**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ**
**Кафедра математичних методів системного аналізу**

До захисту допущено
Завідувач кафедри
_____ Оксана ТИМОЩУК
«__»_____ 2024 р.

# Дипломна робота

**на здобуття ступеня бакалавра**
**за освітньо-професійною програмою «Системний аналіз і управління»**
**спеціальності 124 "Системний аналіз"**
**на тему: «Прогнозування часових рядів для складських запасів»**

Виконала:
Студентка IV курсу, групи КА-04
Осадча Софія Олександрівна _____

Керівник:
Завідувач кафедри ММСА,
доцент, к.т.н., Тимощук Оксана Леонідівна _____

Консультант з економічного розділу:
Доцент кафедри економічної кібернетики,
к.е.н., Рощина Надія Василівна _____

Консультант з нормоконтролю:
Доцент кафедри ММСА,
к.ф.-м.н. Статкевич Віталій Михайлович _____

Рецензент:
Старший науковий співробітник ІТКГІП НАН України,
доцент, к.ек.н., Просянкіна-Жарова Тетяна Іванівна _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студентка _____

Київ – 2024

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Навчально-науковий Інститут прикладного системного аналізу**

**Кафедра математичних методів системного аналізу**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітня програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ:

Завідувач кафедри

_____ Оксана ТИМОЩУК

«__» _____ 2024 р.

**ЗАВДАННЯ**

**на дипломну роботу студентці**

**Осадчій Софії Олександрівні**

1. Тема роботи «Прогнозування часових рядів для складських запасів», керівник роботи завідувач кафедри ММСА, к.т.н. Тимощук О. Л., затверджені наказом по університету від «__»_____ 2024 р. №_____

2. Термін подання студентом роботи ____11.06.2024_____

3. Вихідні дані до роботи: історичні дані попиту на товар.

4. Зміст роботи: теоретичні відомості, методи прогнозування, програмна реалізація.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): «Переваги моделей ARIMA та SARIMA», «ARIMA», «SARIMA», «Аналіз набору даних».

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Економічний | Рощина Н.В., доцент | | |

7. Дата видачі завдання: _____13.09.2023_____

Календарний план

| № з/п | Назва етапів виконання дипломної роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Формулювання тематики (напряму) дослідження. | 13.09.2023 – 30.09.2023 | виконано |
| 2 | Аналіз актуальності задачі стосовно тематики дослідження | 01.10.2023 – 31.10.2023 | виконано |
| 3 | Аналіз відомих результатів стосовно тематики дослідження | 01.11.2023 – 30.11.2023 | виконано |
| 4 | Формулювання задач дослідження | 01.12.2023 – 31.12.2023 | виконано |
| 5 | Уточнення теми дипломної роботи | 21.02.2024 | виконано |
| 6 | Збір статичних даних, попередній аналіз даних | 01.03.2024 – 31.03.2024 | виконано |
| 7 | Розробка програмного продукту для виконання обчислювальних експериментів | 01.03.2024 – 30.04.2024 | виконано |

| | | | |
|---|---|---|---|
| 8 | Виконання обчислювальних експериментів, аналіз та оформлення результатів | 01.05.2024 –20.05.2024 | виконано |
| 9 | Оформлення пояснювальної записки у цілому | 21.05.2024– 27.05.2024 | виконано |
| 10 | Підготовка презентації для захисту | 28.05.2024– 02.05.2024 | виконано |
| 11 | Попередній захист дипломної роботи | 04.06.2024– 05.06.2024 | виконано |
| 12 | Захист дипломної роботи | 17.06.2021– 21.06.2021 | виконано |

Студентка Софія ОСАДЧА

Керівник Оксана ТИМОЩУК

# РЕФЕРАТ

Дипломна робота: 135 с., 6 табл., 18 рис., 2 дод., 23 джерела.

УПРАВЛІННЯ СКЛАДСЬКИМИ ЗАПАСАМИ, СЕЗОННІСТЬ, ЧАСОВІ РЯДИ, МОДЕЛІ ПРОГНОЗУВАННЯ ARIMA, SARIMA.

Об’єкт дослідження – процес управління запасами на підприємстві.

Предмет дослідження – методи прогнозування на основі регресійних моделей ARIMA, SARIMA.

Мета дослідження – проаналізувати підходи до прогнозування сезонного попиту товарів на підприємстві за допомогою часових рядів, створити програмний продукт з простим та зручним користувацьким інтерфейсом, який прогнозуватиме сезонний попит на товар.

Методи дослідження – аналіз наукової та методичної літератури, часові ряди, регресійні методи прогнозування.

Вкажемо основні результати виконаних досліджень: узагальнено теоретичні аспекти та методи прогнозування часових рядів; серед цих методів було обрано моделі ARIMA та SARIMA та запрограмовано за допомогою мови програмування Python; розроблене програмне забезпечення прогнозує попит з використанням перелічених методів.

# ABSTRACT

Thesis: 135 p., 6 tables, 18 figures, 2 appendices, 23 references.

INVENTORY MANAGEMENT, SEASONALITY, TIME SERIES, FORECASTING MODELS ARIMA, SARIMA.

The object of research is the process of inventory management at the enterprise.

The subject of the study is forecasting methods based on ARIMA and SARIMA regression models.

Purpose of the study - to analyze approaches to forecasting the seasonal demand for goods at the enterprise using time series, to create a software product with a simple and convenient user interface that will forecast the seasonal demand for goods.

Research methods: analysis of scientific and methodological literature, time series, regression forecasting methods.

The main results of the research are the following: we summarized theoretical aspects and methods of time series forecasting; we chose and developed the ARIMA and SARIMA models using Python programming language; the developed software performs demand forecasting using the listed methods.

# ЗМІСТ

# ПЕРЕЛІК СКОРОЧЕНЬ

JIT - Just in Time (якраз вчасно)

VMI - Vendor Managed Inventory (Інвентаризація, керована постачальником)

AR model - autoregressive model

I - integrated

MA - moving average model

ARIMA - Autoregressive Integrated Moving Average

SARIMA - Seasonal Autoregressive Integrated Moving Average

# ВСТУП

У сучасному світі стрімкого розвитку, деякі аспекти життя залишаються незмінними. Однією з таких речей є використання товарів у повсякденному житті. Ці товари є невід'ємною частиною нашого щоденного життя і допомагають нам здійснювати різноманітні дії, від простого приготування сніданку до розважальних запитів, тобто товари відіграють ключову роль у забезпеченні наших базових потреб. Постійний попит створює стабільний ринок, який змушує виробників вдосконалювати свою продукцію, адаптуватися до змінюваних вимог споживачів та впроваджувати нові технології. Таким чином, навіть у епоху цифрових інновацій, традиційні товари не втрачають свого значення і залишаються необхідними для забезпечення комфортного та повноцінного життя сучасної людини.

Розглядаючи цю тему з іншого боку, стає очевидним, що успішне зберігання товарів потребує відмінного управління запасами. Виробники повинні постійно оцінювати потреби ринку, прогнозувати зміни у попиті та ефективно використовувати складський простір. Тільки таким чином вони можуть забезпечити оптимальні умови для зберігання товарів і підтримувати надійну роботу в цілому ланцюгу постачання.

Також варто враховувати, що процес сезонного попиту є важливим фактором у процесі будь-якого підприємства. Від сезонних коливань попиту залежить ефективність управління запасами та планування виробництва. Підприємства повинні бути готові до великого збільшення попиту в певний період часу та вміло реагувати на зміни у ринковій ситуації. Тому врахування сезонності стає необхідною складовою управління запасами, що дозволяє підтримувати стабільну роботу підприємства навіть у змінних умовах ринку.

# РОЗДІЛ 1 ТЕОРЕТИЧНІ ВІДОМОСТІ

## 1.1 Управління складськими запасами

Під час своєї діяльності підприємства повинні вирішувати ряд питань, одним з яких є управління складськими запасами. Для досягнення мети забезпечення підприємства необхідними ресурсами важливо мати матеріальні запаси. Ці запаси гарантують нормальне функціонування підприємства дозволяючи здійснювати його діяльність без перешкод.

Запаси – товари та матеріали, що постачаються та зберігаються на підприємстві. Вони утворюються кожен раз коли ресурси, що надходять чи виходять на підприємство не використовуються, хоч і доступні [1].

У літературі є декілька варіантів даного терміну, але в роботі будемо користуватися трактуванням, що наведено.

Запаси є важливими для різних організацій, включаючи виробничі підприємства, оптові компанії, роздрібні торговельні підприємства, компанії сфери послуг, логістичних посередників та операторів, банки, біржі, страхові компанії, порти тощо. У кожній з цих організацій запаси забезпечують товарно-матеріальні цінності для підтримки як основної, так і допоміжної діяльності.

Управління запасами включає визначення операційних цілей управління запасами, планування потреби в запасах, організацію роботи складських працівників, розстановку і налагодження взаємодії працівників, їх мотивацію шляхом створення оптимальних умов праці та відпочинку, виплати належної заробітної плати та премій, налагодження зв’язків із постачальниками і споживачами, контроль виконання замовлень та утримання запасів на підприємстві, просування запасів по логістичному

ланцюгу з метою задоволення потреб виробництва і споживачів готової продукції за оптимальних логістичних витрат [2].

Політика управління товарними запасами є складовою загальної політики управління ресурсами підприємства. Вона спрямована на оптимізацію загального обсягу і структури запасів, мінімізацію витрат на їх обслуговування і забезпечення ефективного контролю за їх рухом. Одним із ключових завдань політики управління товарними запасами є створення механізму, що дозволяє досягти оптимальних капіталовкладень у товарно-матеріальні цінності. Ефективне управління запасами сприяє мінімізації їх обсягу, зниженню витрат і підвищенню прибутковості. Політика управління запасами визначає цілі і підходи підприємства до управління на стратегічному рівні, які повинні узгоджуватися із загальною стратегією підприємства. [3].

### 1.1.1 Стратегії управління складськими запасами

Оперативне управління запасами базується на різноманітних методах управління та контролю. Вибір та комбінування цих методів здійснюється в межах кожного підприємства, враховуючи його цільові орієнтації, умови та обмеження, а також особливості його функціонування. Поговоримо про деякі стратегії управління запасами, де дуже важлива точність прогнозування попиту.

Першою розглянемо стратегію Just in Time (JIT) — популярна система управління запасами, метод планування виробництва, при якому матеріали закуповуються та виготовляються лише тоді, коли вони потрібні. Метою JIT є зниження рівня запасів і витрат шляхом виробництва товарів лише тоді, коли на них є попит, і мінімізація відходів у процесі виробництва.

Однією з головних переваг JIT є те, що він допомагає підприємствам зменшити витрати на запаси, оскільки немає необхідності накопичувати матеріали, які можуть бути непотрібними. Це дозволяє компаніям зменшити витрати на зберігання, транспортування та вартість матеріалів, які можуть застаріти.

Загалом, JIT є корисним методом для компаній, щоб зменшити витрати на запаси, стати більш ефективними та покращити процеси контролю якості. Виробляючи товари лише тоді, коли на них є попит, підприємства можуть гарантувати, що вони виробляють високоякісні продукти, які відповідають вимогам клієнтів. Впроваджуючи JIT, підприємства можуть досягти більшої продуктивності та прибутковості, що допоможе їм розвиватися та досягати успіху в сучасному конкурентному бізнес-середовищі [4].

Оскільки, головним аспектом цієї стратегії є попит, то від того наскільки точне буде прогнозування залежить робота всього підприємства.

Другою розглянемо стратегію "Safety Stock", далі будемо називати як страховий запас. Страховий запас є важливою складовою управління ланцюгами поставок, що забезпечує неперервну діяльність компаній. Він використовується як захист від ризиків, пов'язаних з управлінням запасами, таких як неочікувані зміни у попиті, дефіцит товарів або затримки в поставках. Хоча наявність додаткових запасів важлива для ефективного реагування на зміни у попиті, важливо уникати надмірностей. Правильний розрахунок страхового запасу допомагає оптимізувати управління запасами, уникаючи ситуацій перевищення, що може збільшити витрати на їх зберігання або призвести до розпродажу запасів у зв'язку з закінченням сезону або придатності.

Стратегія "Safety Stock" відіграє ключову роль у управлінні запасами, забезпечуючи компаніям надійність та стабільність у виробничих процесах і постачаннях. Однією з основних переваг цієї стратегії є її здатність захистити компанію від непередбачуваних ризиків. Наявність додаткових запасів

дозволяє компенсувати можливі затримки у поставках або внутрішні зміни у виробничому процесі, що забезпечує стабільність і неперервність виробництва [5].

Крім того, забезпечується неперервне обслуговування клієнтів. У такому випадку стратегія дозволяє компаніям забезпечити продукти чи послуги своїм клієнтам вчасно, навіть у випадку непередбачених обставин, таких як стрибок попиту або зміни в поставках від постачальників. Це сприяє підвищенню рівня задоволення клієнтів і підтримує довгострокові відносини з ними.

Третя важлива стратегія це стратегія "Vendor Managed Inventory" (VMI). Інвентаризація, керована постачальником, часто скорочена як VMI, — це стратегія спільного управління запасами, у якій постачальник бере на себе відповідальність за керування рівнями запасів клієнта. Простіше кажучи, постачальник отримує прямий доступ до даних про запаси клієнта та бере на себе відповідальність за поповнення запасів у разі потреби. Цей підхід означає перехід від традиційного управління запасами, керованого покупцями, до моделі, орієнтованої на постачальника.

Перш за все, VMI сприяє підвищенню ефективності ланцюга поставок шляхом зменшення часу і зусиль, необхідних для управління запасами. Постачальник, відстежуючи рівень запасів у своїх клієнтів, може реагувати на зміни в попиті швидше та ефективніше, сприяючи підтримці оптимальних запасів та уникненню нестач. Також, VMI допомагає знизити витрати обох сторін, адже він дозволяє уникнути зайвих запасів і зв'язаних з ними витрат на утримання. Постачальник може оптимізувати свої поставки, враховуючи реальні потреби клієнтів, тим самим знижуючи витрати на складське утримання та ризики пов'язані з непроданою продукцією [6].

### 1.1.2 Класифікація складських запасів

Запаси класифікуються на виробничі, що ув'язують неперервність споживання ресурсів з дискретністю їх надходження від постачальників, та товарні, що ув'язують інтервали надходження продукції від постачальників з інтервалами відпускання її споживачам.

На підприємствах розрізняють три рівні виробничих запасів.

1. Запаси готової продукції - дозволяють службі збуту забезпечувати більш короткі строки поставок, ніж повний цикл постачання та виготовлення цієї продукції. Вони згладжують нерегулярності або зупинки на виробництві. Достатні запаси готової продукції дозволяють уникнути або відстрочити наслідки призупинення виробництва через ремонт, простої, страйки тощо. Крім того, вони є регулятором виробництва під час сезонних коливань попиту, що дозволяє, якщо це необхідно, підтримувати постійний рівень продуктивності.

2. Запаси незавершеного виробництва – формуються на різних стадіях виробництва таким чином, що зупинка процесу на будь-якій стадії не призводить до раптової зупинки всіх наступних операцій виробничого процесу.

3. Запаси купованих матеріальних ресурсів – виробничі запаси дозволяють шляхом зниження періодичності замовлень користуватись торгівельними знижками для одержання великих партій ресурсів. Вони забезпечують захист від перебоїв у постачанні, наприклад, при закупівлях у монополістів.

За функціональним призначенням розрізняють такі запаси.

1. Поточні – забезпечують неперервність постачання виробничого процесу між двома поставками, а також організацію торгівлі та

обслуговування; становлять основну частину виробничих та товарних запасів: це постійно-змінні запаси.

2. Підготовчі – виокремлюються з виробничих запасів при необхідності додаткової їх підготовки перед використанням у виробництві (наприклад, сушіння лісу). Формуються при необхідності підготувати товари до видачі споживачам.
3. Гарантійні – призначені для забезпечення безперервного постачання та обслуговування за непередбачених обставин: відхилення в періодичності та розмірі партії поставок від запланованих, зміни інтенсивності споживання, затримання поставок тощо. У межах цих запасів існує резервний запас, який враховує можливі помилки при розрахунку обсягу попиту чи пропозиції. У виробництві вони убезпечують від страйків, запізнень поставок тощо. Резерви готової продукції захищають від непередбаченого попиту або виробничих помилок. Ці запаси є постійними і недоторканними під час роботи.
4. Сезонні запаси формуються при сезонному характері виробництва, споживання або транспортування продукції. Вони повинні забезпечити нормальну роботу підприємства під час сезонних перерв у виробництві, споживанні або транспортуванні продукції. У межах цих запасів можуть бути запаси, створені у зв'язку з очікуванням певних подій. Такі запаси створюються для відомих наперед потреб, перед певними подіями і тому меншою мірою піддаються ризику [7].

### 1.1.3 Оптимізація запасів

У сучасному світі управління запасами стає все більш важливим елементом успішної діяльності підприємств. Оптимізація цього процесу відіграє вирішальну роль у забезпеченні ефективності виробничих процесів та здатності компаній задовольняти потреби своїх клієнтів. Управління складськими запасами є складною задачею, оскільки вимагає балансу між мінімізацією витрат на утримання запасів та максимізацією обслуговування клієнтів. Розглянемо одні з головних аспектів оптимізації.

Мінімізація затрат на утримання запасів. Цей принцип орієнтує компанії на мінімізацію витрат, пов'язаних із зберіганням запасів, таких як витрати на землю, приміщення, електроенергію, страхування та витрати на обслуговування складів. Це досягається за допомогою ефективного управління запасами, включаючи оптимізацію рівня запасів, вибір оптимальних стратегій замовлення та використання передових технологій управління запасами, які дозволяють покращити ефективність складського обліку та моніторингу [8].

Максимізація обслуговування клієнтів. Цей принцип ставить на перший план потреби та очікування клієнтів і спрямований на максимальне задоволення їхніх потреб через швидке та точне обслуговування. Це означає, що компанії повинні мати необхідні запаси, щоб вчасно виконувати замовлення та уникати нестач, які можуть призвести до втрати довіри клієнтів і втрати ринкової позиції. Максимізація обслуговування клієнтів вимагає добре організованої системи управління запасами, прогнозування попиту та високого рівня логістики [9].

Баланс між затратами та обсягами запасів. Цей принцип полягає в тому, щоб забезпечити оптимальний рівень запасів, який дозволяє забезпечити максимальне обслуговування клієнтів за мінімальні витрати на їх утримання.

Це вимагає постійного аналізу та оцінки вартості запасів та ризику недостачі, а також застосування стратегій управління запасами, які дозволяють забезпечити оптимальний баланс між витратами та обсягами запасів [10].

## 1.2 Сутність явища сезонності

Жоден бізнес не звільнений від впливу сезонності. Навіть якщо бізнес не вважається сезонним, сезонність все одно впливає на інтерес і попит на продукти й послуги. Це цілком природний, а часом, неминучий аспект бізнесу. Це означає, що його можна вивчити і зрозуміти. І завдяки такому розумінню ці абсолютно неминучі випадки можна використати з користю.

Сезонність притаманна майже всім виробникам не тільки за регіональною ознакою (курорти, туристичні місця), але й у залежності від виду продукції чи послуг які виробляються або надаються, а також притаманна високотехнологічним галузям змушуючи їх координувати роботу із постачальниками та споживачами. Тому керівництву потрібно володіти певними навичками та знаннями інструментів, що діють у ситуаціях із високим попитом на продукцію чи послуги і відповідними навичками у ситуаціях із низьким попитом та вміти їх передбачати. Також, важливе розуміння впливу сезонних факторів на споживання та виробництво як можливості прогнозування ефективної діяльності підприємства [11].

Сезонність – опис характеристика часового ряду, в якому дані періодично зазнають регулярних та передбачуваних змін, які повторюються щорічно за календарем. Любі циклічні коливання або закономірності, які регулярно відтворюються впродовж одного року, вважаються сезонними.

Основні причини сезонності попиту на товари та послуги є пори року, святкові дні, бізнес-активність протягом року, фінансування бюджетних

програм, світові події та безпосередня сезонність самих товарів та послуг [12].

У свою чергу, до економічних факторів сезонних коливань відноситься структура споживання товарів і послуг, формування платоспроможності попиту за допомогою пропозиції. До демографічних факторів відноситься диференційований попит за статевим складом та іншими ознаками. Психологічні фактори складаються з традицій, моди, наслідування. У свою чергу, матеріально-технічні складаються з розвитку мережі розміщення, харчування, транспорту, культурно-оздоровчого обслуговування, а технологічним фактором є комплексний підхід у наданні якісних послуг [13].

Неправильне прогнозування попиту, закупівлі або виробництва може призвести до виникнення надлишкових та мертвих запасів, або ж до їх нестачі. Усе це негативно вплине на діяльність підприємства. Якщо є занадто багато товарів, то потрібно витратити більше грошей на їх зберігання, а якщо товару недостатньо, то можна втратити прибуток.. Мертві запаси є найгіршими для підприємства, бо такі товари або ж неможливо продати, або ж можна продати через тривалий період.

Щоб згладити сезонні коливання, можна використовувати ефективну маркетингову стратегію, яка стимулює попит у періоди його зниження.

Щоб зменшити різкі коливання прибутку внаслідок сезонності, розширення асортименту є ефективним методом. Це означає додавання товарів, які популярні в різні періоди року, а також товарів, які не залежать від сезонності.

## 1.3 Постановка задачі

Метою даної дипломної роботи є розробка програми для прогнозування сезонного попиту складських запасів. Будуть виконуватися підзадачі.

1. Пошук та підготовка тестових даних на основі часових рядів.
2. Проведення аналізу зібраних даних за допомогою статистичних методів.
3. Вибір та тестування моделі прогнозування на основі регресійних моделей.
4. Створення програмного продукту, який має інтуїтивний користувацький інтерфейс.

Врахувавши всі задачі, котрі буде мати програмний продукт, у результаті він повинен відповідати наступним вимогам:

- програмний продукт повинен забезпечувати високу точність прогнозів, щоб допомогти управлінцям приймати обґрунтовані рішення щодо управління запасами;
- продукт повинен дозволяти аналізувати та оброблювати різні дані, які необхідні користувачу;
- програмний продукт має мати інтуїтивний та легкий у використанні для будь-якого користувача.

## 1.4 Висновки до розділу 1

У першому розділі дипломної роботи було розглянуто важливі аспекти для розуміння процесу управління складськими запасами. У сучасних умовах ринкової економіки ефективне управління запасами відіграє критичну роль у

конкурентоспроможності та фінансовій стабільності компанії. Процес управління складськими запасами полягає у підтримці такого рівня запасів, який би забезпечував своєчасне задоволення попиту клієнтів та водночас мінімізував витрати на зберігання і обслуговування запасів. Вивчення історичних даних про попит на продукцію дозволяє прогнозувати майбутні потреби та уникати дефіциту або надлишку товарів.

Також було описано явище сезонності. Сезонність визначається як систематичні коливання в попиті на товари або послуги внаслідок змін у погодних умовах, святкових або інших календарних подій. Розуміння сутності цього явища є ключовим для вдалих стратегій управління запасами та прогнозування попиту, оскільки дозволяє адаптувати планування та розподіл ресурсів до циклічних змін у попиті.

Було визначено постановку задачі, та які кроки потрібно виконати, щоб у результати отримати якісний продукт.

# РОЗДІЛ 2 МЕТОДИ ПРОГНОЗУВАННЯ

## 2.1 Визначення часового ряду

Часові ряди – це послідовність точок даних, виміряних у дискретні моменти часу, які впорядковані за зростаючим часовим індексом, тобто хронологічно.

Однією з особливостей часових рядів є їх послідовність у часі. Важливо враховувати порядок спостережень при аналізі, оскільки час виступає ключовим фактором. Це розрізняє їх від звичайних випадкових вибірок, де індекси можуть використовуватися лише для зручності. Давайте розглянемо основні відмінності між часовими рядами та простими статистичними сукупностями.

1. По-перше, рівні часового ряду не є незалежними. Іншими словами, якщо можна передбачити майбутні значення змінної, то вони залежать від минулих значень цієї змінної.
2. По-друге, рівні часового ряду неоднаково розподілені. Закон розподілу ймовірностей цих випадкових величин, а також їхні математичні сподівання та дисперсії, можуть змінюватися з часом [14].

Незалежно від природи кожного часового ряду, можна виділити основні типи задач, які зазвичай вирішуються при проведенні аналізу початкових даних.

На першому етапі намагаються побудувати просту математичну систему або модель, яка описує поведінку часового ряду в короткій формі. Потім робиться спроба пояснити його поведінку за допомогою інших змінних і з'ясувати ступінь зв'язку як між спостереженнями одного ряду, так і між різними рядами [15].

Виходячи з цілей дослідження, кожний часовий ряд звичайно розглядають як суміш компонент.

1. Тренд. Тренд показує загальну тенденцію даних до збільшення або зменшення протягом тривалого періоду часу. Тренд - це плавна, загальна, довгострокова, середня тенденція. Не завжди необхідно, щоб збільшення або зменшення відбувалося в одному напрямку протягом певного періоду часу.
2. Сезонність. Закономірність, яка часто повторюється через регулярні проміжки часу. Наприклад: високі продажі кожні вихідні.
3. Циклічність. Циклічність - це коли є повторювана модель, але немає фіксованого періоду [16].

Класичний підхід до побудови моделі часового ряду полягає у розкладі його на декілька компонент, природа кожної з яких принципово відмінна від інших. Далі кожна складова аналізується специфічними для неї методами. Найчастіше на практиці розглядають розклад часового ряду $y_t$, за формулою (2.1):

$$y_t = tr_t + s_t + r_t \tag{2.1}$$

Компонента $tr_t$ відповідає складовій, яка змінюється повільно і називається трендом. Це може бути як просто стала, так і многочлен, експонента чи інша подібна функція. Тренд звичайно описує довготривалі тенденції явища. Компонента $s_t$ відповідає складовій, яка змінюється періодично і називається сезонною або циклічною компонентою. Ця компонента часто описує сезонні явища у виробництві протягом року або економічні циклічності. Звичайно складові $tr_t$ та $s_t$ детерміністичні функції. За випадковість у спостереженнях відповідає $r_t$. Основний напрямок розвитку теорії часових рядів якраз є моделювання і аналіз процесу $r_t$ [17].

У задачах прогнозування часові ряди використовуються при наявності значної кількості реальних даних про показник з минулого і при умові, що тенденція, яка спостерігалася у минулому, є чіткою і відносно стабільною. Це передбачає, що минуле може служити дорогою для прогнозування майбутніх подій. Аналіз часових рядів дозволяє здогадатися, що станеться у відсутності зовнішнього впливу, і тому не може передбачити зміни у тенденції. [14].

## 2.2 Стаціонарність

Для будь-якого часового ряду, як і для будь-якої вибірки, можна визначити деякі числові характеристики, подібно до методів математичної статистики.

Для аналізу часових рядів основними параметрами є математичне сподівання, дисперсія і коваріація.

Математичним сподіванням часового ряду $\{Y_T\}$ є функція, яка записується формулою (2.2):

$$\mu_t = Ey_t = \int_{-\infty}^{\infty} x dF_t(x), \tag{2.2}$$

де $F_t(x) = P(y_t < x)$ – функція розподілу $y_t, t = \overline{1,T}$.

Дисперсія часового ряду $\{Y_T\}$ визначається за формулою (2.3):

$$D(y_t) = E(y_t - \mu_t)^2 \tag{2.3}$$

Автоковаріація часового ряду $\{Y_T\}$ визначається за формулою (2.4):

$$\operatorname{cov}(y_t, y_{t-j}) = E(y_t - \mu_t)(y_{t-j} - \mu_{t-j}),\ j = 1,2 \ldots \tag{2.4}$$

Часовий ряд є стаціонарним, якщо:

– Математичне сподівання $\mu_t = \mu < \infty$ для всіх $t = \overline{1,T}$.

– Дисперсія $D(y_t) = \gamma_0 < \infty$ для всіх $t = \overline{1,T}$.

– Автоковаріація j-того порядку $\text{cov}(y_t, y_{t-j}) = \gamma_j < \infty$ для всіх $t = \overline{1,T}$, $j = 1,2$ ...[17].

Отже, поняття стаціонарного часового ряду означає, що його середнє значення не змінюється в часі, тобто часовий ряд не має тренду, дисперсія також не змінюється. Крім того, коваріація між різними елементами часового ряду (як між випадковими величинами) залежить тільки від того, наскільки сильно вони віддалені один від одного в часі [18].

Нестаціонарним часовим рядом називається ряд, який не задовольняє перерахованим вище властивостям.

Таким чином, стаціонарний часовий ряд повинен мати сталі та скінченні математичні сподівання та дисперсію для кожного періоду часу, а також сталу і скінчену автоковаріацію будь-якого порядку в усі моменти часу.

Стаціонарність означає, що розподіл даних не змінюється з часом.

Приклад стаціонарних та нестаціонарних рядів зображено на рисунку 2.1.

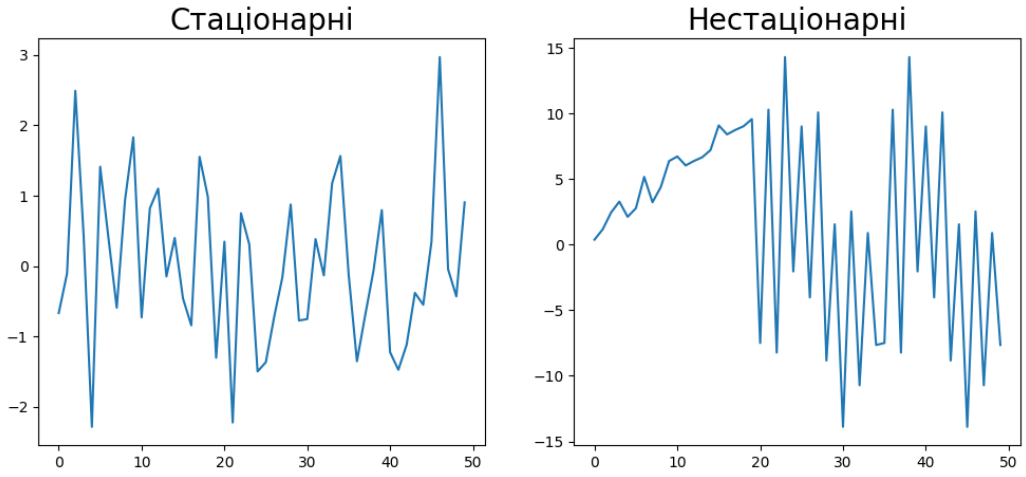



Рисунок 2.1 – Приклад стаціонарного і нестаціонарного ряду

– Немає тренду: ніщо не зростає і не зменшується.

– Середнє значення та дисперсія постійні: середня відстань точок даних від нульової лінії не змінюється.

– Постійна автокореляція: те, як кожне значення в часовому ряді пов'язане з його сусідами, залишається незмінним.

Прикладом стаціонарного процесу є білий шум. Ряд можна вважати білим шумом, якщо математичне сподівання дорівнює нулю. Елементи є некорельованими (незалежними один від одного) однаково розподіленими величинами, і дисперсія є постійною величиною [16].

Одним з найважливіших аспектів часових рядів є поняття стаціонарності. Звичайно, жоден ряд, що представляє реальну інформацію про процес, який відбувається в часі, не може бути ідеально стаціонарним. Однак, якщо для певного часового ряду з деяким наближенням виконуються умови стаціонарності, то для його аналізу можна використовувати багато різних методів аналізу та прогнозування. [17].

## 2.3 Методи прогнозування часових рядів

Прогнозування часових рядів – це ефективний і широко використовуваний метод передбачення майбутніх значень на основі минулих даних. Він знаходить своє застосування в різних галузях, від економіки до науки про клімат, дозволяючи аналізувати тенденції, виявляти закономірності і приймати обґрунтовані рішення на основі цієї інформації.

### 2.3.1 Метод ARIMA

Кожен часовий ряд має зазвичай декілька основних властивостей. Саме на основі характеристик тренду, циклічності та сезонності й базуються всі

методи прогнозування часових рядів. Одними з найпопулярніших моделей прогнозування є моделі ARIMA (авторегресійне інтегроване ковзне середнє). ARIMA є поєднанням авторегресійної моделі (англ. autoregressive model, AR), інтегрованої частини (англ. integrated, I) та моделі ковзного середнього (англ. moving average model, MA).

Авторегресійні моделі (AR) виконують прогнозування часового ряду використовуючи його минулі значення. Моделі ковзного середнього (MA) прогнозують майбутні значення часового ряду на основі похибок минулих прогнозів. Інтегрована компонента (I) відповідає за процес диференціювання, який забезпечує стаціонарність часового ряду для всіх інших застосованих моделей. Основним обмеженням авторегресійних моделей і моделей ковзного середнього є стаціонарність часового ряду. Таке саме обмеження мають і модель ARIMA.

Модель ARIMA – це модель ARMA, але з кроком диференціювання, включеним в модель, який ми представляємо за допомогою I(d). d – це порядок різниць, тобто необхідна кількість перетворень для того, щоб зробити вхідні дані стаціонарними. Отже, модель ARIMA - це просто модель ARMA на різницевих часових рядах.

Модель ARIMA описується за формулою (2.5):

$$\Delta^d y_t = c + \phi_1 \Delta^d y_{t-1} + \cdots + \phi_p \Delta^d y_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t \quad (2.5)$$

де $y_t$ – значення часового ряду на момент часу t, $\Delta^d y_t$ – різниця ряду ступеня d, різниця першого ступеня записується за формулою (2.6):

$$\Delta y_t = y_t - y_{t-1}, \quad (2.6)$$

де c – константа,

$\varepsilon_t$ – значення білого шуму,

$\phi_1 \ldots . \phi_p$ – коефіцієнти авторегресійної моделі,

$\theta_1 \ldots \theta_q$ – коефіцієнти моделі ковзного середнього [19].

Визначення моделі ARIMA для певного часового ряду полягає у виборі значень параметрів p, d і q, що є досить складним завданням. Параметр p позначає порядок авторегресійної частини авторегресійної, d – ступінь різниці ряду, q – порядок частини ковзного середнього [20].

У ході використання моделі прогнозування отримуються точкові прогнози на визначену кількість кроків прогнозування. Для оцінки отриманих значень використовують багато різних функцій помилок, наприклад, середня абсолютна похибка (англ. mean absolute error, MAE) та середня абсолютна відсоткова похибка (англ. mean absolute percentage error, MAPE) [20]. Похибка MAE обчислюється як середнє значення абсолютних похибок прогнозу на кожному кроці та використовує ті самі одиниці виміру, що й значення часового ряду. Помилка MAPE обчислюється як середнє значення абсолютних відсоткових похибок, які подають значення у відсотках і не залежать від початкових одиниць виміру часового ряду [19].

### 2.3.2 Метод SARIMA

Дані часових рядів оточують нас усюди: від цін на акції та погодних умов до прогнозування попиту та сезонних тенденцій у продажах. Щоб зрозуміти сенс цих даних і спрогнозувати майбутні значення, ми звертаємося до потужних моделей, таких як сезонне авторегресійне інтегроване ковзне середнє, або SARIMA [21].

SARIMA, що розшифровується як Seasonal Autoregressive Integrated Moving Average (сезонна авторегресійна інтегрована ковзна середня), є універсальною і широко використовуваною моделлю прогнозування часових

рядів. Вона є розширенням несезонної моделі ARIMA, призначеної для обробки даних з сезонним характером. SARIMA враховує як короткострокові, так і довгострокові залежності в даних, що робить її надійним інструментом для прогнозування. Вона поєднує в собі концепції авторегресійної (AR), інтегрованої (I) та ковзної середньої (MA) моделей з сезонними компонентами.

Складові SARIMA, а також SARIMA(p, d, q) (P, D, Q, s):

Сезонний компонент: "S" в SARIMA означає сезонність, тобто повторюваність даних. Це може бути щоденний, щомісячний, річний або будь-який інший регулярний інтервал. Ідентифікація та моделювання сезонного компоненту є ключовою перевагою SARIMA.

s: Сезонний період.

Авторегресійний (AR) компонент: "AR" в SARIMA означає авторегресійний компонент, який моделює зв'язок між поточною точкою даних та її минулими значеннями. Вона відображає автокореляцію даних, тобто те, наскільки дані корелюють самі з собою в часі.

AR (p): Авторегресійний компонент порядку p.

Сезонний AR(P): Сезонний авторегресійний компонент порядку P.

Інтегрований (I) компонент: "I" в SARIMA вказує на диференціювання, яке перетворює нестаціонарні дані на стаціонарні. Стаціонарність має вирішальне значення для моделювання часових рядів. Інтегрована компонента вимірює, скільки різниць потрібно для досягнення стаціонарності.

I(d): Інтегрований компонент порядку d.

Сезонний I(D): Сезонний інтегрований компонент порядку D.

Компонент ковзного середнього (MA): "MA" в SARIMA представляє компонент ковзного середнього, який моделює залежність між поточною точкою даних і минулими помилками прогнозування. Це допомагає вловити короткостроковий шум у даних.

MA(q): Ковзний середній компонент порядку q.

MA (Q): Сезонний ковзний середній компонент порядку Q [21].

Математичне представлення SARIMA описується формулою (2.7):

$$(1-\phi_1 \mathrm{B})(1-\Phi_1 \mathrm{B}^s)(1-\mathrm{B})(1-\mathrm{B}^s)y_t = (1+\theta_1 \mathrm{B})(1+\Theta_1 \mathrm{B}^s)\varepsilon_t \qquad (2.7)$$

де $y_t$ – спостережуваний часовий ряд у момент часу $t$;

B – оператор зсуву назад,що представляє оператор затримки (тобто $\mathrm{B}y_t = y_{t-1}$);

$\phi_1$ – несезонний авторегресійний коефіцієнт;

$\Phi_1$ – сезонний авторегресійний коефіцієнт;

$\theta_1$ – несезонний ковзний середній коефіцієнт;

$\Theta_1$ – сезонний ковзний середній коефіцієнт;

$s$ – сезонний період;

$\varepsilon_t$ – є членом помилки білого шуму в момент часу $t$.

Розглянемо кожен компонент рівняння:

– Компонент авторегресії (AR): несезонний компонент авторегресії, представлений $(1-\phi_1 \mathrm{B})$ фіксацією зв'язку між поточним спостереженням і певною кількістю спостережень із запізненням (попередні значення в часовому ряді). Термін B представляє оператор зворотного зсуву, який зазвичай використовується в аналізі часових рядів. Він являє собою оператор затримки, який зсуває часовий ряд назад на певну кількість періодів часу. Порядок компоненти авторегресії, позначений (p), визначає кількість минулих значень, які розглядаються в моделі.

– Компонент сезонної авторегресії (SAR): Сезонний компонент авторегресії представлено $(1-\Phi_1 \mathrm{B}^s)$ таким компонентом, який фіксує зв'язок між поточним спостереженням і певною кількістю спостережень із запізненням у сезонні проміжки часу. Термін $\mathrm{B}^s$

представляє оператор зворотного зсуву, застосований до сезонних спостережень із запізненням.

- Несезонний компонент різниці: несезонний компонент різниці представлено як $(1 - \mathrm{B})$, де d є порядком несезонної різниці. Цей компонент використовується, щоб зробити часовий ряд стаціонарним, розрізняючи його певну кількість разів.
- Компонент сезонної різниці: Сезонна компонента різниці представлена як , де D є порядком сезонної різниці та. Цей компонент використовується, щоб зробити часовий ряд стаціонарним, розрізняючи його на сезонні інтервали $(1 - \mathrm{B}^{s})$.
- Спостережуваний часовий ряд: спостережуваний часовий ряд позначається $y_t$ . Він представляє історичні дані, які ми маємо та хочемо спрогнозувати.
- Компонент ковзного середнього (MA): компонент ковзного середнього представлено $(1 + \theta_1 \mathrm{B})$. Цей компонент фіксує зв'язок між поточним спостереженням і залишковими помилками моделі ковзного середнього, застосованої до спостережень із запізненням.
- Компонент сезонного ковзного середнього (SMA): Сезонний компонент ковзного середнього представлено $(1 + \Theta_1 \mathrm{B}^{s})$. Цей компонент фіксує зв'язок між поточним спостереженням і залишковими помилками моделі ковзного середнього, застосованої до спостережень із запізненням у сезонні інтервали.
- Термін помилки : Термін помилки позначається $\varepsilon_t$. Він представляє випадковий шум або незрозумілу зміну в часовому ряді [21].

## 2.4 Висновки до розділу 2

У другому розділі були розглянуті методи прогнозування, що найкраще підходять для аналізу наших даних. Зокрема, було детально проаналізовано метод ARIMA (Autoregressive Integrated Moving Average), який є одним з найпопулярніших підходів для прогнозування часових рядів. Цей метод включає кілька ключових компонентів: автокореляцію (AR), диференціювання (I) та зважену суму попередніх помилок (MA). Використання автокореляції дозволяє моделі враховувати вплив попередніх значень на поточні, що є важливим для розуміння внутрішніх закономірностей часового ряду. Диференціювання допомагає зробити часовий ряд стаціонарним, тобто усуває тренди та сезонні коливання, що спрощує аналіз. Зважена сума попередніх помилок забезпечує корекцію прогнозу на основі попередніх помилок, що підвищує точність прогнозування.

Метод ARIMA дозволяє моделювати широкий спектр складних часових рядів, враховуючи їхні специфічні особливості. Завдяки своїй гнучкості, ARIMA може адаптуватися до різних типів даних і забезпечувати високоякісні прогнози. Цей метод є особливо корисним, коли часові ряди не мають вираженої сезонності або коли сезонність не є основним фактором. Однак, якщо сезонні коливання суттєво впливають на дані, необхідно використовувати більш складні моделі.

Для врахування сезонних коливань у наших даних був обраний метод SARIMA (Seasonal ARIMA). SARIMA розширює базову модель ARIMA, додаючи до неї сезонні компоненти, що дозволяє більш точно моделювати часові ряди з вираженою сезонністю. У моделі SARIMA враховуються сезонні автокореляція, диференціювання та зважені помилки, аналогічно до несезонних компонентів у ARIMA. Це дозволяє моделі враховувати

повторювані патерни, що виникають через сезонні зміни, такі як місячні або квартальні цикли.

Застосування методу SARIMA є особливо важливим для нашого прогнозування, оскільки наші дані піддаються впливу сезонних коливань. Наприклад, попит на певні товари може значно змінюватися в залежності від пори року або святкових періодів. Використання SARIMA дозволяє моделі враховувати ці циклічні зміни, що підвищує точність та надійність прогнозів. Це, в свою чергу, сприяє ефективнішому управлінню складськими запасами та плануванню ресурсів, оскільки підприємство може завчасно підготуватися до сезонних коливань попиту.

# РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

## 3.1 Обґрунтування вибору мови програмування і середовища

Для реалізації програмного продукту, було прийнято рішення використовувати мову програмування Python. Python – одна з найвідоміших мов програмування, що використовують програмісти по всьому світу. Вона є гнучкою і проста у використанні, що в свою чергу дозволяє легко адаптувати програму до змінних потреб користувачів. Дана мова має велику кількість бібліотек, що справно працюють з часовими рядами, забезпечують великий вибір інструментів для обробки та моделювання даних.

Крім того, Python має велику спільноту розробників, яка надає безліч ресурсів, таких як документація, форуми та підручники, що сприяє швидкому розвитку та вирішенню проблем, які можуть виникати під час розробки. Така комбінація можливостей робить Python ідеальним вибором для створення ефективного програмного продукту.

Під час використання Python, було використані такі бібліотеки як: Pandas, matplotlib, statmodels, PySide6.

Бібліотека Pandas надає потужні та ефективні засоби для обробки та аналізу даних, зокрема часових рядів, що є критичним аспектом у прогнозуванні попиту. Ця бібліотека дозволяє легко обробляти, фільтрувати та агрегувати дані, виконувати операції з індексами та стовпцями, а також використовувати різноманітні методи для візуалізації даних. Завдяки своїй простоті та гнучкості, Pandas допомагає вирішувати складні завдання аналізу даних та підготовки їх до моделювання, що робить її незамінним інструментом у процесі розробки програмного продукту.

Бібліотека Matplotlib надає можливості для візуалізації даних у вигляді графіків, діаграм та інших візуальних представлень, що є необхідним для

аналізу та візуалізації часових рядів, результатів прогнозування та інших даних, що використовуються у процесі прогнозування попиту. Matplotlib дозволяє створювати якісні візуальні представлення даних, які полегшують аналіз та прийняття рішень.

Бібліотека Statsmodels надає різноманітні статистичні моделі та методи для аналізу та прогнозування даних, включаючи моделі часових рядів, такі як ARIMA (Autoregressive Integrated Moving Average) та SARIMA (Seasonal Autoregressive Integrated Moving Average). Statsmodels дозволяє побудувати моделі, оцінювати їхні параметри та здійснювати прогнози на основі отриманих результатів. Ця бібліотека забезпечує широкі можливості для аналізу статистичних взаємозв'язків та вивчення залежностей у даних, що допомагає покращити якість та точність прогнозів попиту на складські товари.

Бібліотека PySide6 необхідна для створення користувацького інтерфейсу. PySide – це аналог бібліотеки PyQt, який не обмежений у використанні, тобто доступно як для власного, так і комерційного використання.

Також для створення продукту було використано середовища: Qt designer, Visual Studio Code.

Qt Designer – це інструмент Qt для проектування та створення графічних інтерфейсів користувача (GUI) за допомогою віджетів Qt, для розробки інтерфейсу. У даній програмі можна створювати й налаштовувати свої вікна чи діалоги за принципом WYSIWYG (англ. what you see is what you get, що ви бачите, те й отримуєте) і тестувати їх за допомогою різних стилів і роздільної здатності. Віджети та форми, створені за допомогою Qt Designer, легко інтегруються з запрограмованим кодом. Усі властивості, встановлені в Qt Designer, можна динамічно змінювати в коді. Крім того, такі функції, як просування віджетів і спеціальні плагіни, дозволяють використовувати власні компоненти з Qt Designer.

Visual Studio Code від Microsoft є інтегрованим середовищем розробки, що надає засоби для створення програмного продукту з використанням мови програмування Python. Його потужність полягає в розширених можливостях розробки, включаючи організацію процесу програмування, керування версіями коду та підтримку складних програмних проєктів. Завдяки інтегрованій підтримці Python, Visual Studio Code дозволяє ефективно використовувати бібліотеки для аналізу даних, що ми вже описали: Pandas та Matplotlib, а також для реалізації моделей прогнозування, таких як ARIMA, SARIMA, що є ключовими для розробки програми прогнозування попиту.

## 3.2 Опис вхідних даних

Для тестування розробленого програмного продукту був використаний набір даних Store Item Demand Forecasting Challenge [19] з Kaggle. Цей набір є часовим рядом попиту на різні види товарів у магазинах з більш ніж 10000 рядків. У наборі присутні такі стовпці: date (дата), store (магазин), item (товар), sales (кількість продажів).

Для нашої програми оберемо продажі товару під номером 1 в усіх магазинах. Графік, який демонструє продажі зображений на рисунку 3.1.

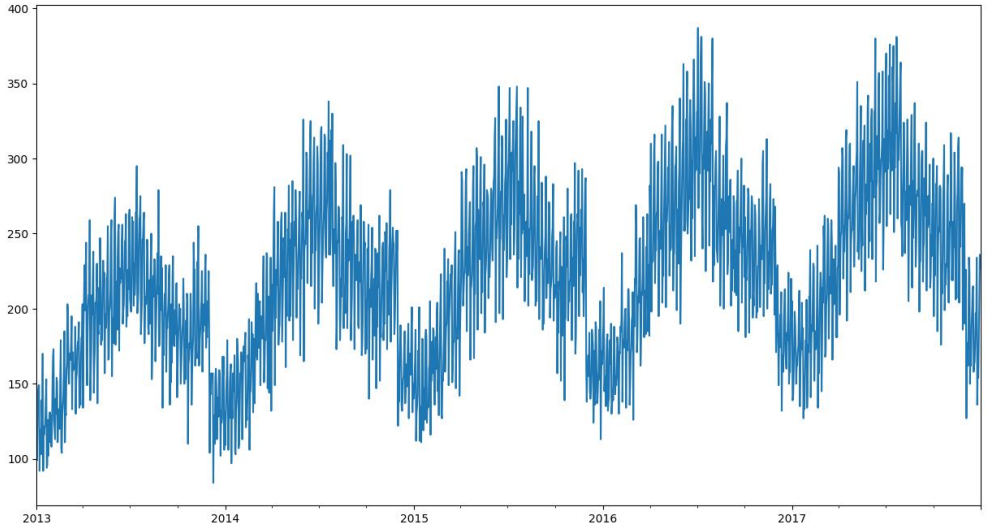



Рисунок 3.1 – Продажі першого товару

Наступним етапом є перевірка стаціонарності за допомогою теста Дікі-Фуллера. Результат тесту зображено на рисунку 3.2.

```
ADF Statistic: -2.292873
p-value: 0.174277
The data is not stationary
```

Рисунок 3.2 – Результат тесту на стаціонарність

У результаті було отримано, що p-значення перевищує 0.05, тобто ряд нестаціонарний.

З рисунку 3.1 можна також побачити наявність сезонного тренду, який триває рік.

## 3.3 Структура програми

Створений програмний продукт має 2 основні частини:

- основний код алгоритму аналізу ряду та прогнозування;
- користувацький інтерфейс.

Програма приймає користувацький набір даних у форматі .csv, корегує назви стовпців: задає індекс у вигляді дати і стовпцю, який відповідає кількості проданого товару присвоює назву target.

Після форматування даних, програма починає аналіз даних. Перевіряє чи є пропуски даних, стаціонарність даних, а також проводить декомпозицію ряду.

Стаціонарність даних перевіряється за допомогою тесту Дікі-Фуллера, який повертає p-значення і якщо це значення менше за 0.05 (чи 5%), то можна вважати, що ряд стаціонарний (отже немає ні тренду, ні сезонності).

Під декомпозицією ряду мається на увазі розділення ряду на його складові – тренд, сезонність та шум. Приклад результату декомпозиції наведений на рисунку 3.3.

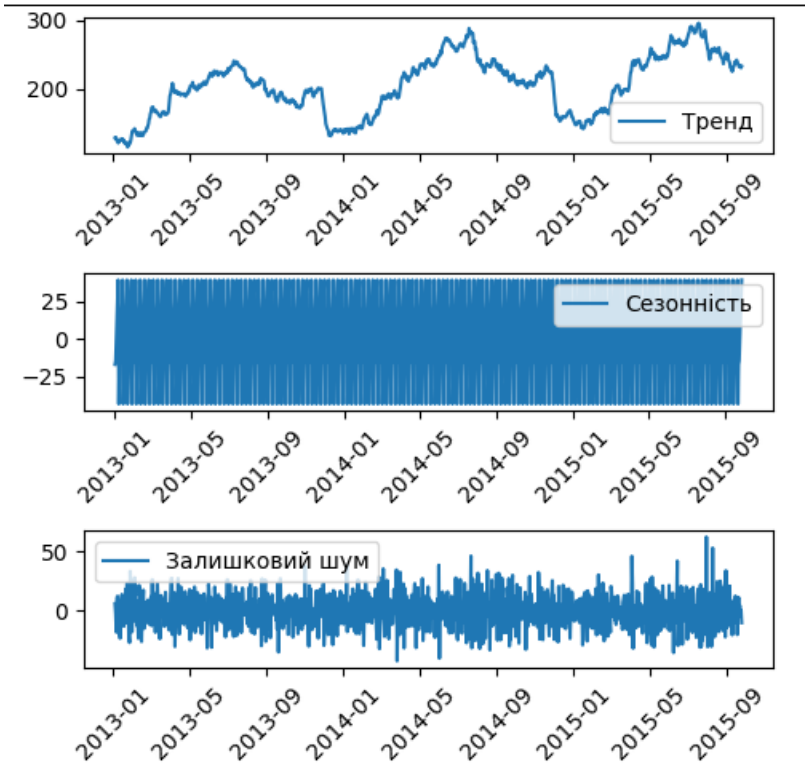



Рисунок 3.3 – Приклад декомпозиції часового ряду

Із декомпозиції можна точно визначити наявність тренду та сезонності. Найлегше визначити сезонність: якщо на графіку різниця максимуму і мінімуму велика, у нашому випадку великою різницею вважається більш ніж

1% від середнього значення по наших даних, то сезонність присутня. Також, якщо наблизити графік, можна помітити постійні збільшення і падіння з часом, а не приблизно пряму лінію.

Тренд визначається за допомогою виконання тесту лінійної регресії, тобто якщо нахил лінії регресії значно відрізняється від нуля (на основі p-значення), то тренд присутній. Тобто, перевіряється гіпотеза, що нахил буде більше за 0. Якщо p-значення більше за 0.05 (5%), то ми вважаємо, що тренд присутній.

У програмі також реалізований механізм оцінки коефіцієнтів для моделі ARIMA та SARIMA. Існує багато способів підбору найкращих параметрів як статистичних, так і прикладних. У програмі застосовано підхід, який спочатку визначає параметр інтегрування d, проводячи тест Дікі-Фуллера, потім починається перебір параметрів p і d. Параметри сезонності шукаються за допомогою проведення тесту Канови-Хансена.

Останнім етапом, для прогнозування часового ряду на задану кількість кроків з раніше знайденими або заданими параметрами, була використана модель SARIMA або ARIMA, якщо сезонні параметри нульові.

Результат виводиться у форматі графіку, на якому зображені як початкові дані, так і прогнозовані, щоб було видно як змінюється кількість продажів відносно заданого періоду часу.

## 3.4 Проектування інтерфейсу

У створеному програмному продукті після відкриття показується головне вікно, на якому є кнопки почати роботу, інформація та вихід. Вигляд початкового вікна показано на рисунку 3.4.

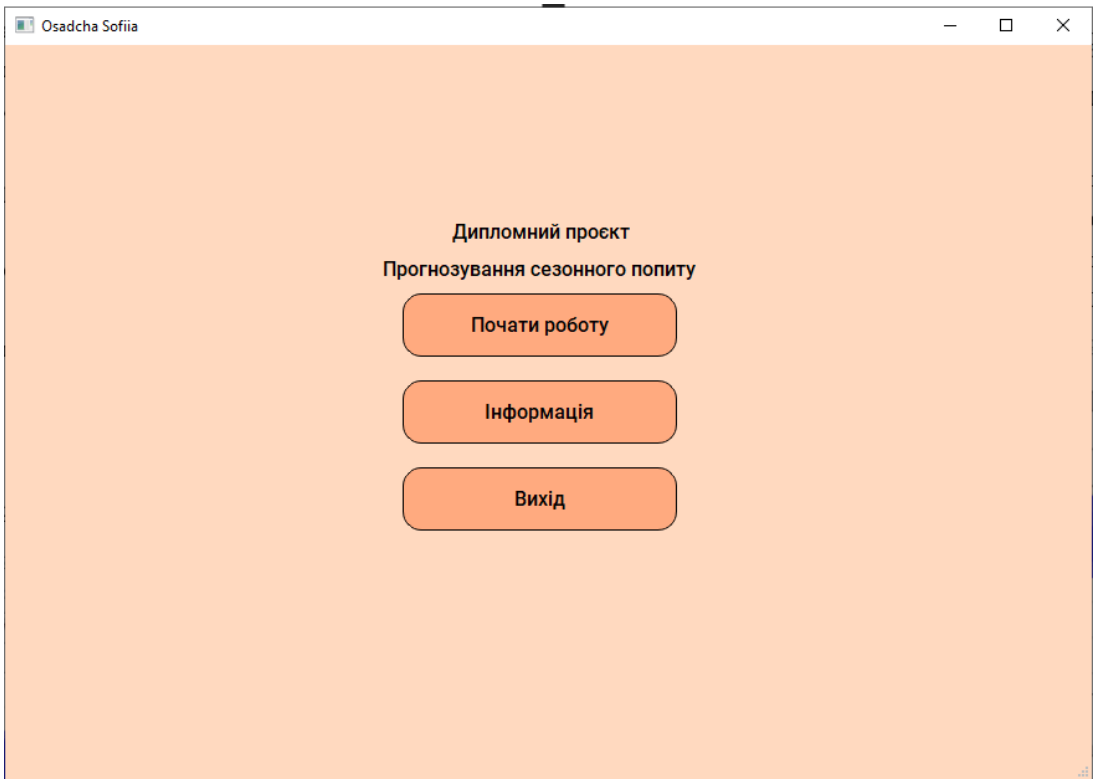



Рисунок 3.4 – Початкове вікно програми

Після натискання на кнопку почати роботу, відкривається головне вікно програми, яке має наступний користувацький інтерфейс, який зображено на рисунку 3.5.

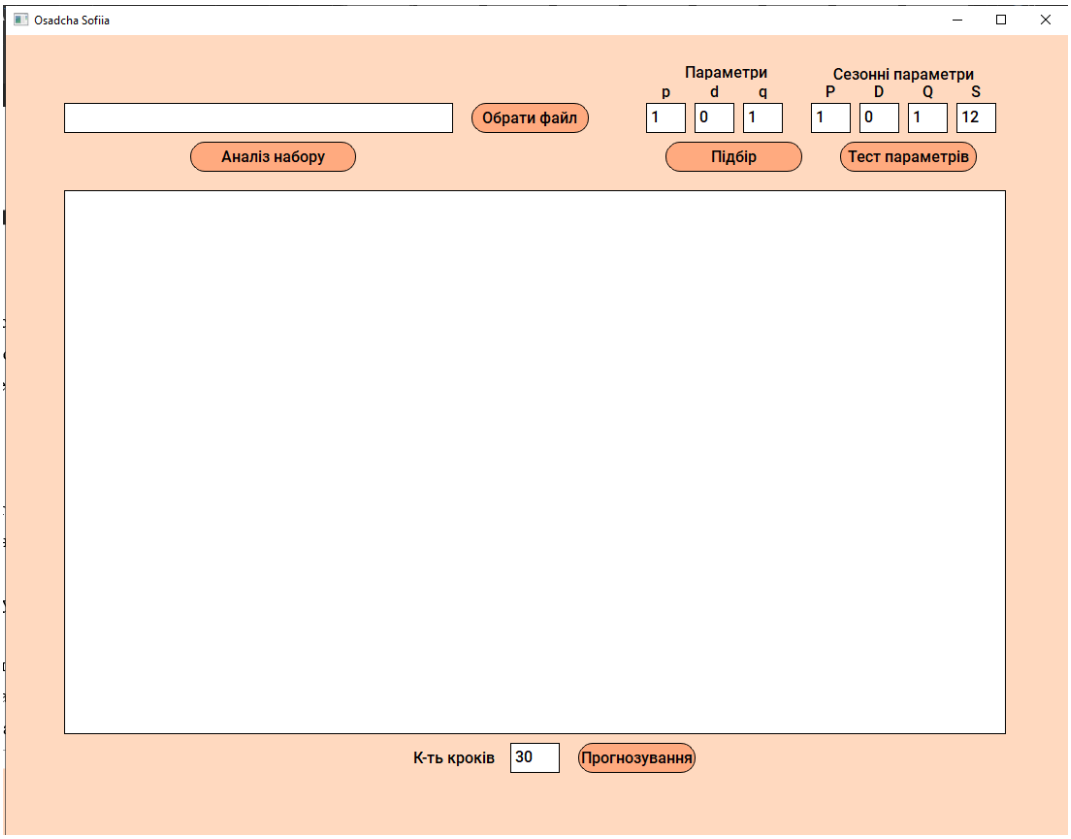



Рисунок 3.5 – Користувацький інтерфейс програми

Програма дає користувачу можливість ввести свої дані шляхом натискання на кнопку Обрати файл, як показано на рисунку 3.6.


Рисунок 3.6 – Приклад вибору даних

Після завантаження даних, на головному екрані з’явиться графічне зображення обраних даних, як показано на рисунку 3.7.

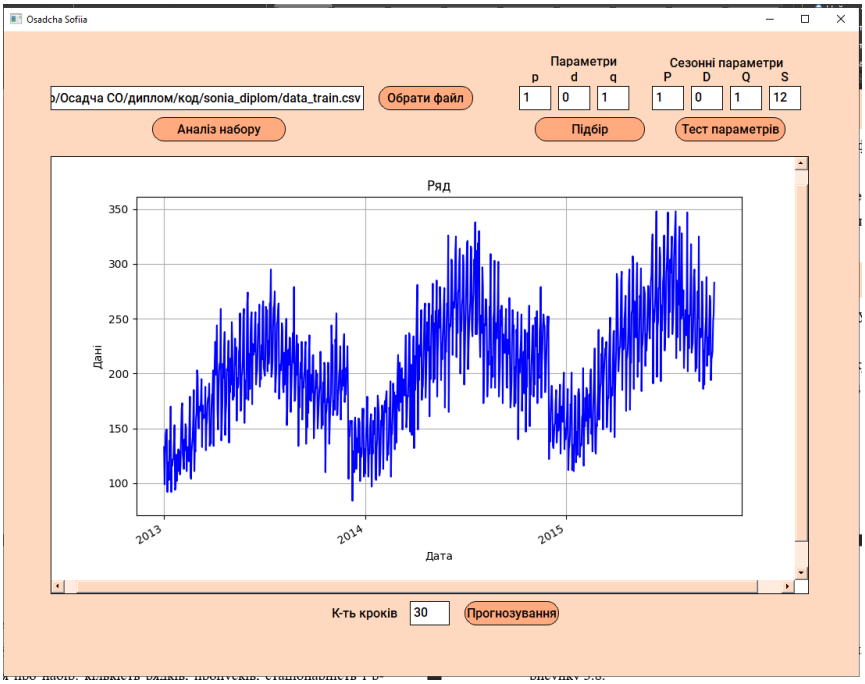



Рисунок 3.7 – Приклад відображення обраних даних

Після вибору даних, користувач може натиснути на кнопку «Аналіз даних» і йому буде відкрито вікно, де показані графік декомпозиції ряду, а також інформація про набір: кількість рядків, пропусків, стаціонарність і p-значення тесту Дікі-Фуллера, а також чи присутній тренд і сезонність у даних. Приклад зображено на рисунку 3.8.

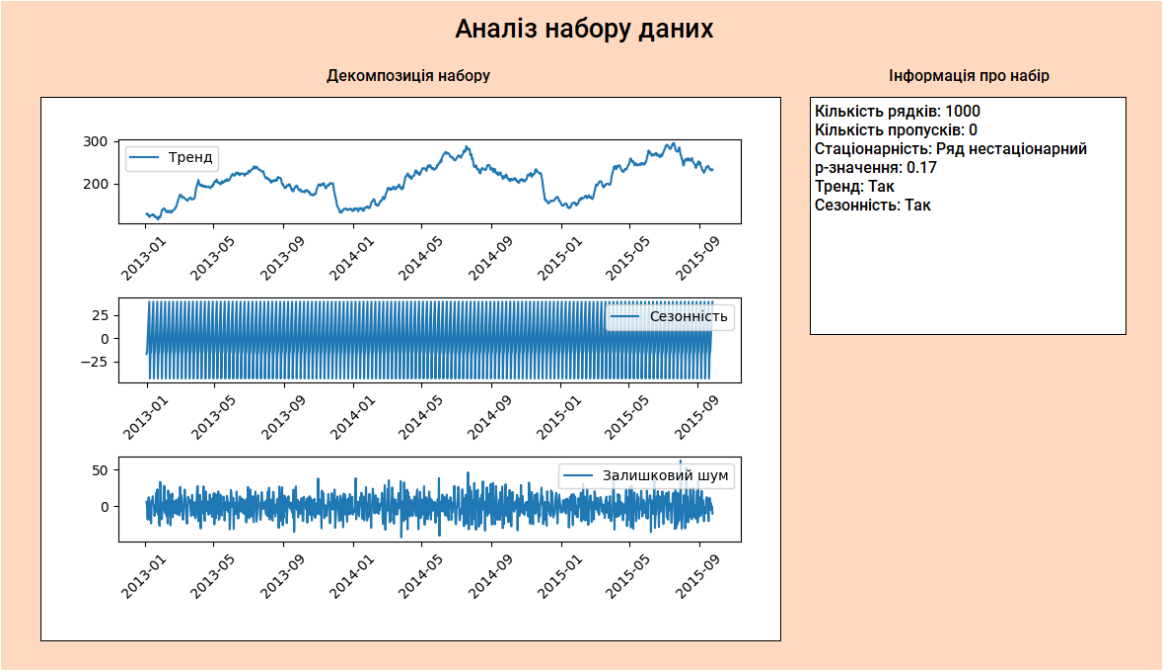



Рисунок 3.8 – Приклад проведеного аналізу даних

Після перегляду аналізу даних, користувач може самостійно обрати параметри для моделі SARIMA чи залишити сезонний порядок нульовим і вибрати модель ARIMA. Також у користувача є можливість натиснути на кнопку підбору параметрів і програма сама порахує оптимальні значення для заданих даних. Приклад зображено на рисунку 3.9.

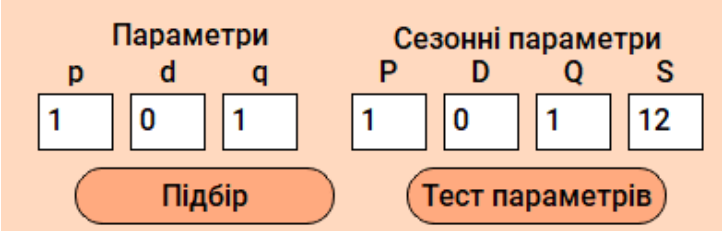



Рисунок 3.9 – Приклад вибору параметрів моделі

Також, користувач може протестувати параметри, які були обрані на частині набору даних шляхом натискання «Тест параметрів», як показано на рисунку 3.9. У результаті буде відкрито вікно з тестом параметрів, а також зі значеннями помилок та невеликим поясненням щодо того, що вони означають. Приклад вікна тесту параметрів показано на рисунку 3.10.


Рисунок 3.10 – Приклад проведеного тесту параметрів

Далі, для побудови прогнозу користувач має обрати кількість кроків прогнозування та натиснути кнопку Прогнозування. Приклад зображено на рисунку 3.11.

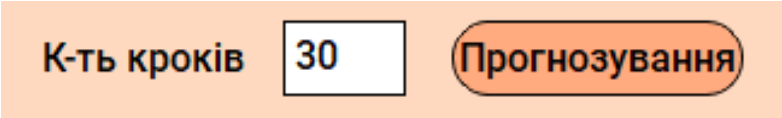



Рисунок 3.11 – Приклад вибору кроків та початку прогнозування

Після натискання на кнопку Прогнозування, користувачу буде виведено вікно з графіком, на якому присутні як його дані, так і прогноз на обрану кількість кроків. Приклад зображено на рисунку 3.12.

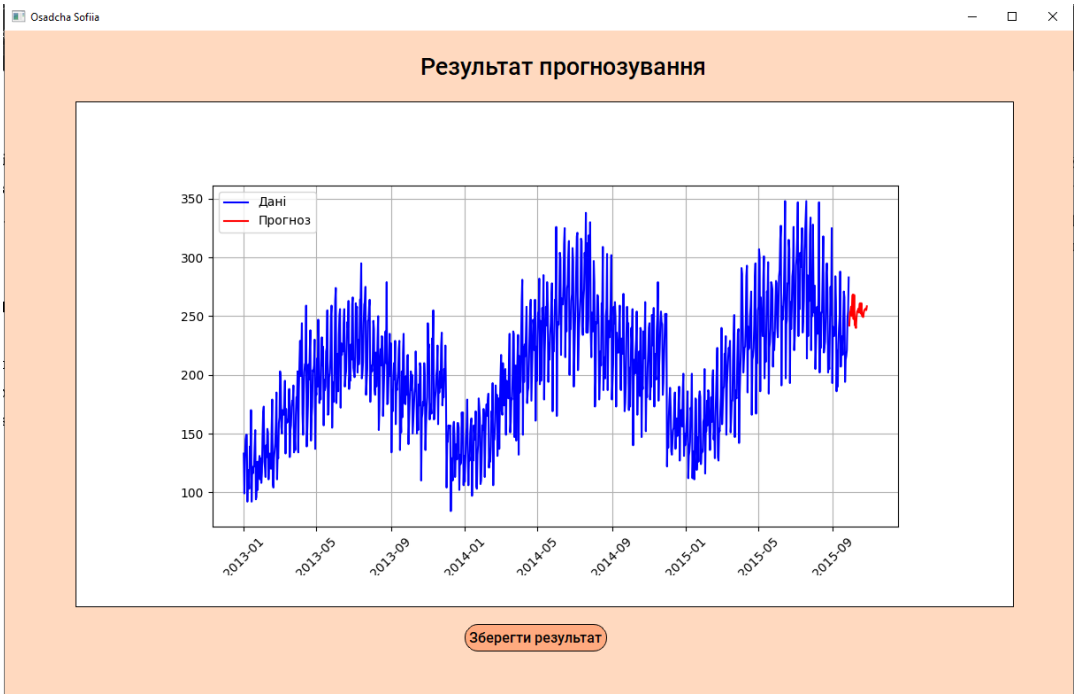



Рисунок 3.12 – Приклад результату прогнозування

В останню чергу, результат прогнозування можна зберегти на диск у форматі .csv чи Excel, натиснувши на кнопку Зберегти результат. Приклад зображено на рисунку 3.12.

## 3.5 Висновки до розділу 3

У третьому розділі було описано вибір мови програмування та програмного середовища. Було вирішено обрати мову програмування Python, яка є простою у використанні і не тільки. Для середовища розробки було обрано Visual Studio Code та Qt Designer, які забезпечили ефективний та зручний процес розробки. Visual Studio Code, завдяки своїй легкості та розширюваності, став основним інструментом для написання та налагодження коду. Цей текстовий редактор підтримує численні розширення, що дозволяють інтегрувати функціонал для роботи з різними мовами програмування та технологіями, включаючи Python, який використовувався у цьому проекті. Для створення графічного інтерфейсу користувача було використано Qt Designer, інтуїтивно зрозумілий інструмент для дизайну візуальних компонентів. Qt Designer дозволив легко створювати діалогові вікна, форми та інші елементи інтерфейсу шляхом перетягування компонентів на робочу область, що забезпечило швидкий та ефективний дизайн інтерфейсу.

Також було описано опис вхідних даних, структуру програми та проектування інтерфейсу. Для тестового набору даних було обрано датасет, на якому була проведена перевірка на стаціонарність. Структура програми дозволяє приймати користувацький набір даних у форматі .csv, потім перевіряє дані на стаціонарність та проводить прогнозування за допомогою моделей ARIMA та SARIMA. Потім результат виводиться у форматі графіку, Кожен з цих модулів виконує специфічні функції, забезпечуючи гнучкість та модульність програми.  У частині про проектування інтерфейсу описано та показано на рисунках усі створені вікна програми. Результат виводиться у форматі графіку.

# РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У цьому розділі здійснюється оцінка ключових характеристик програмного продукту, розробленого для прогнозування складських запасів за допомогою часових рядів.

Далі наведено дослідження, у якому враховуються не тільки економічні фактори, а ще й сумісність з майбутнім програмним продуктом. Для цього було розглянуто різні варіанти реалізації для забезпечення найбільш точної та оптимальної стратегії вибору. Щоб це реалізувати було використано функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка застосовується для оцінки вартості та функціональності продукту з метою підвищення його ефективності. Цей метод включає аналіз функцій продукту, визначення їхньої цінності та пошук шляхів зниження вартості без втрати якості чи продуктивності. ФВА допомагає виявити найекономічніші рішення, які забезпечують оптимальне співвідношення вартості і функціональності. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

## 4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи прогнозу стійкості фінансових показників. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Технічні вимоги до програмного продукту є наступні:

- функціонування на персональних комп'ютерах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

## 4.2 Обґрунтування функцій програмного продукту

Головна функція $F0$ – розробка програмного продукту, який вирішує задачу прогнозування інсульту. Беручи за основу цю функцію, можна виділити наступні:

$F1$ – вибір мови програмування;

$F2$ – якісний аналіз даних;

$F3$ – вибір середовища розробки.

Кожна з цих функцій має декілька варіантів реалізації:

Функція $F1$:

а) Мова програмування Python;

б) Мова програмування C++.

Функція $F2$:

а) Застосування statmodels;

б) Створення власної моделі.

Функція $F3$:

а) Visual Studio Code;

б) Jupyter Notebook.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1).

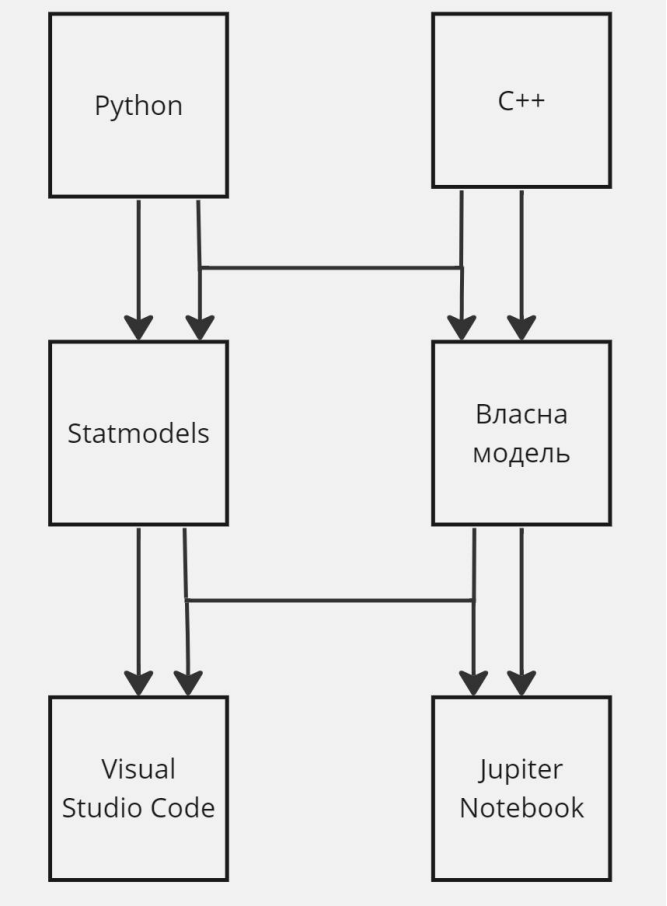



Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. На основі цієї карти будуємо позитивно-негативну матрицю варіантів основних функцій (див. табл. 4.1).

Таблиця 4.1 Позитивно-негативна матриця

| Функції | Варіанти реалізації | Переваги | Недоліки |
|---|---|---|---|
| $F1$ | А | Простий та зрозумілий синтаксис, що робить мову доступною для користувачів. | Може використовувати більше пам'яті через динамічну типізацію та інші особливості |
| | Б | Підтримка як об'єктно-орієнтованого, так і процедурного програмування, а також шаблонів | Складний синтаксис і концепції, що вимагають більше часу на освоєння |
| $F2$ | А | Інтуїтивний інтерфейс для створення статистичних моделей, що полегшує аналіз даних. | Працює добре для типових статистичних моделей, але може бути складно налаштувати для специфічних або нестандартних завдань. |
| | Б | Повний контроль над архітектурою моделі, що дозволяє реалізовувати унікальні алгоритми та структури. | Створення власної моделі з нуля вимагає значного часу і ресурсів. |

Продовження таблиці 4.1

| | | | |
|---|---|---|---|
| $F3$ | А | Широкий спектр плагінів, що розширюють функціональність, такі як автодоповнення, linting, і форматування коду. | У порівнянні з іншими легкими текстовими редакторами, VS Code може запускатися довше. |
| | Б | Підтримка інтерактивного виконання коду, що дозволяє швидко перевіряти результати та виконувати аналіз даних у реальному часі. | Не завжди легко керувати версіями коду у порівнянні з традиційними середовищами розробки. |

У результаті бачимо, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція $F1$:

Для оптимальної робити обираємо варіант А, оскільки він має такі переваги: наявність великої кількості бібліотек, простота використання та швидкість вивчення.

Функція $F2$:

Обидва варіанти можна використовувати в розробці.

Функція $F3$:

Віддаємо перевагу варіанту А через наявність великої кількості зручних плагінів, що полегшують розробку.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

$F1а – F2а – F3а$

$F1а – F2б – F3а$

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

## 4.3 Обґрунтування системи параметрів ПП

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- X1 – швидкодія мови програмування;
- X2 – об'єм пам'яті;
- X3 – час обробки даних;
- X4 – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у таблиці 4.2.

Таблиця 4.2 Основні параметри ПП

| Назва Параметра | Умовні позначення | Одиниці виміру | Значення параметра | | |
|---|---|---|---|---|---|
| | | | гірші | середні | кращі |
| Швидкодія мови програмування | X1 | оп/мс | 100 | 150 | 200 |
| Об'єм пам'яті | X2 | МБ | 200 | 150 | 100 |
| Час обробки даних | X3 | с | 10 | 6 | 4 |
| Потенційний об’єм програмного коду | X4 | кількість рядків коду | 1500 | 1000 | 700 |

За даними таблиці 4.2 будуються графічні характеристики параметрів –

рис. 4.2 – рис. 4.5.

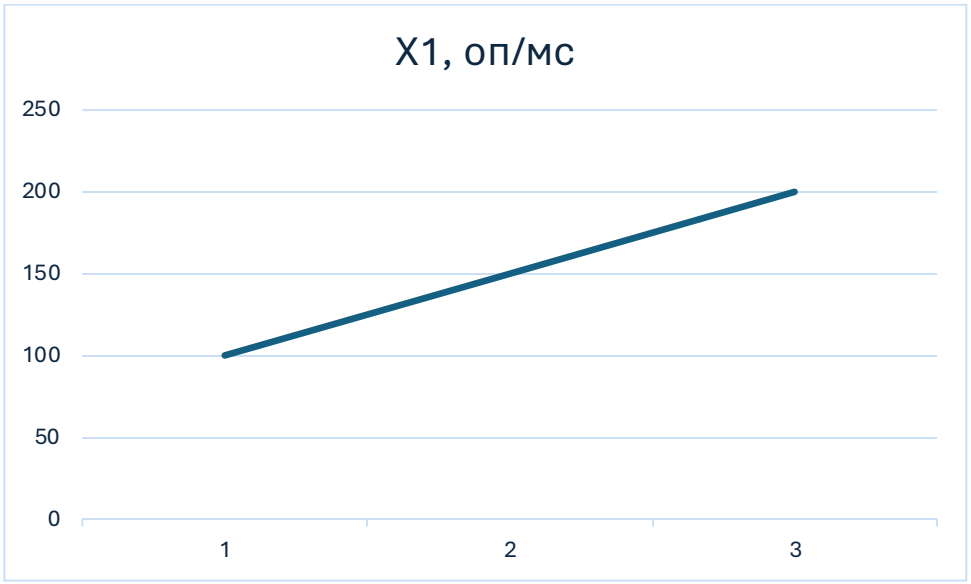



Рисунок 4.2 – X1, швидкодія мови програмування

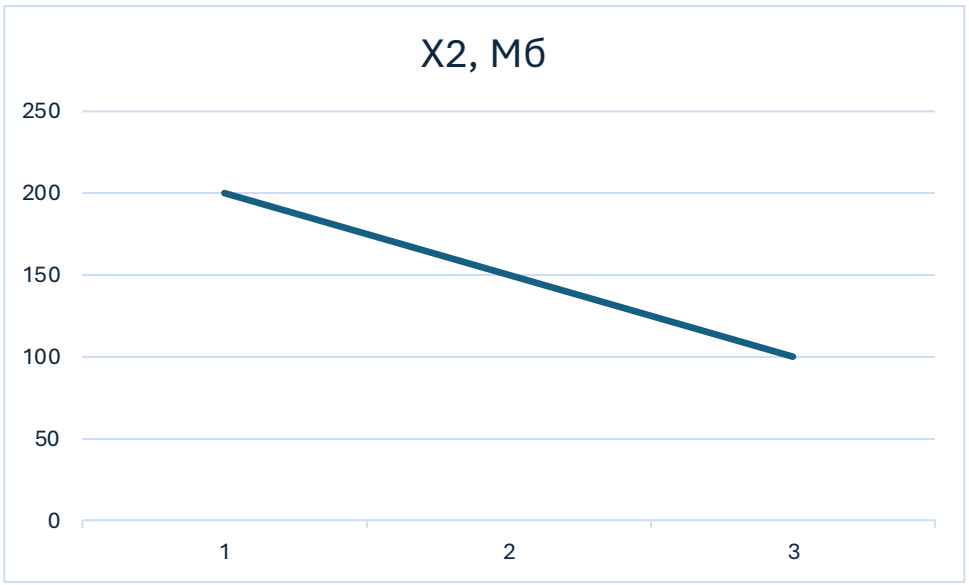



Рисунок 4.3 – X2, Об'єм пам'яті

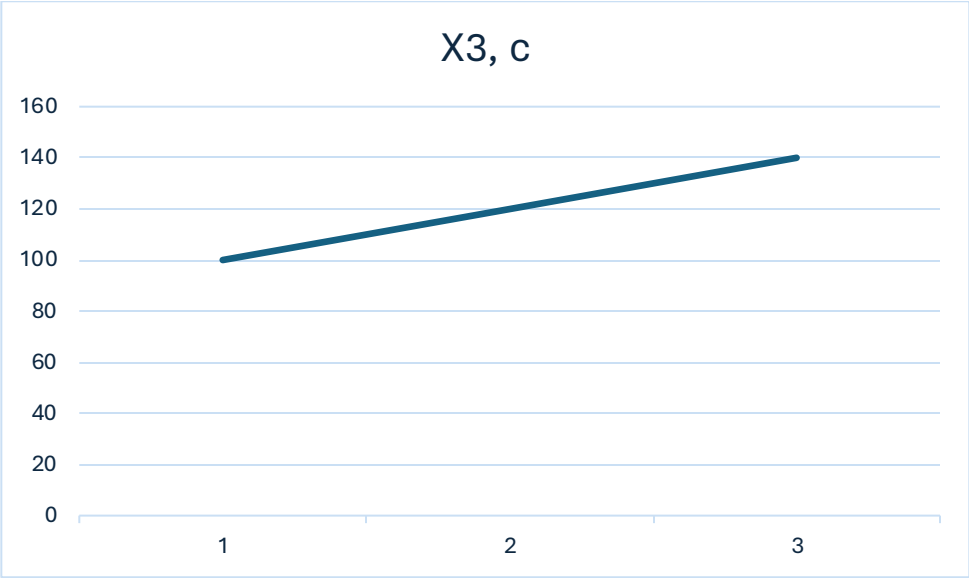



Рисунок 4.4 – X3, Час обробки даних

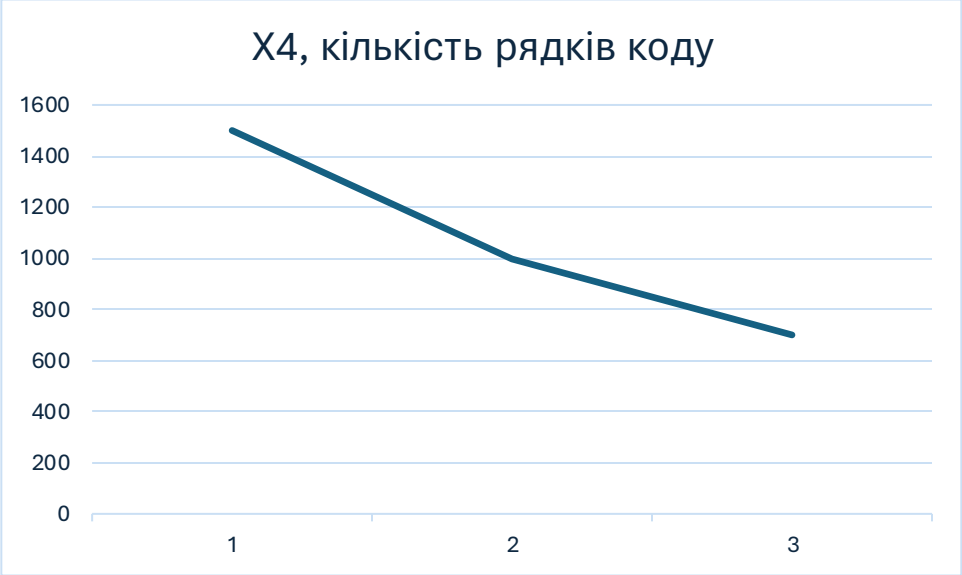



Рисунок 4.5 – X4, потенційний об’єм програмного коду

## 4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні

параметрів моделей адаптивного прогнозування і обчислення прогнозних значень. Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого
- використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 Результати ранжування параметрів

| Позначення параметра | Назва параметра | Одиниці виміру | Ранг параметра за оцінкою експерта | | | | | | | Сума рангів $R_i$ | Відхи-лення $\Delta_i$ | $\Delta_i^2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | | |
| X1 | Швидкодія мови програмування | оп/мс | 2 | 1 | 1 | 2 | 2 | 2 | 1 | 11 | -6,5 | 42,25 |
| X2 | Об’єм пам’яті | МБ | 1 | 2 | 2 | 1 | 1 | 1 | 2 | 10 | -7,5 | 56,25 |
| X3 | Час обробки даних | с | 3 | 4 | 3 | 3 | 4 | 3 | 4 | 24 | 6,5 | 42,25 |
| X4 | Потенційний об’єм програмного коду | кількість рядків коду | 4 | 3 | 4 | 4 | 3 | 4 | 3 | 25 | 7,5 | 56,25 |
| | Разом | | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 70 | 0 | 197 |

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{i_j} R_{ij} = \frac{Nn(n+1)}{2} = 70, \quad (4.1)$$

де $N$ – число експертів,

$n$ – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17{,}5, \quad (4.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T. \quad (4.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 197. \quad (4.4)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 197}{7^2(4^3-4)} = 0{,}804 > Wk = 0{,}67. \quad (4.5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 Попарне порівняння параметрів.

| Параметри | Експерти | | | | | | | Кінцева оцінка | Числове значення |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | |
| X1 і X2 | > | < | < | > | > | > | < | > | 1,5 |
| X1 і X3 | < | < | < | < | < | < | < | < | 0,5 |
| X1 і X4 | < | < | < | < | < | < | < | < | 0,5 |
| X2 і X3 | < | < | < | < | < | < | < | < | 0,5 |
| X2 і X4 | < | < | < | < | < | < | < | < | 0,5 |
| X3 і X4 | < | > | < | < | > | < | > | < | 0,5 |

Числове значення, що визначає ступінь переваги і–го параметра над j–тим, $a_{ij}$ визначається по формулі:

$$a_{ij} = \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1{,}0 \text{ при } X_i = X_j \\ 0{,}5 \text{ при } X_i < X_j \end{cases}. \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю A=$\|a_{ij}\|$.

Для кожного параметра зробимо розрахунок вагомості $K_{B_i}$ за наступними формулами:

$$K_{Bi} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \quad (4.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{Bi} = \frac{b'_i}{\sum_{i=1}^{n} b'_i}, \quad (4.9)$$

$$\acute{b_\iota} = \sum_{i=1}^{N} a_{ij} b_j \qquad (4.10)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 Розрахунок вагомості параметрів

| Параметри $x_i$ | Параметри $x_j$ | | | | Перша ітер. | | Друга ітер. | | Третя ітер. | |
|---|---|---|---|---|---|---|---|---|---|---|
| | X1 | X2 | X3 | X4 | $b_i$ | $K_{Bi}$ | $b_i^1$ | $K_{Bi}^1$ | $b_i^2$ | $K_{Bi}^2$ |
| X1 | 1 | 1,5 | 0,5 | 0,5 | 3,5 | 0,2188 | 12,25 | 0,2076 | 44,875 | 0,2078 |
| X2 | 0,5 | 1 | 0,5 | 0,5 | 2,5 | 0,1563 | 9,25 | 0,1568 | 34,125 | 0,1580 |
| X3 | 1,5 | 1,5 | 1 | 0,5 | 4,5 | 0,2813 | 16,25 | 0,2754 | 59,125 | 0,2737 |
| X4 | 1,5 | 1,5 | 1,5 | 1 | 5,5 | 0,3438 | 21,25 | 0,3602 | 77,875 | 0,3605 |
| Всього: | | | | | 16 | 1 | 59 | 1 | 216 | 1 |

## 4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо. Абсолютні значення параметрів X2 (об'єм пам'яті), X3 (час обробки даних) та X4 (потенційний об'єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП. Абсолютне значення параметра X1 (швидкість роботи мови програмування) обрано не найгіршим. Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується як у таблиці 4.6:

$$K_k(j) = \sum_{i=1}^{n} K_{Bi,j} B_{ij}, \qquad (4.11)$$

де $n$ – кількість параметрів;

$K_{Bi}$ – коефіцієнт вагомості $i$– го параметра;

$B_i$ – оцінка і– го параметра в балах.

Таблиця 4.6 Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

| Основні функції | Варіант реалізації функції | Параметри | Абсолютне значення параметра | Бальна оцінка параметра | Коефіцієнт вагомості параметра | Коефіцієнт рівня якості |
|---|---|---|---|---|---|---|
| F1 | А | X1 | 130 | 4 | 0,21 | 0,84 |
| F2 | А | X2 | 120 | 7 | 0,16 | 1,12 |
| | Б | X2 | 150 | 5 | 0,16 | 0,8 |
| F3 | А | X3 | 5 | 8 | 0,27 | 2,16 |
| F4 | А | X4 | 700 | 10 | 0,36 | 3,6 |

За даними з таблиці 4.6 за формулою:

$$K_k = K_{TY}[F_{1k}] + K_{TY}[F_{2k}] + \cdots + K_{TY}[F_{zk}], \qquad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{k1} = 0{,}84 + 1{,}12 + 0{,}8 + 3{,}6 = 7{,}72$$

$$K_{k2} = 0{,}84 + 1{,}12 + 0{,}8 + 3{,}6 = 7{,}4$$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

## 4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту.

2. Розробка програмної оболонки.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$T_0 = T_p \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТМ}, \quad (4.13)$$

де $T_p$ – трудомісткість розробки ПП;

$K_П$ – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

$K_М$ – коефіцієнт рівня мови програмування;

$K_{СТ}$ –коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТМ}$– коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму

1, трудомісткість дорівнює: $T_p = 80$ людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П = 1{,}4$. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ} = 0.6$. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 80 \cdot 1{,}4 \cdot 0{,}6 = 75{,}6 \text{ людино} - \text{днів}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $T_p$ =35 людино-днів, $K_П = 0{,}5$, $K_{СК}$ = 1, $K_{СТ}$ =0.8:

$$T_1 = 35 \cdot 0{,}5 \cdot 0{,}8 = 14 \text{ людино} - \text{днів}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_{\text{I}} = (75{,}6 + 14 + 4{,}8 + 14) \cdot 8 = 865{,}6 \text{ людино} - \text{годин}$$

$$\text{T}_{\text{II}} = (75{,}6 + 14 + 6{,}91 + 14) \cdot 8 = 884{,}08 \text{ людино} - \text{годин}$$

Найбільш високу трудомісткість має варіант II.

У розробці бере участь один програміст з окладом 16500 грн., один аналітик в області даних з окладом 20446. Визначимо середню зарплату за годину за формулою:

$$Сч = \frac{M}{Tm \cdot t} \text{грн.}, \qquad (4.14)$$

де М – місячний оклад працівників;

$T_m$ – кількість робочих днів тиждень;

$t$ – кількість робочих годин в день.

$$Сч = \frac{16500+20446}{3 \cdot 21 \cdot 8} = 73{,}3 \text{ грн.} \quad (4.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$C_{ЗП} = Сч \cdot T_i \cdot K_Д, \quad (4.16)$$

де СЧ– величина погодинної оплати праці програміста;

$T_i$ – трудомісткість відповідного завдання;

$K_Д$ – норматив, який враховує додаткову заробітну плату. Зарплата розробників за варіантами становить:

I. $C_{ЗП} = 73{,}3 \cdot 1416{,}64 \cdot 1{,}2 = 125117{,}6$грн.

II. $C_{ЗП} = 73{,}3 \cdot 1524{,}15 \cdot 1{,}2 = 134612{,}9$грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $C_{ВІД} = C_{ЗП} \cdot 0{,}22 = 27525{,}9$ грн.

II. $C_{ВІД} = C_{ЗП} \cdot 0{,}22 = 29614{,}8$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($C_M$)

Так як одна ЕОМ обслуговує одного програміста з окладом 16500 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$Cr = 12 \cdot M \cdot K_З = 12 \cdot 16500 \cdot 0{,}2 = 39600 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{ЗП} = Cr \cdot (1 + K_З) = 39600 \cdot (1 + 0{,}2) = 47520 \text{ грн.}$$

Відрахування на соціальний внесок:

$$C_{ВІД} = C_{ЗП} \cdot 0{,}22 = 47520 \cdot 0{,}22 = 10454{,}4 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 12000 грн.

$$C_A = K_{ТМ} \cdot K_A \cdot Ц_{ПР} = 1{,}5 \cdot 0{,}25 \cdot 12000 = 4500 \text{ грн.,}$$

де $K_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$K_A$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$C_P = K_{ТМ} \cdot Ц_{ПР} \cdot K_P = 1{,}5 \cdot 12000 \cdot 0{,}05 = 900 \text{ грн.,}$$

де $K_P$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$T_{ЕФ} = (Д_k - Д_В - Д_С - Д_Р) \cdot t_З \cdot K_B = (365 - 104 - 12 - 9) \cdot 8 \cdot 0{,}9 = \\ = 1634 \text{ годин,}$$

де $Д_k$– календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$K_B$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$C_{ЕЛ} = T_{ЕФ} \cdot N_C \cdot K_З \cdot Ц_{ЕН} = 1634 \cdot 0{,}3 \cdot 0{,}5 \cdot 5{,}23 = 1281{,}9 \text{ грн.},$$

де $N_C$ – середньо-споживча потужність приладу;

$K_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$C_Н = Ц_{ПР} \cdot 0{,}67 = 12000 \cdot 0{,}67 = 8040 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$C_{ЕКС} = C_{ЗП} + C_{ВІД} + C_А + C_Р + C_{ЕЛ} + C_Н, \qquad (4.17)$$

$$C_{ЕКС} = 47520 + 10454{,}4 + 4500 + 900 + 1281{,}9 + 8040 = 72696{,}3 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{М-Г} = \frac{C_{ЕКС}}{T_{ЕФ}} = \frac{72696{,}3}{1634} = 44{,}5 \frac{\text{грн}}{\text{год}}.$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_{\text{М}} = C_{\text{М}-\text{Г}} \cdot T, \tag{4.18}$$

I. $C_{\text{М}} = 44{,}5 \cdot 1416{,}64 = 63040{,}4$грн.

II. $C_{\text{М}} = 44{,}5 \cdot 1524{,}15 = 67824{,}7$ грн.

Накладні витрати складають 67% від заробітної плати:

$$C_{\text{Н}} = C_{\text{ЗП}} \cdot 0{,}67, \tag{4.19}$$

I. $C_{\text{Н}} = 125117{,}6 \cdot 0{,}67 = 83828{,}8$ грн.

II. $C_{\text{Н}} = 134612{,}9 \cdot 0{,}67 = 90190{,}6$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$C_{\text{ПП}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_{\text{М}} + C_{\text{Н}}, \tag{4.20}$$

I. $C_{\text{ПП}} = 125117{,}6 + 27525{,}9 + 63040{,}4 + 83828{,}8 = 299512{,}7$ грн.

II. $C_{\text{ПП}} = 134612{,}9 + 29614{,}8 + 67824{,}7 + 90190{,}6 = 322243$грн.

## 4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$K_{ТЕРj} = K_{Кj}/C_{Фj}, \quad (4.21)$$

$$K_{ТЕР1} = \frac{7{,}72}{299512{,}7} = 2{,}57 \cdot 10^{-5}$$

$$K_{ТЕР2} = \frac{7{,}4}{322243} = 2{,}296 \cdot 10^{-5}$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $K_{ТЕР1} = 2{,}57 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $K_{ТЕР} = 2{,}57 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- мова програмування – Python;
- застосування Statsmodels;
- використання середовища Visual Studio Code.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, непоганий функціонал і швидкодію.

## 4.8 Висновки до розділу 4

Проведено повний функціонально-вартісний аналіз програмного продукту. Визначено та проведено оцінку основних функцій програмного продукту. Визначено параметри, які характеризують програмний продукт. Проведено експертне оцінювання параметрів та аналіз якості варіантів реалізації функцій.

Проведено економічний аналіз варіантів розробки – трудомісткість, витрати на заробітну плату та інші витрати.

На основі аналізу вибрано варіант реалізації програмного продукту.

# ВИСНОВКИ

В умовах глобалізації та швидких змін на ринку, підприємства стикаються з необхідністю оперативного реагування на коливання попиту, які можуть суттєво впливати на їхню діяльність. Здатність точно передбачити ці коливання дозволяє компаніям ефективніше управляти запасами, знижуючи витрати на зберігання і уникати дефіциту товарів. Це особливо важливо для компаній, що працюють в галузях з високою конкуренцією, де навіть невеликі помилки в управлінні запасами можуть призвести до значних фінансових втрат.

Сучасні методи, такі як ARIMA та SARIMA, дозволяють враховувати складні сезонні тенденції, що робить прогнози більш надійними та точними. Це важливо не тільки для великих корпорацій, але й для малих та середніх підприємств, які можуть використовувати такі інструменти для оптимізації своїх процесів без значних інвестицій в аналітичні системи. Використання доступних і ефективних програмних рішень дозволяє цим компаніям конкурувати на рівних умовах з більшими гравцями ринку.

Метою даної дипломної роботи було створити програмне забезпечення, що дозволяє робити прогноз попиту складських запасів, щоб покращити планування роботи підприємств. Під час розробки програмного продукту було враховано необхідність створення простого і зрозумілого користувацького інтерфейсу. Все, що потрібно зробити користувачу, це завантажити історичні дані попиту на деякий товар та отримати прогноз на найближчий час. Цей підхід значно спрощує процес прогнозування і робить його доступним навіть для користувачів без спеціальних знань в області аналізу даних.

Розроблений програмний продукт забезпечує підприємства інструментом для більш точного планування запасів. Це, в свою чергу, допомагає знизити витрати, уникати дефіциту товарів і підвищувати рівень задоволеності клієнтів завдяки кращому обслуговуванню. Крім того, ефективне управління запасами сприяє оптимізації всього логістичного ланцюжка, що дозволяє підприємствам бути більш гнучкими та адаптивними в умовах мінливого ринку.

Таким чином, створений програмний продукт є важливим інструментом для підприємств різного масштабу, допомагаючи їм знижувати витрати, підвищувати ефективність і конкурентоспроможність на ринку. Використання сучасних методів прогнозування та зручного інтерфейсу забезпечує високу точність і надійність отриманих прогнозів, що є ключовим фактором успішного управління складськими запасами в сучасних умовах бізнесу.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ


1. Waters D. Logistics: an introduction to supply chain management. New York: PALGRAVE MACMILLAN, 2003. 384 p.
2. Луценко І.С. Логістичне управління запасами: навчально-методичний комплекс дисципліни: навч. посіб. для студ. спеціальності 073 «Менеджмент». Київ : КПІ ім. Ігоря Сікорського, 2021. 69 с.
3. Воскобоєва О.В., Воскобоєва О.С. Стратегія управління товарними запасами. Вісник ЖДТУ. 2011. № 4(58). С. 197-199.
4. Визначення та переваги методу Just In Time в логістиці. URL: https://firmao.com.ua/blog_net/ua/management/definition-and-advantages-of-the-just-in-time-method-in-logistics
5. Safety stock: how to calculate the ideal safety stock level?. URL: https://www.colibri-snop.com/safety-stock/
6. Guide to Understanding Vendor Managed Inventory (VMI). URL: https://qoblex.com/learning-center/vendor-managed-inventory/
7. Савченко Л.В. Логістика: Конспект лекцій. Київ: НАУ, 2006. 140 с.
8. Understanding and Minimizing Inventory Holding Costs Effectively. URL: https://cashflowinventory.com/blog/holding-costs/
9. Maximizing the Customer Service Experience. URL: https://www.greatnesswithin.com/customer-service
10. Inventory-management-with-mrp-balancing-cost-v.-inventory. URL: https://www.visibility.com/blog/inventory-management-with-mrp-balancing-cost-v.-inventory
11. Бутенко О.П., Опікунова Н.В., Гончарова А.С. Синхромаркетинг: Сутність, методи та компенсаторні інструменти. Ефективна економіка.: електрон. наук. фахове вид. 2020. Вип. 4. 8 с. URL: http://www.economy.nayka.com.ua/pdf/4_2020/94.pdf

12. Шевченко С. Як впливає сезонність на попит на товари і послуги? URL: https://adwservice.com.ua/uk/sezonnist-popytu (08.10.2021).
13. Назарова Г. Г., Хакимов Д.С. Сезонність у туризмі: проблеми і измерения. Вісник ДІТБ. 2001. № 5. С. 60-63.
14. Основи моделювання ринкових ситуацій. URL: https://web.posibnyky.vntu.edu.ua/fksa/12kocubynsky,kyslycia_osn_model_rynk_sytuac/p4.html
15. Box G., Jenkins G. Time Series Analysis: Forecasting and Control. Revised Edition, Holden Day, San Francisco, 1976. 575 p.
16. Introduction to Time Series Analysis and key concepts. Medium. URL: https://medium.com/analytics-vidhya/introduction-to-time-series-analysis-and-key-concepts-dbf6c394984f
17. Кафедра статистики і вищої математики. Методичні матеріали. Часові ряди.URL: https://kstat.pnu.edu.ua/wp-content/uploads/sites/63/2018/04/%D0%A7%D0%B0%D1%81%D0%BE%D0%B2%D1%96-%D1%80%D1%8F%D0%B4%D0%B8.pdf
18. Бідюк П.І., Романенко В.Д., Тимощук О.Л. Аналіз часових рядів. Київ: «Політехніка», 2012. 608 с.
19. Воробець І. Використання моделей ARIMA для прогнозування часових рядів із властивістю циклічності: матеріали VI Міжнародної студентської науково-технічної конференції "Природничі та гуманітарні науки. Актуальні питання" (Тернопіль, 27-28 кві. 2023). Тернопіль: ТНТУ ім. І. Пулюя. С. 122-123.
20. Hyndman R.J., Athanasopoulos G. Forecasting: principles and practice. 3rd ed. Melbourne, Australia : OTexts, 2021. 442 p.
21. SARIMA (Seasonal Autoregressive Integrated Moving Average). URL: https://www.geeksforgeeks.org/sarima-seasonal-autoregressive-integrated-moving-average/

22. Store Item Demand Forecasting Challenge. URL: https://www.kaggle.com/competitions/demand-forecasting-kernels-only

23. Воронко Р.М., Грезенталь В.О. Класифікація товарних запасів на підприємствах роздрібної торгівлі. Науковий вісник НЛТУ України. 2012. Вип. 22.7. С. 164-169.

# ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

## Модуль main.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error
import statsmodels.api as sm
import itertools
from pmdarima import auto_arima

def load_data(path):
    return pd.read_csv(path)

def format_data(data):
    # columns = list(data.columns)
    data.rename(columns={list(data.columns)[0]: 'date'}, inplace=True)
    data['date'] = pd.to_datetime(data['date'])
    data = data.set_index('date')
    data.index.name = None
    columns = list(data.columns)
    columns[0] = "target"
    data.columns = columns
    data = data.asfreq('D')
    return data


def analize_data(data, decomposition):
    line_count = data.shape[0]
```

```
            # Check for missing values
            missing_values = data.isnull().sum()

            # Check for stationarity
            is_stationary = "Ряд стаціонарний"
            result = adfuller(data)
            if result[1] > 0.05:
                is_stationary = "Ряд нестаціонарний"
            else:
                is_stationary = "Ряд стаціонарний"

            # trend
            trend = decomposition.trend.dropna()  # Remove NaN values
            trend_p_value                    =                    sm.OLS(trend,
sm.add_constant(np.arange(len(trend)))).fit().pvalues[1]
            has_trend = trend_p_value < 0.05
            trend_outp = "Так" if has_trend else "Ні"

            # seasonality
            seasonal = decomposition.seasonal
            #  Check  if  the  range  (max  -  min)  of  seasonal  component  is
significantly large
            seasonal_amplitude = (seasonal.max() - seasonal.min()) / 2
            has_seasonality = seasonal_amplitude > 0.01 * np.abs(data).mean()

            seasonality_outp = "Так" if has_seasonality.iloc[0] else "Ні"

            info  =  f"Кількість  рядків:  {line_count}\nКількість  пропусків:
{int(missing_values.iloc[0])}\nСтаціонарність:   {is_stationary}\np-значення:
{round(result[1],2)}\nТренд: {trend_outp}\nСезонність: {seasonality_outp}"

            return info


        def predict(data, order, seasonal_order, n_steps):
            model = SARIMAX(data,
```

```
                                        order=order,
                                        seasonal_order=seasonal_order,
                                        enforce_stationarity=False,
                                        enforce_invertibility=False)
        model_fit = model.fit()
        result = model_fit.predict(start=len(data), end=len(data)+n_steps-
1)
        return result

    def auto_params(data, seasonal_order):
        model = auto_arima(data, seasonal=True, m=seasonal_order,
trace=True,
                           error_action='ignore', suppress_warnings=True,
                           stepwise=True)
        order = model.order  # (p, d, q)
        seasonal_order = model.seasonal_order  # (P, D, Q, S)
        return order, seasonal_order
```

Модуль mainwindow.py

```
    # This Python file uses the following encoding: utf-8
    import sys
    from PySide6.QtWidgets import QApplication, QMainWindow, QFileDialog,
QGraphicsScene, QMessageBox
    from PySide6.QtCore import Qt
    from PySide6.QtGui import QPen, QColor
    import pandas as pd
    import matplotlib.pyplot as plt
    from matplotlib.backends.backend_qtagg import FigureCanvasQTAgg as
FigureCanvas
    import matplotlib.dates as mdates
    from matplotlib.figure import Figure
    from statsmodels.tsa.seasonal import seasonal_decompose
    from sklearn.metrics import mean_squared_error,
mean_absolute_percentage_error, mean_absolute_error, r2_score
    import ui_resultwindow
```

```
import ui_startwindow
import ui_infowindow
import ui_mainwindow
import ui_finalwindow
import ui_edawindow
import main


class StartWindow(QMainWindow):
    def __init__(self, parent=None):
        super(StartWindow, self).__init__(parent)
        self.ui = ui_startwindow.Ui_MainWindow()
        self.ui.setupUi(self)
        self.ui.pushButton.clicked.connect(self.start_program)
        self.ui.pushButton_2.clicked.connect(self.info_program)
        self.ui.pushButton_3.clicked.connect(self.exit_program)
        self.main_window = MainWindow()
        self.info_window = InfoWindow()

    def start_program(self):
        self.main_window.show()
        self.close()

    def info_program(self):
        self.info_window.show()

    def exit_program(self):
        sys.exit(app.exec())

class InfoWindow(QMainWindow):
    def __init__(self, parent=None):
        super(InfoWindow, self).__init__(parent)
        self.ui = ui_infowindow.Ui_MainWindow()
        self.ui.setupUi(self)

    def exit_window(self):
```

```
            self.close()

    class MainWindow(QMainWindow):
        def __init__(self, parent=None):
            super(MainWindow, self).__init__()
            self.ui = ui_mainwindow.Ui_MainWindow()
            self.ui.setupUi(self)
            self.ui.pushButton.clicked.connect(self.loadCSV)
            self.ui.pushButton_3.clicked.connect(self.prediction)
            self.ui.pushButton_2.clicked.connect(self.auto_params)
            self.ui.pushButton_5.clicked.connect(self.test_model)
            self.ui.pushButton_6.clicked.connect(self.open_eda)
            self.ui.textEdit.setText("1")
            self.ui.textEdit_2.setText("0")
            self.ui.textEdit_3.setText("1")
            self.ui.textEdit_4.setText("30")
            self.ui.textEdit_5.setText("1")
            self.ui.textEdit_6.setText("0")
            self.ui.textEdit_7.setText("1")
            self.ui.textEdit_8.setText("12")
            self.data = []
            self.result = []
            self.decomposition = []
            self.errors = []

            self.eda_window = EdaWindow()
            self.final_window = FinalWindow()
            self.result_window = ResultWindow()


        def loadCSV(self):
            file_path, _ = QFileDialog.getOpenFileName(self, "Оберіть CSV
файл", "", "CSV files (*.csv)")
            if file_path:
                self.ui.lineEdit.setText(file_path)
                self.data = main.format_data(main.load_data(file_path))
```

```
                self.plotData(self.data)
                self.eda_window.decompose_data(self.data)

        def open_eda(self):
            self.eda_window.show()

        def auto_params(self):
            # Display a warning message
            msgBox = QMessageBox()
            msgBox.setIcon(QMessageBox.Information)
            msgBox.setText("Підбір параметрів може зайняти кілька хвилин.")
            msgBox.setWindowTitle("Почекайте")
            msgBox.setStandardButtons(QMessageBox.Ok)
            msgBox.exec()  # This will block the GUI and wait for the user
to click OK


            S = int(self.ui.textEdit_8.toPlainText())
            order, seasonal_order = main.auto_params(self.data, S)
            self.ui.textEdit.setText(str(order[0]))
            self.ui.textEdit_2.setText(str(order[1]))
            self.ui.textEdit_3.setText(str(order[2]))
            self.ui.textEdit_5.setText(str(seasonal_order[0]))
            self.ui.textEdit_6.setText(str(seasonal_order[1]))
            self.ui.textEdit_7.setText(str(seasonal_order[2]))


        def plotData(self, data):
            # Assuming data columns 'x' and 'y'
            x = data.index
            y = data['target']

            # Create a figure and transfer it to QGraphicsView
            fig, ax = plt.subplots(figsize = (10,6))
```

```
            ax.xaxis.set_major_locator(mdates.YearLocator())     #   Locate
ticks every year
            ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y'))      #
Format ticks as 'year'
            ax.tick_params(axis='x', rotation=45)
            ax.grid()
            ax.plot(x, y, 'b-')  # Plotting with red line
            ax.set_title('Ряд')
            ax.set_xlabel('Дата')
            ax.set_ylabel('Дані')



            fig.autofmt_xdate()

            # Convert Figure to Qt Widget
            canvas = FigureCanvas(fig)
            scene = QGraphicsScene(self)
            scene.addWidget(canvas)
            self.ui.graphicsView.setScene(scene)
            self.ui.graphicsView.show()

        def prediction(self):
            order          =          (int(self.ui.textEdit.toPlainText()),
int(self.ui.textEdit_2.toPlainText()), int(self.ui.textEdit_3.toPlainText()))
            seasonal_order         =(int(self.ui.textEdit_5.toPlainText()),
int(self.ui.textEdit_6.toPlainText()), \
                                  int(self.ui.textEdit_7.toPlainText()),
int(self.ui.textEdit_8.toPlainText()))
            n_steps = int(self.ui.textEdit_4.toPlainText())
            result   =   main.predict(self.data,   order,   seasonal_order,
n_steps)
            self.result = result

            self.final_window.plot(self.data, self.result)
            self.final_window.show()
```

```
        def test_model(self):
            order          =          (int(self.ui.textEdit.toPlainText()),
int(self.ui.textEdit_2.toPlainText()), int(self.ui.textEdit_3.toPlainText()))

            seasonal_order         =(int(self.ui.textEdit_5.toPlainText()),
int(self.ui.textEdit_6.toPlainText()), \
                                  int(self.ui.textEdit_7.toPlainText()),
int(self.ui.textEdit_8.toPlainText()))

            n_steps = int(self.ui.textEdit_4.toPlainText())
            test_data = self.data[:-n_steps]
            orig_data = self.data[-n_steps:]

            test_result  =  main.predict(test_data,  order,  seasonal_order,
n_steps)
            mse         =         mean_squared_error(test_result.to_numpy(),
orig_data.to_numpy())
            mae        =        mean_absolute_error(test_result.to_numpy(),
orig_data.to_numpy())
            mape  =  mean_absolute_percentage_error(test_result.to_numpy(),
orig_data.to_numpy())
            r2 = r2_score(test_result.to_numpy(), orig_data.to_numpy())
            self.errors = [mse, mae, mape, r2]
            self.result_window.data_orig = orig_data
            self.result_window.test_result = test_result
            self.result_window.display_results(self.errors)
            self.result_window.show()


    class ResultWindow(QMainWindow):
        def __init__(self, parent=None):
            super(ResultWindow, self).__init__(parent)
            self.ui = ui_resultwindow.Ui_ResultWindow()
            self.ui.setupUi(self)
            self.data_orig = []
```

```
            self.test_result = []

        def display_results(self, errors):
            mse, mae, mape, r2 = errors
            mse = round(mse, 2)
            mae = round(mae, 2)
            mape = round(mape, 2)*100
            r2 = round(r2, 2)
            explanation = (
                "MSE (Середня квадратична помилка): Чим менше значення, тим краще модель передбачає дані.\n"
                f"MSE: {mse}\n\n"
                "MAE (Середня абсолютна помилка): Чим менше значення, тим краще модель передбачає дані. "
                "Вона показує середню різницю між передбаченими та фактичними значеннями.\n"
                f"MAE: {mae}\n\n"
                "MAPE (Середня абсолютна відносна помилка): Чим менше значення, тим краще модель передбачає дані. "
                "Це відсоткове значення, яке показує середню абсолютну помилку відносно фактичних значень.\n"
                f"MAPE: {mape}%\n\n"
                "R2 (Коефіцієнт детермінації): Значення близьке до 1 означає, що модель добре підходить для даних.\n"
                f"R2: {r2}"
            )

            self.ui.textBrowser.setText(explanation)
            # Plot the original data and test results
            fig, ax = plt.subplots()

            ax.plot(self.data_orig, label='Оригінальні дані', color='blue')
            ax.plot(self.test_result, label='Прогноз', color='red')

            ax.legend()
            ax.grid()
```

```
            ax.set_title('Оригінальні дані і Прогноз')
            ax.set_xlabel('Час')
            ax.set_ylabel('Значення')

            # Convert the plot to a canvas
            canvas = FigureCanvas(fig)
            scene = QGraphicsScene(self)
            scene.addWidget(canvas)

            # Set the scene to the QGraphicsView
            self.ui.graphicsView.setScene(scene)
            self.ui.graphicsView.show()

    class EdaWindow(QMainWindow):
        def __init__(self, parent=None):
            super(EdaWindow, self).__init__(parent)
            self.ui = ui_edawindow.Ui_MainWindow()
            self.ui.setupUi(self)
            self.data_info = ""

        def decompose_data(self, data):

            decomposition = seasonal_decompose(data, model='additive')
            self.ui.textBrowser.setText(main.analize_data(data,
decomposition))

            # Create a figure with adjusted size
            fig = Figure(figsize=(7, 5))
            canvas = FigureCanvas(fig)

            ax1 = fig.add_subplot(311)  # Trend
            ax2 = fig.add_subplot(312)  # Seasonality
            ax3 = fig.add_subplot(313)  # Residual Noise

            # Set major tick locator and formatter for each subplot
            for ax in [ax1, ax2, ax3]:
```

```
                ax.tick_params(axis='x', rotation=45)   # Optional: Rotate
labels for better fit

            ax1.plot(decomposition.trend, label='Тренд')
            ax1.legend()

            ax2.plot(decomposition.seasonal, label='Сезонність')
            ax2.legend()

            ax3.plot(decomposition.resid, label='Залишковий шум')
            ax3.legend()

            # Improve the spacing and appearance of the subplots
            fig.tight_layout()  # Adjust layout

            # Create a QGraphicsScene and add the FigureCanvas
            scene = QGraphicsScene()
            scene.addWidget(canvas)
            self.ui.graphicsView_2.setScene(scene)
            self.ui.graphicsView_2.show()


    class FinalWindow(QMainWindow):
        def __init__(self, parent=None):
            super(FinalWindow, self).__init__(parent)
            self.ui = ui_finalwindow.Ui_ResultWindow()
            self.ui.setupUi(self)
            self.ui.pushButton_4.clicked.connect(self.save)
            self.result = []

        def plot(self, data, result):
            # Створення фігури та осі
            self.result = result
            fig, ax = plt.subplots(figsize=(10,5))

            # Побудова графіку
```

```
            ax.plot(pd.DataFrame(data), label='Дані', color='b')
            ax.plot(pd.DataFrame(result), label='Прогноз', color='r')
            ax.tick_params(axis='x', rotation=45)
            ax.grid()
            ax.legend()

            # Конвертування фігури у FigureCanvas
            canvas = FigureCanvas(fig)
            scene = QGraphicsScene()
            scene.addWidget(canvas)
            self.ui.graphicsView.setScene(scene)
            self.ui.graphicsView.show()

        def save(self):
            options = QFileDialog.Options()
            fileName, _ = QFileDialog.getSaveFileName(self,
                                "Save File", "", "CSV Files (*.csv);;Excel
Files (*.xlsx)", options=options)
            if fileName:
                data = pd.DataFrame(self.result)
                if fileName.endswith('.csv'):
                    data.to_csv(fileName, index=False)
                elif fileName.endswith('.xlsx'):
                    data.to_excel(fileName, index=False)


    if __name__ == "__main__":
        app = QApplication([])
        window = StartWindow()
        window.show()
        sys.exit(app.exec())
```

Модуль ui_startwindow.py

```
# This Python file uses the following encoding: utf-8
import sys
```

```
    from PySide6.QtWidgets import QApplication, QMainWindow, QFileDialog,
QGraphicsScene, QMessageBox
    from PySide6.QtCore import Qt
    from PySide6.QtGui import QPen, QColor
    import pandas as pd
    import matplotlib.pyplot as plt
    from matplotlib.backends.backend_qtagg import FigureCanvasQTAgg as
FigureCanvas
    import matplotlib.dates as mdates
    from matplotlib.figure import Figure
    from statsmodels.tsa.seasonal import seasonal_decompose
    from sklearn.metrics import mean_squared_error,
mean_absolute_percentage_error, mean_absolute_error, r2_score
    import ui_resultwindow
    import ui_startwindow
    import ui_infowindow
    import ui_mainwindow
    import ui_finalwindow
    import ui_edawindow
    import main


    class StartWindow(QMainWindow):
        def __init__(self, parent=None):
            super(StartWindow, self).__init__(parent)
            self.ui = ui_startwindow.Ui_MainWindow()
            self.ui.setupUi(self)
            self.ui.pushButton.clicked.connect(self.start_program)
            self.ui.pushButton_2.clicked.connect(self.info_program)
            self.ui.pushButton_3.clicked.connect(self.exit_program)
            self.main_window = MainWindow()
            self.info_window = InfoWindow()

        def start_program(self):
            self.main_window.show()
            self.close()
```

```
    def info_program(self):
        self.info_window.show()

    def exit_program(self):
        sys.exit(app.exec())

class InfoWindow(QMainWindow):
    def __init__(self, parent=None):
        super(InfoWindow, self).__init__(parent)
        self.ui = ui_infowindow.Ui_MainWindow()
        self.ui.setupUi(self)

    def exit_window(self):
        self.close()

class MainWindow(QMainWindow):
    def __init__(self, parent=None):
        super(MainWindow, self).__init__()
        self.ui = ui_mainwindow.Ui_MainWindow()
        self.ui.setupUi(self)
        self.ui.pushButton.clicked.connect(self.loadCSV)
        self.ui.pushButton_3.clicked.connect(self.prediction)
        self.ui.pushButton_2.clicked.connect(self.auto_params)
        self.ui.pushButton_5.clicked.connect(self.test_model)
        self.ui.pushButton_6.clicked.connect(self.open_eda)
        self.ui.textEdit.setText("1")
        self.ui.textEdit_2.setText("0")
        self.ui.textEdit_3.setText("1")
        self.ui.textEdit_4.setText("30")
        self.ui.textEdit_5.setText("1")
        self.ui.textEdit_6.setText("0")
        self.ui.textEdit_7.setText("1")
        self.ui.textEdit_8.setText("12")
        self.data = []
        self.result = []
```

```
            self.decomposition = []
            self.errors = []

            self.eda_window = EdaWindow()
            self.final_window = FinalWindow()
            self.result_window = ResultWindow()


        def loadCSV(self):
            file_path, _ = QFileDialog.getOpenFileName(self, "Оберіть CSV
файл", "", "CSV files (*.csv)")
            if file_path:
                self.ui.lineEdit.setText(file_path)
                self.data = main.format_data(main.load_data(file_path))
                self.plotData(self.data)
                self.eda_window.decompose_data(self.data)

        def open_eda(self):
            self.eda_window.show()

        def auto_params(self):
            # Display a warning message
            msgBox = QMessageBox()
            msgBox.setIcon(QMessageBox.Information)
            msgBox.setText("Підбір параметрів може зайняти кілька хвилин.")
            msgBox.setWindowTitle("Почекайте")
            msgBox.setStandardButtons(QMessageBox.Ok)
            msgBox.exec()  # This will block the GUI and wait for the user
to click OK


            S = int(self.ui.textEdit_8.toPlainText())
            order, seasonal_order = main.auto_params(self.data, S)
            self.ui.textEdit.setText(str(order[0]))
            self.ui.textEdit_2.setText(str(order[1]))
            self.ui.textEdit_3.setText(str(order[2]))
```

```
            self.ui.textEdit_5.setText(str(seasonal_order[0]))
            self.ui.textEdit_6.setText(str(seasonal_order[1]))
            self.ui.textEdit_7.setText(str(seasonal_order[2]))


        def plotData(self, data):
            # Assuming data columns 'x' and 'y'
            x = data.index
            y = data['target']

            # Create a figure and transfer it to QGraphicsView
            fig, ax = plt.subplots(figsize = (10,6))


            ax.xaxis.set_major_locator(mdates.YearLocator())     #   Locate
ticks every year
            ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y'))      #
Format ticks as 'year'
            ax.tick_params(axis='x', rotation=45)
            ax.grid()
            ax.plot(x, y, 'b-')  # Plotting with red line
            ax.set_title('Ряд')
            ax.set_xlabel('Дата')
            ax.set_ylabel('Дані')


            fig.autofmt_xdate()

            # Convert Figure to Qt Widget
            canvas = FigureCanvas(fig)
            scene = QGraphicsScene(self)
            scene.addWidget(canvas)
            self.ui.graphicsView.setScene(scene)
            self.ui.graphicsView.show()

        def prediction(self):
```

```
            order          =          (int(self.ui.textEdit.toPlainText()),
int(self.ui.textEdit_2.toPlainText()), int(self.ui.textEdit_3.toPlainText()))
            seasonal_order         =(int(self.ui.textEdit_5.toPlainText()),
int(self.ui.textEdit_6.toPlainText()), \
                                  int(self.ui.textEdit_7.toPlainText()),
int(self.ui.textEdit_8.toPlainText()))
            n_steps = int(self.ui.textEdit_4.toPlainText())
            result   =   main.predict(self.data,   order,   seasonal_order,
n_steps)
            self.result = result

            self.final_window.plot(self.data, self.result)
            self.final_window.show()


        def test_model(self):
            order          =          (int(self.ui.textEdit.toPlainText()),
int(self.ui.textEdit_2.toPlainText()), int(self.ui.textEdit_3.toPlainText()))

            seasonal_order         =(int(self.ui.textEdit_5.toPlainText()),
int(self.ui.textEdit_6.toPlainText()), \
                                  int(self.ui.textEdit_7.toPlainText()),
int(self.ui.textEdit_8.toPlainText()))

            n_steps = int(self.ui.textEdit_4.toPlainText())
            test_data = self.data[:-n_steps]
            orig_data = self.data[-n_steps:]

            test_result  =  main.predict(test_data,  order,  seasonal_order,
n_steps)
            mse         =         mean_squared_error(test_result.to_numpy(),
orig_data.to_numpy())
            mae        =        mean_absolute_error(test_result.to_numpy(),
orig_data.to_numpy())
            mape  =  mean_absolute_percentage_error(test_result.to_numpy(),
orig_data.to_numpy())
```

```
            r2 = r2_score(test_result.to_numpy(), orig_data.to_numpy())
            self.errors = [mse, mae, mape, r2]
            self.result_window.data_orig = orig_data
            self.result_window.test_result = test_result
            self.result_window.display_results(self.errors)
            self.result_window.show()


    class ResultWindow(QMainWindow):
        def __init__(self, parent=None):
            super(ResultWindow, self).__init__(parent)
            self.ui = ui_resultwindow.Ui_ResultWindow()
            self.ui.setupUi(self)
            self.data_orig = []
            self.test_result = []

        def display_results(self, errors):
            mse, mae, mape, r2 = errors
            mse = round(mse, 2)
            mae = round(mae, 2)
            mape = round(mape, 2)*100
            r2 = round(r2, 2)
            explanation = (
                "MSE (Середня квадратична помилка): Чим менше значення, тим
краще модель передбачає дані.\n"
                f"MSE: {mse}\n\n"
                "MAE (Середня абсолютна помилка): Чим менше значення, тим
краще модель передбачає дані. "
                "Вона показує середню різницю між передбаченими та
фактичними значеннями.\n"
                f"MAE: {mae}\n\n"
                "MAPE (Середня абсолютна відносна помилка): Чим менше
значення, тим краще модель передбачає дані. "
                "Це відсоткове значення, яке показує середню абсолютну
помилку відносно фактичних значень.\n"
                f"MAPE: {mape}%\n\n"
```

```
                "R2 (Коефіцієнт детермінації): Значення близьке до 1 означає, що модель добре підходить для даних.\n"
                f"R2: {r2}"
            )

            self.ui.textBrowser.setText(explanation)
            # Plot the original data and test results
            fig, ax = plt.subplots()

            ax.plot(self.data_orig, label='Оригінальні дані', color='blue')
            ax.plot(self.test_result, label='Прогноз', color='red')

            ax.legend()
            ax.grid()
            ax.set_title('Оригінальні дані і Прогноз')
            ax.set_xlabel('Час')
            ax.set_ylabel('Значення')

            # Convert the plot to a canvas
            canvas = FigureCanvas(fig)
            scene = QGraphicsScene(self)
            scene.addWidget(canvas)

            # Set the scene to the QGraphicsView
            self.ui.graphicsView.setScene(scene)
            self.ui.graphicsView.show()

    class EdaWindow(QMainWindow):
        def __init__(self, parent=None):
            super(EdaWindow, self).__init__(parent)
            self.ui = ui_edawindow.Ui_MainWindow()
            self.ui.setupUi(self)
            self.data_info = ""

        def decompose_data(self, data):
```

```
            decomposition = seasonal_decompose(data, model='additive')
            self.ui.textBrowser.setText(main.analize_data(data,
decomposition))

            # Create a figure with adjusted size
            fig = Figure(figsize=(7, 5))
            canvas = FigureCanvas(fig)

            ax1 = fig.add_subplot(311)  # Trend
            ax2 = fig.add_subplot(312)  # Seasonality
            ax3 = fig.add_subplot(313)  # Residual Noise

            # Set major tick locator and formatter for each subplot
            for ax in [ax1, ax2, ax3]:
                ax.tick_params(axis='x', rotation=45)   # Optional: Rotate
labels for better fit

            ax1.plot(decomposition.trend, label='Тренд')
            ax1.legend()

            ax2.plot(decomposition.seasonal, label='Сезонність')
            ax2.legend()

            ax3.plot(decomposition.resid, label='Залишковий шум')
            ax3.legend()

            # Improve the spacing and appearance of the subplots
            fig.tight_layout()  # Adjust layout

            # Create a QGraphicsScene and add the FigureCanvas
            scene = QGraphicsScene()
            scene.addWidget(canvas)
            self.ui.graphicsView_2.setScene(scene)
            self.ui.graphicsView_2.show()
```

```
class FinalWindow(QMainWindow):
    def __init__(self, parent=None):
        super(FinalWindow, self).__init__(parent)
        self.ui = ui_finalwindow.Ui_ResultWindow()
        self.ui.setupUi(self)
        self.ui.pushButton_4.clicked.connect(self.save)
        self.result = []

    def plot(self, data, result):
        # Створення фігури та осі
        self.result = result
        fig, ax = plt.subplots(figsize=(10,5))

        # Побудова графіку
        ax.plot(pd.DataFrame(data), label='Дані', color='b')
        ax.plot(pd.DataFrame(result), label='Прогноз', color='r')
        ax.tick_params(axis='x', rotation=45)
        ax.grid()
        ax.legend()

        # Конвертування фігури у FigureCanvas
        canvas = FigureCanvas(fig)
        scene = QGraphicsScene()
        scene.addWidget(canvas)
        self.ui.graphicsView.setScene(scene)
        self.ui.graphicsView.show()

    def save(self):
        options = QFileDialog.Options()
        fileName, _ = QFileDialog.getSaveFileName(self,
                        "Save File", "", "CSV Files (*.csv);;Excel
Files (*.xlsx)", options=options)
        if fileName:
            data = pd.DataFrame(self.result)
            if fileName.endswith('.csv'):
                data.to_csv(fileName, index=False)
```

```
                elif fileName.endswith('.xlsx'):
                    data.to_excel(fileName, index=False)


    if __name__ == "__main__":
        app = QApplication([])
        window = StartWindow()
        window.show()
        sys.exit(app.exec())
```

Модуль ui_infowindow.py

```
    # -*- coding: utf-8 -*-

    ################################################################################
    ## Form generated from reading UI file 'infowindow.ui'
    ##
    ## Created by: Qt User Interface Compiler version 6.7.0
    ##
    ## WARNING! All changes made in this file will be lost when recompiling UI file!
    ################################################################################

    from PySide6.QtCore import (QCoreApplication, QDate, QDateTime, QLocale,
        QMetaObject, QObject, QPoint, QRect,
        QSize, QTime, QUrl, Qt)
    from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,
        QFont, QFontDatabase, QGradient, QIcon,
        QImage, QKeySequence, QLinearGradient, QPainter,
        QPalette, QPixmap, QRadialGradient, QTransform)
    from PySide6.QtWidgets import (QApplication, QLabel, QMainWindow, QMenuBar,
        QSizePolicy, QStatusBar, QWidget)
```

```
        class Ui_MainWindow(object):
            def setupUi(self, MainWindow):
                if not MainWindow.objectName():
                    MainWindow.setObjectName(u"MainWindow")
                MainWindow.setEnabled(True)
                MainWindow.resize(557, 240)
                palette = QPalette()
                brush = QBrush(QColor(0, 0, 0, 255))
                brush.setStyle(Qt.SolidPattern)
                palette.setBrush(QPalette.Active, QPalette.WindowText, brush)
                brush1 = QBrush(QColor(255, 217, 191, 255))
                brush1.setStyle(Qt.SolidPattern)
                palette.setBrush(QPalette.Active, QPalette.Button, brush1)
                brush2 = QBrush(QColor(255, 255, 255, 255))
                brush2.setStyle(Qt.SolidPattern)
                palette.setBrush(QPalette.Active, QPalette.Light, brush2)
                palette.setBrush(QPalette.Active, QPalette.Midlight, brush2)
                brush3 = QBrush(QColor(127, 127, 127, 255))
                brush3.setStyle(Qt.SolidPattern)
                palette.setBrush(QPalette.Active, QPalette.Dark, brush3)
                brush4 = QBrush(QColor(170, 170, 170, 255))
                brush4.setStyle(Qt.SolidPattern)
                palette.setBrush(QPalette.Active, QPalette.Mid, brush4)
                palette.setBrush(QPalette.Active, QPalette.Text, brush)
                palette.setBrush(QPalette.Active, QPalette.BrightText, brush2)
                palette.setBrush(QPalette.Active, QPalette.ButtonText, brush)
                palette.setBrush(QPalette.Active, QPalette.Base, brush1)
                palette.setBrush(QPalette.Active, QPalette.Window, brush1)
                palette.setBrush(QPalette.Active, QPalette.Shadow, brush)
                palette.setBrush(QPalette.Active,       QPalette.AlternateBase,
brush2)
                brush5 = QBrush(QColor(255, 255, 220, 255))
                brush5.setStyle(Qt.SolidPattern)
                palette.setBrush(QPalette.Active, QPalette.ToolTipBase, brush5)
                palette.setBrush(QPalette.Active, QPalette.ToolTipText, brush)
```

```
            brush6 = QBrush(QColor(0, 0, 0, 127))
            brush6.setStyle(Qt.SolidPattern)
      #if QT_VERSION >= QT_VERSION_CHECK(5, 12, 0)
            palette.setBrush(QPalette.Active,     QPalette.PlaceholderText,
brush6)
      #endif
            palette.setBrush(QPalette.Active, QPalette.Accent, brush2)
            palette.setBrush(QPalette.Inactive, QPalette.WindowText, brush)
            palette.setBrush(QPalette.Inactive, QPalette.Button, brush1)
            palette.setBrush(QPalette.Inactive, QPalette.Light, brush2)
            palette.setBrush(QPalette.Inactive, QPalette.Midlight, brush2)
            palette.setBrush(QPalette.Inactive, QPalette.Dark, brush3)
            palette.setBrush(QPalette.Inactive, QPalette.Mid, brush4)
            palette.setBrush(QPalette.Inactive, QPalette.Text, brush)
            palette.setBrush(QPalette.Inactive,        QPalette.BrightText,
brush2)
            palette.setBrush(QPalette.Inactive, QPalette.ButtonText, brush)
            palette.setBrush(QPalette.Inactive, QPalette.Base, brush1)
            palette.setBrush(QPalette.Inactive, QPalette.Window, brush1)
            palette.setBrush(QPalette.Inactive, QPalette.Shadow, brush)
            palette.setBrush(QPalette.Inactive,     QPalette.AlternateBase,
brush2)
            palette.setBrush(QPalette.Inactive,       QPalette.ToolTipBase,
brush5)
            palette.setBrush(QPalette.Inactive,       QPalette.ToolTipText,
brush)
      #if QT_VERSION >= QT_VERSION_CHECK(5, 12, 0)
            palette.setBrush(QPalette.Inactive,   QPalette.PlaceholderText,
brush6)
      #endif
            palette.setBrush(QPalette.Inactive, QPalette.Accent, brush2)
            palette.setBrush(QPalette.Disabled,        QPalette.WindowText,
brush3)
            palette.setBrush(QPalette.Disabled, QPalette.Button, brush1)
            palette.setBrush(QPalette.Disabled, QPalette.Light, brush2)
            palette.setBrush(QPalette.Disabled, QPalette.Midlight, brush2)
```

```
            palette.setBrush(QPalette.Disabled, QPalette.Dark, brush3)
            palette.setBrush(QPalette.Disabled, QPalette.Mid, brush4)
            palette.setBrush(QPalette.Disabled, QPalette.Text, brush3)
            palette.setBrush(QPalette.Disabled,        QPalette.BrightText,
brush2)
            palette.setBrush(QPalette.Disabled,        QPalette.ButtonText,
brush3)
            palette.setBrush(QPalette.Disabled, QPalette.Base, brush1)
            palette.setBrush(QPalette.Disabled, QPalette.Window, brush1)
            palette.setBrush(QPalette.Disabled, QPalette.Shadow, brush)
            palette.setBrush(QPalette.Disabled,     QPalette.AlternateBase,
brush2)
            palette.setBrush(QPalette.Disabled,       QPalette.ToolTipBase,
brush5)
            palette.setBrush(QPalette.Disabled,       QPalette.ToolTipText,
brush)
            brush7 = QBrush(QColor(127, 127, 127, 127))
            brush7.setStyle(Qt.SolidPattern)
      #if QT_VERSION >= QT_VERSION_CHECK(5, 12, 0)
            palette.setBrush(QPalette.Disabled,   QPalette.PlaceholderText,
brush7)
      #endif
            palette.setBrush(QPalette.Disabled, QPalette.Accent, brush2)
            MainWindow.setPalette(palette)
            MainWindow.setAutoFillBackground(False)
            MainWindow.setStyleSheet(u"background:#ffd9bf")
            self.centralwidget = QWidget(MainWindow)
            self.centralwidget.setObjectName(u"centralwidget")
            self.centralwidget.setStyleSheet(u"background:#ffd9bf")
            self.label = QLabel(self.centralwidget)
            self.label.setObjectName(u"label")
            self.label.setGeometry(QRect(190, 50, 191, 20))
            self.label.setStyleSheet(u"font: 500 12pt \"Roboto Medium\";\n"
      "color: black;")
            self.label_2 = QLabel(self.centralwidget)
            self.label_2.setObjectName(u"label_2")
```

```
            self.label_2.setGeometry(QRect(120, 80, 331, 20))
            self.label_2.setStyleSheet(u"font:     500     12pt     \"Roboto
Medium\";\n"
      "color: black;")
            self.label_2.setAlignment(Qt.AlignCenter)
            self.label_4 = QLabel(self.centralwidget)
            self.label_4.setObjectName(u"label_4")
            self.label_4.setGeometry(QRect(80, 110, 411, 20))
            self.label_4.setStyleSheet(u"font:     500     12pt     \"Roboto
Medium\";\n"
      "color: black;")
            MainWindow.setCentralWidget(self.centralwidget)
            self.menubar = QMenuBar(MainWindow)
            self.menubar.setObjectName(u"menubar")
            self.menubar.setGeometry(QRect(0, 0, 557, 21))
            MainWindow.setMenuBar(self.menubar)
            self.statusbar = QStatusBar(MainWindow)
            self.statusbar.setObjectName(u"statusbar")
            MainWindow.setStatusBar(self.statusbar)

            self.retranslateUi(MainWindow)

            QMetaObject.connectSlotsByName(MainWindow)
        # setupUi

        def retranslateUi(self, MainWindow):

MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow",  u"Osadcha
Sofiia", None))
            self.label.setText(QCoreApplication.translate("MainWindow",
u"\u0414\u0438\u043f\u043b\u043e\u043c\u043d\u0438\u0439
\u043f\u0440\u043e\u0454\u043a\u0442 2024", None))
            self.label_2.setText(QCoreApplication.translate("MainWindow",
u"\u041e\u0441\u0430\u0434\u0447\u0430         \u0421\u043e\u0444\u0456\u044f
\u041e\u043b\u0435\u043a\u0441\u0430\u043d\u0434\u0440\u0456\u0432\u043d\u043
0, \u0433\u0440\u0443\u043f\u0430 \u041a\u0410-04", None))
```

```
            self.label_4.setText(QCoreApplication.translate("MainWindow",
u"\u041f\u0440\u043e\u0433\u043d\u043e\u0437\u0443\u0432\u0430\u043d\u043d\u0
44f                    \u0441\u0435\u0437\u043e\u043d\u043d\u043e\u0433\u043e
\u043f\u043e\u043f\u0438\u0442\u0443
\u0441\u043a\u043b\u0430\u0434\u0441\u044c\u043a\u0438\u0445
\u0437\u0430\u043f\u0430\u0441\u0456\u0432", None))
        # retranslateUi
```

Модуль ui_mainwindow.py

```
    # -*- coding: utf-8 -*-

    ################################################################################
    ## Form generated from reading UI file 'mainwindow.ui'
    ##
    ## Created by: Qt User Interface Compiler version 6.7.0
    ##
    ## WARNING! All changes made in this file will be lost when recompiling
UI file!
    ################################################################################

    from PySide6.QtCore import (QCoreApplication, QDate, QDateTime,
QLocale,
        QMetaObject, QObject, QPoint, QRect,
        QSize, QTime, QUrl, Qt)
    from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,
        QFont, QFontDatabase, QGradient, QIcon,
        QImage, QKeySequence, QLinearGradient, QPainter,
        QPalette, QPixmap, QRadialGradient, QTransform)
    from PySide6.QtWidgets import (QApplication, QGraphicsView, QLabel,
QLineEdit,
        QMainWindow, QMenuBar, QPushButton, QSizePolicy,
        QStatusBar, QTextEdit, QWidget)
```

```
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        if not MainWindow.objectName():
            MainWindow.setObjectName(u"MainWindow")
        MainWindow.setEnabled(True)
        MainWindow.resize(1095, 826)
        palette = QPalette()
        brush = QBrush(QColor(0, 0, 0, 255))
        brush.setStyle(Qt.SolidPattern)
        palette.setBrush(QPalette.Active, QPalette.WindowText, brush)
        brush1 = QBrush(QColor(255, 217, 191, 255))
        brush1.setStyle(Qt.SolidPattern)
        palette.setBrush(QPalette.Active, QPalette.Button, brush1)
        brush2 = QBrush(QColor(255, 255, 255, 255))
        brush2.setStyle(Qt.SolidPattern)
        palette.setBrush(QPalette.Active, QPalette.Light, brush2)
        palette.setBrush(QPalette.Active, QPalette.Midlight, brush2)
        brush3 = QBrush(QColor(127, 127, 127, 255))
        brush3.setStyle(Qt.SolidPattern)
        palette.setBrush(QPalette.Active, QPalette.Dark, brush3)
        brush4 = QBrush(QColor(170, 170, 170, 255))
        brush4.setStyle(Qt.SolidPattern)
        palette.setBrush(QPalette.Active, QPalette.Mid, brush4)
        palette.setBrush(QPalette.Active, QPalette.Text, brush)
        palette.setBrush(QPalette.Active, QPalette.BrightText, brush2)
        palette.setBrush(QPalette.Active, QPalette.ButtonText, brush)
        palette.setBrush(QPalette.Active, QPalette.Base, brush1)
        palette.setBrush(QPalette.Active, QPalette.Window, brush1)
        palette.setBrush(QPalette.Active, QPalette.Shadow, brush)
        palette.setBrush(QPalette.Active,       QPalette.AlternateBase,
brush2)
        brush5 = QBrush(QColor(255, 255, 220, 255))
        brush5.setStyle(Qt.SolidPattern)
        palette.setBrush(QPalette.Active, QPalette.ToolTipBase, brush5)
        palette.setBrush(QPalette.Active, QPalette.ToolTipText, brush)
        brush6 = QBrush(QColor(0, 0, 0, 127))
```

```
            brush6.setStyle(Qt.SolidPattern)
      #if QT_VERSION >= QT_VERSION_CHECK(5, 12, 0)
            palette.setBrush(QPalette.Active,     QPalette.PlaceholderText,
brush6)
      #endif
            palette.setBrush(QPalette.Active, QPalette.Accent, brush2)
            palette.setBrush(QPalette.Inactive, QPalette.WindowText, brush)
            palette.setBrush(QPalette.Inactive, QPalette.Button, brush1)
            palette.setBrush(QPalette.Inactive, QPalette.Light, brush2)
            palette.setBrush(QPalette.Inactive, QPalette.Midlight, brush2)
            palette.setBrush(QPalette.Inactive, QPalette.Dark, brush3)
            palette.setBrush(QPalette.Inactive, QPalette.Mid, brush4)
            palette.setBrush(QPalette.Inactive, QPalette.Text, brush)
            palette.setBrush(QPalette.Inactive,        QPalette.BrightText,
brush2)
            palette.setBrush(QPalette.Inactive, QPalette.ButtonText, brush)
            palette.setBrush(QPalette.Inactive, QPalette.Base, brush1)
            palette.setBrush(QPalette.Inactive, QPalette.Window, brush1)
            palette.setBrush(QPalette.Inactive, QPalette.Shadow, brush)
            palette.setBrush(QPalette.Inactive,     QPalette.AlternateBase,
brush2)
            palette.setBrush(QPalette.Inactive,       QPalette.ToolTipBase,
brush5)
            palette.setBrush(QPalette.Inactive,       QPalette.ToolTipText,
brush)
      #if QT_VERSION >= QT_VERSION_CHECK(5, 12, 0)
            palette.setBrush(QPalette.Inactive,   QPalette.PlaceholderText,
brush6)
      #endif
            palette.setBrush(QPalette.Inactive, QPalette.Accent, brush2)
            palette.setBrush(QPalette.Disabled,        QPalette.WindowText,
brush3)
            palette.setBrush(QPalette.Disabled, QPalette.Button, brush1)
            palette.setBrush(QPalette.Disabled, QPalette.Light, brush2)
            palette.setBrush(QPalette.Disabled, QPalette.Midlight, brush2)
            palette.setBrush(QPalette.Disabled, QPalette.Dark, brush3)
```

```
            palette.setBrush(QPalette.Disabled, QPalette.Mid, brush4)
            palette.setBrush(QPalette.Disabled, QPalette.Text, brush3)
            palette.setBrush(QPalette.Disabled,        QPalette.BrightText,
brush2)
            palette.setBrush(QPalette.Disabled,        QPalette.ButtonText,
brush3)
            palette.setBrush(QPalette.Disabled, QPalette.Base, brush1)
            palette.setBrush(QPalette.Disabled, QPalette.Window, brush1)
            palette.setBrush(QPalette.Disabled, QPalette.Shadow, brush)
            palette.setBrush(QPalette.Disabled,     QPalette.AlternateBase,
brush2)
            palette.setBrush(QPalette.Disabled,       QPalette.ToolTipBase,
brush5)
            palette.setBrush(QPalette.Disabled,       QPalette.ToolTipText,
brush)
            brush7 = QBrush(QColor(127, 127, 127, 127))
            brush7.setStyle(Qt.SolidPattern)
      #if QT_VERSION >= QT_VERSION_CHECK(5, 12, 0)
            palette.setBrush(QPalette.Disabled,   QPalette.PlaceholderText,
brush7)
      #endif
            palette.setBrush(QPalette.Disabled, QPalette.Accent, brush2)
            MainWindow.setPalette(palette)
            MainWindow.setAutoFillBackground(False)
            MainWindow.setStyleSheet(u"background:#ffd9bf")
            self.centralwidget = QWidget(MainWindow)
            self.centralwidget.setObjectName(u"centralwidget")
            self.centralwidget.setStyleSheet(u"background:#ffd9bf")
            self.lineEdit = QLineEdit(self.centralwidget)
            self.lineEdit.setObjectName(u"lineEdit")
            self.lineEdit.setGeometry(QRect(60, 70, 401, 31))
            self.lineEdit.setAutoFillBackground(False)
            self.lineEdit.setStyleSheet(u"\n"
      "    border: 1px solid black;\n"
      "    background-color: white;\n"
      "     color: black;\n"
```

```
"     font: 500 12pt \"Roboto Medium\";\n"
"\n"
"")
        self.lineEdit.setReadOnly(True)
        self.pushButton = QPushButton(self.centralwidget)
        self.pushButton.setObjectName(u"pushButton")
        self.pushButton.setGeometry(QRect(480, 70, 121, 31))
        self.pushButton.setAutoFillBackground(False)
        self.pushButton.setStyleSheet(u"background-color: #FFAA7F;\n"
"border: 1px solid black;\n"
"color: black;\n"
"font: 500 12pt \"Roboto Medium\";\n"
"border-radius: 15px;\n"
"\n"
"\n"
"")
        self.label = QLabel(self.centralwidget)
        self.label.setObjectName(u"label")
        self.label.setGeometry(QRect(700, 30, 91, 16))
        self.label.setStyleSheet(u"font: 500 12pt \"Roboto Medium\";\n"
"color: black;")
        self.textEdit = QTextEdit(self.centralwidget)
        self.textEdit.setObjectName(u"textEdit")
        self.textEdit.setGeometry(QRect(660, 70, 41, 31))
        self.textEdit.setStyleSheet(u"\n"
"    border: 1px solid black;\n"
"     background-color: white;\n"
"     color: black;\n"
"     font: 500 12pt \"Roboto Medium\";\n"
"")
        self.textEdit_2 = QTextEdit(self.centralwidget)
        self.textEdit_2.setObjectName(u"textEdit_2")
        self.textEdit_2.setGeometry(QRect(710, 70, 41, 31))
        self.textEdit_2.setStyleSheet(u"\n"
"    border: 1px solid black;\n"
"     background-color: white;\n"
```

```
"     color: black;\n"
"     font: 500 12pt \"Roboto Medium\";\n"
"")
        self.textEdit_3 = QTextEdit(self.centralwidget)
        self.textEdit_3.setObjectName(u"textEdit_3")
        self.textEdit_3.setGeometry(QRect(760, 70, 41, 31))
        self.textEdit_3.setStyleSheet(u"\n"
"    border: 1px solid black;\n"
"     background-color: white;\n"
"     color: black;\n"
"     font: 500 12pt \"Roboto Medium\";\n"
"")
        self.pushButton_2 = QPushButton(self.centralwidget)
        self.pushButton_2.setObjectName(u"pushButton_2")
        self.pushButton_2.setGeometry(QRect(680, 110, 141, 31))
        self.pushButton_2.setAutoFillBackground(False)
        self.pushButton_2.setStyleSheet(u"background-color: #FFAA7F;\n"
"border: 1px solid black;\n"
"color: black;\n"
"font: 500 12pt \"Roboto Medium\";\n"
"border-radius: 15px;\n"
"\n"
"\n"
"")
        self.textEdit_4 = QTextEdit(self.centralwidget)
        self.textEdit_4.setObjectName(u"textEdit_4")
        self.textEdit_4.setGeometry(QRect(520, 730, 51, 31))
        self.textEdit_4.setTabletTracking(False)
        self.textEdit_4.setStyleSheet(u"\n"
"    border: 1px solid black;\n"
"     background-color: white;\n"
"     color: black;\n"
"     font: 500 12pt \"Roboto Medium\";\n"
"")
        self.label_2 = QLabel(self.centralwidget)
        self.label_2.setObjectName(u"label_2")
```

```
            self.label_2.setGeometry(QRect(420, 730, 91, 31))
            self.label_2.setStyleSheet(u"font:     500     12pt     \"Roboto
Medium\";\n"
      "color: black;")
            self.graphicsView = QGraphicsView(self.centralwidget)
            self.graphicsView.setObjectName(u"graphicsView")
            self.graphicsView.setGeometry(QRect(60, 160, 971, 561))
            self.graphicsView.setStyleSheet(u"QGraphicsView {\n"
      "    border: 1px solid black;\n"
      "     background-color: white;\n"
      "}\n"
      "")
            self.pushButton_3 = QPushButton(self.centralwidget)
            self.pushButton_3.setObjectName(u"pushButton_3")
            self.pushButton_3.setGeometry(QRect(590, 730, 121, 31))
            self.pushButton_3.setAutoFillBackground(False)
            self.pushButton_3.setStyleSheet(u"background-color: #FFAA7F;\n"
      "border: 1px solid black;\n"
      "color: black;\n"
      "font: 500 12pt \"Roboto Medium\";\n"
      "border-radius: 15px;\n"
      "\n"
      "\n"
      "")
            self.label_5 = QLabel(self.centralwidget)
            self.label_5.setObjectName(u"label_5")
            self.label_5.setGeometry(QRect(850, 30, 151, 20))
            self.label_5.setStyleSheet(u"font:     500     12pt     \"Roboto
Medium\";\n"
      "color: black;")
            self.label_5.setAlignment(Qt.AlignCenter)
            self.textEdit_5 = QTextEdit(self.centralwidget)
            self.textEdit_5.setObjectName(u"textEdit_5")
            self.textEdit_5.setGeometry(QRect(830, 70, 41, 31))
            self.textEdit_5.setStyleSheet(u"\n"
      "    border: 1px solid black;\n"
```

```
"     background-color: white;\n"
"     color: black;\n"
"     font: 500 12pt \"Roboto Medium\";\n"
"")
        self.textEdit_6 = QTextEdit(self.centralwidget)
        self.textEdit_6.setObjectName(u"textEdit_6")
        self.textEdit_6.setGeometry(QRect(880, 70, 41, 31))
        self.textEdit_6.setStyleSheet(u"\n"
"    border: 1px solid black;\n"
"     background-color: white;\n"
"     color: black;\n"
"     font: 500 12pt \"Roboto Medium\";\n"
"")
        self.textEdit_7 = QTextEdit(self.centralwidget)
        self.textEdit_7.setObjectName(u"textEdit_7")
        self.textEdit_7.setGeometry(QRect(930, 70, 41, 31))
        self.textEdit_7.setStyleSheet(u"\n"
"    border: 1px solid black;\n"
"     background-color: white;\n"
"     color: black;\n"
"     font: 500 12pt \"Roboto Medium\";\n"
"")
        self.textEdit_8 = QTextEdit(self.centralwidget)
        self.textEdit_8.setObjectName(u"textEdit_8")
        self.textEdit_8.setGeometry(QRect(980, 70, 41, 31))
        self.textEdit_8.setStyleSheet(u"\n"
"    border: 1px solid black;\n"
"     background-color: white;\n"
"     color: black;\n"
"     font: 500 12pt \"Roboto Medium\";\n"
"")
        self.label_6 = QLabel(self.centralwidget)
        self.label_6.setObjectName(u"label_6")
        self.label_6.setGeometry(QRect(670, 50, 21, 16))
        self.label_6.setLayoutDirection(Qt.LeftToRight)
```

```
            self.label_6.setStyleSheet(u"font:     500     12pt     \"Roboto
Medium\";\n"
      "color: black;")
            self.label_6.setAlignment(Qt.AlignCenter)
            self.label_7 = QLabel(self.centralwidget)
            self.label_7.setObjectName(u"label_7")
            self.label_7.setGeometry(QRect(720, 50, 21, 16))
            self.label_7.setLayoutDirection(Qt.LeftToRight)
            self.label_7.setStyleSheet(u"font:     500     12pt     \"Roboto
Medium\";\n"
      "color: black;")
            self.label_7.setAlignment(Qt.AlignCenter)
            self.label_9 = QLabel(self.centralwidget)
            self.label_9.setObjectName(u"label_9")
            self.label_9.setGeometry(QRect(770, 50, 21, 16))
            self.label_9.setLayoutDirection(Qt.LeftToRight)
            self.label_9.setStyleSheet(u"font:     500     12pt     \"Roboto
Medium\";\n"
      "color: black;")
            self.label_9.setAlignment(Qt.AlignCenter)
            self.label_10 = QLabel(self.centralwidget)
            self.label_10.setObjectName(u"label_10")
            self.label_10.setGeometry(QRect(840, 50, 21, 16))
            self.label_10.setLayoutDirection(Qt.LeftToRight)
            self.label_10.setStyleSheet(u"font:     500     12pt     \"Roboto
Medium\";\n"
      "color: black;")
            self.label_10.setAlignment(Qt.AlignCenter)
            self.label_11 = QLabel(self.centralwidget)
            self.label_11.setObjectName(u"label_11")
            self.label_11.setGeometry(QRect(890, 50, 21, 16))
            self.label_11.setLayoutDirection(Qt.LeftToRight)
            self.label_11.setStyleSheet(u"font:     500     12pt     \"Roboto
Medium\";\n"
      "color: black;")
            self.label_11.setAlignment(Qt.AlignCenter)
```

```
                self.label_12 = QLabel(self.centralwidget)
                self.label_12.setObjectName(u"label_12")
                self.label_12.setGeometry(QRect(940, 50, 21, 16))
                self.label_12.setLayoutDirection(Qt.LeftToRight)
                self.label_12.setStyleSheet(u"font:    500    12pt    \"Roboto
Medium\";\n"
      "color: black;")
                self.label_12.setAlignment(Qt.AlignCenter)
                self.label_13 = QLabel(self.centralwidget)
                self.label_13.setObjectName(u"label_13")
                self.label_13.setGeometry(QRect(990, 50, 21, 16))
                self.label_13.setLayoutDirection(Qt.LeftToRight)
                self.label_13.setStyleSheet(u"font:    500    12pt    \"Roboto
Medium\";\n"
      "color: black;")
                self.label_13.setAlignment(Qt.AlignCenter)
                self.pushButton_5 = QPushButton(self.centralwidget)
                self.pushButton_5.setObjectName(u"pushButton_5")
                self.pushButton_5.setGeometry(QRect(860, 110, 141, 31))
                self.pushButton_5.setAutoFillBackground(False)
                self.pushButton_5.setStyleSheet(u"background-color: #FFAA7F;\n"
      "border: 1px solid black;\n"
      "color: black;\n"
      "font: 500 12pt \"Roboto Medium\";\n"
      "border-radius: 15px;\n"
      "\n"
      "\n"
      "")
                self.pushButton_6 = QPushButton(self.centralwidget)
                self.pushButton_6.setObjectName(u"pushButton_6")
                self.pushButton_6.setGeometry(QRect(190, 110, 171, 31))
                self.pushButton_6.setAutoFillBackground(False)
                self.pushButton_6.setStyleSheet(u"background-color: #FFAA7F;\n"
      "border: 1px solid black;\n"
      "color: black;\n"
      "font: 500 12pt \"Roboto Medium\";\n"
```

```
        "border-radius: 15px;\n"
        "\n"
        "\n"
        "")
                MainWindow.setCentralWidget(self.centralwidget)
                self.menubar = QMenuBar(MainWindow)
                self.menubar.setObjectName(u"menubar")
                self.menubar.setGeometry(QRect(0, 0, 1095, 21))
                MainWindow.setMenuBar(self.menubar)
                self.statusbar = QStatusBar(MainWindow)
                self.statusbar.setObjectName(u"statusbar")
                MainWindow.setStatusBar(self.statusbar)

                self.retranslateUi(MainWindow)

                QMetaObject.connectSlotsByName(MainWindow)
            # setupUi

            def retranslateUi(self, MainWindow):

MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow",  u"Osadcha
Sofiia", None))

self.pushButton.setText(QCoreApplication.translate("MainWindow",
u"\u041e\u0431\u0440\u0430\u0442\u0438 \u0444\u0430\u0439\u043b", None))
                self.label.setText(QCoreApplication.translate("MainWindow",
u"\u041f\u0430\u0440\u0430\u043c\u0435\u0442\u0440\u0438", None))

self.pushButton_2.setText(QCoreApplication.translate("MainWindow",
u"\u041f\u0456\u0434\u0431\u0456\u0440", None))
                self.label_2.setText(QCoreApplication.translate("MainWindow",
u"\u041a-\u0442\u044c \u043a\u0440\u043e\u043a\u0456\u0432", None))

self.pushButton_3.setText(QCoreApplication.translate("MainWindow",
u"\u041f\u0440\u043e\u0433\u043d\u043e\u0437\u0443\u0432\u0430\u043d\u043d\u0
44f", None))
```

```
            self.label_5.setText(QCoreApplication.translate("MainWindow",
u"\u0421\u0435\u0437\u043e\u043d\u043d\u0456
\u043f\u0430\u0440\u0430\u043c\u0435\u0442\u0440\u0438", None))
            self.label_6.setText(QCoreApplication.translate("MainWindow",
u"p", None))
            self.label_7.setText(QCoreApplication.translate("MainWindow",
u"d", None))
            self.label_9.setText(QCoreApplication.translate("MainWindow",
u"q", None))
            self.label_10.setText(QCoreApplication.translate("MainWindow",
u"P", None))
            self.label_11.setText(QCoreApplication.translate("MainWindow",
u"D", None))
            self.label_12.setText(QCoreApplication.translate("MainWindow",
u"Q", None))
            self.label_13.setText(QCoreApplication.translate("MainWindow",
u"S", None))

self.pushButton_5.setText(QCoreApplication.translate("MainWindow",
u"\u0422\u0435\u0441\u0442
\u043f\u0430\u0440\u0430\u043c\u0435\u0442\u0440\u0456\u0432", None))

self.pushButton_6.setText(QCoreApplication.translate("MainWindow",
u"\u0410\u043d\u0430\u043b\u0456\u0437 \u043d\u0430\u0431\u043e\u0440\u0443",
None))
        # retranslateUi
```

Модуль ui_edawindow.py

```
# -*- coding: utf-8 -*-

################################################################################
## Form generated from reading UI file 'edawindow.ui'
##
## Created by: Qt User Interface Compiler version 6.7.0
```

```
      ##
      ## WARNING! All changes made in this file will be lost when recompiling
UI file!
      ################################################################################
#########

      from PySide6.QtCore import (QCoreApplication, QDate, QDateTime,
QLocale,
          QMetaObject, QObject, QPoint, QRect,
          QSize, QTime, QUrl, Qt)
      from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,
          QFont, QFontDatabase, QGradient, QIcon,
          QImage, QKeySequence, QLinearGradient, QPainter,
          QPalette, QPixmap, QRadialGradient, QTransform)
      from PySide6.QtWidgets import (QApplication, QGraphicsView, QLabel,
QMainWindow,
          QMenuBar, QSizePolicy, QStatusBar, QTextBrowser,
          QWidget)

      class Ui_MainWindow(object):
          def setupUi(self, MainWindow):
              if not MainWindow.objectName():
                  MainWindow.setObjectName(u"MainWindow")
              MainWindow.setEnabled(True)
              MainWindow.resize(1177, 757)
              palette = QPalette()
              brush = QBrush(QColor(0, 0, 0, 255))
              brush.setStyle(Qt.SolidPattern)
              palette.setBrush(QPalette.Active, QPalette.WindowText, brush)
              brush1 = QBrush(QColor(255, 217, 191, 255))
              brush1.setStyle(Qt.SolidPattern)
              palette.setBrush(QPalette.Active, QPalette.Button, brush1)
              brush2 = QBrush(QColor(255, 255, 255, 255))
              brush2.setStyle(Qt.SolidPattern)
              palette.setBrush(QPalette.Active, QPalette.Light, brush2)
              palette.setBrush(QPalette.Active, QPalette.Midlight, brush2)
```

```
            brush3 = QBrush(QColor(127, 127, 127, 255))
            brush3.setStyle(Qt.SolidPattern)
            palette.setBrush(QPalette.Active, QPalette.Dark, brush3)
            brush4 = QBrush(QColor(170, 170, 170, 255))
            brush4.setStyle(Qt.SolidPattern)
            palette.setBrush(QPalette.Active, QPalette.Mid, brush4)
            palette.setBrush(QPalette.Active, QPalette.Text, brush)
            palette.setBrush(QPalette.Active, QPalette.BrightText, brush2)
            palette.setBrush(QPalette.Active, QPalette.ButtonText, brush)
            palette.setBrush(QPalette.Active, QPalette.Base, brush1)
            palette.setBrush(QPalette.Active, QPalette.Window, brush1)
            palette.setBrush(QPalette.Active, QPalette.Shadow, brush)
            palette.setBrush(QPalette.Active,       QPalette.AlternateBase,
brush2)
            brush5 = QBrush(QColor(255, 255, 220, 255))
            brush5.setStyle(Qt.SolidPattern)
            palette.setBrush(QPalette.Active, QPalette.ToolTipBase, brush5)
            palette.setBrush(QPalette.Active, QPalette.ToolTipText, brush)
            brush6 = QBrush(QColor(0, 0, 0, 127))
            brush6.setStyle(Qt.SolidPattern)
      #if QT_VERSION >= QT_VERSION_CHECK(5, 12, 0)
            palette.setBrush(QPalette.Active,     QPalette.PlaceholderText,
brush6)
      #endif
            palette.setBrush(QPalette.Active, QPalette.Accent, brush2)
            palette.setBrush(QPalette.Inactive, QPalette.WindowText, brush)
            palette.setBrush(QPalette.Inactive, QPalette.Button, brush1)
            palette.setBrush(QPalette.Inactive, QPalette.Light, brush2)
            palette.setBrush(QPalette.Inactive, QPalette.Midlight, brush2)
            palette.setBrush(QPalette.Inactive, QPalette.Dark, brush3)
            palette.setBrush(QPalette.Inactive, QPalette.Mid, brush4)
            palette.setBrush(QPalette.Inactive, QPalette.Text, brush)
            palette.setBrush(QPalette.Inactive,        QPalette.BrightText,
brush2)
            palette.setBrush(QPalette.Inactive, QPalette.ButtonText, brush)
            palette.setBrush(QPalette.Inactive, QPalette.Base, brush1)
```

```
            palette.setBrush(QPalette.Inactive, QPalette.Window, brush1)
            palette.setBrush(QPalette.Inactive, QPalette.Shadow, brush)
            palette.setBrush(QPalette.Inactive,     QPalette.AlternateBase,
brush2)
            palette.setBrush(QPalette.Inactive,       QPalette.ToolTipBase,
brush5)
            palette.setBrush(QPalette.Inactive,       QPalette.ToolTipText,
brush)
      #if QT_VERSION >= QT_VERSION_CHECK(5, 12, 0)
            palette.setBrush(QPalette.Inactive,   QPalette.PlaceholderText,
brush6)
      #endif
            palette.setBrush(QPalette.Inactive, QPalette.Accent, brush2)
            palette.setBrush(QPalette.Disabled,        QPalette.WindowText,
brush3)
            palette.setBrush(QPalette.Disabled, QPalette.Button, brush1)
            palette.setBrush(QPalette.Disabled, QPalette.Light, brush2)
            palette.setBrush(QPalette.Disabled, QPalette.Midlight, brush2)
            palette.setBrush(QPalette.Disabled, QPalette.Dark, brush3)
            palette.setBrush(QPalette.Disabled, QPalette.Mid, brush4)
            palette.setBrush(QPalette.Disabled, QPalette.Text, brush3)
            palette.setBrush(QPalette.Disabled,        QPalette.BrightText,
brush2)
            palette.setBrush(QPalette.Disabled,        QPalette.ButtonText,
brush3)
            palette.setBrush(QPalette.Disabled, QPalette.Base, brush1)
            palette.setBrush(QPalette.Disabled, QPalette.Window, brush1)
            palette.setBrush(QPalette.Disabled, QPalette.Shadow, brush)
            palette.setBrush(QPalette.Disabled,     QPalette.AlternateBase,
brush2)
            palette.setBrush(QPalette.Disabled,       QPalette.ToolTipBase,
brush5)
            palette.setBrush(QPalette.Disabled,       QPalette.ToolTipText,
brush)
            brush7 = QBrush(QColor(127, 127, 127, 127))
            brush7.setStyle(Qt.SolidPattern)
```

```
      #if QT_VERSION >= QT_VERSION_CHECK(5, 12, 0)
              palette.setBrush(QPalette.Disabled,    QPalette.PlaceholderText,
brush7)
      #endif
              palette.setBrush(QPalette.Disabled, QPalette.Accent, brush2)
              MainWindow.setPalette(palette)
              MainWindow.setAutoFillBackground(False)
              MainWindow.setStyleSheet(u"background:#ffd9bf")
              self.centralwidget = QWidget(MainWindow)
              self.centralwidget.setObjectName(u"centralwidget")
              self.centralwidget.setStyleSheet(u"background:#ffd9bf")
              self.textBrowser = QTextBrowser(self.centralwidget)
              self.textBrowser.setObjectName(u"textBrowser")
              self.textBrowser.setGeometry(QRect(820, 120, 321, 241))
              self.textBrowser.setStyleSheet(u"QTextEdit {\n"
      "    border: 1px solid black;\n"
      "     background-color: white;\n"
      "     color: black;\n"
      "     font: 500 12pt \"Roboto Medium\";\n"
      "}\n"
      "")
              self.label_3 = QLabel(self.centralwidget)
              self.label_3.setObjectName(u"label_3")
              self.label_3.setGeometry(QRect(900, 90, 171, 16))
              self.label_3.setStyleSheet(u"font:     500     12pt     \"Roboto
Medium\";\n"
      "color: black;")
              self.label_4 = QLabel(self.centralwidget)
              self.label_4.setObjectName(u"label_4")
              self.label_4.setGeometry(QRect(330, 90, 171, 16))
              self.label_4.setStyleSheet(u"font:     500     12pt     \"Roboto
Medium\";\n"
      "color: black;")
              self.graphicsView_2 = QGraphicsView(self.centralwidget)
              self.graphicsView_2.setObjectName(u"graphicsView_2")
              self.graphicsView_2.setGeometry(QRect(40, 120, 751, 551))
```

```
            self.graphicsView_2.setStyleSheet(u"QGraphicsView {\n"
"    border: 1px solid black;\n"
"     background-color: white;\n"
"}\n"
"")
            self.label_5 = QLabel(self.centralwidget)
            self.label_5.setObjectName(u"label_5")
            self.label_5.setGeometry(QRect(460, 20, 271, 61))
            font = QFont()
            font.setFamilies([u"Roboto Medium"])
            font.setPointSize(20)
            font.setWeight(QFont.Medium)
            font.setItalic(False)
            self.label_5.setFont(font)
            self.label_5.setStyleSheet(u"font:     500     20pt     \"Roboto
Medium\";\n"
"color: black;")
            MainWindow.setCentralWidget(self.centralwidget)
            self.menubar = QMenuBar(MainWindow)
            self.menubar.setObjectName(u"menubar")
            self.menubar.setGeometry(QRect(0, 0, 1177, 21))
            MainWindow.setMenuBar(self.menubar)
            self.statusbar = QStatusBar(MainWindow)
            self.statusbar.setObjectName(u"statusbar")
            MainWindow.setStatusBar(self.statusbar)

            self.retranslateUi(MainWindow)

            QMetaObject.connectSlotsByName(MainWindow)
        # setupUi

        def retranslateUi(self, MainWindow):

MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow",  u"Osadcha
Sofiia", None))
```

```
            self.label_3.setText(QCoreApplication.translate("MainWindow",
u"\u0406\u043d\u0444\u043e\u0440\u043c\u0430\u0446\u0456\u044f
\u043f\u0440\u043e \u043d\u0430\u0431\u0456\u0440\u0440", None))
            self.label_4.setText(QCoreApplication.translate("MainWindow",
u"\u0414\u0435\u043a\u043e\u043c\u043f\u043e\u0437\u0438\u0446\u0456\u044f
\u043d\u0430\u0431\u043e\u0440\u0443", None))
            self.label_5.setText(QCoreApplication.translate("MainWindow",
u"\u0410\u043d\u0430\u043b\u0456\u0437    \u043d\u0430\u0431\u043e\u0440\u0443
\u0434\u0430\u043d\u0438\u0445", None))
        # retranslateUi
```

Модуль ui_finalwindow.py

```
    # -*- coding: utf-8 -*-

    ################################################################################
    ## Form generated from reading UI file 'edawindow.ui'
    ##
    ## Created by: Qt User Interface Compiler version 6.7.0
    ##
    ## WARNING! All changes made in this file will be lost when recompiling
UI file!
    ################################################################################

    from PySide6.QtCore import (QCoreApplication, QDate, QDateTime,
QLocale,
        QMetaObject, QObject, QPoint, QRect,
        QSize, QTime, QUrl, Qt)
    from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,
        QFont, QFontDatabase, QGradient, QIcon,
        QImage, QKeySequence, QLinearGradient, QPainter,
        QPalette, QPixmap, QRadialGradient, QTransform)
    from PySide6.QtWidgets import (QApplication, QGraphicsView, QLabel,
QMainWindow,
```

```
            QMenuBar, QSizePolicy, QStatusBar, QTextBrowser,
            QWidget)

        class Ui_MainWindow(object):
            def setupUi(self, MainWindow):
                if not MainWindow.objectName():
                    MainWindow.setObjectName(u"MainWindow")
                MainWindow.setEnabled(True)
                MainWindow.resize(1177, 757)
                palette = QPalette()
                brush = QBrush(QColor(0, 0, 0, 255))
                brush.setStyle(Qt.SolidPattern)
                palette.setBrush(QPalette.Active, QPalette.WindowText, brush)
                brush1 = QBrush(QColor(255, 217, 191, 255))
                brush1.setStyle(Qt.SolidPattern)
                palette.setBrush(QPalette.Active, QPalette.Button, brush1)
                brush2 = QBrush(QColor(255, 255, 255, 255))
                brush2.setStyle(Qt.SolidPattern)
                palette.setBrush(QPalette.Active, QPalette.Light, brush2)
                palette.setBrush(QPalette.Active, QPalette.Midlight, brush2)
                brush3 = QBrush(QColor(127, 127, 127, 255))
                brush3.setStyle(Qt.SolidPattern)
                palette.setBrush(QPalette.Active, QPalette.Dark, brush3)
                brush4 = QBrush(QColor(170, 170, 170, 255))
                brush4.setStyle(Qt.SolidPattern)
                palette.setBrush(QPalette.Active, QPalette.Mid, brush4)
                palette.setBrush(QPalette.Active, QPalette.Text, brush)
                palette.setBrush(QPalette.Active, QPalette.BrightText, brush2)
                palette.setBrush(QPalette.Active, QPalette.ButtonText, brush)
                palette.setBrush(QPalette.Active, QPalette.Base, brush1)
                palette.setBrush(QPalette.Active, QPalette.Window, brush1)
                palette.setBrush(QPalette.Active, QPalette.Shadow, brush)
                palette.setBrush(QPalette.Active,       QPalette.AlternateBase,
brush2)
                brush5 = QBrush(QColor(255, 255, 220, 255))
                brush5.setStyle(Qt.SolidPattern)
```

```
            palette.setBrush(QPalette.Active, QPalette.ToolTipBase, brush5)
            palette.setBrush(QPalette.Active, QPalette.ToolTipText, brush)
            brush6 = QBrush(QColor(0, 0, 0, 127))
            brush6.setStyle(Qt.SolidPattern)
      #if QT_VERSION >= QT_VERSION_CHECK(5, 12, 0)
            palette.setBrush(QPalette.Active,     QPalette.PlaceholderText,
brush6)
      #endif
            palette.setBrush(QPalette.Active, QPalette.Accent, brush2)
            palette.setBrush(QPalette.Inactive, QPalette.WindowText, brush)
            palette.setBrush(QPalette.Inactive, QPalette.Button, brush1)
            palette.setBrush(QPalette.Inactive, QPalette.Light, brush2)
            palette.setBrush(QPalette.Inactive, QPalette.Midlight, brush2)
            palette.setBrush(QPalette.Inactive, QPalette.Dark, brush3)
            palette.setBrush(QPalette.Inactive, QPalette.Mid, brush4)
            palette.setBrush(QPalette.Inactive, QPalette.Text, brush)
            palette.setBrush(QPalette.Inactive,        QPalette.BrightText,
brush2)
            palette.setBrush(QPalette.Inactive, QPalette.ButtonText, brush)
            palette.setBrush(QPalette.Inactive, QPalette.Base, brush1)
            palette.setBrush(QPalette.Inactive, QPalette.Window, brush1)
            palette.setBrush(QPalette.Inactive, QPalette.Shadow, brush)
            palette.setBrush(QPalette.Inactive,     QPalette.AlternateBase,
brush2)
            palette.setBrush(QPalette.Inactive,       QPalette.ToolTipBase,
brush5)
            palette.setBrush(QPalette.Inactive,       QPalette.ToolTipText,
brush)
      #if QT_VERSION >= QT_VERSION_CHECK(5, 12, 0)
            palette.setBrush(QPalette.Inactive,   QPalette.PlaceholderText,
brush6)
      #endif
            palette.setBrush(QPalette.Inactive, QPalette.Accent, brush2)
            palette.setBrush(QPalette.Disabled,        QPalette.WindowText,
brush3)
            palette.setBrush(QPalette.Disabled, QPalette.Button, brush1)
```

```
            palette.setBrush(QPalette.Disabled, QPalette.Light, brush2)
            palette.setBrush(QPalette.Disabled, QPalette.Midlight, brush2)
            palette.setBrush(QPalette.Disabled, QPalette.Dark, brush3)
            palette.setBrush(QPalette.Disabled, QPalette.Mid, brush4)
            palette.setBrush(QPalette.Disabled, QPalette.Text, brush3)
            palette.setBrush(QPalette.Disabled,        QPalette.BrightText,
brush2)
            palette.setBrush(QPalette.Disabled,        QPalette.ButtonText,
brush3)
            palette.setBrush(QPalette.Disabled, QPalette.Base, brush1)
            palette.setBrush(QPalette.Disabled, QPalette.Window, brush1)
            palette.setBrush(QPalette.Disabled, QPalette.Shadow, brush)
            palette.setBrush(QPalette.Disabled,     QPalette.AlternateBase,
brush2)
            palette.setBrush(QPalette.Disabled,       QPalette.ToolTipBase,
brush5)
            palette.setBrush(QPalette.Disabled,       QPalette.ToolTipText,
brush)
            brush7 = QBrush(QColor(127, 127, 127, 127))
            brush7.setStyle(Qt.SolidPattern)
      #if QT_VERSION >= QT_VERSION_CHECK(5, 12, 0)
            palette.setBrush(QPalette.Disabled,   QPalette.PlaceholderText,
brush7)
      #endif
            palette.setBrush(QPalette.Disabled, QPalette.Accent, brush2)
            MainWindow.setPalette(palette)
            MainWindow.setAutoFillBackground(False)
            MainWindow.setStyleSheet(u"background:#ffd9bf")
            self.centralwidget = QWidget(MainWindow)
            self.centralwidget.setObjectName(u"centralwidget")
            self.centralwidget.setStyleSheet(u"background:#ffd9bf")
            self.textBrowser = QTextBrowser(self.centralwidget)
            self.textBrowser.setObjectName(u"textBrowser")
            self.textBrowser.setGeometry(QRect(820, 120, 321, 241))
            self.textBrowser.setStyleSheet(u"QTextEdit {\n"
      "    border: 1px solid black;\n"
```

```
      "     background-color: white;\n"
      "     color: black;\n"
      "     font: 500 12pt \"Roboto Medium\";\n"
      "}\n"
      "")
              self.label_3 = QLabel(self.centralwidget)
              self.label_3.setObjectName(u"label_3")
              self.label_3.setGeometry(QRect(900, 90, 171, 16))
              self.label_3.setStyleSheet(u"font:     500     12pt     \"Roboto
Medium\";\n"
      "color: black;")
              self.label_4 = QLabel(self.centralwidget)
              self.label_4.setObjectName(u"label_4")
              self.label_4.setGeometry(QRect(330, 90, 171, 16))
              self.label_4.setStyleSheet(u"font:     500     12pt     \"Roboto
Medium\";\n"
      "color: black;")
              self.graphicsView_2 = QGraphicsView(self.centralwidget)
              self.graphicsView_2.setObjectName(u"graphicsView_2")
              self.graphicsView_2.setGeometry(QRect(40, 120, 751, 551))
              self.graphicsView_2.setStyleSheet(u"QGraphicsView {\n"
      "    border: 1px solid black;\n"
      "     background-color: white;\n"
      "}\n"
      "")
              self.label_5 = QLabel(self.centralwidget)
              self.label_5.setObjectName(u"label_5")
              self.label_5.setGeometry(QRect(460, 20, 271, 61))
              font = QFont()
              font.setFamilies([u"Roboto Medium"])
              font.setPointSize(20)
              font.setWeight(QFont.Medium)
              font.setItalic(False)
              self.label_5.setFont(font)
              self.label_5.setStyleSheet(u"font:     500     20pt     \"Roboto
Medium\";\n"
```

```
"color: black;")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QMenuBar(MainWindow)
        self.menubar.setObjectName(u"menubar")
        self.menubar.setGeometry(QRect(0, 0, 1177, 21))
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QStatusBar(MainWindow)
        self.statusbar.setObjectName(u"statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)

        QMetaObject.connectSlotsByName(MainWindow)
    # setupUi

    def retranslateUi(self, MainWindow):

MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow",  u"Osadcha
Sofiia", None))
        self.label_3.setText(QCoreApplication.translate("MainWindow",
u"\u0406\u043d\u0444\u043e\u0440\u043c\u0430\u0446\u0456\u044f
\u043f\u0440\u043e \u043d\u0430\u0431\u0456\u0440", None))
        self.label_4.setText(QCoreApplication.translate("MainWindow",
u"\u0414\u0435\u043a\u043e\u043c\u043f\u043e\u0437\u0438\u0446\u0456\u044f
\u043d\u0430\u0431\u043e\u0440\u0443", None))
        self.label_5.setText(QCoreApplication.translate("MainWindow",
u"\u0410\u043d\u0430\u043b\u0456\u0437   \u043d\u0430\u0431\u043e\u0440\u0443
\u0434\u0430\u043d\u0438\u0445", None))
        # retranslateUi
```

## Модуль ui_resultwindow.py

```
# -*- coding: utf-8 -*-

################################################################################
#########
```

```
      ## Form generated from reading UI file 'edawindow.ui'
      ##
      ## Created by: Qt User Interface Compiler version 6.7.0
      ##
      ## WARNING! All changes made in this file will be lost when recompiling
UI file!
      ################################################################################
#########

      from PySide6.QtCore import (QCoreApplication, QDate, QDateTime,
QLocale,
          QMetaObject, QObject, QPoint, QRect,
          QSize, QTime, QUrl, Qt)
      from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,
          QFont, QFontDatabase, QGradient, QIcon,
          QImage, QKeySequence, QLinearGradient, QPainter,
          QPalette, QPixmap, QRadialGradient, QTransform)
      from PySide6.QtWidgets import (QApplication, QGraphicsView, QLabel,
QMainWindow,
          QMenuBar, QSizePolicy, QStatusBar, QTextBrowser,
          QWidget)

      class Ui_MainWindow(object):
          def setupUi(self, MainWindow):
              if not MainWindow.objectName():
                  MainWindow.setObjectName(u"MainWindow")
              MainWindow.setEnabled(True)
              MainWindow.resize(1177, 757)
              palette = QPalette()
              brush = QBrush(QColor(0, 0, 0, 255))
              brush.setStyle(Qt.SolidPattern)
              palette.setBrush(QPalette.Active, QPalette.WindowText, brush)
              brush1 = QBrush(QColor(255, 217, 191, 255))
              brush1.setStyle(Qt.SolidPattern)
              palette.setBrush(QPalette.Active, QPalette.Button, brush1)
              brush2 = QBrush(QColor(255, 255, 255, 255))
```

```
            brush2.setStyle(Qt.SolidPattern)
            palette.setBrush(QPalette.Active, QPalette.Light, brush2)
            palette.setBrush(QPalette.Active, QPalette.Midlight, brush2)
            brush3 = QBrush(QColor(127, 127, 127, 255))
            brush3.setStyle(Qt.SolidPattern)
            palette.setBrush(QPalette.Active, QPalette.Dark, brush3)
            brush4 = QBrush(QColor(170, 170, 170, 255))
            brush4.setStyle(Qt.SolidPattern)
            palette.setBrush(QPalette.Active, QPalette.Mid, brush4)
            palette.setBrush(QPalette.Active, QPalette.Text, brush)
            palette.setBrush(QPalette.Active, QPalette.BrightText, brush2)
            palette.setBrush(QPalette.Active, QPalette.ButtonText, brush)
            palette.setBrush(QPalette.Active, QPalette.Base, brush1)
            palette.setBrush(QPalette.Active, QPalette.Window, brush1)
            palette.setBrush(QPalette.Active, QPalette.Shadow, brush)
            palette.setBrush(QPalette.Active,       QPalette.AlternateBase,
brush2)
            brush5 = QBrush(QColor(255, 255, 220, 255))
            brush5.setStyle(Qt.SolidPattern)
            palette.setBrush(QPalette.Active, QPalette.ToolTipBase, brush5)
            palette.setBrush(QPalette.Active, QPalette.ToolTipText, brush)
            brush6 = QBrush(QColor(0, 0, 0, 127))
            brush6.setStyle(Qt.SolidPattern)
      #if QT_VERSION >= QT_VERSION_CHECK(5, 12, 0)
            palette.setBrush(QPalette.Active,     QPalette.PlaceholderText,
brush6)
      #endif
            palette.setBrush(QPalette.Active, QPalette.Accent, brush2)
            palette.setBrush(QPalette.Inactive, QPalette.WindowText, brush)
            palette.setBrush(QPalette.Inactive, QPalette.Button, brush1)
            palette.setBrush(QPalette.Inactive, QPalette.Light, brush2)
            palette.setBrush(QPalette.Inactive, QPalette.Midlight, brush2)
            palette.setBrush(QPalette.Inactive, QPalette.Dark, brush3)
            palette.setBrush(QPalette.Inactive, QPalette.Mid, brush4)
            palette.setBrush(QPalette.Inactive, QPalette.Text, brush)
```

```
            palette.setBrush(QPalette.Inactive,        QPalette.BrightText,
brush2)
            palette.setBrush(QPalette.Inactive, QPalette.ButtonText, brush)
            palette.setBrush(QPalette.Inactive, QPalette.Base, brush1)
            palette.setBrush(QPalette.Inactive, QPalette.Window, brush1)
            palette.setBrush(QPalette.Inactive, QPalette.Shadow, brush)
            palette.setBrush(QPalette.Inactive,     QPalette.AlternateBase,
brush2)
            palette.setBrush(QPalette.Inactive,       QPalette.ToolTipBase,
brush5)
            palette.setBrush(QPalette.Inactive,       QPalette.ToolTipText,
brush)
      #if QT_VERSION >= QT_VERSION_CHECK(5, 12, 0)
            palette.setBrush(QPalette.Inactive,   QPalette.PlaceholderText,
brush6)
      #endif
            palette.setBrush(QPalette.Inactive, QPalette.Accent, brush2)
            palette.setBrush(QPalette.Disabled,        QPalette.WindowText,
brush3)
            palette.setBrush(QPalette.Disabled, QPalette.Button, brush1)
            palette.setBrush(QPalette.Disabled, QPalette.Light, brush2)
            palette.setBrush(QPalette.Disabled, QPalette.Midlight, brush2)
            palette.setBrush(QPalette.Disabled, QPalette.Dark, brush3)
            palette.setBrush(QPalette.Disabled, QPalette.Mid, brush4)
            palette.setBrush(QPalette.Disabled, QPalette.Text, brush3)
            palette.setBrush(QPalette.Disabled,        QPalette.BrightText,
brush2)
            palette.setBrush(QPalette.Disabled,        QPalette.ButtonText,
brush3)
            palette.setBrush(QPalette.Disabled, QPalette.Base, brush1)
            palette.setBrush(QPalette.Disabled, QPalette.Window, brush1)
            palette.setBrush(QPalette.Disabled, QPalette.Shadow, brush)
            palette.setBrush(QPalette.Disabled,     QPalette.AlternateBase,
brush2)
            palette.setBrush(QPalette.Disabled,       QPalette.ToolTipBase,
brush5)
```

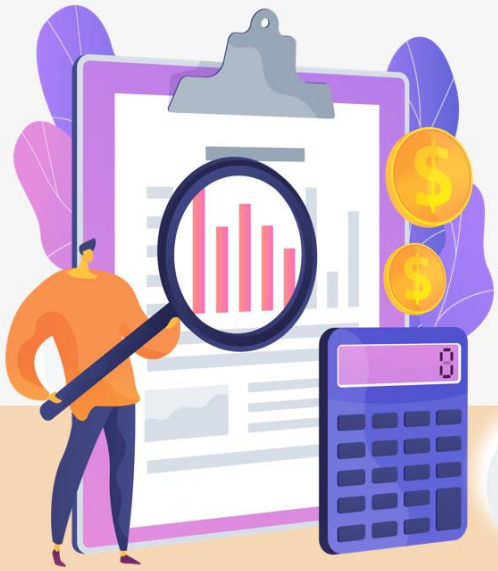
```
            palette.setBrush(QPalette.Disabled,       QPalette.ToolTipText,
brush)
            brush7 = QBrush(QColor(127, 127, 127, 127))
            brush7.setStyle(Qt.SolidPattern)
      #if QT_VERSION >= QT_VERSION_CHECK(5, 12, 0)
            palette.setBrush(QPalette.Disabled,   QPalette.PlaceholderText,
brush7)
      #endif
            palette.setBrush(QPalette.Disabled, QPalette.Accent, brush2)
            MainWindow.setPalette(palette)
            MainWindow.setAutoFillBackground(False)
            MainWindow.setStyleSheet(u"background:#ffd9bf")
            self.centralwidget = QWidget(MainWindow)
            self.centralwidget.setObjectName(u"centralwidget")
            self.centralwidget.setStyleSheet(u"background:#ffd9bf")
            self.textBrowser = QTextBrowser(self.centralwidget)
            self.textBrowser.setObjectName(u"textBrowser")
            self.textBrowser.setGeometry(QRect(820, 120, 321, 241))
            self.textBrowser.setStyleSheet(u"QTextEdit {\n"
      "    border: 1px solid black;\n"
      "     background-color: white;\n"
      "     color: black;\n"
      "     font: 500 12pt \"Roboto Medium\";\n"
      "}\n"
      "")
            self.label_3 = QLabel(self.centralwidget)
            self.label_3.setObjectName(u"label_3")
            self.label_3.setGeometry(QRect(900, 90, 171, 16))
            self.label_3.setStyleSheet(u"font:     500     12pt     \"Roboto
Medium\";\n"
      "color: black;")
            self.label_4 = QLabel(self.centralwidget)
            self.label_4.setObjectName(u"label_4")
            self.label_4.setGeometry(QRect(330, 90, 171, 16))
            self.label_4.setStyleSheet(u"font:     500     12pt     \"Roboto
Medium\";\n"
```

```
        "color: black;")
                self.graphicsView_2 = QGraphicsView(self.centralwidget)
                self.graphicsView_2.setObjectName(u"graphicsView_2")
                self.graphicsView_2.setGeometry(QRect(40, 120, 751, 551))
                self.graphicsView_2.setStyleSheet(u"QGraphicsView {\n"
        "    border: 1px solid black;\n"
        "     background-color: white;\n"
        "}\n"
        "")
                self.label_5 = QLabel(self.centralwidget)
                self.label_5.setObjectName(u"label_5")
                self.label_5.setGeometry(QRect(460, 20, 271, 61))
                font = QFont()
                font.setFamilies([u"Roboto Medium"])
                font.setPointSize(20)
                font.setWeight(QFont.Medium)
                font.setItalic(False)
                self.label_5.setFont(font)
                self.label_5.setStyleSheet(u"font:     500     20pt     \"Roboto
Medium\";\n"
        "color: black;")
                MainWindow.setCentralWidget(self.centralwidget)
                self.menubar = QMenuBar(MainWindow)
                self.menubar.setObjectName(u"menubar")
                self.menubar.setGeometry(QRect(0, 0, 1177, 21))
                MainWindow.setMenuBar(self.menubar)
                self.statusbar = QStatusBar(MainWindow)
                self.statusbar.setObjectName(u"statusbar")
                MainWindow.setStatusBar(self.statusbar)

                self.retranslateUi(MainWindow)

                QMetaObject.connectSlotsByName(MainWindow)
            # setupUi

            def retranslateUi(self, MainWindow):
```

```
MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow",  u"Osadcha
Sofiia", None))
              self.label_3.setText(QCoreApplication.translate("MainWindow",
u"\u0406\u043d\u0444\u043e\u0440\u043c\u0430\u0446\u0456\u044f
\u043f\u0440\u043e \u043d\u0430\u0431\u0456\u0440", None))
              self.label_4.setText(QCoreApplication.translate("MainWindow",
u"\u0414\u0435\u043a\u043e\u043c\u043f\u043e\u0437\u0438\u0446\u0456\u044f
\u043d\u0430\u0431\u043e\u0440\u0443", None))
              self.label_5.setText(QCoreApplication.translate("MainWindow",
u"\u0410\u043d\u0430\u043b\u0456\u0437   \u043d\u0430\u0431\u043e\u0440\u0443
\u0434\u0430\u043d\u0438\u0445", None))
          # retranslateUi
```

# ДОДАТОК Б ГРАФІЧНІ МАТЕРІАЛИ

1



## Прогнозування часових рядів для складських запасів

Виконала:
студентка IV курсу,
групи КА-04
Осадча Софія

Керівник:
в.о завідувача кафедри,
доцент к.т.н Тимощук Оксана
Леонідівна

2

## Постановка задачі

**01**
Пошук та підготовка тестових даних на основі часових рядів

**02**
Проведення аналізу зібраних даних за допомогою статистичних методів.

**03**
Вибір та тестування моделі прогнозування на основі регресійних моделей.

**04**
Створення програмного продукту, який має інтуїтивний користувацький інтерфейс.

3

# Актуальність задачі

Управління складськими запасами є критично важливим для успішної роботи будь-якого підприємства, оскільки точне прогнозування допомагає уникнути багатьох проблем(надлишкові запаси, нестача запасів тощо.)

Прогнозування сезонного попиту допомагає мінімізувати ризики, забезпечуючи оптимальний рівень запасів. Це сприяє зниженню витрат, підвищенню обслуговування клієнтів і покращенню конкурентоспроможності підприємства.

04

**Мета**

Проаналізувати підходи до прогнозування сезонного попиту товарів на підприємстві за допомогою часових рядів

**Об'єкт дослідження**

Процес управління запасами на підприємстві

**Предмет дослідження:**

- Модель ARIMA (Autoregressive Integrated Moving Average, ARIMA)
- Модель SARIMA(Seasonal Autoregressive Integrated Moving Average)

05

# Переваги моделей ARIMA та SARIMA

ARIMA моделі можуть захоплювати широкий спектр часових рядів, включаючи дані з трендами, автокореляцією та іншими структурними особливостями.

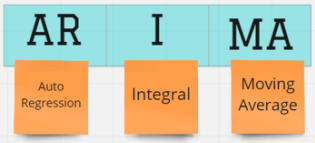
Модель SARIMA є більш точною для даних з явною сезонністю.

Інтегрована компонента (I) дозволяє моделі ARIMA обробляти нестаціонарні ряди, перетворюючи їх у стаціонарні, що спрощує прогнозування.

SARIMA ефективно прогнозує довгострокові сезонні тенденції, що є корисним для бізнес-планування, фінансового аналізу, управління запасами та інших сфер, де важливі сезонні коливання.

6

Модель прогнозування




# ARIMA

$$\Delta^d y_t = c + \phi_1 \Delta^d y_{t-1} + \cdots + \phi_p \Delta^d y_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

де $y_t$ – значення часового ряду на момент часу t,
$\Delta^d y_t$ – різниця ряду ступеня d, різниця першого ступеня записується за формулою $\Delta y_t = y_t - y_{t-1}$

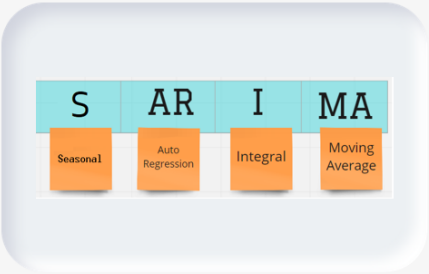
де c – константа,

$\varepsilon_t$ – значення білого шуму,

$\phi_1 \ldots \phi_p$ – коефіцієнти авторегресійної моделі,
$\theta_1 \ldots \theta_q$ – коефіцієнти моделі ковзного середнього.

7

Модель прогнозування




## SARIMA

$$(1-\phi_1 B)(1-\Phi_1 B^s)(1-B)(1-B^s)y_t = (1+\theta_1 B)(1+\Theta_1 B^s)\varepsilon_t$$

де $y_t$ – спостережуваний часовий ряд у момент часу $t$;

B – оператор зсуву назад,що представляє оператор затримки (тобто $By_t = y_{t-1}$);

$\phi_1$ – несезонний авторегресійний коефіцієнт;

$\Phi_1$ – сезонний авторегресійний коефіцієнт;

$\theta_1$ – несезонний ковзний середній коефіцієнт;

$\Theta_1$ – сезонний ковзний середній коефіцієнт;

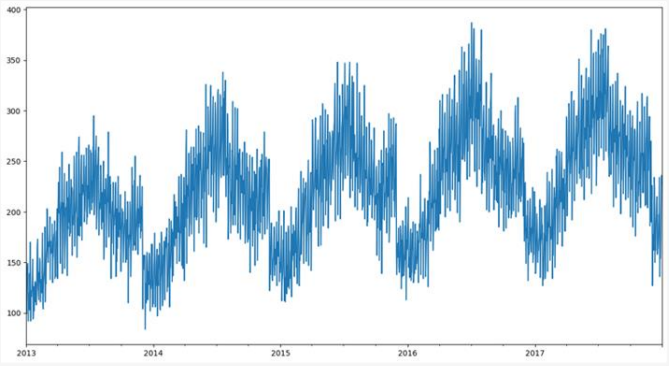
$s$ – сезонний період;

$\varepsilon_t$ – є членом помилки білого шуму в момент часу $t$.

8

## Вибір та опис датасету

Для тестування розробленого програмного продукту був використаний набір даних Store Item Demand Forecasting Challenge з Kaggle. Цей набір є часовим рядом попиту на різні види товарів у магазинах



9

# Аналіз набору даних

Після форматування даних, програма починає аналіз даних.

Починається перевірка чи є пропуски даних, стаціонарність даних, а також проводить декомпозицію ряду.

- Форматування даних: Ми привели дані до вигляду, що маємо дві колонки (Дата, Кількість продажів). Де дата стає індексом, а кількість продажів стовпчиком target.
- Стаціонарність: дані перевіряються за допомогою тесту Дікі-Фуллера.
- Декомпозиція ряду: розділення ряду на його складові – тренд, сезонність та шум.

10

## Тест параметрів моделі

Для того, щоб зрозуміти наскільки добре підібрані параметри, робимо тестування

11

## Формули помилок

**MSE**

Середньоквадратична помилка: Це середнє значення квадратів різниць між передбаченими і фактичними значеннями.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- n – кількість спостережень.
- $y_i$ – фактичне значення для $i$-го спостереження.
- $\hat{y}_i$ – передбачене значення для $i$-го спостереження.
- $(y_i - \hat{y}_i)^2$ – квадрат різниці між фактичним і передбаченим значеннями для $i$-го спостереження.

**MAE**

Середня абсолютна помилка: Це середнє значення абсолютних значень різниць між передбаченими і фактичними значеннями.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- n – кількість спостережень.
- $y_i$ – фактичне значення для $i$-го спостереження.
- $\hat{y}_i$ – передбачене значення для $i$-го спостереження.
- $|y_i - \hat{y}_i|$ – – абсолютне значення різниці між фактичним і передбаченим значеннями для $i$-го спостереження.

12

## Формули помилок

**MAPE**

Середня абсолютна процентна помилка: Це середнє значення абсолютних процентних помилок.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i^2}\right| \times 100\%$$

- n – кількість спостережень.
- $y_i$ – фактичне значення для $i$-го спостереження.
- $\hat{y}_i$ – передбачене значення для $i$-го спостереження.
- $\left|\frac{y_i - \hat{y}_i}{y_i}\right|$ – абсолютне значення відсоткової різниці між фактичним і передбаченим значеннями для $i$-го спостереження.

**R2**

Коефіцієнт детермінації: Вимірює, яка частка варіації залежної змінної пояснюється незалежними змінними.

$$R2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

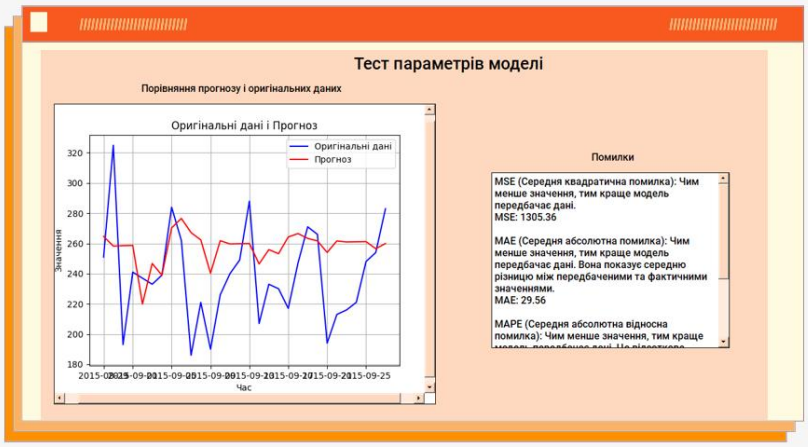
- $y_i$ – фактичне значення для $i$-го спостереження.
- $\hat{y}_i$ – передбачене значення для $i$-го спостереження.
- $\bar{y}_i$ – середнє значення всіх фактичних значень.
- $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ – сума квадратів різниць між фактичними і передбаченими значеннями.
- $\sum_{i=1}^{n}(y_i - \bar{y}_i)^2$ – сума квадратів різниць між фактичними значеннями і середнім фактичним значенням.

13

# Інтерфейс програми


14

# Інтерфейс програми



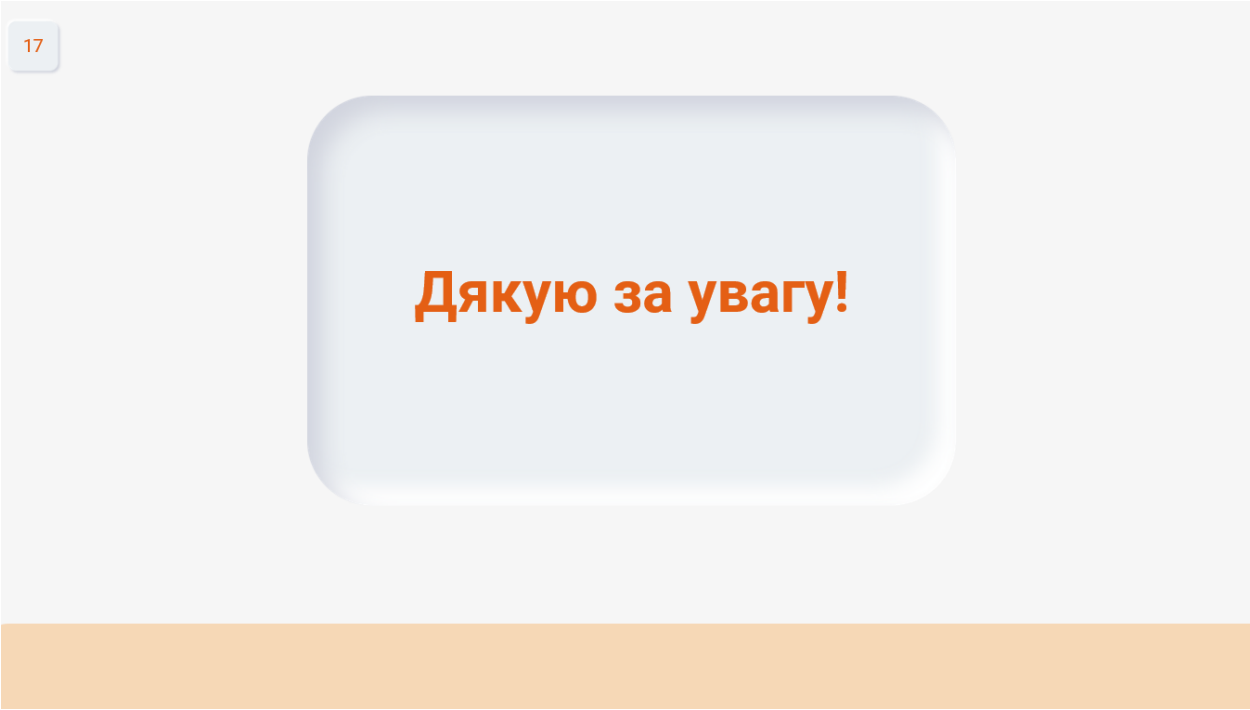
15

## Інтерфейс програми


16

## ВИСНОВКИ

Практичне значення програми полягає у можливості ефективного планування обсягів закупівель, зменшення витрат на зберігання, уникнення надмірних запасів та підвищення задоволеності клієнтів завдяки забезпеченню наявності необхідних товарів у потрібний час.


17
Дякую за увагу!