

**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
«__» _____ 2026 р.

Дипломна робота

на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»**

**на тему: «Система підтримки прийняття рішень для аналізу табличних
наборів даних»**

Виконав:

студент IV курсу, групи КА-22

Бондаренко Владислав Михайлович _____

Керівник:

Старший викладач кафедри ММСА, доктор філософії

Левенчук Людмила Борисівна _____

Консультант з економічного розділу:

Доцент кафедри ТПЕ, к.е.н.

Рощина Надія Василівна _____

Консультант з нормоконтролю:

Доцент кафедри ММСА, к.ф.-м.н.

Статкевич Віталій Михайлович _____

Рецензент:

Доцент кафедри ШІ, доктор філософії, доцент

Гуськова Віра Геннадіївна _____

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

Студент _____

Київ – 2026 року

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
 «__» _____ 2026 р.

ЗАВДАННЯ
на дипломну роботу студенту
Бондаренку Владиславу Михайловичу

1. Тема роботи «Система підтримки прийняття рішень для аналізу табличних наборів даних», керівник роботи Левенчук Людмила Борисівна, доктор філософії, затверджені наказом по університету від 27.05.2026 р. № 1944-с.
2. Термін подання студентом роботи «10» червня 2026 р.
3. Вихідні дані до роботи: коректні результати користування системою.
4. Зміст роботи: дослідження предметної області систем підтримки прийняття рішень та методів аналізу табличних даних; аналіз існуючих програмних засобів обробки та візуалізації даних; проектування та розробка системи підтримки прийняття рішень для аналізу табличних наборів даних; дослідження ефективності роботи розробленої системи на практичних наборах даних; функціонально-вартісний аналіз програмного продукту.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Функціонально-вартісний аналіз	Рощина Н.В., доц., к.е.н.		

7. Дата видачі завдання 20.04.2026

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Затвердження теми бакалаврської дипломної роботи (БДР), опанування ДСТУ 3008:2015	26.02.2026-28.02.2026	виконано
2	Вивчення літератури за темою	20.04.2026-26.04.2026	виконано
3	Підготовка першого розділу	26.04.2026-03.05.2026	виконано
4	Підготовка другого розділу	03.05.2026-10.05.2026	виконано
5	Розробка програмного продукту	10.05.2026-20.05.2026	виконано
6	Підготовка третього розділу	20.05.2026-23.05.2026	виконано
7	Функціонально-вартісний аналіз	23.05.2026-25.05.2026	виконано
8	Оформлення дипломної роботи	25.05.2026-29.05.2026	виконано

Студент

Владислав БОНДАРЕНКО

Керівник

Людмила ЛЕВЕНЧУК

РЕФЕРАТ

Дипломна робота містить 108 сторінок, 21 рисунок, 27 таблиць, 2 додатки, 15 джерел.

СИСТЕМА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ, АНАЛІЗ ТАБЛИЧНИХ НАБОРІВ ДАНИХ, АНАЛІЗ ЧАСОВИХ РЯДІВ, КЛАСТЕРИЗАЦІЯ ДАНИХ, ВИЯВЛЕННЯ АНОМАЛІЙ, K-MEANS, Z-SCORE, STREAMLIT, GEMINI API.

Метою роботи є дослідження та реалізація методів аналізу табличних наборів даних із використанням сучасних засобів аналізу даних та машинного навчання.

Об'єктом дослідження є процес аналізу табличних наборів даних у системах підтримки прийняття рішень. Предметом дослідження є методи аналізу часових рядів, кластеризації даних та виявлення аномалій.

У роботі проведено аналіз методів SMA, ЕМА, лінійного прогнозування, методу Z-score, алгоритму K-Means та методу ліктя. Досліджено сучасні програмні засоби аналізу даних, зокрема Python, pandas, NumPy, scikit-learn, Plotly та Streamlit.

У межах роботи реалізовано систему підтримки прийняття рішень для аналізу табличних наборів даних. Система забезпечує завантаження та обробку даних, аналіз часових рядів, виявлення аномалій, кластеризацію даних та інтерактивну візуалізацію результатів.

Результатом роботи є веб-застосунок, реалізований мовою програмування Python із використанням Streamlit та бібліотек аналізу даних. Для автоматизованої генерації аналітичних висновків використано Gemini API.

ABSTRACT

The thesis consists of 108 pages, 21 figures, 27 tables, 2 appendices, 15 references.

DECISION SUPPORT SYSTEM, TABULAR DATA ANALYSIS, TIME SERIES ANALYSIS, DATA CLUSTERING, ANOMALY DETECTION, K-MEANS, Z-SCORE, STREAMLIT, GEMINI API.

The purpose of this work is to investigate and implement methods for tabular data analysis using modern data analysis and machine learning tools.

The object of the research is the process of tabular data analysis in decision support systems. The subject of the research includes methods of time series analysis, data clustering, and anomaly detection.

The work analyzes the SMA, EMA, linear forecasting, Z-score method, K-Means algorithm, and Elbow Method. Modern data analysis tools, including Python, pandas, NumPy, scikit-learn, Plotly, and Streamlit, were also studied.

As part of the work, a decision support system for tabular data analysis was developed. The system provides data loading and processing, time series analysis, anomaly detection, data clustering, and interactive visualization of analysis results.

The result of the work is a web application implemented in Python using Streamlit and data analysis libraries. Gemini API was used for automated generation of analytical conclusions.

Further development of the work involves expanding the system's functionality, improving forecasting methods, and integrating additional machine learning algorithms.

ЗМІСТ

ВСТУП.....	8
РОЗДІЛ 1 ТЕОРЕТИЧНИЙ АПАРАТ.....	9
1.1 Аналіз табличних наборів даних.....	9
1.2 Аналіз часових рядів.....	10
1.2.1 Метод простого ковзного середнього (SMA).....	12
1.2.2 Метод експоненційного ковзного середнього (EMA).....	13
1.2.3 Лінійне прогнозування часових рядів.....	14
1.3 Аномалії в даних.....	16
1.3.1 Метод Z-score.....	17
1.4 Кластеризація даних.....	18
1.4.1 Метод K-Means.....	19
1.4.2 Метод ліктя.....	21
1.5 Програмні засоби аналізу даних.....	22
1.5.1 Мова програмування Python.....	23
1.5.2 Бібліотека pandas.....	24
1.5.3 Бібліотека NumPy.....	25
1.5.4 Бібліотека scikit-learn.....	26
1.5.5 Бібліотека Plotly.....	27
1.5.6 Streamlit.....	28
1.5.7 Використання Gemini API для генерації аналітичних висновків..	29
Висновки до розділу 1.....	30
РОЗДІЛ 2 РЕАЛІЗАЦІЯ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ..	32
2.1 Загальна характеристика розробленої системи.....	32

2.2 Реалізація підготовки даних.....	34
2.3 Реалізація аналізу часових рядів.....	36
2.4 Реалізація виявлення аномалій.....	40
2.5 Реалізація кластерного аналізу.....	42
2.6 Генерація AI-висновків.....	46
Висновки до розділу 2.....	48
РОЗДІЛ 3 ПРОЄКТУВАННЯ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ	
РІШЕНЬ.....	49
3.1 Архітектура системи.....	49
3.2 Переваги реалізованої архітектури.....	51
Висновки до розділу 3.....	52
РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ.....	
54	
4.1 Постановка задачі проєктування.....	55
4.2 Обґрунтування функцій програмного продукту.....	56
4.3 Обґрунтування системи параметрів програмного продукту.....	60
4.4 Аналіз експертного оцінювання параметрів.....	63
4.5 Аналіз рівня якості варіантів реалізації функцій.....	67
4.6 Економічний аналіз варіантів розробки ПП.....	69
4.7 Вибір кращого варіанту ПП техніко-економічного рівня.....	74
Висновки до розділу 4.....	75
ВИСНОВКИ.....	76
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....	78
ДОДАТОК А ЛІСТИНГ ПРОГРАМИ.....	80
ДОДАТОК Б ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ.....	99

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій та цифровізації різних сфер діяльності значно зростає обсяг даних, що накопичуються та обробляються інформаційними системами. Аналіз табличних наборів даних є важливою складовою процесу прийняття рішень у багатьох галузях, зокрема у сфері економіки, фінансів та інформаційних технологій. У зв'язку з цим виникає необхідність використання сучасних методів аналізу даних для автоматизації обробки інформації, виявлення прихованих закономірностей та пошуку аномалій.

Одним із перспективних напрямів аналізу даних є використання методів машинного навчання та статистичного аналізу. Особливу увагу приділяють задачам кластеризації, виявлення аномалій та аналізу часових рядів, оскільки такі методи дозволяють ефективно досліджувати структуру даних, знаходити нетипові значення та прогнозувати розвиток процесів.

Актуальність роботи полягає у необхідності створення систем підтримки прийняття рішень, здатних автоматизувати процес аналізу табличних наборів даних та формування аналітичних висновків.

Метою роботи є дослідження та реалізація підходів до аналізу табличних наборів даних із використанням методів кластеризації, виявлення аномалій та аналізу часових рядів.

У роботі використовуються методи статистичного аналізу, алгоритми машинного навчання та інструменти обробки даних із використанням мови програмування Python.

РОЗДІЛ 1 ТЕОРЕТИЧНИЙ АПАРАТ

1.1 Аналіз табличних наборів даних

У сучасних інформаційних системах значна частина даних зберігається у табличному вигляді. Табличні набори даних широко використовуються у фінансовій сфері, медицині, маркетингу, логістиці, інформаційних технологіях та інших галузях діяльності. Такий формат подання інформації забезпечує зручність зберігання, обробки та подальшого аналізу даних [3].

Табличний набір даних являє собою структуру, що складається з рядків та стовпців. Кожний рядок зазвичай описує окремий об'єкт або спостереження, а стовпці містять характеристики цих об'єктів. Дані можуть бути числовими, текстовими, часовими або категоріальними, що дозволяє виконувати різні види аналізу залежно від поставленої задачі.

Аналіз табличних наборів даних є важливим етапом процесу обробки інформації, оскільки дозволяє виявляти закономірності, визначати взаємозв'язки між параметрами та отримувати інформацію, необхідну для прийняття рішень [3]. Застосування методів аналізу даних дає можливість здійснювати оцінку стану об'єктів дослідження, знаходити відхилення від нормальної поведінки та прогнозувати розвиток певних процесів.

Процес аналізу даних зазвичай включає декілька основних етапів:

- 1) збір даних;
- 2) попередню обробку та очищення даних;
- 3) аналіз структури набору даних;
- 4) візуалізацію даних;
- 5) інтерпретацію отриманих результатів.

Одним із важливих етапів є попередня обробка даних. На цьому етапі виконуються перевірка коректності даних, усунення пропущених значень,

видалення дублікатів та приведення даних до необхідного формату. Якість підготовки даних безпосередньо впливає на точність подальшого аналізу та достовірність отриманих результатів.

Важливу роль у процесі аналізу даних відіграє візуалізація інформації. Графічне подання результатів дозволяє спростити сприйняття даних, виявити закономірності та полегшити аналіз структури набору даних. Для цього використовуються різні типи графіків, діаграм та гістограм [5].

Аналіз табличних наборів даних є важливою складовою сучасних систем підтримки прийняття рішень. Використання методів аналізу даних дозволяє підвищити ефективність обробки інформації, автоматизувати процес пошуку закономірностей та забезпечити більш обґрунтоване прийняття рішень.

1.2 Аналіз часових рядів

Часовий ряд являє собою послідовність значень певного показника, впорядкованих у часі. Аналіз часових рядів широко застосовується у сфері економіки, фінансів, інформаційних технологій, метеорології та інших галузях, де необхідно досліджувати зміну показників у часі та прогнозувати їх подальший розвиток [1].

Основною особливістю часових рядів є залежність значень показника від часу. На відміну від звичайних наборів даних, елементи часового ряду мають певний порядок, а зміна значень може містити закономірності, тенденції та повторювані компоненти.

Під час аналізу часових рядів зазвичай виділяють такі основні компоненти:

- 1) тренд;
- 2) сезонність;

- 3) циклічність;
- 4) випадкові коливання.

Тренд характеризує загальний напрям зміни показника протягом певного проміжку часу. Сезонність відображає регулярні зміни, що повторюються через однакові проміжки часу. Циклічність описує довгострокові коливання показників, а випадкові коливання характеризують випадкові зміни, які неможливо точно передбачити [1].

Одним із основних завдань аналізу часових рядів є прогнозування майбутніх значень показників [1, 5]. Для цього використовуються статистичні методи та моделі прогнозування, які дозволяють виявляти закономірності у даних та оцінювати можливий розвиток процесів у майбутньому.

Важливим етапом аналізу часових рядів є попередня обробка даних. Вона включає перевірку коректності даних, усунення пропущених значень, згладжування випадкових коливань та визначення основних характеристик ряду. Якість підготовки даних безпосередньо впливає на точність подальшого аналізу та достовірність результатів прогнозування.

Для дослідження часових рядів часто використовуються методи згладжування даних, які дозволяють зменшити вплив випадкових коливань та виділити основну тенденцію зміни показників. Також застосовуються методи статистичного аналізу для оцінки структури ряду та виявлення залежностей між значеннями показників у різні моменти часу.

Аналіз часових рядів є важливою складовою сучасних систем аналізу даних та підтримки прийняття рішень. Використання методів аналізу часових рядів дозволяє оцінювати динаміку процесів, прогнозувати зміни показників та підвищувати ефективність прийняття рішень на основі отриманих результатів.

1.2.1 Метод простого ковзного середнього (SMA)

Одним із найбільш поширених методів аналізу часових рядів є метод простого ковзного середнього (Simple Moving Average, SMA). Даний метод використовується для згладжування часових рядів та виявлення основної тенденції зміни показників у часі [1].

Принцип роботи методу полягає у послідовному обчисленні середнього значення за певний проміжок спостережень, який називається вікном. Для кожної точки часового ряду розраховується середнє значення декількох сусідніх елементів ряду. У результаті випадкові коливання зменшуються, а загальна тенденція зміни показника стає більш помітною.

Формально просте ковзне середнє обчислюється за формулою [1]:

$$SMA_t = \frac{1}{n} \sum_{i=0}^{n-1} x_{t-i}$$

де:

SMA_t – значення простого ковзного середнього;

n – розмір вікна;

x_{t-i} – значення часового ряду.

Метод простого ковзного середнього дозволяє зменшити вплив випадкових відхилень та шуму в даних, що є важливим під час аналізу економічних, фінансових та інших часових рядів. Застосування SMA спрощує виявлення тренду та полегшує подальший аналіз даних.

Основним параметром методу є розмір вікна. При малому розмірі вікна часовий ряд залишається більш чутливим до змін, тоді як збільшення розміру вікна забезпечує сильніше згладжування даних. Вибір оптимального розміру вікна залежить від структури даних та поставленої задачі аналізу.

До переваг методу простого ковзного середнього можна віднести простоту реалізації, зрозумілість інтерпретації результатів та ефективність під час аналізу загальної тенденції часового ряду. Недоліком методу є те, що SMA однаково враховує всі значення у межах вікна та може повільно реагувати на різкі зміни показників.

Метод простого ковзного середнього широко використовується у системах аналізу даних та прогнозування, оскільки дозволяє здійснювати базове згладжування часових рядів та покращує візуальне сприйняття зміни показників у часі [1].

1.2.2 Метод експоненційного ковзного середнього (ЕМА)

Метод експоненційного ковзного середнього (Exponential Moving Average, ЕМА) є одним із методів згладжування часових рядів, який використовується для аналізу динаміки зміни показників у часі. На відміну від простого ковзного середнього, ЕМА надає більшу вагу новим значенням часового ряду, завдяки чому метод швидше реагує на зміни даних [1].

Основна ідея методу полягає у тому, що останні значення часового ряду мають більший вплив на результат обчислення, ніж попередні спостереження. Це дозволяє ефективніше відстежувати зміни тенденції та швидше виявляти нові закономірності у даних.

Експоненційне ковзне середнє обчислюється за формулою [1]:

$$EMA_t = \alpha x_t + (1 - \alpha)EMA_{t-1}$$

де:

EMA_t – значення експоненційного ковзного середнього у момент часу t ;

x_t – поточне значення часового ряду;

EMA_{t-1} – попереднє значення експоненційного ковзного середнього;

α – коефіцієнт згладжування, значення якого знаходиться в межах від 0 до 1.

Коефіцієнт згладжування визначає ступінь впливу нових значень на результат обчислення. При більшому значенні α метод швидше реагує на зміни у часовому ряді, тоді як менші значення забезпечують більш плавне згладжування даних.

Метод ЕМА широко використовується під час аналізу фінансових показників, економічних процесів та інших часових рядів, де важливо швидко реагувати на зміну тенденцій. Порівняно з методом SMA, експоненційне ковзне середнє дозволяє ефективніше враховувати актуальні зміни у даних та забезпечує більш гнучкий аналіз часового ряду [1, 5].

До переваг методу ЕМА можна віднести швидку реакцію на зміни даних, простоту реалізації та ефективність під час аналізу трендів. Недоліком методу є залежність результатів від правильно обраного коефіцієнта згладжування, оскільки надто великі значення можуть призводити до підвищеної чутливості до випадкових коливань.

1.2.3 Лінійне прогнозування часових рядів

Лінійне прогнозування є одним із базових методів аналізу часових рядів, який використовується для оцінки подальшої зміни показників на основі наявних даних. Даний метод ґрунтується на припущенні, що зміна значень часового ряду може бути описана лінійною залежністю [1].

Основною метою лінійного прогнозування є побудова математичної моделі, яка дозволяє оцінити тенденцію зміни показника та спрогнозувати його майбутні значення. Метод широко використовується у сфері економіки, фінансів, аналітики даних та інформаційних систем завдяки простоті реалізації та зрозумілості інтерпретації результатів.

Лінійна модель прогнозування описується рівнянням [5]:

$$y = kx + b$$

де:

y – прогнозоване значення показника;

x – порядковий номер або момент часу;

k – коефіцієнт нахилу прямої;

b – початкове значення моделі.

Коефіцієнт k визначає напрям та швидкість зміни показника. Якщо значення коефіцієнта є додатним, показник має тенденцію до зростання, а при від'ємному значенні спостерігається спадання.

Під час побудови лінійного прогнозу виконується апроксимація часового ряду прямою лінією, яка найбільш точно описує загальну тенденцію зміни даних. На основі отриманої моделі здійснюється прогнозування майбутніх значень на певну кількість кроків уперед.

Перевагами лінійного прогнозування є простота реалізації, невисока обчислювальна складність та можливість швидкого отримання прогнозних значень. Метод дозволяє ефективно аналізувати часові ряди з вираженим трендом та отримувати базову оцінку подальшої зміни показників.

Недоліком методу є те, що лінійне прогнозування не враховує складні залежності, сезонні компоненти та випадкові коливання у даних. У випадку складних часових рядів точність прогнозу може знижуватись, тому для більш точного аналізу можуть використовуватись складніші методи прогнозування [1, 5].

1.3 Аномалії в даних

Основною метою виявлення аномалій є знаходження значень або об'єктів, які суттєво відрізняються від основної структури даних та можуть свідчити про помилки, нестандартну поведінку або критичні зміни у досліджуваному процесі [2].

Аномаліями називають спостереження, які мають значні відхилення від більшості інших значень набору даних. Причинами виникнення аномалій можуть бути помилки введення даних, технічні несправності, різкі зміни процесів або нетипова поведінка об'єктів дослідження.

Під час аналізу даних виявлення аномалій дозволяє:

- 1) визначати помилки та некоректні значення у наборах даних;
- 2) виявляти нестандартні зміни показників;
- 3) підвищувати достовірність результатів аналізу;
- 4) покращувати якість прийняття рішень на основі даних [2].

Для виявлення аномалій використовуються статистичні методи, алгоритми машинного навчання та методи аналізу часових рядів. Одним із найбільш поширених підходів є використання статистичних методів оцінки відхилення значень від середнього рівня.

Ефективність методів виявлення аномалій значною мірою залежить від структури набору даних, кількості шуму, типу даних та правильно обраних параметрів аналізу. Тому під час дослідження даних важливим є вибір методу, який найбільш відповідає особливостям поставленої задачі [2].

1.3.1 Метод Z-score

Одним із найбільш поширених статистичних методів виявлення аномалій є метод Z-score. Даний метод використовується для оцінки ступеня відхилення окремих значень від середнього значення набору даних. Основна ідея методу полягає у визначенні того, наскільки далеко певне значення знаходиться від середнього значення відносно стандартного відхилення [2].

Метод Z-score дозволяє виявляти значення, які суттєво відрізняються від більшості інших спостережень та можуть вважатися аномальними. Такий підхід широко використовується у статистичному аналізі, фінансових системах, аналізі часових рядів та системах підтримки прийняття рішень.

Значення Z-score обчислюється за формулою [2]:

$$z = \frac{x - \mu}{\sigma}$$

де:

z – стандартизоване значення;

x – поточне значення набору даних;

μ – середнє значення вибірки;

σ – стандартне відхилення.

Результат обчислення показує, на скільки стандартних відхилень певне значення відрізняється від середнього. Чим більшим є абсолютне значення Z-score, тим більш нетиповим вважається спостереження.

Для визначення аномалій зазвичай використовується порогове значення. У більшості випадків значення вважається аномальним, якщо [2]:

$$|z| > 3$$

Таке правило ґрунтується на властивостях нормального розподілу, відповідно до яких переважна більшість значень знаходиться в межах трьох стандартних відхилень від середнього.

До переваг методу Z-score можна віднести простоту реалізації, невисоку обчислювальну складність та швидкість аналізу даних. Метод є ефективним для виявлення різких відхилень у числових наборах даних та може використовуватись у системах автоматизованого аналізу інформації.

Недоліком методу є залежність результатів від розподілу даних. У випадку наявності великої кількості аномалій або значного шуму ефективність методу може знижуватись. Також метод чутливий до сильних викидів, які можуть впливати на середнє значення та стандартне відхилення вибірки [2].

1.4 Кластеризація даних

Кластеризація даних є одним із методів аналізу даних, основною метою якого є автоматичний поділ об'єктів на групи відповідно до їх схожості. Об'єкти, які мають подібні характеристики, об'єднуються в один кластер, тоді як об'єкти з різними характеристиками потрапляють до різних груп [5].

Методи кластеризації широко застосовуються у сфері аналізу даних, машинного навчання, маркетингу, фінансових системах, медицині та інших галузях, де необхідно виконувати сегментацію великих наборів даних. Використання кластерного аналізу дозволяє виявляти приховані закономірності, аналізувати структуру даних та спрощувати подальшу інтерпретацію результатів [5].

Основною особливістю кластеризації є відсутність попередньо визначених класів або категорій. Алгоритм самостійно визначає структуру даних та формує групи об'єктів на основі обраних критеріїв схожості.

Під час кластерного аналізу важливу роль відіграє вибір ознак, за якими здійснюється групування об'єктів. Якість обраних параметрів безпосередньо

впливає на точність формування кластерів та достовірність отриманих результатів.

Для оцінки схожості між об'єктами зазвичай використовуються різні метрики відстані, серед яких найбільш поширеною є евклідова відстань [5]. Чим меншою є відстань між об'єктами, тим більш схожими вони вважаються.

Перед виконанням кластеризації часто здійснюється попередня обробка даних, яка включає очищення даних, заповнення пропущених значень та масштабування ознак. Це дозволяє зменшити вплив різних масштабів значень на результати кластерного аналізу та підвищити точність роботи алгоритмів.

Методи кластеризації є важливою складовою сучасних систем аналізу даних та підтримки прийняття рішень. Використання кластерного аналізу дозволяє автоматизувати процес сегментації даних, виявляти групи схожих об'єктів та спрощувати аналіз великих наборів інформації.

1.4.1 Метод K-Means

Одним із найбільш поширених методів кластеризації даних є метод K-Means. Даний алгоритм використовується для автоматичного поділу набору даних на визначену кількість груп, які називаються кластерами. Основною метою алгоритму є об'єднання схожих об'єктів у межах одного кластера та максимальне розділення різних груп об'єктів [5, 7].

Принцип роботи методу K-Means полягає у визначенні центрів кластерів, які називаються центроїдами. На початковому етапі алгоритм випадковим чином обирає задану кількість центрів кластерів. Після цього кожний об'єкт набору даних відноситься до того кластера, центр якого знаходиться на найменшій відстані від об'єкта.

Для визначення схожості між об'єктами найчастіше використовується евклідова відстань, яка обчислюється за формулою [5]:

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

де:

$d(x, y)$ – відстань між об'єктами;

x_i, y_i – значення ознак об'єктів;

n – кількість ознак.

Після розподілу об'єктів між кластерами алгоритм виконує перерахунок центрів кластерів. Новий центр визначається як середнє значення всіх об'єктів, що належать до відповідного кластера. Даний процес повторюється до моменту, поки положення центрів кластерів не перестане суттєво змінюватися.

Основним параметром алгоритму є кількість кластерів K , яку необхідно визначити перед початком роботи алгоритму [5]. Вибір оптимальної кількості кластерів безпосередньо впливає на якість результатів кластеризації.

До переваг методу K-Means можна віднести простоту реалізації, високу швидкість роботи та ефективність під час аналізу великих наборів даних. Алгоритм дозволяє швидко виконувати сегментацію об'єктів та виявляти структуру даних [7].

Недоліком методу є необхідність попереднього визначення кількості кластерів, а також чутливість до початкового розташування центрів кластерів. Крім того, результати роботи алгоритму можуть погіршуватися у випадку наявності шуму або аномальних значень у даних.

Метод K-Means широко використовується у системах аналізу даних, маркетингових дослідженнях, фінансовій аналітиці та системах підтримки прийняття рішень для сегментації об'єктів та дослідження структури наборів даних.

1.4.2 Метод ліктя

Однією з важливих задач під час використання алгоритму K-Means є визначення оптимальної кількості кластерів. Для вирішення цієї задачі широко застосовується метод ліктя (Elbow Method), який дозволяє оцінити якість кластеризації при різній кількості кластерів та обрати найбільш доцільне значення параметра K [7].

Принцип роботи методу полягає у послідовному виконанні кластеризації для різної кількості кластерів та обчисленні суми квадратів відстаней між об'єктами та центрами відповідних кластерів. Даний показник характеризує ступінь компактності кластерів та дозволяє оцінити якість групування об'єктів.

Для оцінки результатів кластеризації використовується функція [7]:

$$WCSS = \sum_{i=1}^K \sum_{x \in C_i} (x - \mu_i)^2$$

де:

$WCSS$ – сума квадратів внутрішньокластерних відстаней;

K – кількість кластерів;

C_i – множина об'єктів i -го кластера;

μ_i – центр відповідного кластера;

x – окремий об'єкт набору даних.

Під час збільшення кількості кластерів значення $WCSS$ зменшується, оскільки об'єкти розподіляються між більшою кількістю груп. Однак після певного моменту швидкість зменшення показника стає меншою. На графіку залежності $WCSS$ від кількості кластерів така точка зазвичай нагадує форму ліктя, звідки і походить назва методу.

Оптимальною вважається така кількість кластерів, у якій подальше збільшення значення K не забезпечує суттєвого покращення результатів

кластеризації. Використання методу ліктя дозволяє зменшити ризик вибору надто малої або надто великої кількості кластерів [7].

Перевагою методу є простота реалізації та наочність інтерпретації результатів. Недоліком є те, що у деяких випадках точка “ліктя” може бути виражена недостатньо чітко, що ускладнює вибір оптимальної кількості кластерів.

1.5 Програмні засоби аналізу даних

Сучасний аналіз даних неможливо уявити без використання спеціалізованих програмних засобів та інструментів автоматизації обробки інформації. Використання сучасних технологій дозволяє ефективно працювати з великими наборами даних, виконувати статистичний аналіз, будувати моделі машинного навчання та візуалізувати результати досліджень.

Програмні засоби аналізу даних широко застосовуються у сфері економіки, фінансів, інформаційних технологій, медицини та інших галузях, де необхідно здійснювати обробку та аналіз великих обсягів інформації. Використання спеціалізованих бібліотек та програмних платформ дозволяє автоматизувати процес підготовки даних, реалізовувати алгоритми аналізу та спрощувати інтерпретацію отриманих результатів.

Одним із найбільш популярних інструментів для аналізу даних є мова програмування Python, яка завдяки великій кількості бібліотек та простоті використання широко застосовується у сфері Data Science та машинного навчання. Для роботи з табличними наборами даних, статистичного аналізу, кластеризації та візуалізації результатів використовуються спеціалізовані бібліотеки, які забезпечують ефективну реалізацію алгоритмів аналізу даних.

У процесі реалізації системи підтримки прийняття рішень для аналізу табличних наборів даних використовувались програмні засоби для обробки

даних, побудови часових рядів, виявлення аномалій, кластеризації та візуалізації результатів аналізу. Використання сучасних програмних інструментів дозволяє підвищити ефективність обробки інформації, автоматизувати виконання аналітичних задач та забезпечити зручну взаємодію користувача із системою.

1.5.1 Мова програмування Python

Python є однією з найбільш популярних мов програмування у сфері аналізу даних, машинного навчання та розробки інформаційних систем. Завдяки простому синтаксису, великій кількості бібліотек та високій продуктивності Python широко використовується для обробки табличних наборів даних, статистичного аналізу та побудови аналітичних систем [5].

Однією з основних переваг мови Python є простота написання та читання коду. Використання зрозумілого синтаксису дозволяє спростити процес розробки програмного забезпечення та прискорити реалізацію алгоритмів аналізу даних. Це є особливо важливим під час створення систем підтримки прийняття рішень та реалізації методів машинного навчання.

Python підтримує велику кількість спеціалізованих бібліотек для роботи з даними, математичних обчислень, статистичного аналізу, візуалізації та побудови моделей машинного навчання. Завдяки цьому мова програмування широко застосовується для реалізації систем аналізу даних різної складності.

Ще однією перевагою Python є підтримка різних форматів даних та можливість інтеграції з іншими програмними засобами. Мова дозволяє працювати з CSV, Excel, базами даних та іншими джерелами інформації, що є важливим під час аналізу табличних наборів даних [4].

Python активно використовується у сфері Data Science, штучного інтелекту та аналітики даних завдяки великій спільноті розробників та

постійному розвитку екосистеми бібліотек. Використання Python дозволяє реалізовувати алгоритми аналізу даних, автоматизувати процес обробки інформації та створювати інтерактивні аналітичні системи.

У процесі розробки системи підтримки прийняття рішень Python використовувався як основний засіб реалізації алгоритмів аналізу часових рядів, виявлення аномалій, кластеризації даних та побудови інтерактивної системи аналізу табличних наборів даних.

1.5.2 Бібліотека pandas

Бібліотека pandas є одним із основних інструментів для роботи з табличними даними у мові програмування Python. Вона призначена для зчитування, обробки, аналізу та структурування інформації у зручному форматі. Завдяки широкому набору функцій pandas активно застосовується у сфері аналітики даних, машинного навчання та створення інформаційних систем [6].

Основною структурою даних у pandas є DataFrame, який являє собою двовимірну таблицю з рядками та стовпцями. Такий формат дозволяє зручно працювати з великими обсягами інформації, здійснювати фільтрацію, сортування, групування та зміну структури наборів даних.

Бібліотека підтримує роботу з різними форматами файлів, зокрема CSV та Excel. Це дозволяє завантажувати інформацію з зовнішніх джерел та виконувати її подальшу обробку. У процесі аналізу даних важливу роль відіграють операції очищення наборів даних, серед яких видалення дублікатів, обробка пропущених значень та зміна типів даних.

Однією з переваг pandas є можливість швидкого виконання статистичних операцій та аналізу структури набору даних. Бібліотека

дозволяє визначати основні характеристики даних, виконувати агрегацію значень та готувати інформацію до подальшого аналізу [6].

Під час реалізації системи підтримки прийняття рішень бібліотека `pandas` використовувалась для завантаження табличних наборів даних, попередньої обробки інформації, зміни типів даних, роботи з часовими рядами та формування структурованих таблиць для подальшого аналізу.

1.5.3 Бібліотека NumPy

NumPy є однією з базових бібліотек мови програмування Python для виконання математичних та числових обчислень. Бібліотека забезпечує ефективну роботу з багатовимірними масивами даних та використовується у багатьох системах аналізу даних, машинного навчання та наукових обчислень [12].

Основою є структура `ndarray`, яка дозволяє зберігати та обробляти великі набори числових даних. Використання масивів NumPy забезпечує високу швидкість обчислень та оптимізацію операцій над даними порівняно зі стандартними структурами Python [12].

Бібліотека містить велику кількість математичних функцій для виконання арифметичних операцій, статистичних розрахунків, лінійної алгебри та роботи з випадковими величинами. NumPy також підтримує векторизовані обчислення, що дозволяє виконувати операції одночасно над усіма елементами масиву без використання циклів.

Під час аналізу даних NumPy часто використовується разом із іншими бібліотеками Python, оскільки забезпечує основу для виконання математичних операцій та підготовки даних до подальшої обробки. Багато інструментів машинного навчання та статистичного аналізу використовують масиви NumPy як основний формат зберігання числових даних.

У процесі реалізації системи підтримки прийняття рішень бібліотека NumPy використовувалась для виконання математичних обчислень, роботи з числовими масивами, обробки значень часових рядів та побудови лінійного прогнозування.

1.5.4 Бібліотека scikit-learn

Scikit-learn є однією з найбільш поширених бібліотек Python для реалізації методів машинного навчання та аналізу даних. Бібліотека містить набір готових алгоритмів для класифікації, регресії, кластеризації, масштабування даних та попередньої обробки інформації [7].

Однією з переваг scikit-learn є простота використання та єдина структура реалізації алгоритмів. Бібліотека дозволяє швидко інтегрувати методи машинного навчання у програмні системи та виконувати аналіз даних без необхідності самостійної реалізації складних математичних алгоритмів.

Scikit-learn підтримує інструменти попередньої обробки даних, що дозволяє підготувати інформацію до подальшого аналізу. Для роботи з числовими даними використовуються методи масштабування та нормалізації, які зменшують вплив різних діапазонів значень на результати роботи алгоритмів.

Для виконання кластерного аналізу у бібліотеці реалізовано алгоритм K-Means, який дозволяє автоматично групувати об'єкти за рівнем їх схожості. Робота алгоритму ґрунтується на визначенні центрів кластерів та мінімізації відстані між об'єктами і центрами відповідних груп [7].

Також бібліотека містить засоби для обробки пропущених значень у наборах даних. Використання методів заповнення пропусків дозволяє підвищити якість підготовки інформації та забезпечити коректну роботу алгоритмів аналізу даних.

Під час реалізації системи підтримки прийняття рішень бібліотека `scikit-learn` використовувалась для масштабування числових ознак, заповнення пропущених значень та виконання кластеризації табличних наборів даних за допомогою алгоритму K-Means.

1.5.5 Бібліотека Plotly

Plotly є сучасною бібліотекою для побудови інтерактивних графіків та візуалізації даних у середовищі Python. Бібліотека використовується для створення графічного представлення результатів аналізу даних та забезпечує можливість інтерактивної взаємодії користувача з побудованими візуалізаціями [10].

Однією з основних переваг Plotly є підтримка інтерактивних елементів, серед яких масштабування графіків, перегляд окремих значень, переміщення по графіку та керування відображенням окремих елементів візуалізації. Це дозволяє спростити аналіз даних та покращити сприйняття результатів дослідження.

Бібліотека підтримує створення різних типів графіків, зокрема лінійних графіків, гістограм, діаграм розсіювання та інших засобів візуалізації. Завдяки цьому Plotly широко використовується у системах аналітики даних та інформаційних системах підтримки прийняття рішень [10].

Під час аналізу часових рядів Plotly дозволяє відображати зміну показників у часі, результати згладжування даних та прогнозування. Для виявлення аномалій бібліотека використовується для побудови графіків, які дозволяють візуально визначати нетипові значення у наборах даних. У задачах кластеризації Plotly забезпечує графічне представлення результатів групування об'єктів та спрощує аналіз структури даних.

Використання інтерактивної візуалізації дозволяє покращити інтерпретацію результатів аналізу та забезпечує більш зручну взаємодію користувача із системою. Завдяки широким можливостям налаштування графіків бібліотека Plotly є ефективним засобом візуального представлення результатів аналізу даних у сучасних інформаційних системах.

1.5.6 Streamlit

Streamlit є фреймворком для створення інтерактивних веб застосунків засобами мови програмування Python [8]. Даний інструмент широко використовується у сфері аналізу даних та машинного навчання для швидкої розробки аналітичних систем і веб інтерфейсів без необхідності використання складних веб технологій.

Однією з основних переваг Streamlit є простота створення користувацького інтерфейсу. Фреймворк дозволяє реалізовувати елементи взаємодії з користувачем, серед яких кнопки, поля введення, меню вибору, повзунки та засоби завантаження файлів. Це значно спрощує процес створення інтерактивних систем аналізу даних.

Streamlit підтримує інтеграцію з бібліотеками Python для роботи з даними, машинного навчання та візуалізації [8]. Завдяки цьому користувач може виконувати аналіз інформації, переглядати результати обробки та взаємодіяти з графіками безпосередньо у веб інтерфейсі застосунку.

Фреймворк забезпечує автоматичне оновлення результатів після зміни параметрів аналізу, що дозволяє оперативно виконувати дослідження наборів даних та оцінювати результати роботи алгоритмів. Такий підхід є зручним під час роботи із системами підтримки прийняття рішень та аналітичними платформами.

У процесі реалізації системи підтримки прийняття рішень Streamlit використовувався для створення веб інтерфейсу користувача, організації взаємодії із системою, завантаження табличних наборів даних, налаштування параметрів аналізу та відображення результатів обробки інформації.

1.5.7 Використання Gemini API для генерації аналітичних висновків

У сучасних системах аналізу даних все більшого поширення набуває використання технологій штучного інтелекту для автоматизації інтерпретації результатів аналізу. Одним із підходів до реалізації таких можливостей є використання API моделей штучного інтелекту, які дозволяють формувати текстові пояснення та аналітичні висновки на основі отриманих результатів обробки даних.

API (Application Programming Interface) являє собою програмний інтерфейс, який забезпечує взаємодію між різними програмними системами [9]. Використання API дозволяє інтегрувати зовнішні сервіси штучного інтелекту у власні програмні застосунки та автоматизувати окремі процеси аналізу інформації.

У розробленій системі підтримки прийняття рішень використовувався Gemini API для генерації текстових пояснень результатів аналізу даних. Після виконання аналізу часових рядів, виявлення аномалій або кластеризації система формує структуровану інформацію про результати дослідження та передає її до моделі штучного інтелекту для подальшої генерації текстового висновку.

Використання Gemini API дозволяє автоматизувати формування коротких аналітичних пояснень для користувача, спрощує інтерпретацію результатів аналізу та покращує взаємодію із системою. Згенеровані висновки

можуть містити інформацію про основні закономірності, тенденції зміни показників та особливості структури даних.

Перевагою використання API моделей штучного інтелекту є можливість швидкої інтеграції сучасних інтелектуальних технологій у програмні системи без необхідності самостійного навчання моделей.

Висновки до розділу 1

У першому розділі було розглянуто теоретичні основи аналізу табличних наборів даних та сучасні підходи до обробки інформації у системах підтримки прийняття рішень. Проведено аналіз особливостей роботи з табличними даними, основних етапів їх підготовки та обробки, а також значення аналізу даних у сучасних інформаційних системах.

У процесі дослідження було розглянуто методи аналізу часових рядів, зокрема метод простого ковзного середнього, метод експоненційного ковзного середнього та лінійне прогнозування. Досліджено принципи їх роботи, основні переваги та особливості використання під час аналізу зміни показників у часі.

Також було проаналізовано методи виявлення аномалій у даних. Особливу увагу приділено методу Z-score, який дозволяє визначати нетипові значення на основі оцінки відхилення даних від середнього значення вибірки.

У межах дослідження методів кластеризації було розглянуто алгоритм K-Means та метод ліктя для визначення оптимальної кількості кластерів. Проведений аналіз показав ефективність використання кластерного аналізу для групування об'єктів за рівнем їх схожості та дослідження структури наборів даних.

Крім того, було досліджено програмні засоби, використані під час реалізації системи підтримки прийняття рішень. Розглянуто можливості мови

програмування Python, бібліотек pandas, NumPy, scikit-learn, Plotly та Streamlit, а також використання Gemini API для автоматизованої генерації аналітичних висновків.

Отримані теоретичні відомості та результати дослідження є основою для подальшої реалізації системи аналізу табличних наборів даних та проведення практичного дослідження результатів роботи розробленої системи.

РОЗДІЛ 2 РЕАЛІЗАЦІЯ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

2.1 Загальна характеристика розробленої системи

У межах виконання роботи було реалізовано систему підтримки прийняття рішень для аналізу табличних наборів даних. Основним призначенням системи є автоматизація процесу обробки, аналізу та візуалізації даних, а також спрощення інтерпретації результатів аналізу для користувача.

Розроблена система дозволяє виконувати аналіз табличних наборів даних без необхідності використання складних спеціалізованих програмних засобів або ручної реалізації алгоритмів аналізу. Користувач має можливість завантажувати власні набори даних та виконувати дослідження інформації за допомогою реалізованих інструментів аналізу.

Система підтримує роботу з файлами у форматах CSV та Excel, що дозволяє використовувати табличні набори даних із різних джерел інформації. Після завантаження даних користувач може виконувати їх попередню обробку, змінювати типи даних, працювати з часовими рядами та здійснювати аналіз окремих параметрів набору даних.

Основний функціонал системи включає:

- 1) завантаження та обробку табличних наборів даних;
- 2) аналіз часових рядів;
- 3) згладжування даних за допомогою методів SMA та EMA;
- 4) лінійне прогнозування часових рядів;
- 5) виявлення аномалій методом Z-score;
- 6) кластеризацію даних за допомогою алгоритму K-Means;
- 7) визначення оптимальної кількості кластерів методом ліктя;

- 8) візуалізацію результатів аналізу;
- 9) генерацію аналітичних висновків за допомогою Gemini API.

Для взаємодії користувача із системою було реалізовано веб інтерфейс, який забезпечує зручне налаштування параметрів аналізу та відображення результатів обробки даних (рисунок 2.1).

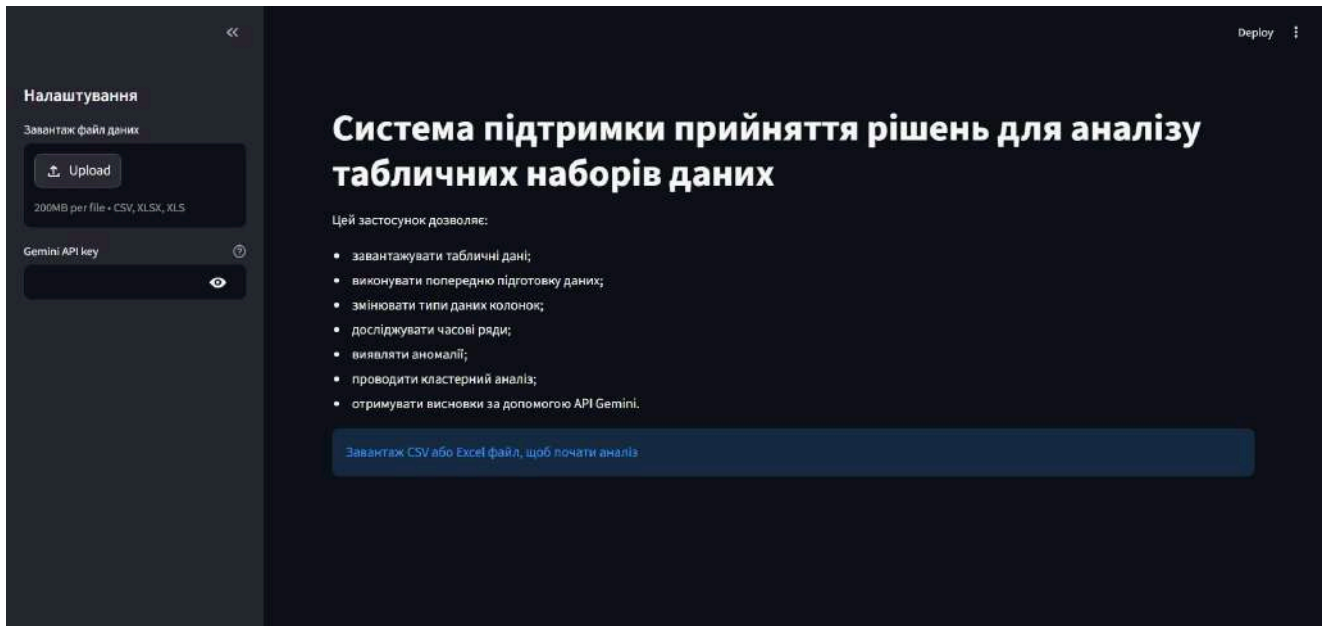


Рисунок 2.1–Головний інтерфейс системи підтримки прийняття рішень

Система дозволяє завантажувати датасети, будувати графіки часових рядів, відображати результати кластеризації та виявлення аномалій, а також отримувати автоматично сформовані текстові пояснення результатів аналізу.

Розроблена система підтримки прийняття рішень дозволяє автоматизувати процес аналізу табличних наборів даних та спрощує дослідження інформації за допомогою сучасних методів аналізу даних і машинного навчання.

2.2 Реалізація підготовки даних

Одним із важливих етапів роботи системи підтримки прийняття рішень є підготовка табличних наборів даних до подальшого аналізу. Якість попередньої обробки безпосередньо впливає на точність результатів аналізу, ефективність роботи алгоритмів та коректність побудови моделей.

У розробленій системі реалізовано функціонал завантаження табличних наборів даних у форматах CSV та Excel. Після завантаження файлу система автоматично виконує зчитування структури таблиці та відображає основну інформацію про набір даних для подальшої роботи користувача.

Після завантаження даних користувач має можливість переглядати структуру таблиці, типи даних колонок та основні характеристики набору даних. Це дозволяє оцінити коректність зчитування інформації та підготувати дані до подальшого аналізу (рисунок 2.2).

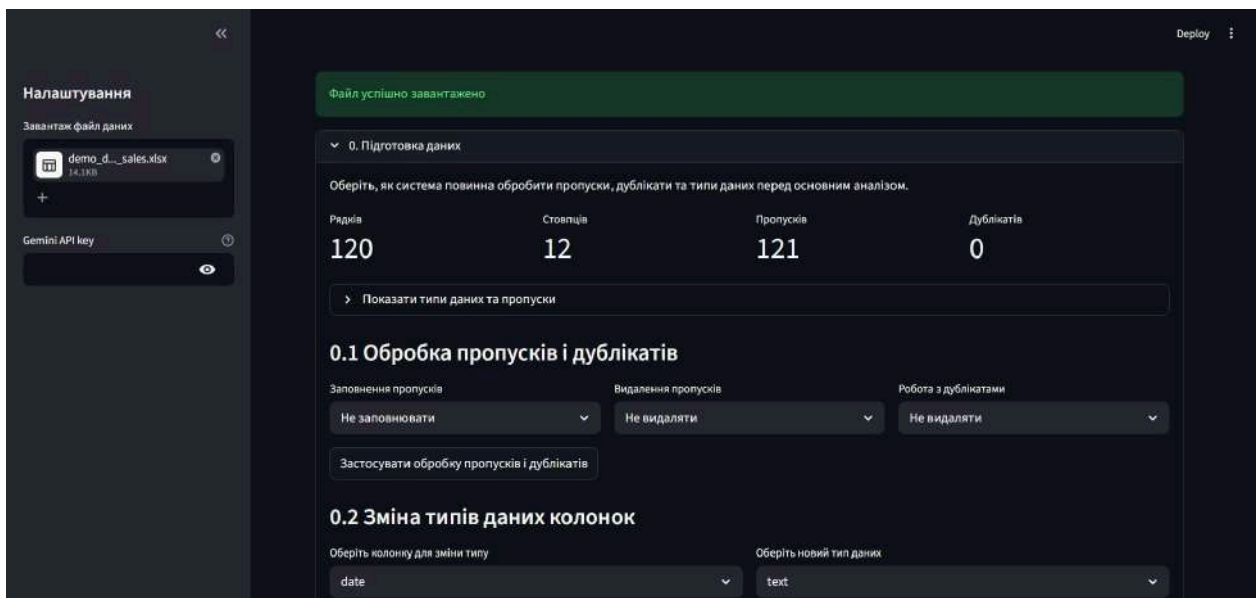


Рисунок 2.2 – Завантаження та первинний перегляд табличного набору даних

У системі реалізовано механізми попередньої обробки даних, які дозволяють виконувати очищення та підготовку інформації перед проведенням аналізу. Одним із важливих елементів є робота з пропущеними

значеннями. Для цього користувач може обирати способи обробки пропусків залежно від особливостей набору даних та поставленої задачі.

Наприклад у цьому випадку було обрано заповнити пропущені значення найчастішим значенням. У результаті кількість пропусків упала до 0 (рисунок 2.3).

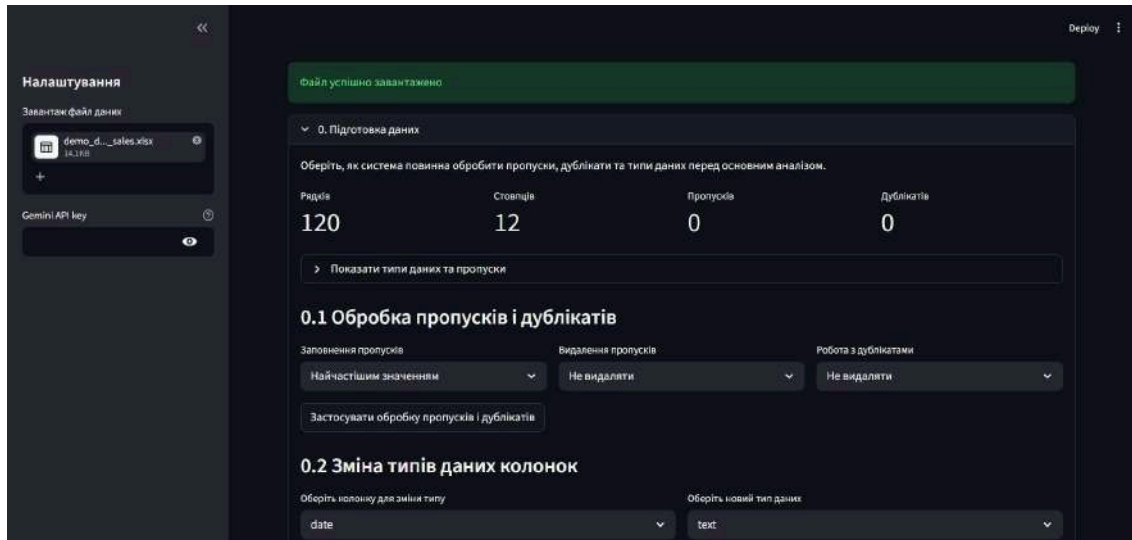


Рисунок 2.3 – Реалізація заповнення пропусків

Також система підтримує зміну типів даних колонок, що дозволяє коректно працювати з числовими, текстовими та часовими значеннями. Використання правильних типів даних є важливим під час аналізу часових рядів, кластеризації та статистичної обробки інформації.

Для забезпечення коректності аналізу у системі реалізовано перевірку дублікатів та засоби очищення даних. Це дозволяє зменшити вплив помилкових або повторюваних записів на результати роботи алгоритмів аналізу даних.

Реалізовані засоби підготовки даних дозволяють автоматизувати процес очищення та структурування інформації, що підвищує ефективність подальшого аналізу табличних наборів даних та забезпечує коректну роботу реалізованих методів дослідження.

Також після обробки датасету є можливість подивитися на вже готовий датасет з можливістю його завантаження і повернення до початкового вигляду (рисунок 2.4).

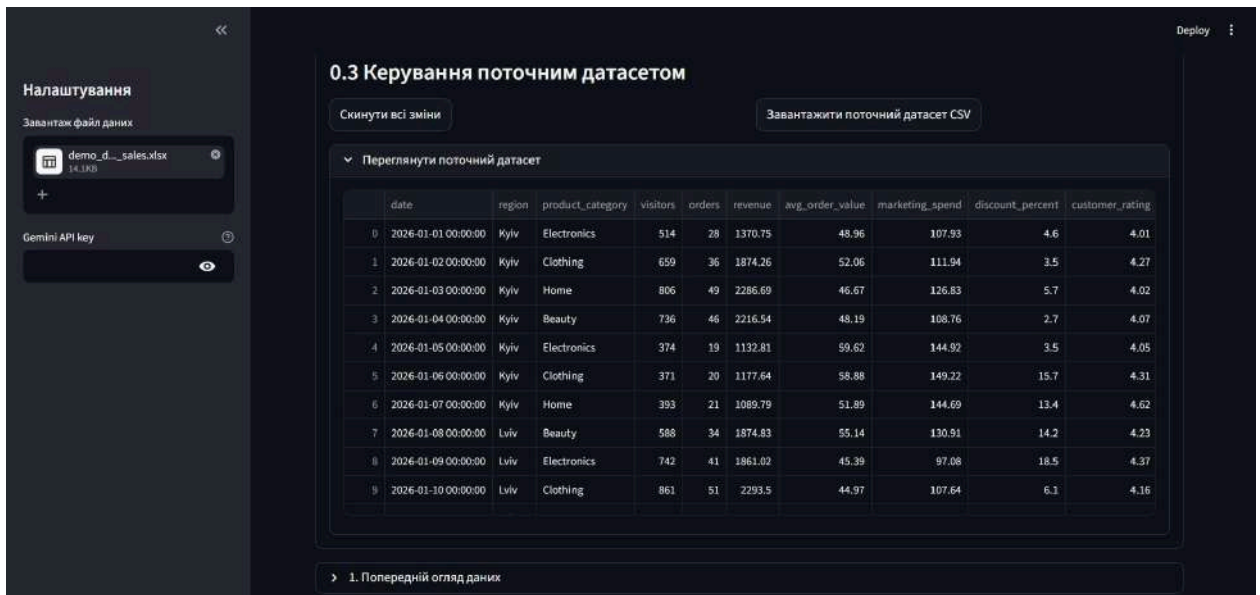


Рисунок 2.4 – Керування поточним датасетом

2.3 Реалізація аналізу часових рядів

Одним із функціональних модулів розробленої системи є аналіз часових рядів, який дозволяє досліджувати зміну показників у часі, виявляти тенденції та виконувати прогнозування майбутніх значень. Для проведення аналізу користувач має можливість обирати колонку з датою та числовий параметр, який буде використовуватись для побудови часового ряду. Також у нього є можливість обрати спосіб агрегування даних та вибрати часовий проміжок.

Після вибору необхідних параметрів система автоматично формує часовий ряд та відображає його у вигляді інтерактивного графіка. Це дозволяє оцінити загальну динаміку зміни показників, виявити можливі

тенденції та проаналізувати поведінку даних протягом визначеного періоду часу (рисунк 2.5).



Рисунок 2.5 – Побудова часового ряду на основі табличного набору даних

Для згладжування випадкових коливань у системі реалізовано метод простого ковзного середнього (SMA). Використання даного методу дозволяє зменшити вплив шуму у даних та більш чітко відобразити основну тенденцію зміни показників. Користувач має можливість змінювати розмір вікна згладжування, що дозволяє налаштовувати рівень деталізації аналізу (рисунк 2.6).



Рисунок 2.6 – Результат застосування методу SMA до часового ряду

Крім того, у системі реалізовано метод експоненційного ковзного середнього (ЕМА), який забезпечує швидшу реакцію на зміну значень часового ряду порівняно з методом SMA. Використання ЕМА дозволяє ефективніше відстежувати актуальні зміни показників та аналізувати короткострокові тенденції (рисунок 2.7).



Рисунок 2.7 – Результат застосування методу ЕМА до часового ряду

Для прогнозування подальшої зміни показників у системі реалізовано лінійне прогнозування. Після побудови моделі система виконує прогнозування значень на визначену кількість кроків уперед та відображає результати у вигляді графіка (рисунк 2.8).



Рисунок 2.8 – Лінійне прогнозування часового ряду

Також користувачу надане коротке пояснення про те, як саме працює аналіз часових рядів. У кінці ж він має можливість завантажити результати прогнозу у вигляді файлу csv (рисунк 2.9).

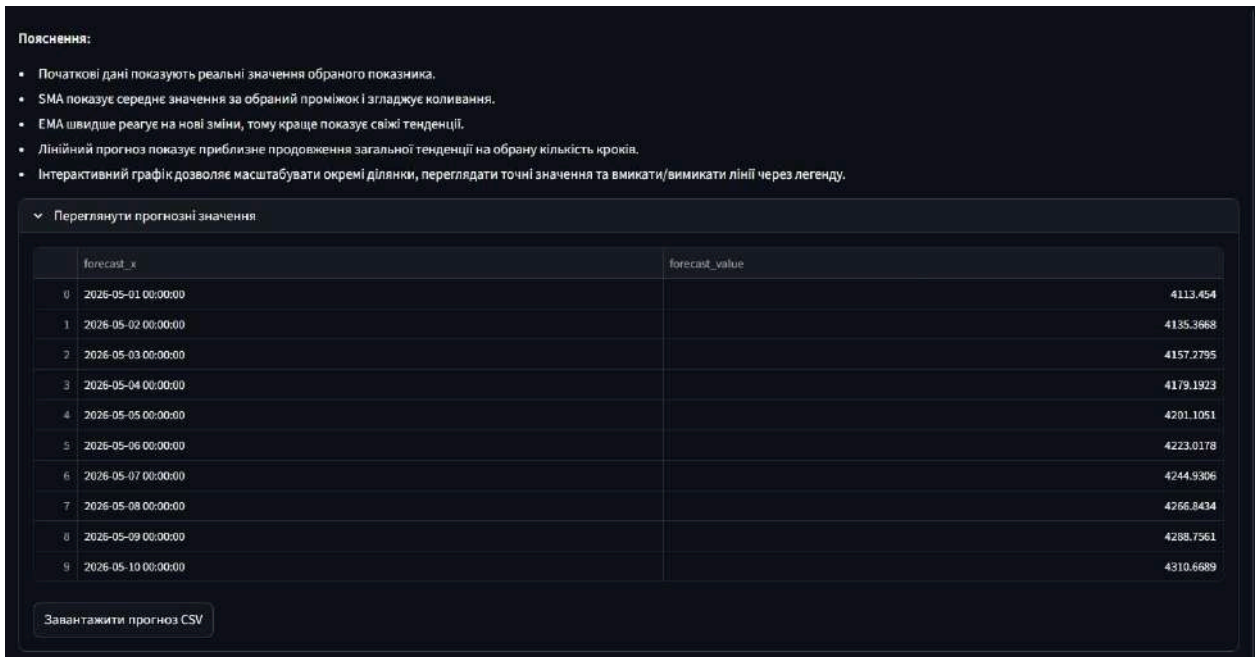


Рисунок 2.9 – Пояснення та завантаження прогнозу

У процесі дослідження часових рядів було встановлено, що використання методів згладжування та прогнозування дозволяє покращити аналіз структури даних, виявити основні тенденції зміни показників та спростити інтерпретацію результатів аналізу для користувача.

2.4 Реалізація виявлення аномалій

У процесі аналізу табличних наборів даних важливим етапом є виявлення аномальних значень, які можуть свідчити про помилки у даних, нестандартну поведінку показників або різкі зміни досліджуваних процесів. Для реалізації даного функціоналу у системі підтримки прийняття рішень використовується метод *Z-score*, який дозволяє оцінювати ступінь відхилення окремих значень від середнього значення вибірки.

Для проведення аналізу користувач обирає числову колонку набору даних та задає порогове значення для визначення аномалій. Після цього

система виконує обчислення стандартизованих значень та автоматично визначає спостереження, які перевищують встановлений поріг (рисунок 2.10).

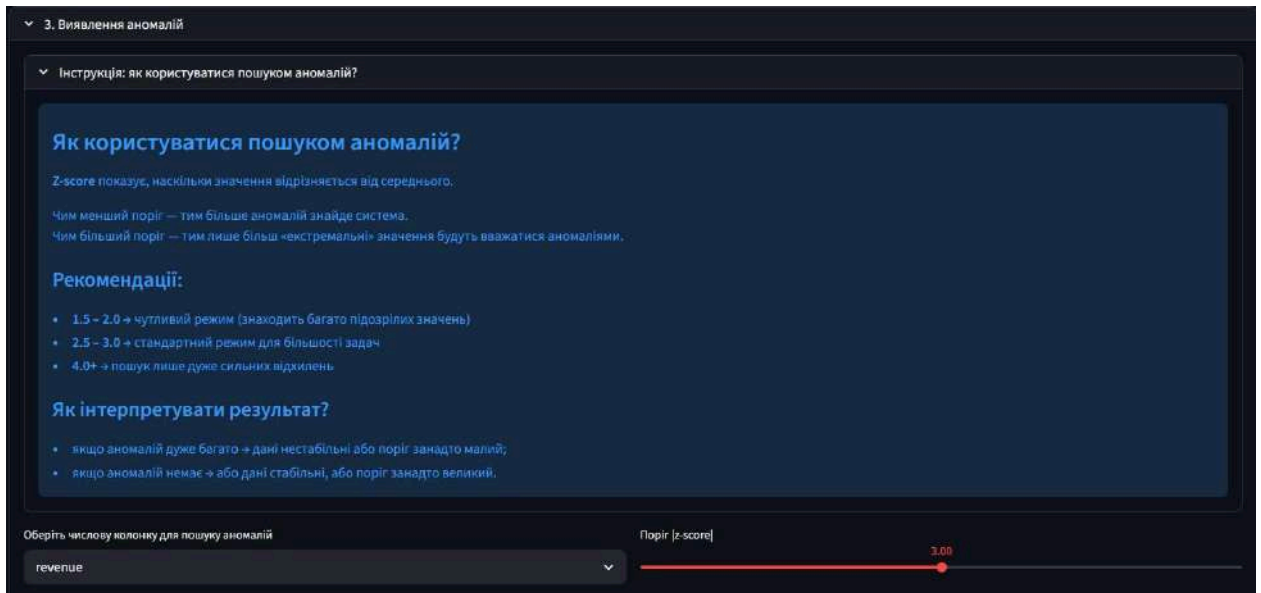


Рисунок 2.10 – Налаштування параметрів виявлення аномалій

Результати аналізу відображаються у вигляді таблиці та інтерактивного графіка, на якому аномальні значення виділяються окремим кольором. Такий підхід дозволяє швидко виявляти нетипові значення та оцінювати їх вплив на загальну структуру даних (рисунок 2.11).



Рисунок 2.11 – Результат виявлення аномальних значень методом Z-score

Під час дослідження тестового набору даних було виявлено декілька значень, які суттєво відрізнялися від основної частини вибірки. Зокрема, аномалії спостерігались у колонці доходу, що може свідчити про різкі зміни активності користувачів або нетипові умови роботи системи.

Використання методу Z-score дозволило ефективно автоматизувати процес пошуку аномальних значень та спростити аналіз структури набору даних. Реалізований функціонал забезпечує швидке виявлення нетипових спостережень та покращує якість подальшого аналізу інформації.

2.5 Реалізація кластерного аналізу

Одним із функціональних модулів розробленої системи є кластерний аналіз, який використовується для автоматичного групування об'єктів за рівнем їх схожості. Реалізація кластеризації дозволяє досліджувати структуру набору даних, виявляти групи подібних об'єктів та спрощувати інтерпретацію результатів аналізу.

Для виконання кластеризації у системі використовується алгоритм K-Means. Перед початком аналізу користувач має можливість обрати числові параметри, які будуть використовуватись для формування кластерів. Після вибору ознак система виконує попередню обробку даних, масштабування значень та побудову кластерної моделі (рисунок 2.12).

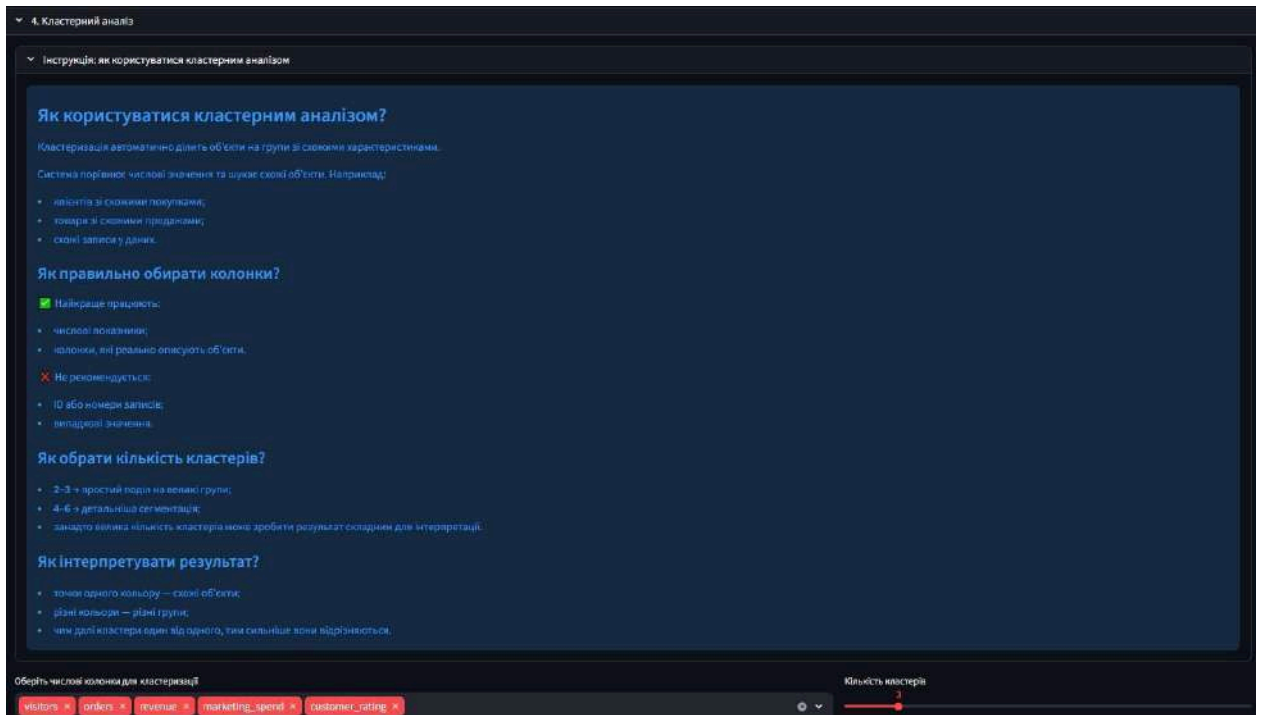


Рисунок 2.12 – Вибір параметрів для кластерного аналізу

Для визначення оптимальної кількості кластерів у системі реалізовано метод ліктя. Під час роботи алгоритму виконується послідовне обчислення значення внутрішньокластерної суми квадратів відстаней для різної кількості кластерів. Результати відображаються у вигляді графіка, який дозволяє визначити оптимальне значення параметра K (рисунок 2.13).



Рисунок 2.13 – Результат застосування методу ліктя

Після визначення кількості кластерів, у нашому випадку це 4, ми обираємо відповідне значення і система виконує кластеризацію набору даних та відображає результати у вигляді інтерактивного графіка. Кожний кластер позначається окремим кольором, що дозволяє візуально оцінити структуру даних та рівень подібності між об'єктами (рисунок 2.14).



Рисунок 2.14 – Результат кластеризації даних методом K-Means

У процесі дослідження тестового набору даних було сформовано чотири кластери, які характеризуються різними показниками активності користувачів, кількості замовлень, доходу та витрат на маркетинг. Отримані результати дозволили виділити групи з низьким, середнім та високим рівнем ефективності.

Найбільш результативним виявився кластер 3, для якого характерні найвищі показники кількості відвідувачів, замовлень та доходу. Водночас даний кластер характеризується значними витратами на маркетинг, що свідчить про високу активність системи просування.

Кластер 0 також демонструє високі показники доходу та кількості замовлень, однак при значно менших витратах на маркетинг порівняно з кластером 3. Це може свідчити про більш ефективне використання маркетингових ресурсів.

Кластери 1 та 2 характеризуються нижчим рівнем активності користувачів та меншою кількістю замовлень. Для даних груп спостерігаються нижчі показники доходу, що відповідає меншій кількості відвідувачів та загальній активності користувачів.

Використання кластерного аналізу дозволило автоматизувати процес сегментації даних та спростити дослідження структури набору даних. Реалізований функціонал забезпечує зручний інструмент для аналізу взаємозв'язків між показниками та виявлення груп об'єктів із подібними характеристиками.

2.6 Генерація AI-висновків

Однією з особливостей розробленої системи підтримки прийняття рішень є можливість автоматичної генерації текстових аналітичних висновків за допомогою Gemini API. Реалізований функціонал дозволяє формувати текстові пояснення результатів аналізу даних та спрощує інтерпретацію отриманої інформації для користувача.

Для використання функції генерації AI-висновків користувачу необхідно ввести власний API-ключ Gemini у відповідне поле веб інтерфейсу системи. Після введення ключа система отримує можливість взаємодії з API моделі штучного інтелекту та виконання генерації текстових пояснень результатів аналізу (рисунок 2.15).

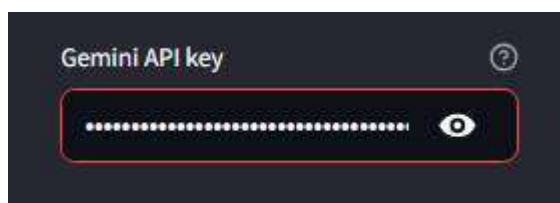


Рисунок 2.15 – Поле введення Gemini API key у веб інтерфейсі системи

Після виконання аналізу часових рядів, виявлення аномалій або кластеризації система формує структуровану інформацію про результати дослідження та передає її до моделі штучного інтелекту. На основі отриманих даних API генерує текстовий висновок, який містить опис основних

закономірностей, тенденцій та особливостей структури набору даних (рисунок 2.16).

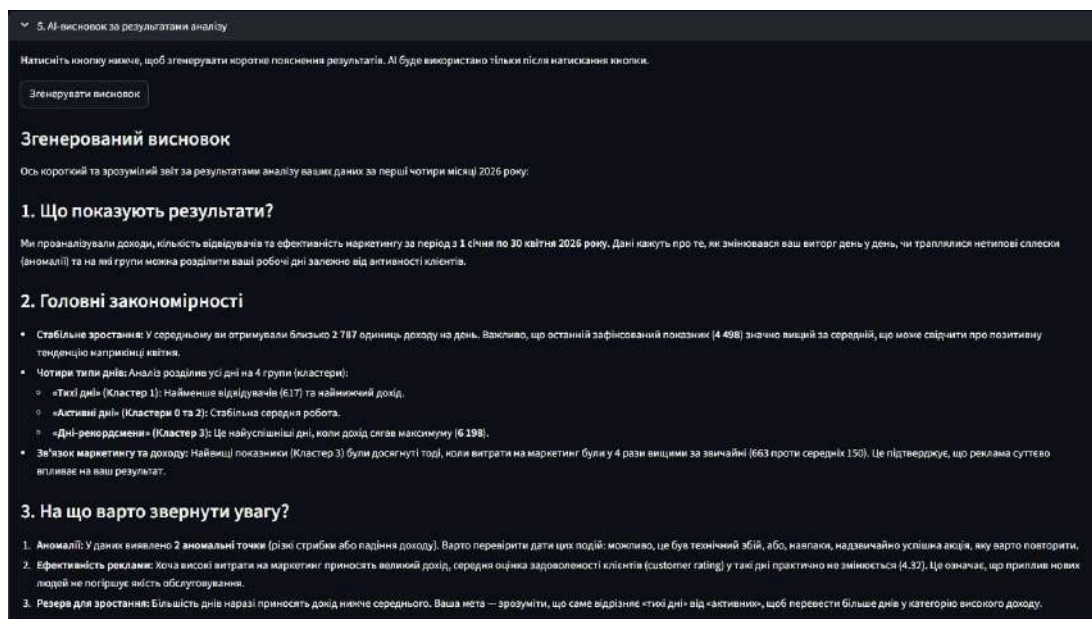


Рисунок 2.16 – Генерація аналітичного висновку за допомогою Gemini API

У процесі тестування системи було встановлено, що використання AI-висновків дозволяє спростити аналіз результатів та зробити інтерпретацію даних більш зрозумілою для користувача. Згенеровані пояснення містили інформацію про динаміку зміни показників, виявлені аномалії та результати кластеризації набору даних.

Під час аналізу тестового набору даних система сформувала висновки щодо загальної тенденції зміни кількості замовлень, стабільності доходу та особливостей окремих кластерів. Зокрема, було визначено групи з високим рівнем доходу та активності користувачів, а також виявлено аномальні значення, які суттєво відрізнялися від основної частини вибірки.

Використання Gemini API дозволило автоматизувати процес формування аналітичних пояснень та підвищити зручність роботи із системою. Реалізований функціонал забезпечує можливість швидкого отримання текстової інтерпретації результатів аналізу без необхідності самостійного детального дослідження усіх отриманих показників.

Висновки до розділу 2

У другому розділі було розглянуто процес реалізації системи підтримки прийняття рішень для аналізу табличних наборів даних та проведено аналіз результатів роботи реалізованих функціональних модулів. Реалізована система забезпечує можливість завантаження табличних наборів даних, виконання попередньої обробки інформації, аналізу часових рядів, виявлення аномалій, кластеризації та генерації аналітичних висновків.

У процесі дослідження було реалізовано функціонал підготовки даних, який включає роботу з пропущеними значеннями, зміну типів даних, очищення інформації та формування часових рядів. Реалізовані механізми підготовки даних дозволяють забезпечити коректну роботу алгоритмів аналізу та підвищити якість отриманих результатів.

Під час аналізу часових рядів було досліджено роботу методів SMA та ЕМА, а також реалізовано функціонал лінійного прогнозування. Використання методів згладжування дозволило зменшити вплив випадкових коливань та більш чітко визначити основні тенденції зміни показників у часі.

Для виявлення аномалій у системі було реалізовано метод Z-score, який дозволив автоматизувати процес пошуку нетипових значень у наборах даних. Проведений аналіз показав ефективність використання даного методу для визначення різких відхилень показників від основної структури вибірки.

У межах кластерного аналізу було реалізовано алгоритм K-Means та метод ліктя для визначення оптимальної кількості кластерів. Результати кластеризації дозволили виділити групи об'єктів із різними характеристиками та дослідити структуру набору даних.

Окрему увагу було приділено реалізації генерації AI-висновків за допомогою Gemini API. Реалізований функціонал забезпечує автоматичне формування текстових пояснень результатів аналізу та спрощує інтерпретацію отриманої інформації для користувача.

РОЗДІЛ 3 ПРОЄКТУВАННЯ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

3.1 Архітектура системи

Архітектура системи підтримки прийняття рішень визначає структуру взаємодії основних програмних компонентів, модулів аналізу даних та засобів взаємодії користувача із системою [13]. Розроблена система побудована за модульним принципом, що забезпечує гнучкість, зручність розширення функціоналу та спрощує процес обробки табличних наборів даних (рисунок 3.1).

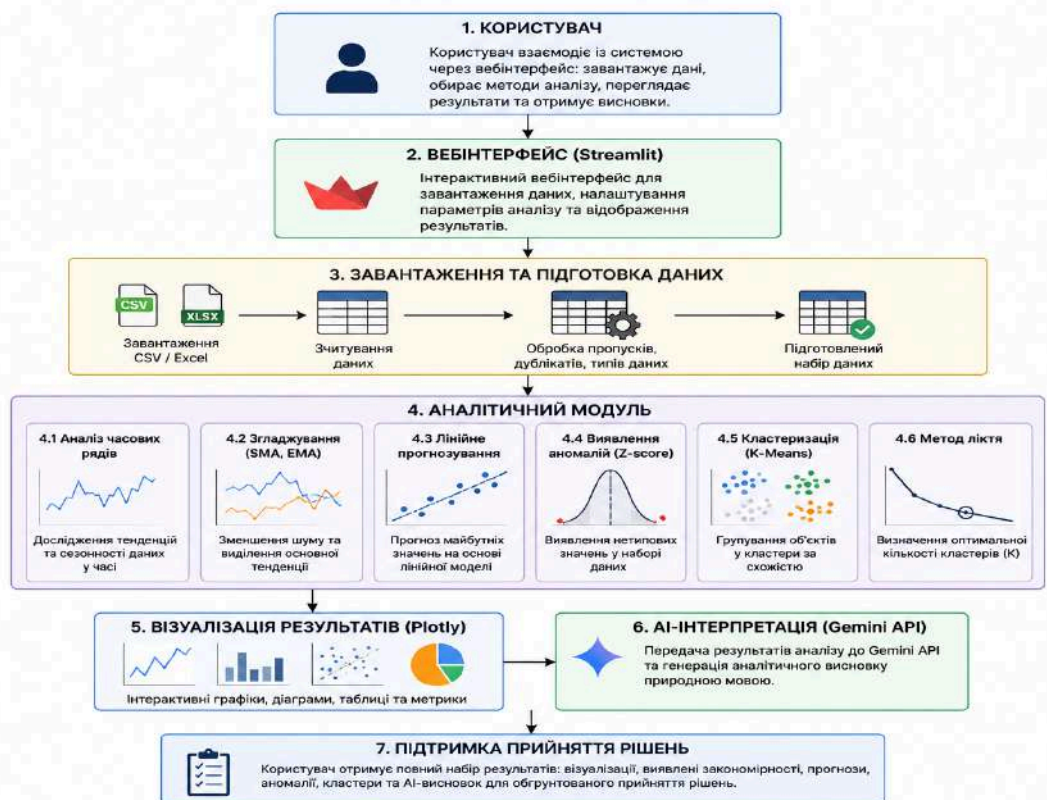


Рисунок 3.1 – Архітектура системи підтримки прийняття рішень для аналізу табличних наборів даних

Основною метою розробленої архітектури є забезпечення повного циклу аналізу даних, який включає завантаження набору даних, його

попередню обробку, виконання аналітичних методів, візуалізацію результатів та генерацію текстових висновків за допомогою технологій штучного інтелекту.

Система взаємодіє з користувачем через веб інтерфейс, реалізований засобами Streamlit. За допомогою інтерфейсу користувач має можливість завантажувати файли у форматах CSV та Excel, обирати параметри аналізу, налаштовувати методи дослідження та переглядати результати роботи системи.

Після завантаження файлу система виконує попередню обробку даних, яка включає перевірку структури таблиці, зміну типів даних, обробку пропущених значень та усунення дублікатів. Підготовлені дані передаються до аналітичного модуля системи. Аналітичний модуль є основною складовою архітектури СППР та забезпечує виконання методів аналізу даних.

Результати роботи аналітичного модуля передаються до модуля візуалізації, де за допомогою бібліотеки Plotly формуються інтерактивні графіки, діаграми та таблиці. Використання інтерактивної візуалізації дозволяє спростити процес аналізу результатів та покращує взаємодію користувача із системою.

Окремим компонентом архітектури є модуль AI-інтерпретації результатів, реалізований за допомогою Gemini API. Даний модуль отримує результати аналізу та формує текстові аналітичні висновки, які допомагають користувачу швидше інтерпретувати отримані результати та приймати обґрунтовані рішення.

Використання модульної архітектури забезпечує можливість подальшого розширення функціоналу системи, інтеграції нових методів аналізу даних та вдосконалення засобів підтримки прийняття рішень.

3.2 Переваги реалізованої архітектури

Реалізована архітектура системи підтримки прийняття рішень забезпечує ефективну організацію процесу аналізу табличних наборів даних та дозволяє поєднати методи обробки інформації, візуалізації результатів і технології штучного інтелекту в межах єдиної програмної системи.

Однією з основних переваг розробленої архітектури є її модульна структура. Використання окремих функціональних модулів дозволяє незалежно реалізовувати та вдосконалювати різні компоненти системи без необхідності зміни всієї структури програмного забезпечення.

До основних переваг реалізованої архітектури належать:

- 1) модульна структура системи;
- 2) можливість масштабування функціоналу;
- 3) зручна взаємодія користувача із системою;
- 4) автоматизація аналізу табличних наборів даних;
- 5) інтерактивна візуалізація результатів аналізу;
- 6) інтеграція технологій штучного інтелекту;
- 7) можливість додавання нових методів аналізу даних.

Модульний підхід дозволяє розширювати функціональні можливості системи шляхом додавання нових алгоритмів аналізу, методів прогнозування або засобів візуалізації без суттєвих змін основної структури СППР. Це забезпечує гнучкість системи та спрощує її подальший розвиток [13].

Важливою перевагою реалізованої архітектури є використання веб інтерфейсу, створеного засобами Streamlit. Завдяки цьому користувач може працювати із системою без необхідності використання складних програмних засобів або спеціальних технічних знань.

Інтеграція бібліотеки Plotly забезпечує побудову інтерактивних графіків та діаграм, що дозволяє спростити аналіз результатів та покращити візуальне сприйняття інформації. Використання інтерактивної візуалізації підвищує

ефективність роботи користувача із системою та полегшує процес аналізу даних.

Окрему увагу в архітектурі системи приділено інтеграції Gemini API, який забезпечує автоматичну генерацію текстових аналітичних висновків. Використання технологій штучного інтелекту дозволяє автоматизувати процес інтерпретації результатів аналізу та підвищує рівень підтримки прийняття рішень.

Реалізована архітектура забезпечує ефективну взаємодію між усіма компонентами системи та створює основу для подальшого вдосконалення СППР і розширення її функціональних можливостей.

Висновки до розділу 3

У третьому розділі було розглянуто архітектуру системи підтримки прийняття рішень для аналізу табличних наборів даних та досліджено особливості організації взаємодії основних компонентів системи. У процесі проєктування СППР було реалізовано модульну структуру, яка забезпечує ефективну обробку даних, виконання аналітичних методів, візуалізацію результатів та генерацію текстових аналітичних висновків.

Було встановлено, що використання модульного підходу дозволяє спростити процес розробки та забезпечує можливість подальшого розширення функціональних можливостей системи. Реалізована архітектура забезпечує зручну взаємодію користувача із системою, автоматизацію аналізу табличних наборів даних та інтеграцію сучасних засобів візуалізації і технологій штучного інтелекту.

У межах архітектури системи реалізовано окремі модулі підготовки даних, аналізу часових рядів, виявлення аномалій, кластеризації та AI-інтерпретації результатів. Взаємодія між модулями забезпечує послідовне

виконання всіх етапів аналізу інформації та формування аналітичних результатів для підтримки прийняття рішень.

Таким чином, реалізована архітектура системи підтримки прийняття рішень забезпечує ефективну організацію процесу аналізу табличних наборів даних та створює основу для подальшого вдосконалення функціональних можливостей розробленої СППР.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ

У даному розділі проводиться функціонально-вартісний аналіз системи підтримки прийняття рішень для аналізу табличних наборів даних, розробленої у межах дипломної роботи. Основною метою аналізу є оцінювання ефективності реалізованого програмного продукту з точки зору його функціональних можливостей, зручності використання та витрат, пов'язаних із процесом розробки та експлуатації системи.

У процесі дослідження розглядаються різні варіанти реалізації програмного продукту, які відрізняються рівнем функціональності, способом взаємодії користувача із системою та можливостями аналізу даних. Зокрема, оцінюються варіанти використання локального програмного застосунку, веб орієнтованої системи з інтерактивною візуалізацією, а також системи з інтеграцією технологій штучного інтелекту для автоматичної генерації аналітичних висновків.

Для визначення найбільш ефективного варіанта реалізації програмного продукту використовується апарат функціонально-вартісного аналізу. Даний підхід дозволяє оцінити співвідношення між функціональними можливостями системи та витратами, необхідними для її створення і використання.

Функціонально-вартісний аналіз є методом комплексного оцінювання програмного продукту, який базується на дослідженні основних функцій системи, параметрів якості та економічних характеристик реалізації.

Використання ФВА дозволяє визначити найбільш доцільний варіант програмного продукту, який забезпечує необхідний рівень функціональності при мінімально можливих витрат.

У межах функціонально-вартісного аналізу досліджуються основні функції системи підтримки прийняття рішень, серед яких завантаження та

обробка табличних даних, аналіз часових рядів, виявлення аномалій, кластеризація даних, інтерактивна візуалізація результатів та генерація текстових висновків за допомогою Gemini API.

Для проведення аналізу використовуються технічні характеристики програмного продукту, оцінки функціональних параметрів, показники якості реалізації окремих модулів системи, а також економічні показники, пов'язані з витратами на розробку програмного забезпечення.

Алгоритм проведення функціонально-вартісного аналізу включає визначення основних функцій програмного продукту, формування системи параметрів оцінювання, аналіз варіантів реалізації системи, оцінювання рівня якості програмного продукту, розрахунок витрат на розробку та визначення найбільш ефективного варіанта реалізації СППР.

4.1 Постановка задачі проектування

У межах дипломної роботи розробляється система підтримки прийняття рішень для аналізу табличних наборів даних, основним призначенням якої є автоматизація процесу обробки, аналізу та інтерпретації інформації. Необхідність створення подібних систем обумовлена постійним зростанням обсягів даних та потребою у швидкому отриманні аналітичних результатів для підтримки прийняття рішень.

Основною задачею проектування є створення програмного продукту, який забезпечує зручне завантаження табличних наборів даних, попередню обробку інформації, виконання аналітичних методів та візуалізацію результатів аналізу у зрозумілому для користувача вигляді.

Під час проектування системи також необхідно враховувати вимоги до зручності користувацького інтерфейсу, швидкодії системи, гнучкості

налаштування параметрів аналізу та можливості подальшого розширення функціональних можливостей програмного продукту.

У межах функціонально-вартісного аналізу розглядаються декілька варіантів реалізації системи підтримки прийняття рішень, які відрізняються рівнем функціональності та складністю реалізації. Аналіз варіантів дозволяє визначити найбільш ефективний підхід до розробки програмного продукту з точки зору співвідношення між функціональними можливостями системи та витратами на її створення і використання.

4.2 Обґрунтування функцій програмного продукту

Головна функція F0 – розробка системи підтримки прийняття рішень для аналізу табличних наборів даних, яка забезпечує автоматизацію процесу обробки, аналізу та інтерпретації інформації. Беручи за основу дану функцію, можна виділити наступні:

- F1 – вибір середовища розробки програмного продукту;
- F2 – вибір мови програмування програмного продукту;
- F3 – реалізація користувацького інтерфейсу системи;
- F4 – реалізація модуля AI-генерації аналітичних висновків.

Кожна з цих функцій має декілька варіантів реалізації (таблиця 4.1):

Таблиця 4.1 – Варіанти реалізацій функцій програмного продукту.

F1	F2	F3	F4
Visual Studio Code	Python	Консольний інтерфейс	Використання Gemini API
PyCharm	Java	Вебінтерфейс Streamlit	Використання локальної мовної моделі
Visual Studio	C#	Desktop-застосунок	

Варіанти реалізації функцій продемонстровано на морфологічній карті системи (рисунок 4.1). Морфологічна карта відображає множину всіх можливих варіантів основних функцій.

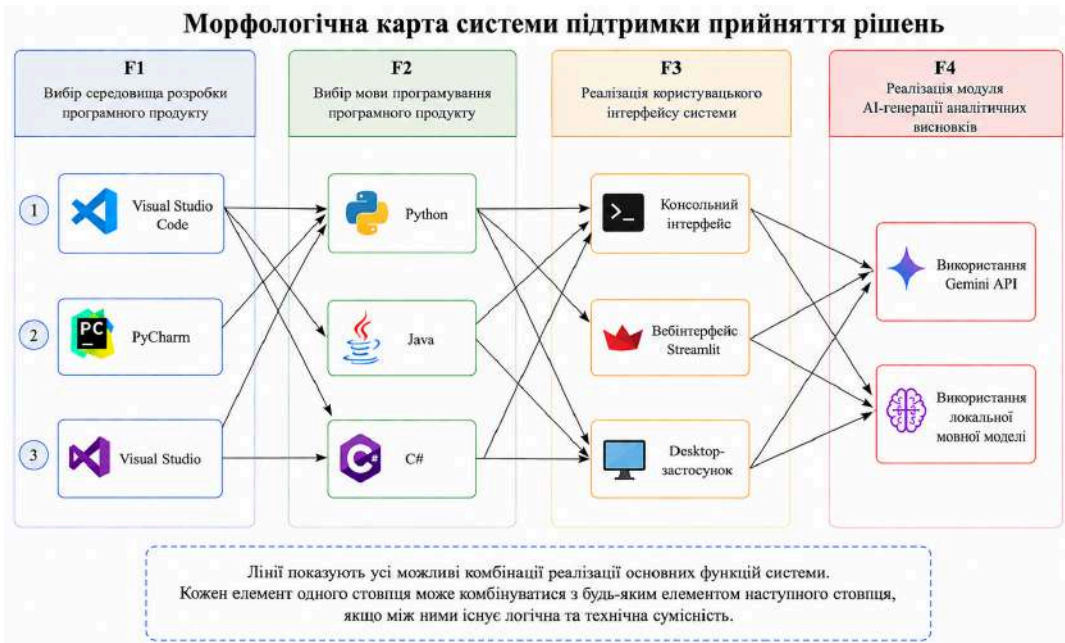


Рисунок 4.1 – Морфологічна карта
Також побудовано позитивно-негативну матрицю (таблиця 4.2):
Таблиця 4.2 – Позитивно-негативна матриця

Функція	Варіант реалізації	Переваги	Недоліки
F1	А	Безкоштовне середовище, підтримка багатьох мов програмування, велика кількість розширень	Потребує додаткового налаштування
	Б	Зручна робота з Python, вбудовані засоби аналізу коду	Частина функціоналу доступна лише у платній версії
	В	Потужні засоби розробки для C#	Великий обсяг системних ресурсів

Продовження таблиці 4.2

F2	A	Простота використання, велика кількість бібліотек для аналізу даних	Нижча швидкодія порівняно з компільованими мовами
	Б	Висока продуктивність та кросплатформність	Складніший синтаксис
	В	Зручна інтеграція з desktop-застосунками	Менша кількість бібліотек для Data Science
F3	A	Простота реалізації, низькі вимоги до ресурсів	Незручність для користувача
	Б	Простота створення інтерактивного інтерфейсу	Обмежені можливості кастомізації
	В	Повноцінний графічний інтерфейс	Складніша реалізація
F4	A	Швидка генерація аналітичних висновків, сучасні AI-можливості	Потреба в інтернет-з'єднанні та API-ключі
	Б	Автономна робота без зовнішніх сервісів	Високі вимоги до апаратного забезпечення

З позитивно-негативної матриці можна зробити висновок, що деякі варіанти реалізації функцій програмного продукту доцільно відкинути, оскільки вони не повністю відповідають вимогам, поставленим перед системою підтримки прийняття рішень для аналізу табличних наборів даних.

Функція *F1*:

Основною вимогою до середовища розробки є підтримка сучасних мов програмування, зручність інтеграції бібліотек аналізу даних та швидкість розробки програмного продукту. PyCharm є зручним середовищем для Python, однак має обмеження щодо роботи з повною версією. Visual Studio

забезпечує потужні засоби розробки, але потребує більших системних ресурсів. Тому залишаємо варіанти А та В.

Функція $F2$:

Основною вимогою до мови програмування є підтримка бібліотек аналізу даних, можливість реалізації алгоритмів машинного навчання та інтеграція інструментів візуалізації. Найбільш придатною мовою є Python. Мова Java має обмеження щодо інтеграції окремих інструментів аналізу даних, а C# менш поширений у сфері Data Science. Тому залишаємо лише варіант А.

Функція $F3$:

Основною вимогою до користувацького інтерфейсу є зручність використання та інтерактивність. Консольний інтерфейс має обмежені можливості взаємодії з користувачем. Desktop-застосунок забезпечує ширші можливості інтерфейсу, однак потребує складнішої реалізації. Найбільш доцільним є використання веб інтерфейсу Streamlit. Тому залишаємо варіанти Б та В.

Функція $F4$:

Для реалізації AI-генерації аналітичних висновків важливими є простота інтеграції та швидкість роботи. Gemini API забезпечує сучасні можливості генерації тексту та просту інтеграцію із системою. Локальна мовна модель забезпечує автономність роботи, однак потребує значних апаратних ресурсів. Тому залишаємо варіанти А та Б.

Таким чином, будемо розглядати наступні варіанти реалізації програмного продукту:

F1A – F2A – F3Б – F4A

F1B – F2A – F3В – F4Б

4.3 Обґрунтування системи параметрів програмного продукту

Отримані в результаті попереднього аналізу показники слугують базою для формування оцінних критеріїв, необхідних для подальшого обчислення індексу технічної досконалості. Комплексна специфікація досліджуваного програмного забезпечення визначається за допомогою такої системи метрик:

- 1) X1 – час обробки набору даних;
- 2) X2 – об’єм використаної оперативної пам’яті;
- 3) X3 – час генерації аналітичного висновку;
- 4) X4 – потенційний об’єм програмного коду.

Формування діапазону оцінок (від мінімально допустимих до еталонних) здійснюється з урахуванням специфікацій технічного завдання та реальних умов функціонування розробленого софту. Детальний розподіл цих критеріїв зафіксовано в матеріалах таблиці 4.3.

Таблиця 4.3 – Основні метрики програмного продукту

Назва метрики	Умовні позначення	Одиниці виміру	Значення метрики		
			гірші	середні	кращі
Час обробки набору даних	X1	мс	1200	700	300
Об’єм пам’яті	X2	МБ	1024	512	256
Час генерації аналітичного висновку	X3	с	12	6	2
Потенційний об’єм коду	X4	кількість рядків коду	2500	1700	900

За даними таблиці 4.3, побудуємо графічні характеристики параметрів (рис. 4.2 – 4.5):

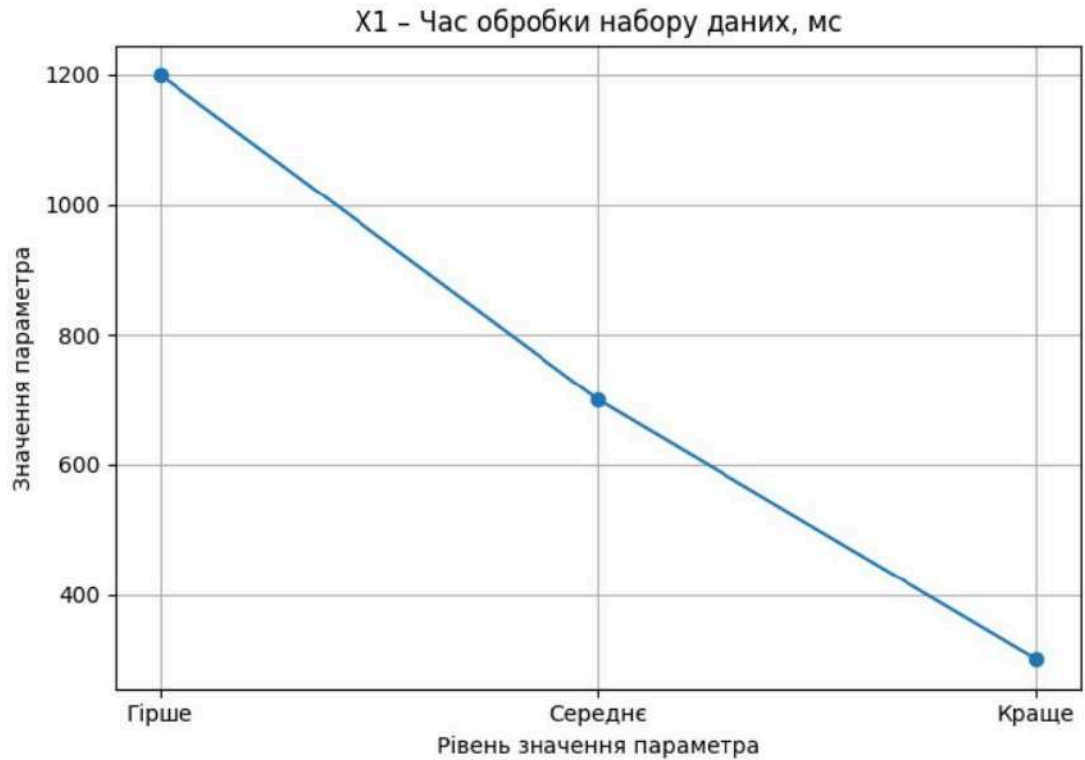


Рисунок 4.2 – X1, час обробки набору даних

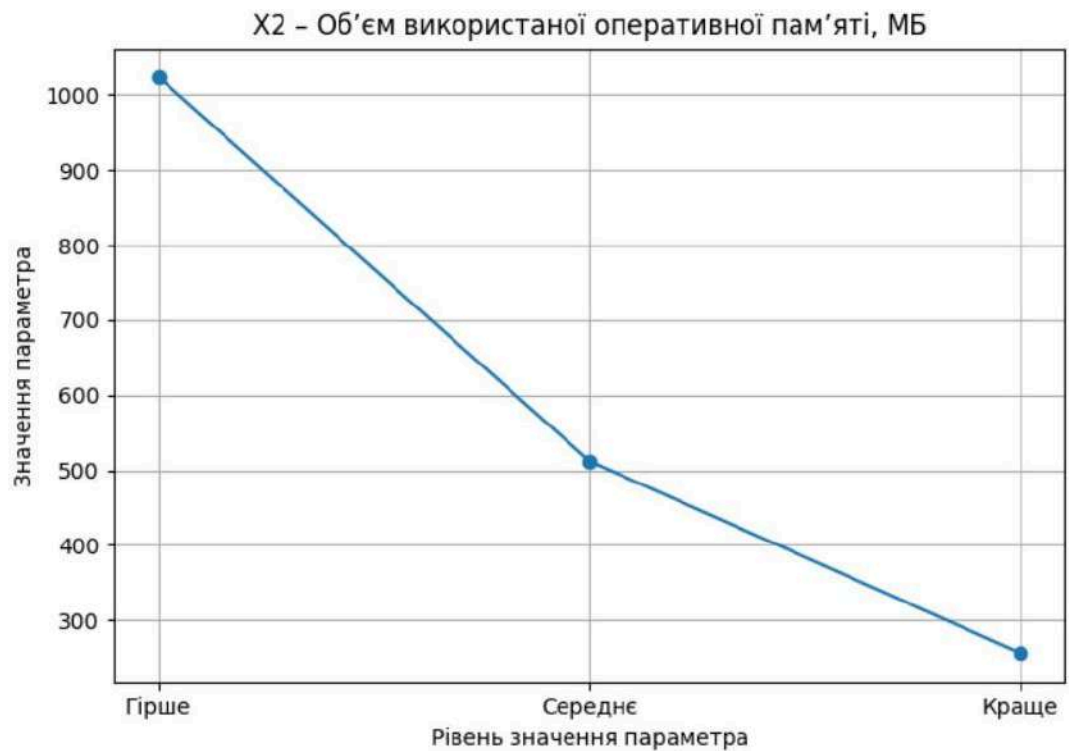


Рисунок 4.3 – X2, об'єм пам'яті

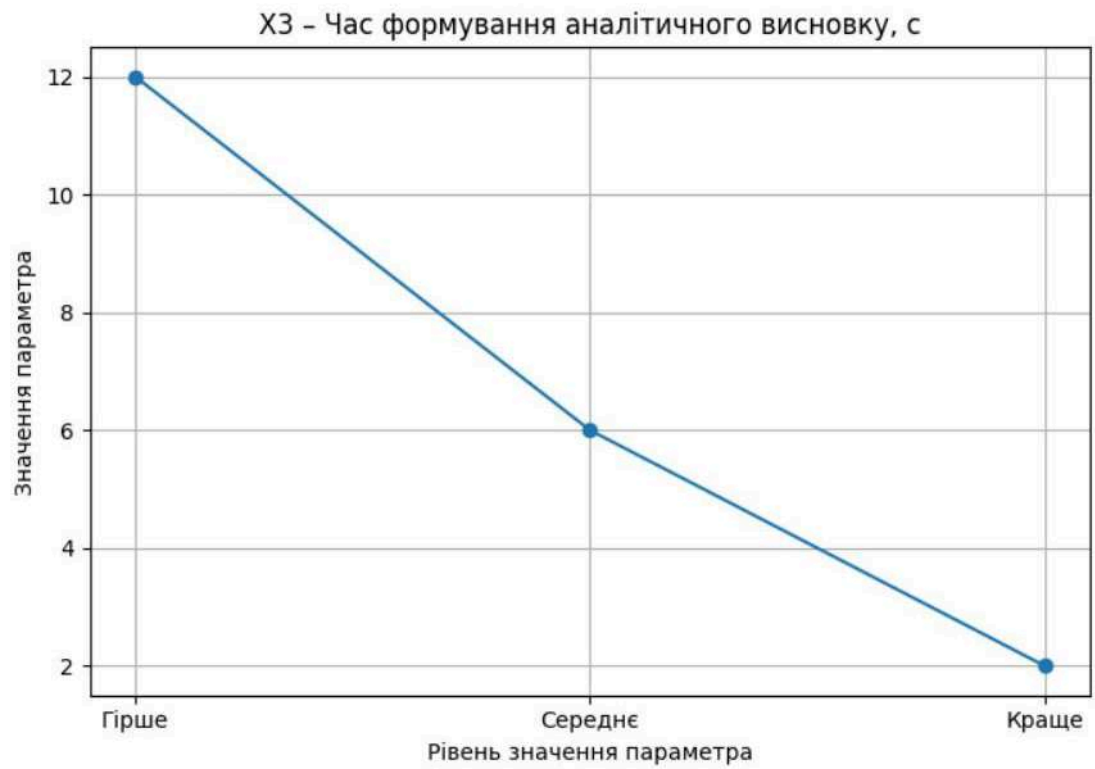


Рисунок 4.4 – Х3, час генерації аналітичного висновку

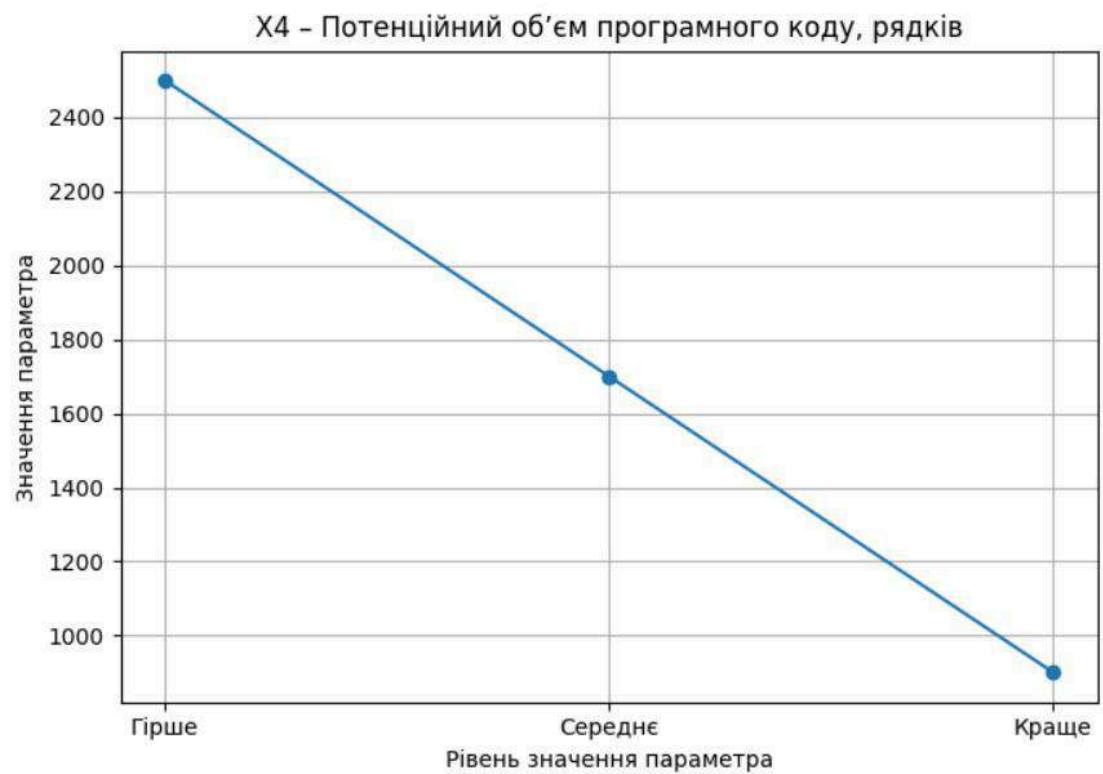


Рисунок 4.5 – Х4, потенційний об'єм коду

4.4 Аналіз експертного оцінювання параметрів

Фіксація вагових коефіцієнтів для кожної технічної характеристики відбувається індивідуально після колегіального вивчення специфіки проєкту.

Пріоритетність досліджуваних чинників встановлюється на основі матриць попарних порівнянь. До цього процесу залучено аналітичну групу, що складається із 7 сертифікованих фахівців. Процедура математичного обчислення вагових індексів охоплює такі етапи:

- 1) диференціацію вагомості критеріїв за допомогою механізму ранжування;
- 2) математичний контроль узгодженості та валідності отриманих експертних висновків;
- 3) розрахунок відносних пріоритетів для кожної пари ознак;
- 4) статистичну обробку зібраних анкет та виведення фінального коефіцієнту значимості.

Підсумкові матричні дані експертного оцінювання систематизовано у таблиці 4.4:

Таблиця 4.4 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Час обробки набору даних	мс	1	2	1	1	2	1	2	10	-7.5	56.25
X2	Об'єм пам'яті	МБ	3	3	3	2	3	3	3	20	2.5	6.25
X3	Час генерації аналітичного висновку	с	2	1	2	3	1	2	1	12	-5.5	30.25
X4	Потенційний об'єм коду	кількість рядків коду	4	4	4	4	4	4	4	28	10.5	110.25
	Разом		10	10	10	10	10	10	10	70	0	203

Для підтвердження об'єктивності та релевантності отриманих експертних висновків пропонується обчислити такий комплекс метрик:

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

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

де N – кількість експертів, n – кількість параметрів;

Розрахунок відносного індексу a_{ij} , який відображає рівень домінування або пріоритетності i -ї характеристики над j -ю, здійснюється за такою математичною залежністю:

$$a_{ij} = \begin{cases} 1.5 & \text{при } X_i > X_j \\ 1.0 & \text{при } X_i = X_j \\ 0.5 & \text{при } X_i < X_j \end{cases} \quad (4.6)$$

На основі обчислених кількісних показників пріоритетності формується матриця парних порівнянь $A = \| a_{ij} \|$. Наступним кроком для кожного визначеного критерію здійснюється обчислення коефіцієнтів вагомості K_{bi} за такими математичними залежностями:

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

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

Процедура обчислення відносних оцінок має ітераційний характер і повторюється до досягнення заданого рівня збіжності, коли похибка між суміжними значеннями не перевищує 2%. Починаючи з другого етапу, розрахунок зазначених показників реалізується на основі таких математичних виразів:

$$K_{bi} = \frac{b_i^{'}}{\sum_{i=1}^n b_i^{'}} \quad (4.9)$$

$$b_i^{' } = \sum_{j=1}^N a_{ij} b_j \quad (4.10)$$

Розрахунки вагомості параметрів показано в таблиці 4.6:

Таблиця 4.6 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітерація		Друга ітерація		Третя ітерація	
	X1	X2	X3	X4	b_i	K_{bi}	b_i^1	K_{bi}^1	b_i^2	K_{bi}^2
X1	1	1.5	1.5	1.5	5.5	0.344	21.25	0.36	78.88	0.362
X2	0.5	1	0.5	1.5	3.5	0.219	12.25	0.208	45.13	0.207
X3	0.5	1.5	1	1.5	4.5	0.281	16.25	0.275	60.13	0.276
X4	0.5	0.5	0.5	1	2.5	0.156	9.25	0.157	33.88	0.155
Всього					16	1	59	1	218	1

Згідно з даними, наведеними в табл. 4.6, дельта між розрахованими коефіцієнтами вагомості знаходиться в межах 2%, що свідчить про досягнення необхідної точності та відсутність потреби у подальшому розгортанні ітераційного процесу.

4.5 Аналіз рівня якості варіантів реалізації функцій

Для подальшого порівняльного аналізу розроблюваного програмного забезпечення виконаємо диференційоване оцінювання якісних характеристик для кожної альтернативи реалізації його ключових функцій. Отримані числові показники таких метрик, як обсяг задіяної оперативної пам'яті X2, тривалість формування аналітичного висновку X3 та прогнозована кількість рядків програмного коду X4, повністю задовольняють критеріям працездатності та технічним обмеженням експлуатації створеної СППР. Водночас базове значення параметра X1, що відображає чистий час обробки табличного набору даних, зафіксовано на оптимальному рівні.

Математичне визначення інтегрального коефіцієнта технічного рівня для кожної з розглянутих архітектурних конфігурацій системи здійснюється за наведеним нижче алгоритмом:

$$K_K(j) = \sum_{i=1}^n K_{bi} B_{ij}, \quad (4.11)$$

де n – кількість параметрів, K_{bi} – коефіцієнт вагомості i -го параметра, B_i – оцінка i -го параметра в балах.

Зведені результати обчислення якісних показників для кожної з розглянутих альтернатив архітектурної реалізації базових функцій системи деталізовано в матеріалах таблиці 4.7:

Таблиця 4.7 – 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	A	X4	1200	8	0.155	1.24
	B	X4	1500	7	0.155	1.085
F2	A	X1	350	9	0.362	3.258
F3	Б	X4	1200	8	0.155	1.24
	B	X4	2100	5	0.155	0.775
F4	A	X3	3	9	0.276	2.484
	Б	X3	9	5	0.276	1.38

За даними з таблиці 4.7 за формулою:

$$K_K = K_{\text{ту}}[F_{1k}] + K_{\text{ту}}[F_{2k}] + \dots + K_{\text{ту}}[F_{zk}] \quad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 1.24 + 3.258 + 1.24 + 2.484 = 8.222;$$

$$K_{K2} = 1.085 + 3.258 + 0.775 + 1.38 = 6.498.$$

Аналіз отриманих розрахункових даних свідчить про те, що оптимальні результати демонструє саме перший варіант конфігурації програмного забезпечення, який характеризується найвищим показником технічної досконалості. Спираючись на сформовану систему критеріїв оцінювання, цей варіант побудови функціональних модулів системи підтримки прийняття рішень є найбільш ефективним для практичного впровадження.

4.6 Економічний аналіз варіантів розробки ПП

Передумовою для обчислення фінансових витрат на створення програмного забезпечення є визначення загальних затрат праці. Процес розробки для кожного з розглянутих архітектурних рішень СППР диференціюється на два ключові етапи:

1. Розробка програмного продукту для аналізу табличних наборів даних та підтримки прийняття рішень.
2. Розробка програмної оболонки та користувацького інтерфейсу системи.

За критерієм технологічної новизни перший етап класифікується як розробка групи А, тоді як другий етап відповідає групі Б. Відповідно до специфіки алгоритмічної складності, розробка програмного продукту для аналізу табличних наборів даних та підтримки прийняття рішень (завдання 1) віднесена до 1-ї групи складності, а розробка програмної оболонки та користувацького інтерфейсу системи (завдання 2) — до 3-ї групи складності. Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних. На основі наведених вихідних характеристик виконаємо розрахунок трудомісткості проектування та

розробки програмного продукту для кожного етапу окремо. Сумарна трудомісткість створення системи визначається за такою залежністю:

$$T_o = T_p \cdot K_{\Pi} \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М} \quad (4.13)$$

де T_p – трудомісткість розробки ПП, K_{Π} – поправочний коефіцієнт, $K_{СК}$ – коефіцієнт на складність вхідної інформації, K_M – коефіцієнт рівня мови програмування, $K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм $K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого етапу розробки, враховуючи нормативні витрати часу на реалізацію завдань розрахунково-аналітичного характеру зі ступенем новизни А та першою групою алгоритмічної складності, базова трудова місткість становить $T_p = 90$ людино-день. З метою врахування специфіки використання нормативно-довідкової інформації (НДІ) під час проектування математичного апарату вводиться поправочний коефіцієнт $K_{\Pi} = 1.7$. Оцінка складності процедур контролю вхідних і вихідних інформаційних потоків для обох етапів проекту є стандартною, тому відповідний коефіцієнт приймається на рівні $K_{СК} = 1$. $K_M = 0.85$ для Python. Оскільки під час реалізації першого завдання залучається готове програмне інструментування та стандартні бібліотечні модулі, цей чинник враховується за допомогою знижувального коефіцієнта $K_{СТ} = 0.8$. І також використовується стандартне математичне забезпечення $K_{СТ.М} = 1.2$. З урахуванням усіх визначених коригувальних індексів, сумарна трудомісткість проектування та програмування для першого завдання обчислюється як:

$$T_1 = 90 \cdot 1.7 \cdot 1 \cdot 0.85 \cdot 0.8 \cdot 1.2 = 124.85 \text{ людино-днів}$$

Аналогічним чином виконаємо розрахунок часових витрат для наступного етапу проекту. Під час реалізації другого завдання, що передбачає створення інтерфейсної частини та оболонки системи, алгоритми проектування класифікуються за третьою групою складності, а ступінь технологічної новизни відповідає категорії Б. Згідно з галузевими

нормативами, базова трудова місткість для таких умов становить $T_p = 19$ людино-днів. Оскільки на цьому етапі як основне інформаційне середовище залучаються структуровані масиви даних (БД), відповідний коефіцієнт коригування становить $K_{\Pi} = 0.75$. Показник складності контролю вхідних та вихідних потоків інформації залишається незмінним і дорівнює $K_{СК} = 1$. Фактор використання готового програмного інструментування (високорівневих бібліотек мови Python) $K_M = 0.85$ та стандартних інтерфейсних модулів враховується за допомогою коефіцієнта $K_{СТ} = 0.8$. У цьому завданні стандартне математичне забезпечення також буде $K_{СТ.М} = 1.2$. З урахуванням зазначених параметрів, підсумкова трудомісткість програмування оболонки та користувацького інтерфейсу обчислюється як:

$$T_2 = 19 \cdot 0.75 \cdot 1 \cdot 0.85 \cdot 0.8 \cdot 1.2 = 11.63 \text{ людино-дні}$$

Для визначення підсумкових часових витрат на створення програмного продукту виконаємо сумування обчислених значень трудомісткості окремих етапів розробки для кожної з розглянутих архітектурних альтернатив системи. Це дозволить отримати повну інтегральну оцінку трудовитрат, необхідних для комплексного проектування та програмування СППР. Загальний баланс витрат праці знаходиться за такою формулою:

$$T_1 = (124.85 + 11.63) \cdot 8 = 1091.84 \text{ людино-годин}$$

$$T_{\Sigma} = (124.85 + 11.63 + 9.9) \cdot 8 = 1171.04 \text{ людино-годин}$$

У розробці програмного продукту беруть участь два програмісти та один QA-інженер. Перший програміст виконує розробку модулів аналізу табличних наборів даних, реалізацію алгоритмів статистичної обробки, кластерного аналізу та прогнозування. Другий програміст відповідає за створення користувацького інтерфейсу системи, реалізацію засобів візуалізації результатів аналізу, а також інтеграцію функціональних модулів програмного продукту. QA-інженер здійснює тестування системи, перевірку коректності роботи програмних модулів та контроль стабільності функціонування програмного продукту.

Для подальших економічних розрахунків приймаємо, що посадовий оклад кожного програміста становить 23000 грн на місяць, а посадовий оклад QA-інженера – 21000 грн на місяць. Середня погодинна ставка, рахується за формулою:

$$C_{\text{ч}} = \frac{M}{T_m \cdot t} \quad (4.14)$$

де M – місячний оклад працівників, T_m – кількість робочих днів тиждень, t – кількість робочих годин в день.

$$C_{\text{ч}} = \frac{23000+23000+21000}{4 \cdot 4 \cdot 5 \cdot 8} = 104.69 \text{ грн.} \quad (4.15)$$

Розрахуємо заробітну плату за формулою:

$$C_{\text{зп}} = C_{\text{ч}} \cdot T_i \cdot K_{\text{д}} \quad (4.16)$$

де $C_{\text{ч}}$ – величина погодинної оплати праці працівника, T_i – трудомісткість відповідного завдання, $K_{\text{д}}$ – норматив, який враховує додаткову заробітну плату. Зарплата розробників з урахуванням додаткової ($K_{\text{д}} = 1.2$) за варіантами становить:

$$\text{І. } C_{\text{зп}} = 104.69 \cdot 1091.84 \cdot 1.2 = 137\,165.67 \text{ грн.}$$

$$\text{ІІ. } C_{\text{зп}} = 104.69 \cdot 1171.04 \cdot 1.2 = 147\,115.41 \text{ грн.}$$

Відрахування на єдиний соціальний внесок (22%):

$$\text{І. } C_{\text{вд}} = 137\,165.67 \cdot 0.22 = 30\,176.44 \text{ грн.}$$

$$\text{ІІ. } C_{\text{вд}} = 147\,115.41 \cdot 0.22 = 32\,365.39 \text{ грн.}$$

Визначимо витрати на оплату однієї машино-години. ЕОМ обслуговує розробника з окладом 23 000 грн, коефіцієнт зайнятості $K_3 = 0.2$:

$$C_{\text{г}} = 12 \cdot 23000 \cdot 0.2 = 55\,200 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{\text{зп}} (\text{ЕОМ}) = 55200 \cdot (1 + 0.2) = 66\,240 \text{ грн.}$$

Відрахування на соціальний внесок:

$$C_{\text{вд}} (\text{ЕОМ}) = 66240 \cdot 0.22 = 14\,572.8 \text{ грн.}$$

Амортизаційні відрахування при нормі амортизації 25% та вартості ноутбука 45 000 грн:

$$C_A = K_{TM} \cdot K_A \cdot C_{PP} = 1.15 \cdot 0.25 \cdot 45\,000 = 12\,937.5 \text{ грн}$$

де K_{TM} – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача, K_A – річна норма амортизації, C_{PP} – договірна ціна приладу.

Витрати на ремонт та профілактику:

$$C_P = K_{TM} \cdot C_{PP} \cdot K_P = 1.15 \cdot 45\,000 \cdot 0.05 = 2\,587.5 \text{ грн}$$

де K_P – відсоток витрат на поточні ремонти.

Ефективний річний фонд часу роботи ПК:

$$T_{EF} = (D_K - D_B - D_C - D_P) \cdot t_z \cdot K_B = (365 - 104 - 12 - 12) \cdot 8 \cdot 0.9 = 1\,706.4 \text{ годин}$$

де D_K – календарна кількість днів у році, D_B , D_C – відповідно кількість вихідних та святкових днів, D_P – кількість днів планових ремонтів устаткування, t_z – кількість робочих годин в день, K_B – коефіцієнт використання приладу у часі протягом зміни.

Витрати на електроенергію (тариф 14.02 грн/кВт·год, потужність 0.3 кВт):

$$C_{EL} = T_{EF} \cdot N_C \cdot K_3 \cdot C_{EH} = 1\,706.4 \cdot 0.3 \cdot 0.2 \cdot 14.02 = 1435.42 \text{ грн}$$

де N_C – середньо-споживча потужність приладу, K_3 – коефіцієнтом зайнятості приладу, C_{EH} – тариф за 1 кВт-годин електроенергії.

Накладні витрати:

$$C_H = C_{PP} \cdot 0.67 = 45\,000 \cdot 0.67 = 30\,150 \text{ грн}$$

Річні експлуатаційні витрати:

$$C_{EKC} = C_{ЗП} + C_{ВІД} + C_A + C_P + C_{EL} + C_H \quad (4.17)$$

$$C_{EKC} = 66\,240 + 14\,572.8 + 12\,937.5 + 2\,587.5 + 1435.42 + 30\,150 = 127\,923.22 \text{ грн}$$

Собівартість однієї машино-години:

$$C_{M-Г} = C_{EKC} / T_{EF} = 127\,923.22 / 1\,706.4 = 74.96 \text{ грн/год}$$

Витрати на машинний час:

$$C_M = C_{M-Г} \cdot T \quad (4.18)$$

$$\text{ї. } C_M = 74.96 \cdot 1091.84 = 81\,844.32 \text{ грн.}$$

$$\text{ІІ. } C_M = 74,96 \cdot 1171.04 = 87\,781.15 \text{ грн.}$$

Накладні витрати (67% від заробітної плати):

$$C_H = C_{3П} \cdot 0.67 \quad (4.19)$$

$$\text{І. } C_H = 137\,165.67 \cdot 0.67 = 91\,900.55 \text{ грн.}$$

$$\text{ІІ. } C_H = 147\,115.41 \cdot 0.67 = 98\,567.32 \text{ грн.}$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{3П} + C_{ВІД} + C_M + C_H \quad (4.20)$$

$$\text{І. } C_{ПП} = 137\,165.67 + 30\,176.44 + 81\,844.32 + 91\,900.55 = 341\,086,98 \text{ грн.}$$

$$\text{ІІ. } C_{ПП} = 147\,115.41 + 32\,365.3 + 87\,781.15 + 98\,567.32 = 365\,829.18 \text{ грн.}$$

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$K_{\text{ТЕР}j} = K_{Kj} / C_{ППj} \quad (4.21)$$

де K_{Kj} – коефіцієнт рівня якості j-го варіанту; $C_{ППj}$ – вартість розробки j-го варіанту, грн

$$K_{\text{ТЕР}1} = 8.222 / 341\,086,98 = 2.41 \cdot 10^{-5}$$

$$K_{\text{ТЕР}2} = 6.498 / 365\,829.18 = 1.77 \cdot 10^{-5}$$

Проведений аналіз наочно демонструє, що максимальну ефективність забезпечує перша альтернатива розробки програмного забезпечення, інтегральний індекс техніко-економічного рівня якої становить $K_{\text{ТЕР}1} = 2.41 \cdot 10^{-5}$. Таким чином, спираючись на результати проведеного дослідження, можна констатувати: серед архітектурних рішень саме перша модифікація є найбільш раціональною та оптимальною для практичного впровадження. Ця конфігурація реалізації СППР характеризується такими базовими параметрами: Visual Studio Code , Python , Streamlit та Gemini API.

Висновки до розділу 4

У четвертому розділі було проведено функціонально-вартісний аналіз системи підтримки прийняття рішень для аналізу табличних наборів даних. У процесі дослідження визначено основні функції програмного продукту, побудовано морфологічну карту системи та проаналізовано можливі варіанти реалізації програмного продукту.

Було обґрунтовано систему параметрів програмного продукту та проведено експертне оцінювання їх вагомості. На основі отриманих результатів виконано аналіз рівня якості варіантів реалізації функцій системи та визначено найбільш доцільні технологічні рішення.

Також у розділі проведено економічний аналіз розробки програмного продукту, визначено ступінь новизни та складності алгоритмів, а також розраховано трудомісткість реалізації системи. Проведений аналіз підтвердив доцільність використання обраних програмних засобів та ефективність розробленої системи підтримки прийняття рішень.

ВИСНОВКИ

У результаті виконання дипломної роботи було досліджено сучасні підходи до аналізу табличних наборів даних та реалізовано систему підтримки прийняття рішень для автоматизації процесу обробки й аналізу інформації. Проведений аналіз теоретичних джерел дозволив дослідити особливості аналізу часових рядів, методи виявлення аномалій та підходи до кластеризації даних, а також визначити програмні засоби, найбільш доцільні для реалізації системи.

У межах роботи було реалізовано програмний продукт на основі мови програмування Python із використанням бібліотек pandas, NumPy, scikit-learn, Plotly та Streamlit. Розроблена система забезпечує завантаження та обробку табличних наборів даних, побудову часових рядів, застосування методів SMA та ЕМА, лінійне прогнозування, виявлення аномалій методом Z-score, кластеризацію даних за допомогою алгоритму K-Means та визначення оптимальної кількості кластерів методом ліктя.

У роботі також реалізовано інтеграцію Gemini API для автоматизованої генерації аналітичних висновків, що дозволило підвищити рівень автоматизації інтерпретації результатів аналізу даних та спростити взаємодію користувача із системою. Розроблений веб інтерфейс забезпечує зручне налаштування параметрів аналізу та інтерактивне відображення результатів дослідження.

Під час виконання функціонально-вартісного аналізу було визначено основні функції програмного продукту, обґрунтовано систему параметрів та проведено економічний аналіз варіантів реалізації системи. Отримані результати підтвердили доцільність використання обраних програмних засобів та ефективність реалізованої системи підтримки прийняття рішень.

Розроблена система може бути використана для аналізу табличних наборів даних у сфері економіки, фінансів, аналітики та інших галузях, де

необхідно автоматизувати процес обробки інформації, виявлення закономірностей та формування аналітичних висновків.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hyndman R. J., Athanasopoulos G. *Forecasting: principles and practice*. 3rd ed. OTexts: Melbourne, Australia, 2021. URL: <https://otexts.com/fpp3/> (дата звернення: 27.04.2026).
2. Aggarwal C. C. *Outlier Analysis*. 2nd ed. Springer International Publishing, 2017. 512 p.
3. Згуровський М. З., Панкратова Н. Д. Системний аналіз: проблеми, методологія, додатки. Київ : Наукова думка, 2011. 728 с.
4. Python 3.12 documentation. *Python Language Reference*. URL: <https://docs.python.org/3/> (дата звернення: 01.05.2026).
5. James G., Witten D., Hastie T., Tibshirani R. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. 426 p.
6. Pandas documentation. *Pandas: powerful Python data analysis toolkit*. URL: <https://pandas.pydata.org/docs/> (дата звернення: 07.05.2026).
7. Scikit-learn: machine learning in Python. *Scikit-learn documentation*. URL: <https://scikit-learn.org/stable/> (дата звернення: 07.05.2026).
8. Streamlit documentation. *Streamlit: The fastest way to build and share data apps*. URL: <https://docs.streamlit.io/> (дата звернення: 07.05.2026).
9. Google AI for Developers. *Gemini API Overview*. URL: <https://ai.google.dev/docs> (дата звернення: 10.05.2026).
10. Plotly Graphing Libraries. *Plotly Python Open Source Graphing Library*. URL: <https://plotly.com/python/> (дата звернення: 09.05.2026).
11. Statsmodels: Econometric and statistical modeling with Python. *Statsmodels documentation*. URL: <https://www.statsmodels.org/stable/index.html> (дата звернення: 09.05.2026).
12. NumPy Documentation. *NumPy: the fundamental package for scientific computing with Python*. URL: <https://numpy.org/doc/> (дата звернення: 07.05.2026).

- 13.Бідюк П. І., Тимощук О. Л., Коваленко А. Є., Коршевніук Л. О. Системи і методи підтримки прийняття рішень : підручник. Київ : КПІ ім. Ігоря Сікорського, 2022. 610 с.
- 14.Пашін В. П., Романов В. В., Єгорова Н. В. Методичні вказівки до виконання економіко-організаційного розділу дипломних проектів (робіт) бакалаврів і спеціалістів для студентів Інституту прикладного системного аналізу з дисципліни «Економіка та організація виробництва». Київ : НТУУ «КПІ», 2011. 118 с.
- 15.Бідюк П. І., Романенко В. Д., Тимощук О. Л. Аналіз часових рядів: навч. посіб. Київ, НТУУ "КПІ", 2013. 600 с.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

main.py

```

from typing import List, Tuple

import numpy as np
import pandas as pd
import plotly.graph_objects as go
import streamlit as st
from sklearn.cluster import KMeans
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from statsmodels.tsa.seasonal import seasonal_decompose
from google import genai

st.set_page_config(page_title="DSS для аналізу табличних даних", layout="wide")

# =====
# Допоміжні функції
# =====

def load_data(uploaded_file) -> pd.DataFrame:
    """Завантаження CSV або Excel у DataFrame."""
    if uploaded_file is None:
        return pd.DataFrame()

    file_name = uploaded_file.name.lower()

    if file_name.endswith(".csv"):
        for enc in ["utf-8", "cp1251", "latin1"]:
            try:
                uploaded_file.seek(0)
                return pd.read_csv(uploaded_file, encoding=enc)
            except Exception:
                uploaded_file.seek(0)
        raise ValueError("Не вдалося визначити кодування файлу")

    if file_name.endswith(".xlsx") or file_name.endswith(".xls"):
        uploaded_file.seek(0)
        return pd.read_excel(uploaded_file)

    raise ValueError("Підтримуються лише CSV та Excel файли")

def get_numeric_columns(df: pd.DataFrame) -> List[str]:
    return df.select_dtypes(include=[np.number]).columns.tolist()

def try_parse_datetime_column(df: pd.DataFrame, column: str) -> pd.DataFrame:
    """Пробує перетворити обрану колонку на datetime."""
    result = df.copy()
    try:
        result[column] = pd.to_datetime(result[column], errors="coerce")
    except Exception:
        pass
    return result

```

```

def moving_average(series: pd.Series, window: int) -> pd.Series:
    return series.rolling(window=window).mean()

def exponential_moving_average(series: pd.Series, span: int) -> pd.Series:
    return series.ewm(span=span, adjust=False).mean()

def detect_anomalies_zscore(series: pd.Series, threshold: float = 3.0) ->
pd.DataFrame:
    """Пошук аномалій через z-score."""
    clean_series = series.dropna()

    if clean_series.empty or clean_series.std(ddof=0) == 0:
        result = pd.DataFrame(index=series.index)
        result["value"] = series
        result["z_score"] = np.nan
        result["is_anomaly"] = False
        return result

    z_scores = (clean_series - clean_series.mean()) / clean_series.std(ddof=0)

    result = pd.DataFrame(index=series.index)
    result["value"] = series
    result["z_score"] = np.nan
    result.loc[clean_series.index, "z_score"] = z_scores
    result["is_anomaly"] = result["z_score"].abs() > threshold
    return result

def prepare_features_for_clustering(df: pd.DataFrame, features: List[str]) ->
Tuple[np.ndarray, pd.DataFrame]:
    """Готує матрицю ознак: заповнює пропуски та масштабує дані."""
    features_df = df[features].copy()

    imputer = SimpleImputer(strategy="mean")
    imputed = imputer.fit_transform(features_df)

    scaler = StandardScaler()
    scaled = scaler.fit_transform(imputed)

    return scaled, features_df

def build_summary_statistics(df: pd.DataFrame) -> pd.DataFrame:
    numeric_cols = get_numeric_columns(df)
    if not numeric_cols:
        return pd.DataFrame()
    return df[numeric_cols].describe().T

def convert_df_to_csv(df: pd.DataFrame) -> bytes:
    return df.to_csv(index=False).encode("utf-8-sig")

def fill_missing_values(df: pd.DataFrame, strategy: str) -> pd.DataFrame:
    """Заповнення пропусків відповідно до обраної стратегії."""
    result = df.copy()
    numeric_cols = get_numeric_columns(result)
    categorical_cols = [col for col in result.columns if col not in
numeric_cols]

    if strategy == "Не заповнювати":

```

```

        return result

    if strategy == "Середнім значенням":
        for col in numeric_cols:
            result[col] = result[col].fillna(result[col].mean())
        for col in categorical_cols:
            if result[col].isna().any():
                mode = result[col].mode(dropna=True)
                if not mode.empty:
                    result[col] = result[col].fillna(mode.iloc[0])
        return result

    if strategy == "Медіаною":
        for col in numeric_cols:
            result[col] = result[col].fillna(result[col].median())
        for col in categorical_cols:
            if result[col].isna().any():
                mode = result[col].mode(dropna=True)
                if not mode.empty:
                    result[col] = result[col].fillna(mode.iloc[0])
        return result

    if strategy == "Найчастішим значенням":
        for col in result.columns:
            if result[col].isna().any():
                mode = result[col].mode(dropna=True)
                if not mode.empty:
                    result[col] = result[col].fillna(mode.iloc[0])
        return result

    if strategy == "Замінити на 0":
        for col in numeric_cols:
            result[col] = result[col].fillna(0)
        return result

    return result

def drop_missing_values(df: pd.DataFrame, mode: str) -> pd.DataFrame:
    """Видалення пропусків за обраним режимом."""
    result = df.copy()

    if mode == "Не видаляти":
        return result
    if mode == "Видалити рядки з пропусками":
        return result.dropna()
    if mode == "Видалити колонки з пропусками":
        return result.dropna(axis=1)

    return result

def remove_duplicates(df: pd.DataFrame, mode: str) -> pd.DataFrame:
    """Робота з дублікатами."""
    result = df.copy()

    if mode == "Не видаляти":
        return result
    if mode == "Видалити дублікати":
        return result.drop_duplicates()

    return result

```

```

def convert_column_type(df: pd.DataFrame, column: str, target_type: str) ->
pd.DataFrame:
    """Змінює тип даних вибраної колонки."""
    result = df.copy()

    if target_type == "text":
        result[column] = result[column].astype("string")

    elif target_type == "integer":
        result[column] = pd.to_numeric(result[column],
errors="coerce").astype("Int64")

    elif target_type == "float":
        result[column] = pd.to_numeric(result[column], errors="coerce")

    elif target_type == "datetime":
        result[column] = pd.to_datetime(result[column], errors="coerce")

    elif target_type == "category":
        result[column] = result[column].astype("category")

    return result

def build_dtype_info(df: pd.DataFrame) -> pd.DataFrame:
    """Інформація про типи даних і кількість пропусків."""
    return pd.DataFrame(
        {
            "column": df.columns,
            "dtype": [str(dtype) for dtype in df.dtypes],
            "missing_values": [int(df[col].isna().sum()) for col in
df.columns],
            "unique_values": [int(df[col].nunique(dropna=True)) for col in
df.columns],
        }
    )

def create_decomposition_plot(x_values, component, title: str, y_title: str):
    """Створює інтерактивний графік для компоненти декомпозиції."""
    fig = go.Figure()
    fig.add_trace(
        go.Scatter(
            x=x_values,
            y=component,
            mode="lines",
            name=title,
            hovertemplate="%{x}<br>Значення: %{y}<extra></extra>",
        )
    )
    fig.update_layout(
        title=title,
        xaxis_title="Дата / час",
        yaxis_title=y_title,
        hovermode="x unified",
        height=350,
    )
    return fig

def build_linear_forecast(x_values, y_values, forecast_steps: int,
use_datetime_axis: bool):

```

```

"""Будує простий лінійний прогноз на основі поточного часового ряду."""
clean_y = pd.to_numeric(y_values, errors="coerce").dropna()

if len(clean_y) < 2:
    return None, None, "Недостатньо даних для прогнозування. Потрібно
мінімум 2 коректні точки."

x_numeric = np.arange(len(clean_y))
slope, intercept = np.polyfit(x_numeric, clean_y.values, 1)

future_x_numeric = np.arange(len(clean_y), len(clean_y) + forecast_steps)
forecast_y = slope * future_x_numeric + intercept

if use_datetime_axis:
    dates = pd.to_datetime(x_values).dropna().sort_values()

    if dates.empty:
        return None, None, "Не вдалося визначити дати для прогнозу."

    last_date = dates.max()

    if len(dates) >= 2:
        date_diffs = dates.diff().dropna()
        step_size = date_diffs.median()

        if pd.isna(step_size) or step_size <= pd.Timedelta(0):
            step_size = pd.Timedelta(days=1)
    else:
        step_size = pd.Timedelta(days=1)

    forecast_x = [last_date + step_size * (i + 1) for i in
range(forecast_steps)]
    else:
        try:
            last_index = int(max(x_values))
        except Exception:
            last_index = len(clean_y) - 1
        forecast_x = list(range(last_index + 1, last_index + forecast_steps +
1))

    return forecast_x, forecast_y, None

def generate_gemini_summary(api_key: str, analysis_text: str) -> str:
    client = genai.Client(api_key=api_key)

    prompt = f"""
Ти аналітик даних. Сформууй короткий і зрозумілий висновок українською мовою
для звичайного користувача, який не дуже розуміється в аналітиці.

Не вигадуй дані, яких немає у результатах.
Поясни:
1. що показують результати;
2. які головні закономірності можна помітити;
3. на що користувачу варто звернути увагу.

Результати аналізу:
{analysis_text}
"""

    response = client.models.generate_content(
        model="gemini-3-flash-preview",
        contents=prompt
    )

```

```

    return response.text

# =====
# Заголовок
# =====

st.title("Система підтримки прийняття рішень для аналізу табличних наборів даних")
st.markdown(
    """
    Цей застосунок дозволяє:
    - завантажувати табличні дані;
    - виконувати попередню підготовку даних;
    - змінювати типи даних колонок;
    - досліджувати часові ряди;
    - виявляти аномалії;
    - проводити кластерний аналіз;
    - отримувати висновки за допомогою API Gemini.
    """
)

# =====
# Бокова панель
# =====

st.sidebar.header("Налаштування")
uploaded_file = st.sidebar.file_uploader(
    "Завантаж файл даних", type=["csv", "xlsx", "xls"]
)
gemini_api_key = st.sidebar.text_input(
    "Gemini API key",
    type="password",
    help="Ключ потрібен лише для генерації AI-висновків."
)

# =====
# Завантаження даних
# =====

if uploaded_file is not None:
    try:
        uploaded_name = uploaded_file.name

        if (
            "original_df" not in st.session_state
            or st.session_state.get("current_file_name") != uploaded_name
        ):
            df_loaded = load_data(uploaded_file)
            st.session_state.original_df = df_loaded.copy()
            st.session_state.working_df = df_loaded.copy()
            st.session_state.current_file_name = uploaded_name

            st.success("Файл успішно завантажено")
        except Exception as exc:
            st.error(f"Не вдалося зчитати файл: {exc}")
            st.stop()
    else:
        st.info("Завантаж CSV або Excel файл, щоб почати аналіз")
        st.stop()

```

```

df_original = st.session_state.original_df.copy()
df = st.session_state.working_df.copy()

if df.empty:
    st.warning("Датасет порожній")
    st.stop()

# =====
# 0. Підготовка даних
# =====

with st.expander("0. Підготовка даних", expanded=True):
    st.markdown("Оберіть, як система повинна обробити пропуски, дублікати та типи даних перед основним аналізом.")

    original_rows = len(df_original)
    original_cols = len(df_original.columns)
    original_missing = int(df_original.isna().sum().sum())
    original_duplicates = int(df_original.duplicated().sum())

    current_rows = len(df)
    current_cols = len(df.columns)
    current_missing = int(df.isna().sum().sum())
    current_duplicates = int(df.duplicated().sum())

    met_col1, met_col2, met_col3, met_col4 = st.columns(4)
    met_col1.metric("Рядків", f"{current_rows}")
    met_col2.metric("Стовпців", f"{current_cols}")
    met_col3.metric("Пропусків", f"{current_missing}")
    met_col4.metric("Дублікатів", f"{current_duplicates}")

    with st.expander("Показати типи даних та пропуски", expanded=False):
        st.dataframe(build_dtype_info(df), use_container_width=True)

    st.subheader("0.1 Обробка пропусків і дублікатів")

    prep_col1, prep_col2, prep_col3 = st.columns(3)

    missing_fill_strategy = prep_col1.selectbox(
        "Заповнення пропусків",
        options=[
            "Не заповнювати",
            "Середнім значенням",
            "Медіаною",
            "Найчастішим значенням",
            "Замінити на 0",
        ],
    )

    missing_drop_strategy = prep_col2.selectbox(
        "Видалення пропусків",
        options=[
            "Не видаляти",
            "Видалити рядки з пропусками",
            "Видалити колонки з пропусками",
        ],
    )

    duplicate_strategy = prep_col3.selectbox(
        "Робота з дублікатами",
        options=[

```

```

        "Не видаляти",
        "Видалити дублікати",
    ],
)

apply_prep = st.button("Застосувати обробку пропусків і дублікатів")

if apply_prep:
    processed_df = st.session_state.working_df.copy()
    processed_df = fill_missing_values(processed_df, missing_fill_strategy)
    processed_df = drop_missing_values(processed_df, missing_drop_strategy)
    processed_df = remove_duplicates(processed_df, duplicate_strategy)

    st.session_state.working_df = processed_df
    st.success("Обробку пропусків і дублікатів застосовано")
    st.rerun()

st.subheader("0.2 Зміна типів даних колонок")

type_col1, type_col2 = st.columns(2)

selected_column = type_col1.selectbox(
    "Оберіть колонку для зміни типу",
    options=df.columns.tolist()
)

target_type = type_col2.selectbox(
    "Оберіть новий тип даних",
    options=[
        "text",
        "integer",
        "float",
        "datetime",
        "category",
    ]
)

apply_type_conversion = st.button("Застосувати зміну типу")

if apply_type_conversion:
    before_dtype = str(st.session_state.working_df[selected_column].dtype)
    updated_df = convert_column_type(st.session_state.working_df,
selected_column, target_type)
    after_dtype = str(updated_df[selected_column].dtype)

    st.session_state.working_df = updated_df
    st.success(
        f"Тип колонки '{selected_column}' змінено: {before_dtype} →
{after_dtype}"
    )
    st.rerun()

st.subheader("0.3 Керування поточним датасетом")

manage_col1, manage_col2 = st.columns(2)

if manage_col1.button("Скинути всі зміни"):
    st.session_state.working_df = st.session_state.original_df.copy()
    st.success("Усі зміни скинуто")
    st.rerun()

with manage_col2:
    st.download_button(

```

```

        label="Завантажити поточний датасет CSV",
        data=convert_df_to_csv(df),
        file_name="processed_dataset.csv",
        mime="text/csv",
    )

    with st.expander("Переглянути поточний датасет", expanded=True):
        st.dataframe(df.head(20), use_container_width=True)

# =====
# 1. Попередній огляд даних
# =====

with st.expander("1. Попередній огляд даних", expanded=False):
    df = st.session_state.working_df.copy()

    col1, col2, col3 = st.columns(3)
    col1.metric("Кількість рядків", len(df))
    col2.metric("Кількість стовпців", len(df.columns))
    col3.metric("Числових колонок", len(get_numeric_columns(df)))

    with st.expander("Переглянути перші рядки таблиці", expanded=True):
        st.dataframe(df.head(20), use_container_width=True)

    with st.expander("Типи даних і пропуски"):
        st.dataframe(build_dtype_info(df), use_container_width=True)

    with st.expander("Описова статистика"):
        stats_df = build_summary_statistics(df)
        if stats_df.empty:
            st.warning("У таблиці немає числових колонок для статистичного
аналізу")
        else:
            st.dataframe(stats_df, use_container_width=True)
            st.download_button(
                label="Завантажити статистику CSV",
                data=convert_df_to_csv(stats_df.reset_index()),
                file_name="summary_statistics.csv",
                mime="text/csv",
            )

# =====
# 2. Аналіз часових рядів
# =====

with st.expander("2. Аналіз часових рядів", expanded=False):
    df = st.session_state.working_df.copy()
    numeric_columns = get_numeric_columns(df)
    all_columns = df.columns.tolist()

    if not numeric_columns:
        st.warning("Немає числових колонок для побудови часового ряду")
    else:
        ts_col1, ts_col2, ts_col3, ts_col4, ts_col5, ts_col6 = st.columns(6)

        datetime_candidate = ts_col1.selectbox(
            "Оберіть колонку з датою/часом або порядкову вісь",
            options=["Індекс рядка"] + all_columns,
            key="datetime_column",
        )

```

```

value_column = ts_col2.selectbox(
    "Оберіть числову колонку",
    options=numeric_columns,
    key="ts_value_column",
)

ma_window = ts_col3.slider("Вікно SMA", min_value=2, max_value=30,
value=5)

aggregation_method = ts_col4.selectbox(
    "Метод агрегації",
    ["sum", "mean", "median", "count"]
)

start_date = None
end_date = None

ts_df = df.copy()

if datetime_candidate != "Індекс рядка":
    ts_df = try_parse_datetime_column(ts_df, datetime_candidate)

    # Прибираємо рядки, де дата не зчиталась
    ts_df = ts_df.dropna(subset=[datetime_candidate])

    if ts_df.empty:
        st.warning("Після перетворення колонки дати не залишилося
коректних значень")
        st.stop()

    min_date = ts_df[datetime_candidate].min().date()
    max_date = ts_df[datetime_candidate].max().date()

    start_date = ts_col5.date_input(
        "Початкова дата",
        value=min_date,
        min_value=min_date,
        max_value=max_date,
    )

    end_date = ts_col6.date_input(
        "Кінцева дата",
        value=max_date,
        min_value=min_date,
        max_value=max_date,
    )

    if start_date > end_date:
        st.warning("Початкова дата не може бути більшою за кінцеву")
        st.stop()

    ts_df = ts_df[
        (ts_df[datetime_candidate].dt.date >= start_date) &
        (ts_df[datetime_candidate].dt.date <= end_date)
    ]

    if ts_df.empty:
        st.warning("За обраний період немає даних")
        st.stop()

    # Агрегація після фільтрації по датах
    if aggregation_method == "sum":

```

```

        ts_df =
ts_df.groupby(datetime_candidate)[value_column].sum().reset_index()
        elif aggregation_method == "mean":
            ts_df =
ts_df.groupby(datetime_candidate)[value_column].mean().reset_index()
        elif aggregation_method == "median":
            ts_df =
ts_df.groupby(datetime_candidate)[value_column].median().reset_index()
        elif aggregation_method == "count":
            ts_df =
ts_df.groupby(datetime_candidate)[value_column].count().reset_index()

        x_values = ts_df[datetime_candidate]
        x_label = datetime_candidate
        use_datetime_axis = True
    else:
        x_values = ts_df.index
        x_label = "Індекс"
        use_datetime_axis = False

    y_values = ts_df[value_column]
    sma = moving_average(y_values, ma_window)
    ema = exponential_moving_average(y_values, ma_window)

    y_axis_title = "Кількість" if aggregation_method == "count" else
value_column

    st.subheader("2.1 Налаштування простого прогнозу")
    forecast_col1, forecast_col2 = st.columns(2)

    forecast_enabled = forecast_col1.checkbox(
        "Показати простий лінійний прогноз",
        value=False,
        help="Прогноз будується як проста лінійна тенденція на основі
поточного часового ряду."
    )

    forecast_steps = forecast_col2.slider(
        "Кількість кроків для прогнозу",
        min_value=1,
        max_value=60,
        value=10,
        help="Кількість майбутніх точок, які система додасть до графіка."
    )

    forecast_x = None
    forecast_y = None
    forecast_error = None

    if forecast_enabled:
        forecast_x, forecast_y, forecast_error = build_linear_forecast(
            x_values=x_values,
            y_values=y_values,
            forecast_steps=forecast_steps,
            use_datetime_axis=use_datetime_axis,
        )

        if forecast_error:
            st.warning(forecast_error)

    fig = go.Figure()

    fig.add_trace(

```

```

        go.Scatter(
            x=x_values,
            y=y_values,
            mode="lines+markers",
            name="Початкові дані",
            hovertemplate="%{x}<br>Значення: %{y}<extra></extra>",
        )
    )

    fig.add_trace(
        go.Scatter(
            x=x_values,
            y=sma,
            mode="lines",
            name=f"SMA ({ma_window})",
            hovertemplate="%{x}<br>SMA: %{y}<extra></extra>",
        )
    )

    fig.add_trace(
        go.Scatter(
            x=x_values,
            y=ema,
            mode="lines",
            name=f"EMA ({ma_window})",
            hovertemplate="%{x}<br>EMA: %{y}<extra></extra>",
        )
    )

    if forecast_enabled and forecast_x is not None and forecast_y is not
None:
        # Додаємо останню фактичну точку на початок прогнозової лінії,
        # щоб прогноз візуально продовжував ряд без розриву.
        forecast_plot_x = [x_values.iloc[-1] if hasattr(x_values, "iloc")
else list(x_values)[-1]] + list(forecast_x)
        forecast_plot_y = [y_values.iloc[-1] if hasattr(y_values, "iloc")
else list(y_values)[-1]] + list(forecast_y)

        fig.add_trace(
            go.Scatter(
                x=forecast_plot_x,
                y=forecast_plot_y,
                mode="lines+markers",
                name="Лінійний прогноз",
                line=dict(dash="dash"),
                hovertemplate="%{x}<br>Прогноз: %{y}<extra></extra>",
            )
        )

    fig.update_layout(
        title=f"Часовий ряд для колонки: {value_column}",
        xaxis_title=x_label,
        yaxis_title=y_axis_title,
        hovermode="x unified",
        legend_title="Показники",
        height=520,
    )

    st.plotly_chart(fig, use_container_width=True)

    st.session_state.time_series_summary = f"""
Розділ: Аналіз часових рядів
Обрана колонка часу: {datetime_candidate}

```

```

Обрана числова колонка: {value_column}
Метод агрегації: {aggregation_method}
Період: {start_date} - {end_date}
Кількість точок: {len(y_values)}
Мінімальне значення: {float(y_values.min()):.2f}
Максимальне значення: {float(y_values.max()):.2f}
Середнє значення: {float(y_values.mean()):.2f}
Останнє значення: {float(y_values.iloc[-1]):.2f}
"""

    st.markdown(
        """
        **Пояснення:**
        - Початкові дані показують реальні значення обраного показника.
        - SMA показує середнє значення за обраний проміжок і згладжує
        коливання.
        - ЕМА швидше реагує на нові зміни, тому краще показує свіжі
        тенденції.
        - Лінійний прогноз показує приблизне продовження загальної
        тенденції на обрану кількість кроків.
        - Інтерактивний графік дозволяє масштабувати окремі ділянки,
        переглядати точні значення та вмикати/вимикати лінії через легенду.
        """
    )

    if forecast_enabled and forecast_x is not None and forecast_y is not
None:
        with st.expander("Переглянути прогнозні значення", expanded=False):
            forecast_result_df = pd.DataFrame(
                {
                    "forecast_x": forecast_x,
                    "forecast_value": forecast_y,
                }
            )
            st.dataframe(forecast_result_df, use_container_width=True)
            st.download_button(
                label="Завантажити прогноз CSV",
                data=convert_df_to_csv(forecast_result_df),
                file_name="forecast_results.csv",
                mime="text/csv",
            )

# =====
# 3. Виявлення аномалій
# =====

with st.expander("3. Виявлення аномалій", expanded=False):

    with st.expander("Інструкція: як користуватися пошуком аномалій?",
expanded=False):
        st.info("""
        ### Як користуватися пошуком аномалій?

        **Z-score** показує, наскільки значення відрізняється від середнього.

        Чим менший поріг – тим більше аномалій знайде система.
        Чим більший поріг – тим лише більш «екстремальні» значення будуть вважатися
        аномаліями.

        #### Рекомендації:
        - **1.5 – 2.0** → чутливий режим (знаходить багато підозрілих значень)
        - **2.5 – 3.0** → стандартний режим для більшості задач
        - **4.0+** → пошук лише дуже сильних відхилень

```

```

#### Як інтерпретувати результат?
- якщо аномалій дуже багато → дані нестабільні або поріг занадто малий;
- якщо аномалій немає → або дані стабільні, або поріг занадто великий.
"""

df = st.session_state.working_df.copy()
numeric_columns = get_numeric_columns(df)

if not numeric_columns:
    st.warning("Немає числових колонок для пошуку аномалій")
else:
    an_col1, an_col2 = st.columns(2)

    anomaly_column = an_col1.selectbox(
        "Оберіть числову колонку для пошуку аномалій",
        options=numeric_columns,
        key="anomaly_column",
    )

    threshold = an_col2.slider(
        "Попир |z-score|",
        min_value=1.0,
        max_value=5.0,
        value=3.0,
        step=0.1,
    )

    anomaly_df = detect_anomalies_zscore(df[anomaly_column], threshold)
    anomalies_only = anomaly_df[anomaly_df["is_anomaly"]].copy()

    fig = go.Figure()

    fig.add_trace(
        go.Scatter(
            x=anomaly_df.index,
            y=anomaly_df["value"],
            mode="lines",
            name="Значення",
            hovertemplate="Індекс: %{x}<br>Значення: %{y}<extra></extra>",
        )
    )

    if not anomalies_only.empty:
        fig.add_trace(
            go.Scatter(
                x=anomalies_only.index,
                y=anomalies_only["value"],
                mode="markers",
                name="Аномалії",
                marker=dict(size=9, symbol="circle"),
                hovertemplate="Індекс: %{x}<br>Аномалія:
%{y}<extra></extra>",
            )
        )

    fig.update_layout(
        title=f"Аномалії для колонки: {anomaly_column}",
        xaxis_title="Індекс",
        yaxis_title=anomaly_column,
        hovermode="x unified",
        legend_title="Показники",

```

```

        height=520,
    )

    st.plotly_chart(fig, use_container_width=True)

    col_a, col_b = st.columns(2)
    col_a.metric("Кількість аномалій", int(anomalies_only.shape[0]))
    col_b.metric("Частка аномалій", f"{(anomalies_only.shape[0] /
len(anomaly_df)) * 100:.2f}%")

    with st.expander("Переглянути знайдені аномалії"):
        st.dataframe(anomalies_only.reset_index(),
            use_container_width=True)
        if not anomalies_only.empty:
            st.download_button(
                label="Завантажити аномалії CSV",
                data=convert_df_to_csv(anomalies_only.reset_index()),
                file_name="anomalies.csv",
                mime="text/csv",
            )

    st.session_state.anomaly_summary = f"""
Розділ: Виявлення аномалій
Обрана колонка: {anomaly_column}
Поріг z-score: {threshold}
Кількість значень: {len(anomaly_df)}
Кількість аномалій: {int(anomalies_only.shape[0])}
Частка аномалій: {(anomalies_only.shape[0] / len(anomaly_df)) * 100:.2f}%
"""

# =====
# 4. Кластерний аналіз
# =====

with st.expander("4. Кластерний аналіз", expanded=False):

    with st.expander("Інструкція: як користуватися кластерним аналізом",
        expanded=False):
        st.info("""
        ### Як користуватися кластерним аналізом?

        Кластеризація автоматично ділить об'єкти на групи зі схожими
        характеристиками.

        Система порівнює числові значення та шукає схожі об'єкти.
        Наприклад:
        - клієнтів зі схожими покупками;
        - товари зі схожими продажами;
        - схожі записи у даних.

        #### Як правильно обирати колонки?
         Найкраще працюють:
        - числові показники;
        - колонки, які реально описують об'єкти.

         Не рекомендується:
        - ID або номери записів;
        - випадкові значення.

        #### Як обрати кількість кластерів?
        - **2-3** → простий поділ на великі групи;
        - **4-6** → детальніша сегментація;

```

- занадто велика кількість кластерів може зробити результат складним для інтерпретації.

```
#### Як інтерпретувати результат?
- точки одного кольору – схожі об'єкти;
- різні кольори – різні групи;
- чим далі кластери один від одного, тим сильніше вони відрізняються.
"""
```

```
df = st.session_state.working_df.copy()
numeric_columns = get_numeric_columns(df)

if len(numeric_columns) < 2:
    st.warning("Для кластеризації потрібно щонайменше 2 числові колонки")
else:
    cl_coll1, cl_col2 = st.columns([2, 1])

    selected_features = cl_coll1.multiselect(
        "Оберіть числові колонки для кластеризації",
        options=numeric_columns,
        default=numeric_columns[: min(3, len(numeric_columns))],
        key="clustering_selected_features",
    )

    n_clusters = cl_col2.slider(
        "Кількість кластерів",
        min_value=2,
        max_value=10,
        value=3,
        key="kmeans_n_clusters_slider",
    )

    if len(selected_features) < 2:
        st.info("Оберіть щонайменше 2 числові колонки")
    else:
        X_scaled, features_df = prepare_features_for_clustering(df,
selected_features)

        if any("id" in col.lower() for col in selected_features):
            st.warning(
                "Ви обрали колонку, схожу на ID. Такі колонки зазвичай не
рекомендується використовувати для кластеризації."
            )

        with st.expander("Підказка для вибору кількості кластерів",
expanded=False):
            st.info("""
            ### Метод ліктя

            Метод ліктя допомагає приблизно оцінити, скільки кластерів
            варто обрати.

            Дивіться на графік і шукайте місце, де лінія починає помітно
            вирівнюватися.
            Це місце часто вважається хорошим варіантом кількості
            кластерів.
            """)

            wcss = []
            max_k = min(10, len(df))

            for k in range(1, max_k + 1):
```

```

n_init=10)

    elbow_model = KMeans(n_clusters=k, random_state=42,

    elbow_model.fit(X_scaled)
    wcss.append(elbow_model.inertia_)

fig_elbow = go.Figure()

fig_elbow.add_trace(
    go.Scatter(
        x=list(range(1, max_k + 1)),
        y=wcss,
        mode="lines+markers",
        name="WCSS"
    )
)

fig_elbow.update_layout(
    title="Метод ліктя (Elbow Method)",
    xaxis_title="Кількість кластерів",
    yaxis_title="WCSS / Inertia",
    height=400
)

st.plotly_chart(fig_elbow, use_container_width=True)

model = KMeans(n_clusters=n_clusters, random_state=42, n_init=10)
clusters = model.fit_predict(X_scaled)

result_df = df.copy()
result_df["cluster"] = clusters

st.success("Кластеризацію виконано")

plot_x = selected_features[0]
plot_y = selected_features[1]

fig = go.Figure()

fig.add_trace(
    go.Scatter(
        x=result_df[plot_x],
        y=result_df[plot_y],
        mode="markers",
        marker=dict(
            color=result_df["cluster"],
            showscale=True,
            colorbar=dict(title="Кластер"),
        ),
        text=[f"Кластер: {cluster}" for cluster in
result_df["cluster"]],
        name="Об'єкти",
        hovertemplate=(
            f"{plot_x}: %{x}<br>"
            f"{plot_y}: %{y}<br>"
            "%{text}<extra></extra>"
        ),
    )
)

fig.update_layout(
    title="Результат кластеризації K-Means",
    xaxis_title=plot_x,
    yaxis_title=plot_y,

```

```

        height=520,
    )

    st.plotly_chart(fig, use_container_width=True)

    with st.expander("Переглянути таблицю з кластерами"):
        st.dataframe(result_df[selected_features + ["cluster"]],
            use_container_width=True)
        st.download_button(
            label="Завантажити результати кластеризації CSV",
            data=convert_df_to_csv(result_df[selected_features +
["cluster"]]),
            file_name="clustering_results.csv",
            mime="text/csv",
        )

        cluster_summary =
result_df.groupby("cluster")[selected_features].mean().reset_index()
        st.subheader("Середні значення ознак у кластерах")
        st.dataframe(cluster_summary, use_container_width=True)
        st.session_state.clustering_summary = f"""
Розділ: Кластерний аналіз
Обрані ознаки: {selected_features}
Кількість кластерів: {n_clusters}
Кількість об'єктів: {len(result_df)}
Середні значення ознак по кластерах:
{cluster_summary.to_string(index=False)}
"""

with st.expander("5. AI-висновок за результатами аналізу", expanded=False):
    st.markdown(
        """
        Натисніть кнопку нижче, щоб згенерувати коротке пояснення результатів.
        AI буде використано тільки після натискання кнопки.
        """
    )

    available_summaries = []

    if "time_series_summary" in st.session_state:
        available_summaries.append(st.session_state.time_series_summary)

    if "anomaly_summary" in st.session_state:
        available_summaries.append(st.session_state.anomaly_summary)

    if "clustering_summary" in st.session_state:
        available_summaries.append(st.session_state.clustering_summary)

    if not available_summaries:
        st.info("Спочатку виконайте хоча б один аналіз: часовий ряд, аномалії
або кластеризацію.")
    else:
        analysis_text = "\n\n".join(available_summaries)

    if st.button("Згенерувати висновок"):
        if not gemini_api_key:
            st.warning("Введіть Gemini API key у боковій панелі.")
        else:
            with st.spinner("Генерується висновок..."):
                try:
                    ai_summary = generate_gemini_summary(
                        api_key=gemini_api_key,
                        analysis_text=analysis_text

```

```
)  
    st.subheader("Згенерований висновок")  
    st.write(ai_summary)  
except Exception as exc:  
    st.error(f"Не вдалося згенерувати висновок: {exc}")
```

ДОДАТОК Б ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

Система підтримки прийняття рішень для аналізу табличних наборів даних

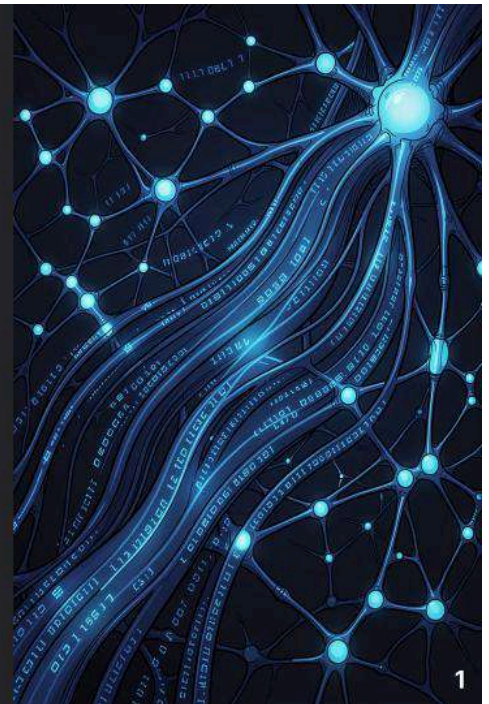
Бакалаврська кваліфікаційна робота

Автор: Бондаренко Владислав Михайлович

Спеціальність: 124 «Системний аналіз»

Науковий керівник: старший викладач кафедри ММСА, доктор філософії Левенчук Людмила Борисівна

КПІ ім. Ігоря Сікорського · 2026



1

Актуальність, мета та завдання

Актуальність

- Експоненційне зростання обсягів даних у всіх галузях
- Складність ручного аналізу великих масивів інформації
- Зростаюча потреба в автоматизації прийняття рішень

Мета роботи

Розробити систему підтримки прийняття рішень для комплексного аналізу табличних наборів даних із використанням методів машинного навчання та AI.

Ключові завдання

01

Аналіз часових рядів

02

Виявлення аномалій

03

Кластеризація даних

04

Веб-застосунок + AI-висновки

2

Використані технології

Сучасний стек Python-екосистеми для повного циклу аналізу даних — від завантаження до візуалізації та AI-висновків.



Python

Основна мова програмування для аналізу даних та ML



Pandas & NumPy

Обробка, трансформація та математичні операції з даними



Scikit-learn

Реалізація алгоритмів ML: регресія, кластеризація, аномалії



Plotly & Streamlit

Інтерактивні графіки та швидке розгортання веб-інтерфейсу

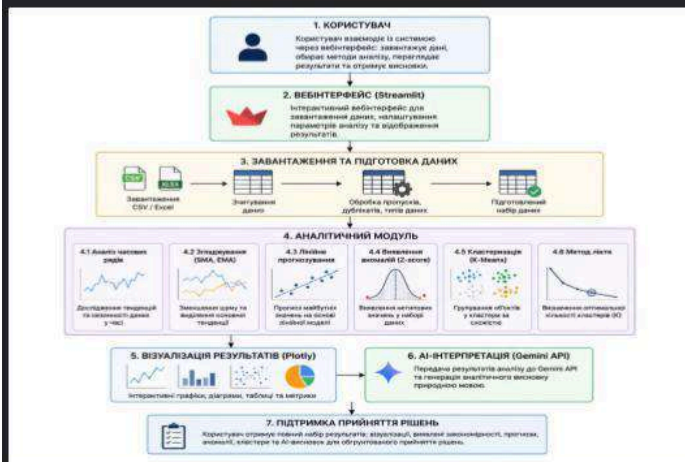


Gemini API

Генерація аналітичних висновків на основі результатів аналізу

3

Архітектура системи



Принцип роботи

Система реалізує повний цикл обробки даних: від завантаження файлу до формування аналітичного звіту з AI-висновками.

Три аналітичні модулі (часові ряди, аномалії, кластеризація) виконуються паралельно на підготовлених даних, забезпечуючи комплексний огляд.

4

Тренувальний набір даних

Для демонстрації роботи розробленої системи було використано тестовий набір даних інтернет-магазину. Датасет містить інформацію про відвідуваність, продажі, маркетингові витрати та показники обслуговування клієнтів. Наявність часової складової дозволяє виконувати аналіз часових рядів і прогнозування, а числові показники використовуються для виявлення аномалій та кластеризації даних.

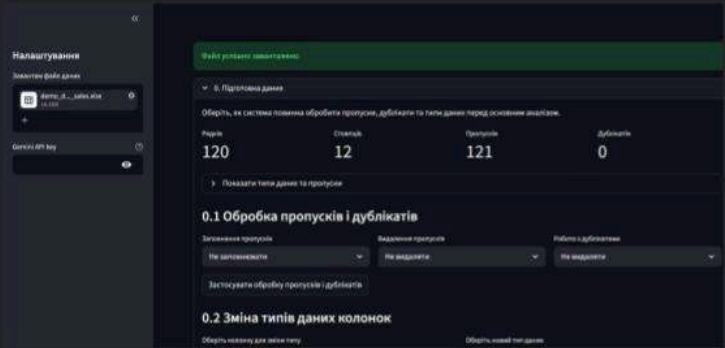
Основні характеристики датасету

- Кількість записів: 120
- Кількість атрибутів: 12
- Період даних: з 01.01.2026 по 30.04.2026
- Тип даних: дані інтернет-магазину
- Регіони: Kyiv, Lviv, Odesa, Dnipro
- Категорії товарів: Electronics, Clothing, Home, Beauty

Поля набору даних

- date — дата
- region — періон
- product_category — категорія товару
- visitors — кількість відвідувачів
- orders — кількість замовлень
- revenue — дохід
- avg_order_value — середній чек
- marketing_spend — витрати на маркетинг
- discount_percent — розмір знижки
- customer_rating — рейтинг клієнтів
- delivery_time_days — час доставки
- event_note — спеціальні події або примітки

Підготовка даних



Основні можливості модуля



Завантаження

Підтримка форматів CSV та XLSX із автоматичним визначенням кодування



Перегляд структури

Відображення перших рядків, типів стовпців та базової статистики



Очищення

Обробка пропусків, видалення дублікатів, зміна типів даних

Виявлення аномалій



Метод Z-Score

Статистичний метод виявлення викидів на основі відхилення від середнього значення.

$$Z = \frac{x - \mu}{\sigma}$$

Точки з $|Z| > 3$ позначаються як аномальні (синіми на графіку).

⚡ Швидкість

🔍 Простота

🤖 Автоматизація

9

Кількісні характеристики модуля пошуку аномалій

Кількість аномалій

2

Частка аномалій

1.67%

▼ Переглянути знайдені аномалії

	index	value	z_score	is_anomaly
0	24	6034.15	3.0571	☒
1	88	6198.74	3.2121	☒

Завантажити аномалії CSV

10

Кластерний аналіз



Реалізовані методи

K-Means

Ітеративний алгоритм групування об'єктів у K кластерів за мінімізацією відстані до центрів

Elbow Method

Автоматичне визначення оптимальної кількості кластерів за «вигином» графіка інерції

🟢 **Результат:** Автоматичне групування схожих об'єктів без попереднього маркування даних.

11

Кількісні характеристики модуля кластеризації

Переглянути таблицю з кластерами

cluster	visitors	orders	revenue	marketing_spend	customer_rating
0	514	28	1370.75	187.93	4.81
1	619	36	1874.26	111.94	4.27
2	806	49	2286.89	126.83	4.82
3	736	46	2218.54	138.76	4.87
4	374	19	1133.81	144.82	4.86
5	371	20	1177.64	149.22	4.31
6	393	21	1089.79	144.89	4.82
7	589	34	1874.83	130.91	4.23
8	742	41	1861.02	97.08	4.37
9	861	51	2283.5	187.64	4.16

Завантажити результати кластеризації CSV

Середні значення ознак у кластерах

cluster	visitors	orders	revenue	marketing_spend	customer_rating
0	0	1158.6818	79.8964	3828.8539	187.7925
1	1	611.5882	33.7941	1874.3676	133.1972
2	2	809.8517	53.3122	2551.9761	156.7621
3	3	1578	108	6186.14	663.32

12

Gemini API — Генерація аналітичних висновків

Призначення та функціональність

Gemini API є інтерфейсом взаємодії з моделлю штучного інтелекту Gemini для автоматизації інтерпретації результатів аналізу даних.

Обробка результатів

Система формує структуровану інформацію про результати аналізу.

Генерація висновків

Gemini API створює зрозумілі текстові висновки на основі даних.

Автоматизація звітів

Скорочує час на підготовку аналітичних звітів.

Рекомендації

Надає рекомендації щодо трендів, аномалій та структури даних.



Результат: Швидка інтерпретація результатів та обґрунтовані рішення на основі AI-аналізу

13

Аналітичний висновок

Згенерований висновок

Ось короткий аналіз ваших даних, підготовлений зрозумілою мовою:

1. Про що ці результати?

Ми проаналізували роботу вашого бізнесу за чотири місяці (з січня по квітень 2026 року). Звіт показує, як змінювався витіток щодня, чи були нетипові сплески продажів, та розділяє всі робочі дні на 4 групи за їхньою ефективністю (від найслабших до найуспішніших).

2. Головні закономірності

- **Загальний стан:** Середній витіток за день складає близько 2 788 грн. Проте останній показник у звіті (4 499 грн) значно вищий за середній, що може свідчити про позитивну тенденцію наприкінці квітня.
- **Зв'язок маркетингу та продажів:** Чітко видно, що найбільший витіток (Кластер 3) досягається за рахунок значного збільшення витрат на маркетинг.
- **Ефективність груп:**
 - «Пікові дні» (Кластер 3): Найвищі показники (витіток ~6 200 грн), але й витрати на рекламу тут у 4 рази вищі, ніж зазвичай.
 - «Стабільна середина» (Кластер 0): Дуже вдалі дні, де витіток високий (~4 000 грн), але витрати на маркетинг залишаються помірними. Це найбільш вигідний режим роботи.
 - «Слабкі дні» (Кластер 1): Дні з найнижчим відвідуванням та витратом (в середньому 1 834 грн).

3. На що варто звернути увагу?

1. **Аномалії:** Система виявила 2 дні з нетиповими показниками (різкі стрибки або падіння). Варто перевірити дати цих аномалій: це міг бути або технічний збій, або надзвичайно успішна акція, досвід якої варто повторити.
2. **Витрати на маркетинг:** У «пікові дні» (Кластер 3) ви витратите на маркетинг 663 грн, щоб отримати 6 199 грн витітку. У «стабільні дні» (Кластер 0) ви витратите лише 166 грн, щоб отримати 4 004 грн. Висновок: дні з Кластеру 0 є більш прибутковими з точки зору окупності реклами.
3. **Клієнтський досвід:** Рейтинг клієнтів (customer_rating) залишається стабільно високим (4.0–4.3) у всіх групах, що є дуже хорошим показником — якість сервісу не падає навіть при великій кількості замовлень.

14

Результати роботи



Аналіз часових рядів
SMA, EMA, прогнозування
трендів



Виявлення аномалій
Z-Score, автоматичне
маркування викидів



Кластеризація
K-Means + Elbow Method



Веб-інтерфейс
Streamlit + Plotly, інтерактивні графіки



AI-висновки
Інтеграція Gemini API для генерації звітів

Висновки

Отримані результати

- Автоматизовано повний цикл аналізу табличних даних
- Реалізовано сучасні методи ML: часові ряди, аномалії, кластеризація
- Створено веб-застосунок на Python + Streamlit
- Інтегровано генерацію AI-висновків через Gemini API

Практична цінність

Розроблена система може бути застосована для аналізу фінансових, бізнесових та наукових даних, скорочуючи час на прийняття рішень.

- Робота відповідає вимогам спеціальності 124 «Системний аналіз» та демонструє комплексний підхід до розробки аналітичних систем.

Дякую за увагу!