів, і надаються через чат у Telegram.

Використання мультиплікативної моделі забезпечило вищу точність прогнозів для даних із варіативною амплітудою, що підтверджується аналізом побудованих графіків та прогнозів.

Розроблений бот демонструє інтегрованість сучасних технологій у фінансовий аналіз і зручність використання для автоматизації задач управління персональними даними.

ВИСНОВКИ

У даній кваліфікаційній роботі досліджено підходи до вирішення завдань автоматизації процесів управління особистими фінансами та вирішено завдання розробки програмного забезпечення для автоматизації управління особистими фінансами, яке базується на методах аналізу часових рядів. Робота охоплює аналіз предметної області, визначення вимог до системи, огляд існуючих підходів і їхніх обмежень, а також розробку алгоритмів і програмного забезпечення з подальшим тестуванням отриманого рішення.

У першому розділі було досліджено ключові поняття часових рядів і основні підходи до їх аналізу та прогнозування. Особливу увагу приділено методам згладжування, розрахунку трендових і сезонних компонент, які стали базою для подальшого проєктування алгоритмів.

Другий розділ присвячено розробці алгоритмічного забезпечення для прогнозування прибутків користувачів. У цьому розділі обґрунтовано вибір адитивної та мультиплікативної моделей, реалізовано алгоритм, який включає метод Крамера і ковзної середньої для аналізу трендів і сезонних коливань.

У третьому розділі охоплено процес створення програмного забезпечення, зокрема інтеграцію алгоритмів у застосунок на базі Python. Реалізовано функціонал для автоматизованого створення прогнозів, їхньої візуалізації та експорту у формат Excel. Окрім того, забезпечено можливість інтеграції програмного забезпечення з Telegram-ботом, що спрощує доступ до системи для кінцевих користувачів.

Загалом, виконана кваліфікаційна робота демонструє, що розроблена система є ефективним інструментом для прогнозування особистих фінансів. Вона забезпечує автоматизацію фінансового аналізу, покращує точність прогнозів і сприяє підвищенню ефективності управління особистими доходами та витратами завдяки зручному і доступному інтерфейсу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бідюк П. І., Романенко В. Д., Тимощук О. Л. *Аналіз часових рядів: Навчальний посібник.* Київ: НТУУ КПІ, 2013. – 599 с.

2. Ейлін Нільсен. *Практичний аналіз часових рядів. Прогнозування зі статистикою і машинне навчання*, 2020. – 532 с.

3. Hyndman R. J., Athanasopoulos G. *Forecasting: Principles and Practice.* OTexts, 2018. – 380 p. URL: https://otexts.com/fpp3/ (date of access: 20.11.2024).

4. Montgomery D. C., Jennings C. L., Kulahci M. *Introduction to Time Series Analysis and Forecasting.* Wiley, 2015. – 672 p.

5. Enders W. *Applied Econometric Time* Series. Wiley, 2010. – 517 p.

6. Chatfield C., Xing H. *The Analysis of Time Series: An Introduction with R*. 2019. – 414 p.

7. Tsay R. S. *Multivariate Time Series Analysis: With R and Financial Applications*. 2014. – 520 p.

8. Kantz H. *Nonlinear Time Series Analysis*. 2015. – 388 p.

9. *Using Asyncio in Python: Understanding Python's Asynchronous Programming Features*. 2020. – 163 p

10. Chan J., Chung R., Huang J. *Python API Development Fundamentals*. 2019. – 372 p.

11. Hillar G. C. *Building RESTful Python Web Services*. 2016. – 418 p.

12. Lubanovic B. *FastAPI: Modern Python Web Development*. 2023. – 277 p.

13. Joseph M., Tackes J. *Modern Time Series Forecasting with Python: Industry-ready Machine Learning and Deep Learning Time Series Analysis with PyTorch and pandas*. 2024. – 658 p.

14. Atwan T. A. *Time Series Analysis with Python Cookbook: Practical Recipes for Exploratory Data Analysis, Data Preparation, Forecasting, and Model Evaluation*. 2022. – 630 p.

15. Joseph M. *Modern Time Series Forecasting with Python: Explore Industry-ready Time Series Forecasting Using Modern Machine Learning and Deep Learning*. 2022. – 552 p.

16. Cerqueira V., Roque L. *Deep Learning for Time Series Cookbook: Use PyTorch and Python Recipes for Forecasting, Classification, and Anomaly Detection*. 2024. – 274 p.

17. Time Series. ScienceDirect. URL: https://www.sciencedirect.com/topics/social-sciences/time-series (date of access: 18.11.2024).

18. Wiley Online Library. Journal of Time Series Analysis. URL: https://onlinelibrary.wiley.com/journal/14679892 (date of access: 18.11.2024).

19. Tableau. What is Time Series Analysis? URL: https://www.tableau.com/analytics/what-is-time-series-analysis (date of access: 18.11.2024).

20. Intelliarts. Time Series Analysis Examples. URL: https://intelliarts.com/blog/time-series-analysis-examples/ (date of access: 18.11.2024).

21. QuantStart. Time Series Analysis Articles. URL: https://www.quantstart.com/articles/topic/time-series-analysis/ (date of access: 18.11.2024).

22. Learn Simplified. Time Series Analysis: What, Why, and How. Medium. URL: https://medium.com/@learn-simplified/time-series-analysis-what-why-and-how-d01a208d2073 (date of access: 18.11.2024).

23. Python Documentation. URL: https://docs.python.org/3/ (date of access: 18.11.2024).

24. Telegram Bot API Documentation. Telegram. URL: https://core.telegram.org/bots/api (date of access: 18.11.2024).

25. PyCharm Documentation. URL: https://www.jetbrains.com/help/pycharm (date of access: 18.11.2024).

main.py

```
from aiogram import Bot, Dispatcher, types
from aiogram.types import Message, ReplyKeyboardMarkup, KeyboardButton
from aiogram.filters import Command
import pandas as pd
import asyncio
from analysis import perform_analysis
import os
from aiogram.types.input_file import FSInputFile


API_TOKEN = "7612743717:AAFysM0ZZT9T0qi4cchfDt9eimXStbHcsvs"
DATA_FILE = "client_finance_data.csv"  # Основний файл для даних
data_buffer = pd.DataFrame()  # Глобальний буфер даних


async def main():
    bot = Bot(token=API_TOKEN)
    dp = Dispatcher()

    # Клавіатура з командами
    keyboard = ReplyKeyboardMarkup(
        keyboard=[
            [KeyboardButton(text="/analyze")],
            [KeyboardButton(text="/upload")],
            [KeyboardButton(text="/summary")],
            [KeyboardButton(text="/view_data"), KeyboardButton(text="/clear_data")],
        ],
        resize_keyboard=True,
    )

    @dp.message(Command("start"))
    async def start_command(message: Message):
        await message.answer(
            "Welcome! Use the buttons below to select a command.",
            reply_markup=keyboard,
        )

    @dp.message(Command("upload"))
    async def upload_file(message: Message):
        global data_buffer
```

```
    if not message.document:
        await message.answer("Please upload a CSV file with the following columns:
Date, Income, Expenses.")
        return

    file_id = message.document.file_id
    file_name = message.document.file_name
    temp_file_path = f"./{file_name}"

    try:
        # Download the file
        file = await bot.get_file(file_id)
        await bot.download_file(file.file_path, destination=temp_file_path)
        await message.answer("File received. Processing...")

        if not file_name.endswith(".csv"):
            raise ValueError("The file is not in CSV format.")

        # Load data
        data = pd.read_csv(temp_file_path)
        print(f"Temp file path: {temp_file_path}")
        print(f"Data preview:\n{data.head()}")  # Debug step 1

        # Validate required columns
        required_columns = {"Date", "Income", "Expenses"}
        if not required_columns.issubset(data.columns):
            raise ValueError(f"Missing required columns: {',
'.join(required_columns)}")

        data["Date"] = pd.to_datetime(data["Date"], errors="coerce")
        if data["Date"].isna().any():
            raise ValueError("Invalid date values detected.")
        if data[["Income", "Expenses"]].isna().any().any():
            raise ValueError("Income or Expenses columns contain invalid values.")

        # Update buffer
        if not data_buffer.empty:
            data_buffer = pd.concat([data_buffer, data],
ignore_index=True).drop_duplicates()
        else:
            data_buffer = data
        data_buffer.sort_values(by="Date", inplace=True)
```

```
        print(f"Updated data buffer:\n{data_buffer}")  # Debug step 2

        # Save to file
        data_buffer.to_csv(DATA_FILE, index=False)
        await message.answer("File successfully uploaded and processed!")

    except Exception as e:
        await message.answer(f"File upload error: {e}")
    finally:
        if os.path.exists(temp_file_path):
            os.remove(temp_file_path)

@dp.message(Command("view_data"))
async def view_data(message: Message):
    global data_buffer

    if data_buffer.empty:
        await message.answer("No data available. Please upload a file first.")
        return

    try:
        # Формування таблиці
        monthly_data = (
            data_buffer.set_index("Date")
            .resample("ME")
            .sum()[["Income", "Expenses"]]
            .assign(Profit=lambda x: x["Income"] - x["Expenses"])
        )
        table_message = monthly_data.to_string()
        await message.answer(f"Monthly Data:\n{table_message}")
    except Exception as e:
        await message.answer(f"Error viewing data: {e}")

@dp.message(Command("clear_data"))
async def clear_data(message: Message):
    global data_buffer

    data_buffer = pd.DataFrame()  # Чистимо буфер
    if os.path.exists(DATA_FILE):
        os.remove(DATA_FILE)

    await message.answer("All data has been cleared.")
```

```
@dp.message(Command("summary"))
async def show_summary(message: Message):
    global data_buffer

    if data_buffer.empty:
        await message.answer("No data available. Please upload a file first.")
        return

    try:
        total_income = data_buffer["Income"].sum()
        total_expenses = data_buffer["Expenses"].sum()
        total_profit = total_income - total_expenses

        summary = (
            f"Summary:\n"
            f"Total Income: {total_income}\n"
            f"Total Expenses: {total_expenses}\n"
            f"Total Profit: {total_profit}\n"
        )
        await message.answer(summary)
    except Exception as e:
        await message.answer(f"Summary error: {e}")

@dp.message(Command("analyze"))
async def analyze_data(message: Message):
    global data_buffer

    if data_buffer.empty:
        await message.answer("No data to analyze. Please upload a file first.")
        return

    try:
        data_buffer.to_csv(DATA_FILE, index=False)  # Зберегаємл буфер перед
аналізом
        graphs, excel_path, forecast_data, adequacy_info =
perform_analysis(DATA_FILE)

        for graph_path in graphs:
            photo = FSInputFile(graph_path)
            await bot.send_photo(
                chat_id=message.chat.id,
                photo=photo,
                caption=f"Graph: {os.path.basename(graph_path)}",
```

```
            )

        excel_document = FSInputFile(excel_path)
        await bot.send_document(
            chat_id=message.chat.id,
            document=excel_document,
            caption="Forecast results in Excel.",
        )

        forecast_table = forecast_data.to_string(index=False)
        adequacy_message = (
            f"Model Adequacy:\n"
            f"Additive Model - R²: {adequacy_info['Additive']['R2']:.2f}, "
            f"DW: {adequacy_info['Additive']['DW']:.2f}, "
            f"MSE: {adequacy_info['Additive']['MSE']:.2f}, "
            f"MAE: {adequacy_info['Additive']['MAE']:.2f}\n"
            f"Multiplicative Model - R²: {adequacy_info['Multiplicative']['R2']:.2f}, "
            f"DW: {adequacy_info['Multiplicative']['DW']:.2f}, "
            f"MSE: {adequacy_info['Multiplicative']['MSE']:.2f}, "
            f"MAE: {adequacy_info['Multiplicative']['MAE']:.2f}\n\n"
            f"Best Model: {adequacy_info['Best Model']}\n\n"
            f"Forecast Table:\n{forecast_table}"
        )

        await message.answer(adequacy_message)

    except Exception as e:
        await message.answer(f"Analysis error: {e}")

try:
    await dp.start_polling(bot)
finally:
    await bot.session.close()


if __name__ == "__main__":
    asyncio.run(main())
```

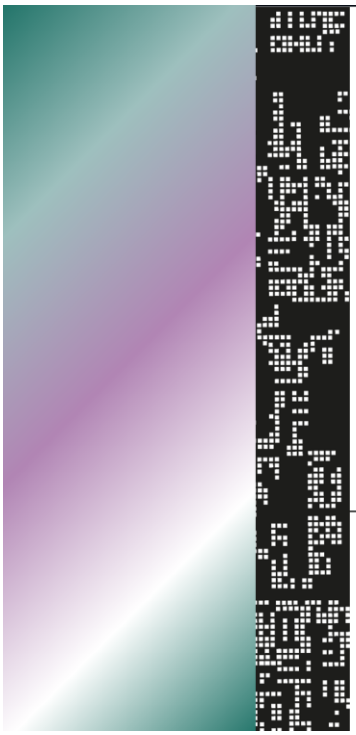
analysis.py

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import os
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from statsmodels.stats.stattools import durbin_watson


# Використовуємо бэкенд Agg для збереження графіків
matplotlib.use("Agg")


def perform_analysis(file_path):
    # Перевірка наявності файлу
    if not os.path.exists(file_path):
        raise FileNotFoundError(f"The file '{file_path}' does not exist.")

    # Завантаження даних
    data = pd.read_csv(file_path, parse_dates=["Date"])
    required_columns = {"Date", "Income", "Expenses"}
    if not required_columns.issubset(data.columns):
        raise ValueError(f"Missing required columns: {', '.join(required_columns)}")

    data["Profit"] = data["Income"] - data["Expenses"]
    data = data.set_index("Date")
    quarterly_data = data.resample("QE").sum()

    # Рухоме середнє (Trend)
    window_size = 2
    quarterly_data["MA_Profit"] =
quarterly_data["Profit"].rolling(window=window_size, center=True).mean()

    # Адитивна та мультиплікативна моделі
    quarterly_data["Seasonality_Add"] = quarterly_data["Profit"] -
quarterly_data["MA_Profit"]
    quarterly_data["Seasonality_Mult"] = quarterly_data["Profit"] /
quarterly_data["MA_Profit"]

    # Трендова модель
    X = np.arange(1, len(quarterly_data) + 1)
    Y = quarterly_data["Profit"]
    a1, a0 = np.polyfit(X, Y, 1)
```

```
    quarterly_data["Trend"] = a0 + a1 * X

    # Оцінка адекватності
    r2_additive = r2_score(Y, quarterly_data["Trend"])
    r2_multiplicative = r2_score(Y, quarterly_data["Trend"] *
quarterly_data["Seasonality_Mult"].mean())
    dw_additive = durbin_watson(Y - quarterly_data["Trend"])
    dw_multiplicative = durbin_watson(Y - (quarterly_data["Trend"] *
quarterly_data["Seasonality_Mult"].mean()))

    # Розрахунок точності (СООП і САО)
    forecast_periods = 4
    forecast_index = pd.date_range(start=quarterly_data.index[-1],
periods=forecast_periods + 1, freq="QE")[1:]
    forecast_trend = a0 + a1 * np.arange(len(quarterly_data) + 1, len(quarterly_data) +
forecast_periods + 1)
    forecast_add = forecast_trend + quarterly_data["Seasonality_Add"].mean()
    forecast_mult = forecast_trend * quarterly_data["Seasonality_Mult"].mean()

    additive_error_mse = mean_squared_error(Y, quarterly_data["Trend"])
    multiplicative_error_mse = mean_squared_error(Y, quarterly_data["Trend"] *
quarterly_data["Seasonality_Mult"].mean())
    additive_error_mae = mean_absolute_error(Y, quarterly_data["Trend"])
    multiplicative_error_mae = mean_absolute_error(Y, quarterly_data["Trend"] *
quarterly_data["Seasonality_Mult"].mean())

    # Визначення моделі з кращою точністю
    best_model = "Additive" if additive_error_mse < multiplicative_error_mse else
"Multiplicative"

    # Шляхи для результатів
    output_dir = "output"
    os.makedirs(output_dir, exist_ok=True)
    additive_graph_path = os.path.join(output_dir, "additive_model_graph.png")
    multiplicative_graph_path = os.path.join(output_dir,
"multiplicative_model_graph.png")
    excel_output_path = os.path.join(output_dir, "forecast_results.xlsx")

    # Збереження графіку для адитивної моделі
    plt.figure(figsize=(14, 7))
    plt.plot(quarterly_data.index, quarterly_data["Profit"], label="Actual Profit",
color="blue")
    plt.plot(quarterly_data.index, quarterly_data["Trend"], label="Trend (Additive)",
```

```
linestyle="--", color="orange")
    plt.plot(forecast_index, forecast_add, label="Forecast (Additive)", linestyle=":",
color="red")
    plt.title("Profit Analysis: Additive Model")
    plt.xlabel("Date")
    plt.ylabel("Profit Amount")
    plt.legend()
    plt.grid()
    plt.tight_layout()
    plt.savefig(additive_graph_path)
    plt.close()

    # Збереження графіку для мультиплікативної моделі
    plt.figure(figsize=(14, 7))
    plt.plot(quarterly_data.index, quarterly_data["Profit"], label="Actual Profit",
color="blue")
    plt.plot(quarterly_data.index, quarterly_data["Trend"], label="Trend
(Multiplicative)", linestyle="--", color="orange")
    plt.plot(forecast_index, forecast_mult, label="Forecast (Multiplicative)",
linestyle=":", color="red", alpha=0.7)
    plt.title("Profit Analysis: Multiplicative Model")
    plt.xlabel("Date")
    plt.ylabel("Profit Amount")
    plt.legend()
    plt.grid()
    plt.tight_layout()
    plt.savefig(multiplicative_graph_path)
    plt.close()

    # Збереження результатів у Excel
    with pd.ExcelWriter(excel_output_path, engine="xlsxwriter") as writer:
        quarterly_data.to_excel(writer, sheet_name="Quarterly Data")
        forecast_df = pd.DataFrame({
            "Forecast Date": forecast_index,
            "Forecast Additive": forecast_add,
            "Forecast Multiplicative": forecast_mult
        })
        forecast_df.to_excel(writer, sheet_name="Forecast")

    adequacy_info = {
        "Additive": {"R2": r2_additive, "DW": dw_additive, "MSE":
additive_error_mse, "MAE": additive_error_mae},
        "Multiplicative": {"R2": r2_multiplicative, "DW": dw_multiplicative, "MSE":
```

```
multiplicative_error_mse, "MAE": multiplicative_error_mae},
        "Best Model": best_model,
    }

    return [additive_graph_path, multiplicative_graph_path], excel_output_path,
forecast_df.head(), adequacy_info
```



ПРЕЗЕНТАЦІЯ ДО ДИПЛОМНОЇ РОБОТИ

На тему:

“Розроблення програмного забезпечення для автоматизації управління особистими фінансовими процесами”

Студент групи МгЗІТ1-23:
Валерій Микитенко

Науковий керівник:
к.т.н., доц. Тетяна Демківська

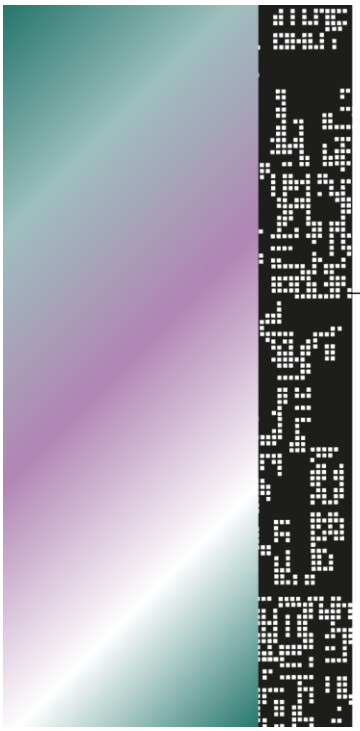
Об’єкт і предмет дослідження

Об’єкт дослідження

Об’єктом дослідження є процес аналізу часових рядів для фінансового прогнозування.

Предмет дослідження

Предметом дослідження є алгоритми та методи аналізу й прогнозування часових рядів для особистих фінансових задач.



Мета і завдання роботи

МЕТА

Розробити алгоритм і програмне забезпечення для прогнозування фінансів користувачів, що засноване на аналізі часових рядів із використанням адитивних і мультиплікативних моделей.

ЗАВДАННЯ

- **Дослідити особливості часових рядів** та основні підходи до їх аналізу;
- **Розробити алгоритмічне забезпечення** для прогнозування;
- **Реалізувати програмне забезпечення**, яке базується на розробленому алгоритмі;
- **Створити зручний і доступний інтерфейс** у вигляді Telegram-бота

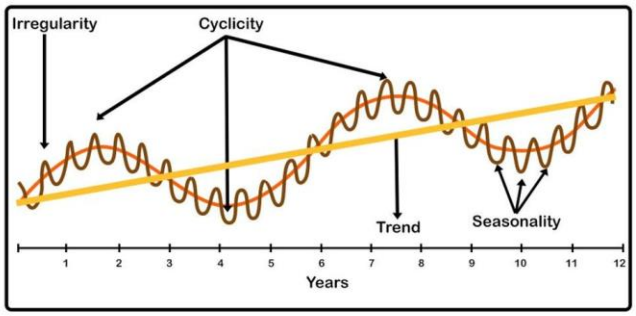
Актуальність

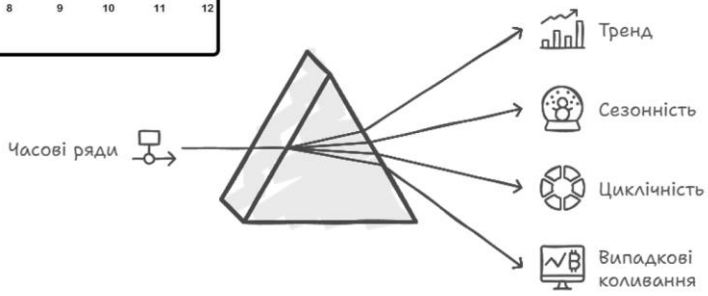
01 В умовах нестабільної економіки особисте фінансове планування стає критично важливим.

02 Існуючі інструменти не завжди враховують сезонність, тренди та індивідуальні особливості користувачів.

03 Сучасні технології, зокрема інтеграція у Telegram, забезпечують доступність фінансового менеджменту для широкої аудиторії.

Часові ряди





Алгоритмічне забезпечення

Алгоритм для прогнозу фінансів користувачів має таку структуру

Попередня обробка даних:	• Формування **показника прибутку** на основі доходів і витрат.
Розрахунок компонент часового ряду:	• Згладжування часового ряду **методом ковзної середньої**. • **Розрахунок тренду** за допомогою методу Крамера. • **Визначення сезонної компоненти** для адитивної і мультиплікативна моделі.
Моделювання рівнів ряду:	• Аналітичне **вирівнювання рівнів ряду** з використанням тренду та сезонності.
Прогнозування	• Розрахунок прогнозних значень на основі тренду та сезонності • Застосування формул для **адитивної** або **мультиплікативної моделі**
Оцінка точності прогнозу:	• Використання метрик **САО** (середнє абсолютне відхилення) та **СООП** (середнє відхилення відносно помилок).

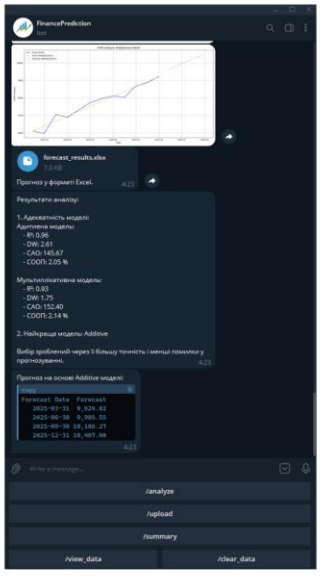
Мова програмування і обрані технології

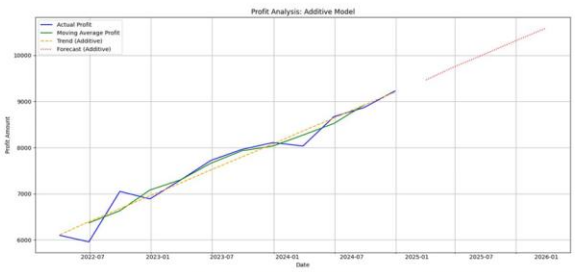
Програмне забезпечення реалізовано на Python, як інструмент для роботи з даними і створення бота.

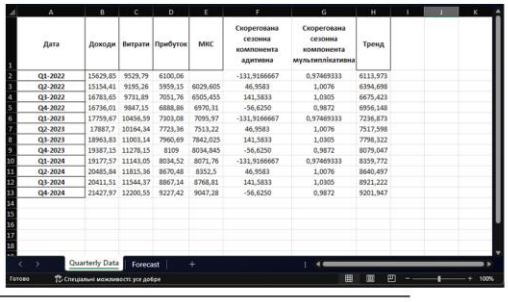
Використані технології:

- **Бібліотека Pandas**: обробка та аналіз таблиць даних.
- **Бібліотека NumPy**: математичні операції, зокрема з часовими рядами.
- **Бібліотека Matplotlib**: побудова графіків і візуалізація результатів.
- **Telegram API** для створення бота, що надає функціонал у вигляді інтерактивного бота.

Відображення інтерфейсу, текстової інформації, графіку і згенерованого excel файлу







	A	B	C	D	E	F	G	H
1	Дата	Доходи	Витрати	Прибуток	МКС	Скорегована сезонна компонента адитивна	Скорегована сезонна компонента мультиплікативна	Тренд
2	Q1-2022	15629,85	9529,79	6100,06		-131,9166667	0,97469333	6113,973
3	Q2-2022	15154,41	9195,26	5959,15	6029,605	46,9583	1,0076	6394,698
4	Q3-2022	16783,65	9731,89	7051,76	6505,455	141,5833	1,0305	6675,423
5	Q4-2022	16736,01	9847,15	6888,86	6970,31	-56,6250	0,9872	6956,148
6	Q1-2023	17759,67	10456,59	7303,08	7095,97	-131,9166667	0,97469333	7236,873
7	Q2-2023	17887,7	10164,34	7723,36	7513,22	46,9583	1,0076	7517,598
8	Q3-2023	18963,83	11003,14	7960,69	7842,025	141,5833	1,0305	7798,322
9	Q4-2023	19387,15	11278,15	8109	8034,845	-56,6250	0,9872	8079,047
10	Q1-2024	19177,57	11143,05	8034,52	8071,76	-131,9166667	0,97469333	8359,772
11	Q2-2024	20485,84	11815,36	8670,48	8352,5	46,9583	1,0076	8640,497
12	Q3-2024	20411,51	11544,37	8867,14	8768,81	141,5833	1,0305	8921,222
13	Q4-2024	21427,97	12200,55	9227,42	9047,28	-56,6250	0,9872	9201,947
14								
15								
16								
17								
18								

Quarterly Data | Forecast

Результати і тестування

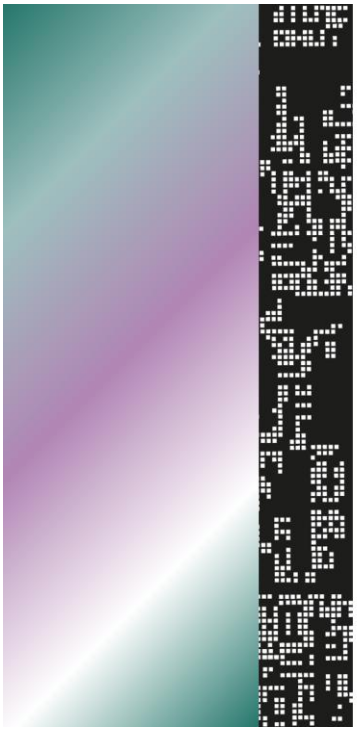
Розроблене програмне забезпечення успішно протестоване на реальних даних. Тестування показало:

Згідно розрахунків найкращою моделлю була обрана адитивна.
Через високу отриману якість і точність моделі.

Ефективність алгоритмів:
Адитивна модель виявилася ефективною для отриманих даних.

Приклади використання:
На основі даних особистих доходів користувач отримав прогноз на наступні три квартали року з врахуванням сезонності та трендів.
Прогнози допомогли покращити планування бюджету та оцінити майбутні фінансові ризики.

Переваги:
Динамічне оновлення прогнозів.
Швидка інтеграція в повсякденне фінансове планування.



Висновки

Досягнення:
- Розроблено ефективний алгоритм, який дозволяє прогнозувати фінансові показники.
- Реалізовано програму на Python, що забезпечує аналіз даних і їх візуалізацію.
- Інтеграція з Telegram створила простий і доступний інструмент для користувачів.

Практичне значення:
- Програмне забезпечення сприяє оптимізації фінансового планування.
- Може бути використане в особистому фінансовому менеджменті або інтегроване в корпоративні системи.

Перспективи розвитку:
- Розширення можливостей, наприклад, додавання модулів для оцінки фінансових ризиків.
- Впровадження прогнозування для груп користувачів (наприклад, сімейних бюджетів).
- Інтеграція з іншими платформами, такими як Google Sheets або фінансові CRM-системи.

Загальний висновок:
Робота досягла поставленої мети: створено програмне рішення для управління фінансами, яке поєднує точність, простоту використання та доступність.