Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Програмна реалізація статистичних методів для виявлення аномалій у даних із застосуванням мови програмування Python»

Виконав: студент 4 курсу
групи 4ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Вараниці М.С.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Катєльніков Д. І.
(прізвище та ініціали)

Рецензент: к.т.н., доц.каф.ЗІ Маліновський В.І.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ
д.т.н., проф. Романюк О. Н.
«04» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
_____ О. Н. Романюк
« 21 » березня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Вараниці Миколі Сергійовичу

1. Тема роботи – програмна реалізація статистичних методів для виявлення аномалій у даних із застосуванням мови програмування Python.

Керівник роботи: Катєльніков Денис Іванович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. №97.

2. Строк подання студентом роботи 2 червня 2025.

3. Вихідні дані до роботи: дії програмної реалізації – завантаження та обробка табличних даних у форматах CSV, Excel, JSON; застосування статистичних методів виявлення аномалій; візуалізація даних та результатів аналізу через гістограми, діаграми розмаху, часові ряди; експорт результатів у табличні файли та графічні зображення; статистичні методи – Z-score аналіз; міжквартильний розмах (IQR); контрольні карти Шухарта; Обробка даних – завантаження через pandas для роботи з різними форматами файлів; статистична обробка через NumPy та SciPy для математичних розрахунків; контроль якості – перевірка типів даних та їх відповідності для статистичного аналізу; валідація порогових значень та параметрів методів; кінцевий результат – інтерактивна програма з графічним інтерфейсом для статистичного аналізу аномалій у даних.

4. Зміст текстової частини: вступ; аналіз розвитку систем виявлення аномалій у даних та формувння завдань дослідження; розробка модульної архітектури системи; розробка програмних модулів реалізації системи; тестування програми; висновки; список використаних джерел; додатки.

5. Перелік ілюстративної частини: титульний слайд; актуальність теми; мета, об'єкт дослідження та предмет дослідження; постановка задач дослідження; наукова новизна отриманих результатів; порівняльний аналіз аналогів; блок-схема загальної роботи системи; блок-схема виявлення аномалій методом Z-score; блок-схема виявлення аномалій методом IQR; блок-схема виявлення аномалій методом карти Шухарта; блок-схема реалізації системи виявлення

аномалій; блок-схема реалізації системи виявлення аномалій; блок-схема узагальненого тестування методів виявлення аномалій; блок-схема візуалізації аномалій у часовому ряді; графічна схема інтерфейсу головного вікна системи; результат тестуваннння програмної реалізації; апробація та публікація матеріалів бакалаврської кваліфікаційної роботи; фінальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Катєльніков Д.І., к.т.н., доц. каф. ПЗ	24.03.25	01.06.25

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз розвитку систем виявлення аномалій у даних та формування завдань дослідження	25.03.25 – 31.03.2025	вик
2	Розробка модульної архітектурної системи	31.03.25 – 05.04.2025	вик
3	Удосконалення статичних методів виявлення аномалій	05.03.2025 – 07.04.2025	вик
4	Розробка адаптивної системи візуалізації результатів аналізу аномалій	07.04.2025 – 12.04.2025	вик
5	Розробка узагальненого підходу до тестування методів виявлення аномалій	12.04.2025 – 23.04.2025	вик
6	Розробка програмної реалізації статистичних методів для виявлення аномалій у даних	23.04.2025 – 01.05.2025	вик
7	Тестування програмної реалізації	01.05.2025 – 10.05.2025	вик
8	Оформлення матеріалів до захисту БКР	10.05.2025 – 30.05.25	вик

Студент _____ **М. С. Варанiця**
(підпис) (ініціали та прізвище)

Керівник бакалаврської кваліфікаційної роботи _____ **Д. І. Катєльніков**
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

УДК 004.4

Вараниця М.С. Програмна реалізація статистичного виявлення аномалій з використанням методів IQR, Z-score та контрольних карт Шухарта: бакалаврська дипломна робота зі спеціальності 121 Інженерія програмного забезпечення, освітнія програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 68 с. На укр. мові. Бібліогр. : 22 назв.; рис. : 21; табл. : 3.

У бакалаврській дипломній роботі було визначено доцільність, практичну цінність та мету розробки, яка полягає у створенні програмного додатку для статистичного виявлення аномалій у даних з використанням методів IQR, Z-score та контрольних карт Шухарта.

Запропоновано алгоритм комплексного аналізу даних, особливість якого полягає у поєднанні трьох надійних статистичних методів виявлення аномалій. Розроблено інтерактивну систему візуалізації результатів, яка забезпечує наочне представлення виявлених аномалій. Удосконалено спосіб організації інтерфейсу користувача, який відрізняється викори+станням системи вкладок.

Програмне забезпечення розроблено із використанням мови програмування Python та бібліотек PyQt5, matplotlib, seaborn, pandas і numpy. Під час розробки використано середовище розробки PyCharm для налагодження та оптимізації коду. У результаті виконання бакалаврської дипломної роботи розроблено програмну реалізацію, працездатність і правильність роботи якого перевірено, підготовлена інструкція користувача та визначено системні вимоги, реалізовано можливість завантаження даних з різних форматів файлів (CSV, Excel, JSON).

Отримані в бакалаврській дипломній роботі результати можна використати для ефективної ідентифікації та аналізу аномалій у різних типах даних з можливістю збереження результатів для подальшого використання.

Ключові слова: виявлення аномалій, статистичний аналіз, IQR метод, Z-score метод, контрольні карти Шухарта, Python-додаток, візуалізація даних.

ABSTRACT

UDC 004.4

Varanytsya M.S. Software implementation of statistical anomaly detection using IQR, Z-score and Shewhart control charts: bachelor's thesis in specialty 121 Software Engineering, educational program - Software Engineering. Vinnytsia: VNTU, 2025. 68 p.

In Ukrainian. Bibliography: 22 titles; fig.: 21; tab.: 3.

The bachelor's thesis determined the feasibility, practical value and purpose of the development, which consists in creating a software application for statistical anomaly detection in data using IQR, Z-score and Shewhart control charts.

An algorithm for complex data analysis is proposed, the peculiarity of which is the combination of three reliable statistical methods for anomaly detection. An interactive system for visualizing the results has been developed, which provides a visual representation of the detected anomalies. The method of organizing the user interface has been improved, which is distinguished by the use of a tab system, which allows for increased ease of navigation between different functional modules.

The software was developed using the Python programming language and the PyQt5, matplotlib, seaborn, pandas, and numpy libraries. During development, the PyCharm development environment was used to debug and optimize the code. As a result of the bachelor's thesis, a software implementation was developed, the operability and correctness of which were tested, user instructions were prepared, and system requirements were determined, and the ability to load data from various file formats (CSV, Excel, JSON) was implemented.

The results obtained in the bachelor's thesis can be used for effective identification and analysis of anomalies in various types of data with the ability to save the results for further use.

Keywords: anomaly detection, statistical analysis, IQR method, Z-score method, Shewhart control charts, Python application, data visualization.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Обґрунтування вибору теми дослідження.

У сучасному цифровому суспільстві обсяг даних, які щодня генеруються, зростає з надзвичайною швидкістю. Від медичних вимірювань і фінансових транзакцій до даних сенсорів в інтернеті речей це усі ці потоки інформації потребують не лише зберігання, а й глибокого аналізу. Проте однією з найважливіших і водночас найвразливіших точок у цьому процесі є наявність аномалій-спостережень, що суттєво відрізняються від решти даних і можуть свідчити як про критичні помилки, так і про нові, ще не вивчені явища.

В умовах стрімкого розвитку автоматизованих систем прийняття рішень, де навіть найменше викривлення вхідної інформації може спричинити значні втрати або збої, виявлення аномалій набуває вирішального значення. Особливо це стосується галузей, що працюють з високочутливою інформацією – фінансового сектора, медицини, кібербезпеки, виробництва. У таких умовах застосування надійних статистичних методів дозволяє своєчасно фіксувати відхилення, які можуть вказувати на шахрайські дії, технічні несправності або загрози безпеці.

У той же час, стрімкий розвиток програмного забезпечення надає нові можливості для інтеграції класичних методів аналізу у сучасні цифрові платформи. Серед мов програмування, що забезпечують баланс між простотою синтаксису, функціональністю та потужною бібліотечною підтримкою, особливе місце посідає Python [1]. Саме ця мова дозволяє поєднувати наукову точність із високою швидкістю розробки, забезпечуючи реалізацію складних статистичних алгоритмів у доступному та зрозумілому вигляді. Актуальність теми також підсилюється потребою в автоматизації процесу аналізу даних у реальному часі [2]. У сучасному світі недостатньо лише виявити проблему – необхідно зробити це якомога швидше, часто у форматі «онлайн».

Саме тому розробка програмного рішення, що забезпечує реалізацію статистичних методів виявлення аномалій із можливістю гнучкого застосування до різних типів даних, відповідає нагальним викликам цифрової епохи.

Таким чином, вибір теми «Розробка програмної реалізації статистичних методів для виявлення аномалій у даних із застосуванням мови програмування Python» є цілком обґрунтованим: вона поєднує актуальність практичної задачі з науковою глибиною, а також надає можливість створення прикладного інструменту, що має широке поле застосування в сучасних інформаційних системах [3].

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення інтерактивності та наочності при виявлення аномалій у даних за рахунок розробки програмного забезпечення, яке реалізує статистичні методи виявлення аномалій у структурованих даних із використанням мови програмування Python.

Основними задачами дослідження є:

1. Визначити оптимальні статистичні методи для виявлення аномалій в різних типах даних.

2. Створити інтуїтивно зрозумілий та функціональний графічний інтерфейс користувача.

3. Розробити програмний додаток, що інтегрує в себе всі необхідні функції для ефективного аналізу аномалій.

4. Провести всебічне тестування розробленої програми для забезпечення її надійності та точності в реальних умовах.

Об’єкт дослідження.

Об’єктом дослідження є процес розробки програмної реалізації статистичних методів виявлення аномалій у структурованих даних із використанням мови програмування Python.

Предмет дослідження - статистичні методи виявлення аномалій у структурованих даних та засоби інтерактивної візуалізації результатів.

Методи дослідження.

У ході дослідження були використані: теорія ймовірностей та математична статистика – для виявлення аномалій у структурованих даних; методи машинного навчання – для порівняння результатів; програмна інженерія – для побудови модульної структури застосунку; об’єктно-орієнтоване програмування – для реалізації програмних компонентів; комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Новизна отриманих результатів.

1. Подальшого розвитку отримав метод інтеграції модулів статистичного виявлення аномалій у структурованих даних, який на відміну від існуючих методів інтеграції, використовує модульну структуру, що дозволяє не тільки інтегрувати нові методи виявлення аномалій, але й модифікувати існуючі модулі, адаптуючи їх до нових форматів даних.

2. Подальшого розвитку отримав метод візуалізації результатів аналізу аномалій, який на відміну від існуючих методів, використовує інтерактивну візуалізацію результатів, що дає змогу не тільки виявляти але й інтерпретувати аномальні спостереження і приймати обґрунтовані рішення.

Практична цінність отриманих результатів полягає в тому, що на основі отриманих в бакалаврській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби для статистичного виявлення аномалій у структурованих даних. Створене програмне забезпечення може бути використане у сфері фінансового моніторингу, біоаналітики, контролю якості технічних процесів, а також в освітньому процесі для навчання студентів основам статистичного аналізу та Python-програмування. Система забезпечує інтерфейс для введення, обробки, аналізу та візуалізації даних, що дає змогу ефективно виявляти та інтерпретувати аномальні спостереження.

Особистий внесок здобувача.

У бакалаврській кваліфікаційній роботі всі результати дослідження отримані автором самостійно.

Апробація матеріалів бакалаврської дипломної роботи. Основні положення бакадаврської кваліфікаційної роботи доповідалися та обговорювалися на конференціях «LIII Всеукраїнській науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (2025)» та Міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» та опубліковані в тезах доповіді [5,6].

Публікації. За тематикою дослідження опубліковано дві наукових праці, що доповідалась на конференціях «LIII Всеукраїнській науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (2025)» і Міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» та опубліковано в тезах доповіді [5,6].

1 АНАЛІЗ РОЗВИТКУ СИСТЕМ ВИЯВЛЕННЯ АНОМАЛІЙ У ДАНИХ ТА ФОРМУЛЮВАННЯ ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Аналіз стану систем виявлення аномалій у даних

У сучасних умовах стрімкого розвитку цифрових технологій та збільшення обсягів даних виявлення аномалій стає важливою складовою забезпечення надійності, безпеки та ефективності інформаційних систем. Аномалії, тобто відхилення від типової поведінки даних, можуть свідчити про потенційні проблеми, збої, шахрайські дії або загрози, тому їх своєчасне виявлення має важливе прикладне значення у багатьох сферах, зокрема у фінансах, охороні здоров’я, кібербезпеці, промисловості, логістиці та інтернет-сервісах. Розробка програмної реалізації для автоматизованого пошуку аномалій дає змогу підвищити якість аналізу даних, своєчасно виявляти відхилення та реагувати на них до того, як вони призведуть до критичних наслідків.

Однією з головних переваг систем виявлення аномалій є здатність автоматично аналізувати великі масиви інформації та виокремлювати підозрілі елементи без потреби у постійному втручанні людини. Це значно оптимізує роботу аналітичних відділів і дозволяє зосередити ресурси на критичних випадках. При цьому ефективність таких систем значною мірою залежить від обраних методів обробки та аналізу даних. У практиці виявлення аномалій застосовуються різноманітні підходи: від класичних статистичних методів, таких як Z-score або міжквартильний розмах (IQR) [4], до сучасних алгоритмів машинного навчання, здатних виявляти приховані зв’язки та складні закономірності.

Сучасні програмні інструменти, зокрема мова програмування Python, забезпечують потужні можливості для реалізації подібних систем. Python активно використовується у сфері аналізу даних завдяки своїй простоті, гнучкості та широкому набору спеціалізованих бібліотек. Наприклад, pandas та numpy забезпечують ефективну роботу з масивами та таблицями, scikit-learn і pyod надають доступ до десятків алгоритмів виявлення аномалій, тоді як

matplotlib і seaborn дозволяють візуалізувати отримані результати у зручній формі. Це дозволяє створювати не лише функціональні, а й інтуїтивно зрозумілі для користувача рішення.

Попри велику кількість наявних бібліотек і методів, у багатьох випадках користувачі стикаються з труднощами у застосуванні таких інструментів через складність налаштування, відсутність зручного інтерфейсу або надлишкову спеціалізацію програм. Тому постає потреба у створенні програмних засобів, які б поєднували простоту використання, гнучкість у виборі методів аналізу та можливість візуалізації результатів. Такі програми повинні бути достатньо універсальними, щоб адаптуватися до різних типів даних, але водночас досить простими, щоб бути доступними для користувачів без глибоких знань у галузі статистики або машинного навчання.

Отже, розробка програмної реалізації для виявлення аномалій у даних із використанням Python є актуальним напрямом у сфері інформаційних технологій. Такий інструмент здатен стати ефективним засобом підтримки прийняття рішень у різних прикладних задачах, забезпечити підвищену надійність систем та скоротити витрати, пов’язані з обробкою помилкових або небажаних сценаріїв. У світі, де якість даних має вирішальне значення, важливо впроваджувати адаптивні та інтелектуальні підходи до їх аналізу, що й визначає актуальність обраної теми.

1.2 Порівняльний аналіз аналогів

Зі стрімким розвитком технологій аналізу даних та зростанням обсягів інформації, що потребує обробки, область виявлення аномалій стає все більш важливою для різних галузей. Ефективна ідентифікація аномальних спостережень допомагає організаціям оперативно реагувати на потенційні проблеми, запобігати помилкам та оптимізувати процеси прийняття рішень.

Завдяки системам виявлення аномалій користувачі можуть швидко знаходити відхилення в даних, аналізувати причини їх виникнення та вживати відповідні заходи. Розвиток у цій області призвів до появи різноманітних

програмних рішень, які відрізняються методами аналізу, інтерфейсом та функціональними можливостями. Такі системи забезпечують автоматизований аналіз даних, вибір оптимальних алгоритмів виявлення аномалій та візуалізацію результатів для полегшення інтерпретації.

Серед багатьох аналогів розроблюваної програми варто виділити декілька популярних рішень: Python Outlier Detection (PyOD), Anomaly Detection Toolkit (ADTK), Isolation Forest від scikit-learn та Elasticsearch Machine Learning, кожне з яких пропонує свої унікальні можливості та підходи до виявлення аномалій у даних.

На рисунку 1.1 зображено Python Outlier Detection (PyOD), який є всеосяжною бібліотекою Python для виявлення аномалій у багатовимірних даних. PyOD включає понад 40 алгоритмів виявлення аномалій, що робить його однією з найбільш повних бібліотек у цій області. Бібліотека підтримує як класичні алгоритми (статистичний аналіз та моделі на основі близькості), так і сучасні методи (глибоке навчання та комбіновані моделі) [7].

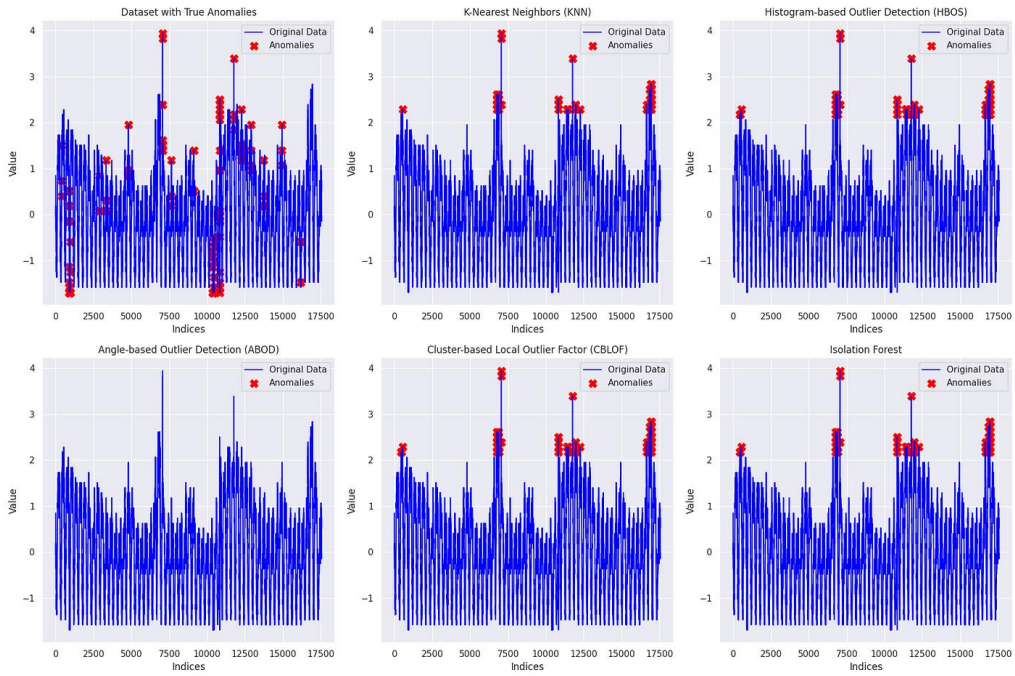

Рисунок 1.1 – Приклад роботи PyOD із візуалізацією аномалій

На рисунку 1.2 зображено Anomaly Detection Toolkit (ADTK), який є Python-пакетом для виявлення аномалій у часових рядах. ADTK підтримує різні типи аномалій у часових рядах, включаючи точкові викиди, зміни рівнів, тренди та сезонні відхилення. Бібліотека містить колекцію готових детекторів аномалій та дозволяє користувачам створювати власні моделі шляхом комбінування існуючих компонентів [8].

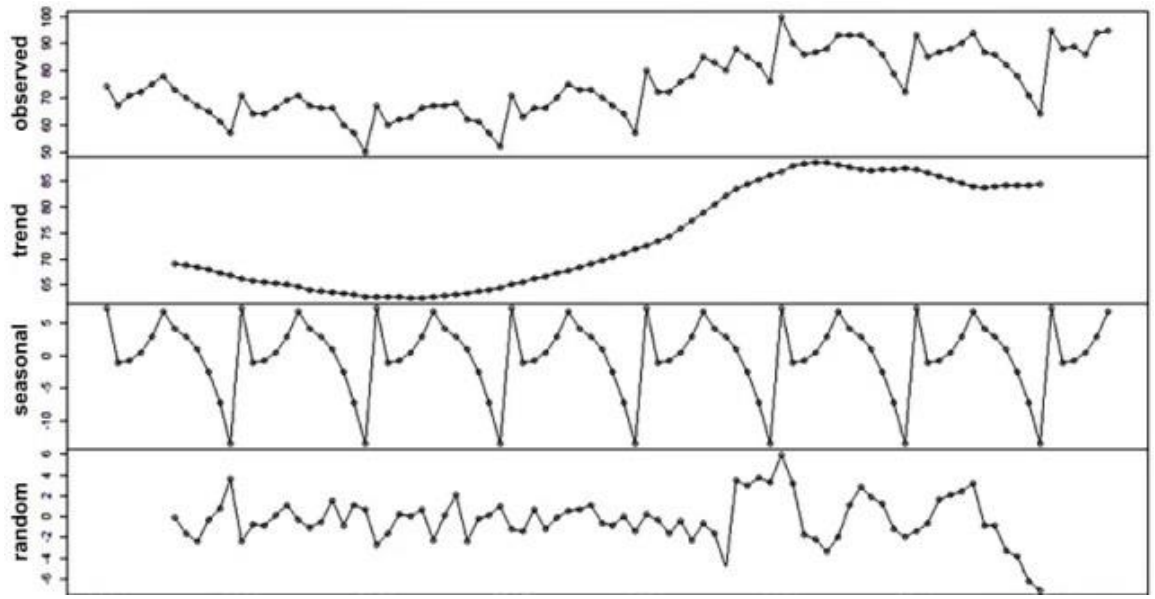

Рисунок 1.2 – Приклад виявлення аномалій у часових рядах з ADTK

На рисунку 1.3 зображено алгоритм Isolation Forest, який є модулем бібліотеки scikit-learn для виявлення аномалій. Цей алгоритм ефективно ізолює спостереження шляхом випадкового вибору функцій та їх випадкового розділення, що робить його особливо ефективним для багатовимірних даних. Алгоритм має високу швидкість роботи та низькі вимоги до пам'яті, що робить його привабливим для великих наборів даних.

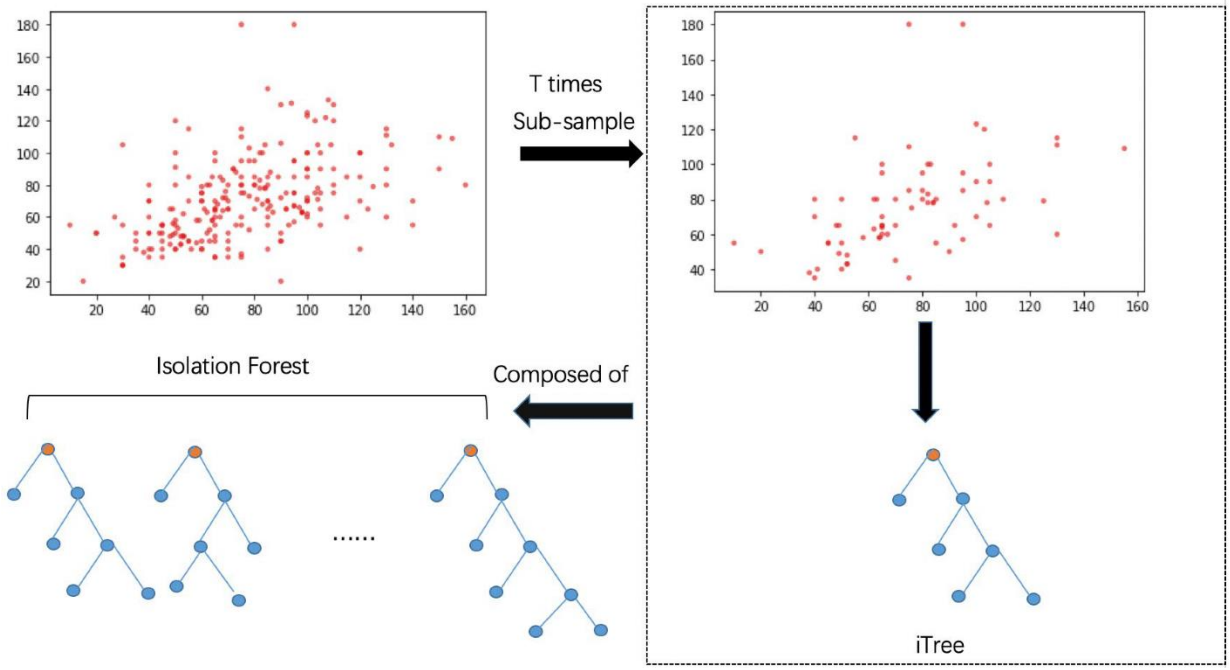

Рисунок 1.3 – Приклад роботи Isolation Forest для виявлення аномалій

На рисунку 1.4 зображено Elasticsearch Machine Learning, який є компонентом Elastic Stack, що забезпечує автоматизоване виявлення аномалій у даних часових рядів. Цей інструмент використовує методи машинного навчання без учителя для моделювання нормальної поведінки даних та виявлення відхилень. Elasticsearch ML особливо корисний для аналізу даних журналів, метрик та бізнес-KPI в режимі реального часу [9].

Elasticsearch ML надає гнучкі інструменти для налаштування чутливості виявлення, дозволяючи користувачам балансувати між виявленням фактичних аномалій та уникненням хибних спрацьовувань. Система також підтримує аналіз сезонності та тренди, що важливо для точного виявлення аномалій у циклічних даних.

Крім того, Elasticsearch ML забезпечує інтеграцію з Kibana для створення інтерактивних панелей моніторингу, які відображають результати виявлення аномалій та дозволяють глибше аналізувати підозрілі випадки. Цей інструмент добре підходить для корпоративних сценаріїв, де необхідно обробляти великі обсяги даних у режимі реального часу.

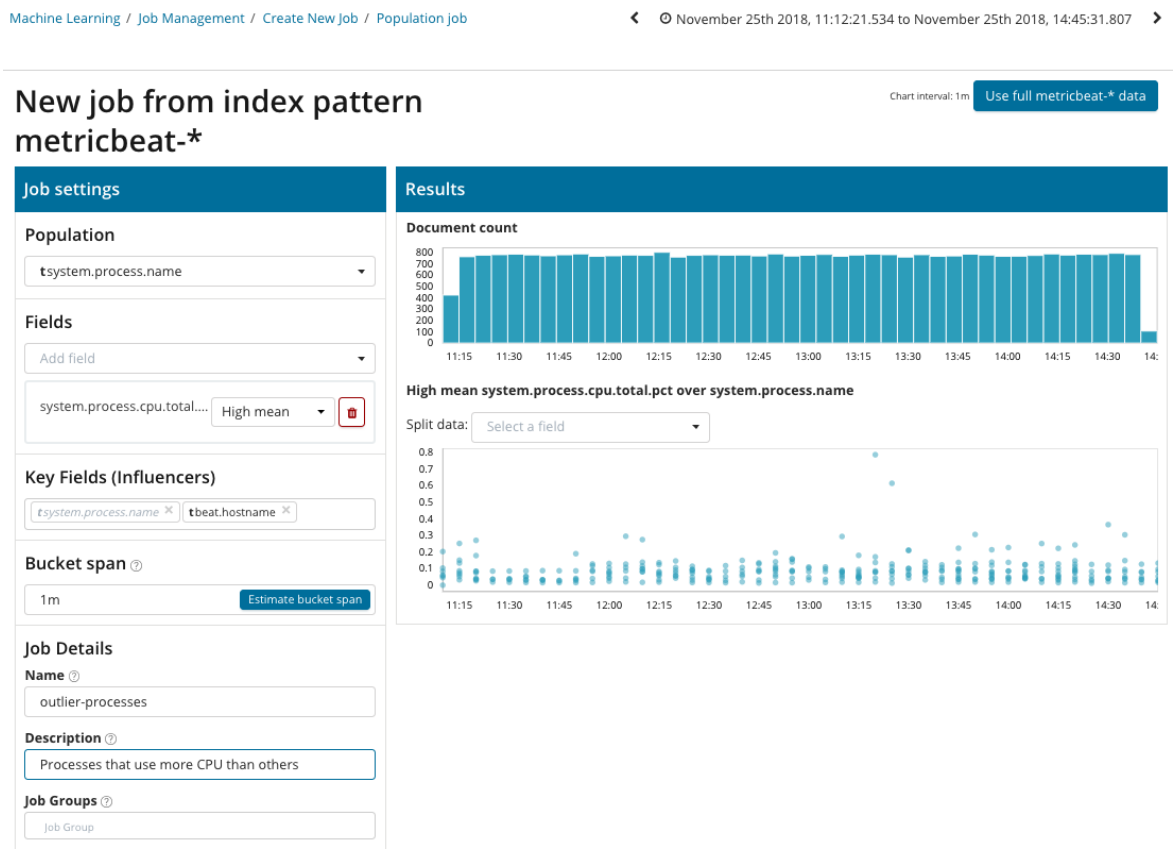

Рисунок 1.4 – Приклад інтерфейсу Elasticsearch ML для виявлення аномалій

На рисунку 1.5 зображено AWS Lookout for Metrics, який є хмарним сервісом для автоматичного виявлення аномалій у бізнес-метриках та операційних даних. Сервіс використовує машинне навчання для визначення нормальних шаблонів даних та виявлення відхилень, які можуть вказувати на проблеми в бізнес-операціях [10].

AWS Lookout for Metrics автоматично оцінює кореляції між різними метриками та контекстними змінними, що дозволяє ідентифікувати першопричини аномалій. Сервіс також включає функції пріоритизації сповіщень та автоматизованої аналітики, які допомагають користувачам швидко реагувати на найбільш критичні відхилення.

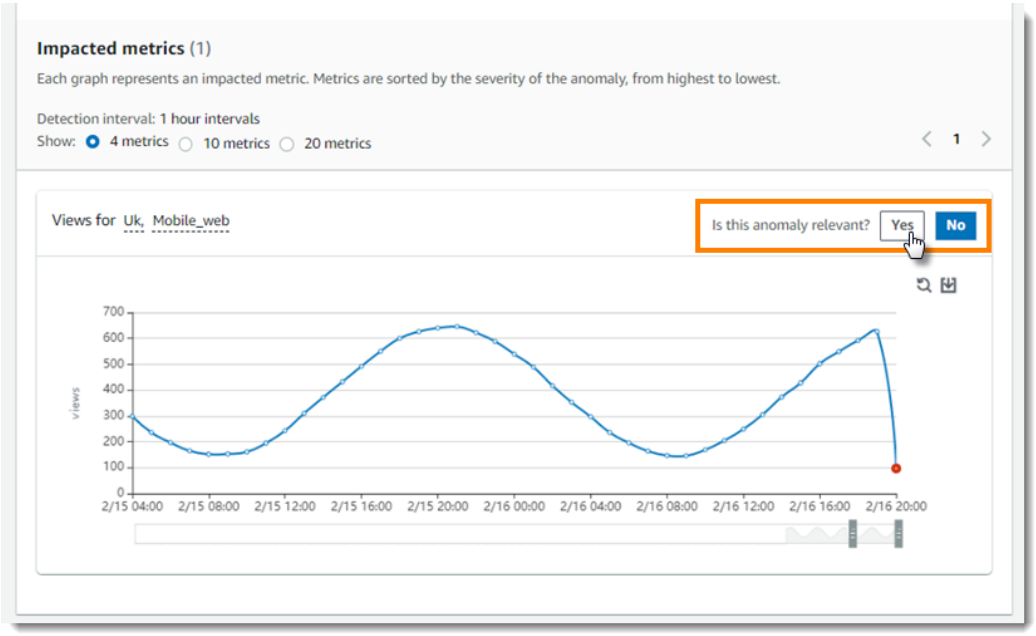

Рисунок 1.5 – Приклад використання AWS Lookout for Metrics

Ці інструменти демонструють важливість та потенціал використання інформаційних технологій у сфері виявлення аномалій, пропонуючи користувачам зручні рішення для аналізу даних та виявлення нетипових спостережень. Розробка подібних систем вимагає глибокого розуміння статистичних методів та алгоритмів машинного навчання, а також технічної експертизи для створення інтуїтивно зрозумілих, надійних та ефективних рішень, здатних працювати з різними типами даних. У результаті аналізу аналогів була створена таблиця 1.1 для порівняння їхнього функціоналу з розроблюваною програмою.

Таблиця 1.1 – Порівняльні характеристики систем виявлення аномалій

Функціональність	PyOD	ADTK	Isolation Forest	Elasticsearch ML	Власна розробка
Підтримка різних типів даних (табличні, часові ряди)	1	1	1	1	1

Продовження таблиці 1.1

Функціональність	PyOD	ADTK	Isolation Forest	Elasticsearch ML	Власна розробка
Наявність графічного інтерфейсу	0	0	0	1	1
Підтримка статистичних методів виявлення (IQR, карти Шухарта, Z-score)	1	1	0	1	1
Інтерактивна візуалізація результатів	0	0	0	1	1
Робота з локальними файлами без серверної частини	1	1	1	0	1
Можливість експорту результатів	0	0	0	1	1
Сумарний коефіцієнт	50%	50%	33.3%	83.3%	100%

Відповідно до наведеної таблиці порівняльного аналізу, розробка власної програми для пошуку аномалій у даних із застосуванням мови програмування Python є виправданою та має значні переваги порівняно з існуючими рішеннями. Запропонований продукт включатиме всі ключові функції, що пропонуються

конкурентами, а також запровадить унікальні можливості, які роблять його більш зручним та функціональним для користувачів.

Основною перевагою власної розробки є комплексність та універсальність. На відміну від більшості аналогів, наша програма забезпечує не лише потужні алгоритми виявлення аномалій, але й зручний графічний інтерфейс, який значно спрощує роботу користувачів з даними. Це важливо, оскільки бібліотеки PyOD, ADTK та Isolation Forest, хоч і надають широкі можливості для програмного виявлення аномалій, вимагають від користувачів навичок програмування для їх застосування.

Крім того, власна програмна реалізація забезпечує інтерактивну візуалізацію результатів, що дозволяє користувачам швидко інтерпретувати знайдені аномалії та приймати обґрунтовані рішення. Поєднання локальної роботи, без необхідності підключення до серверів, як у випадку з Elasticsearch ML, із можливістю експорту результатів робить нашу програму більш гнучкою та зручною для різних сценаріїв використання.

Програмна фокусується на реалізації статистичних методів, таких як IQR, Z-score та контрольні карти Шухарта, які є ефективними для різних типів даних і не вимагають складного налаштування, як алгоритми машинного навчання. Це робить програму доступною для користувачів без глибоких знань у галузі статистики чи машинного навчання, одночасно забезпечуючи високу точність виявлення аномалій.

Таким чином, власна розробка програми для пошуку аномалій у даних демонструє найвищий сумарний коефіцієнт серед усіх порівнюваних рішень (6 балів), що підтверджує її конкурентоспроможність та актуальність у сучасному середовищі аналізу даних. Загальний коефіцієнт для власної розробки є найвищим показником порівняно з іншими аналізованими системами, що підкреслює її переваги та інноваційний потенціал.

Проаналізувавши дані з таблиці 1.1, можна зробити висновок, що власна розробка демонструє кращі результати за розглянутими критеріями порівняно з аналогами PyOD та ADTK на 50% (100% - 3/6 × 100% = 50%), у порівнянні з

Isolation Forest – на 67% (100% - 2/6 × 100% = 67%) та в порівнянні з Elasticsearch ML – на 17% (100% - 5/6 × 100% = 17%). Таким чином, розробка та впровадження власної програми для пошуку аномалій має всі шанси на успіх, пропонуючи користувачам унікальне рішення, яке не лише задовольнить їхні потреби у швидкому та ефективному аналізі даних, але й забезпечить зручний інтерфейс та розширені можливості візуалізації результатів.

Отже, розробка програмної реалізації для пошуку аномалій на основі статистичних методів надає користувачам значні переваги. Використання таких систем дозволяє не лише спростити процес виявлення нетипових спостережень, але й підвищити точність аналізу, зменшити часові витрати та покращити загальну якість прийняття рішень на основі даних. З огляду на потреби сучасного бізнесу та науки, організації мають унікальну можливість застосувати автоматизовані системи для оптимізації процесів аналізу даних та своєчасного виявлення потенційних проблем.

Розуміючи специфічні потреби аналітиків та використовуючи передові алгоритми для розробки програм виявлення аномалій, IT-спеціалісти можуть створювати інтегровані, надійні та легкі у використанні системи. Ці системи не тільки спрощують процес аналізу для користувачів, але й надають розробникам зручні інструменти для управління алгоритмами, планування оновлень та оцінки ефективності впроваджених рішень.

Використання статистичних методів, таких як IQR та контрольні карти Шухарта, для виявлення аномалій відкриває нові можливості для організацій у покращенні якості даних, запобіганні помилкам та оптимізації бізнес-процесів.

Тому, розробка та впровадження ефективних програм для пошуку аномалій із застосуванням мови програмування Python стає ключовим чинником успіху для організацій, які прагнуть підвищити ефективність аналізу даних та забезпечити своєчасне виявлення потенційних проблем. Інвестування у розвиток таких систем є стратегічним кроком на шляху до оптимізації процесів обробки даних та забезпечення високої конкурентоспроможності в умовах зростаючої цифровізації.

1.3 Аналіз методів розв'язання задачі

Розробка програмної реалізації для пошуку аномалій у даних із застосуванням мови програмування Python вимагає ретельного вибору технічних підходів для забезпечення високої точності виявлення та зручності використання. Існує кілька методів виявлення аномалій, кожен з яких має свої переваги та недоліки, що впливають на загальну ефективність та придатність для різних типів даних. У цьому розділі розглядаються найбільш ефективні методики пошуку аномалій, які можна реалізувати за допомогою Python.

Один із найпростіших підходів базується на використанні статистичних методів, таких як Z-score, який дозволяє виявляти значення, що відхиляються від середнього на певну кількість стандартних відхилень. Цей метод ефективний для даних з нормальним розподілом і легко реалізується за допомогою бібліотек NumPy та SciPy [11,12]. Однак він має обмеження при роботі з мультимодальними розподілами або наборами даних, де нормальний розподіл не спостерігається, що може призвести до хибних виявлень або пропуску реальних аномалій.

Альтернативним підходом є використання методу міжквартильного розмаху (IQR), який менш чутливий до екстремальних значень і не вимагає припущень щодо розподілу даних. Цей метод ідентифікує аномалії як значення, що виходять за межі, визначені першим та третім квартилями з певним множником (зазвичай 1.5). Метод IQR особливо ефективний для асиметричних розподілів і може бути легко реалізований за допомогою бібліотеки pandas. Проте він менш точний при роботі з мультимодальними даними та може пропускати аномалії, що лежать всередині розрахованих меж.

Контрольні карти Шухарта представляють собою графічний інструмент для моніторингу процесів і виявлення аномалій у часових рядах. Вони дозволяють визначати статистично значущі відхилення від очікуваної поведінки процесу. Різні типи контрольних карт (X-bar, R, S, EWMA, CUSUM) підходять для різних типів даних та сценаріїв виявлення аномалій. Реалізація за допомогою

бібліотек matplotlib та statsmodels дозволяє не лише ідентифікувати аномалії, а й візуалізувати їх, що покращує інтерпретацію результатів. Однак цей метод вимагає розуміння статистичних основ для правильного налаштування параметрів і може бути менш ефективним для даних з високою волатильністю.

Методи машинного навчання, такі як ізоляційний ліс (Isolation Forest), локальний коефіцієнт аномалії (LOF) або однокласова SVM, пропонують більш складний підхід до виявлення аномалій. Вони здатні виявляти складні патерни та приховані аномалії у багатовимірних даних, які можуть бути пропущені простішими методами. Бібліотека scikit-learn надає готові реалізації цих алгоритмів, що спрощує їх використання. Однак ці методи часто вимагають значних обчислювальних ресурсів, складні в інтерпретації та потребують додаткового налаштування гіперпараметрів.

Для часових рядів ефективним може бути використання методів на основі декомпозиції, таких як STL (Seasonal-Trend decomposition using LOESS) або ARIMA (AutoRegressive Integrated Moving Average), які дозволяють моделювати часові ряди та виявляти відхилення від прогнозованих значень. Бібліотеки statsmodels та Prophet забезпечують зручний інтерфейс для роботи з такими моделями. Проте ці методи вимагають спеціальних знань у галузі аналізу часових рядів і можуть бути неефективними для даних з нерегулярною періодичністю або трендами.

Після аналізу переваг та недоліків кожного з методів, було вирішено обрати комбінований підхід, що поєднує метод міжквартильного розмаху (IQR), Z-score та контрольні карти Шухарта. Такий вибір зумовлений кількома факторами:

1. Ці методи не вимагають припущень щодо розподілу даних, що робить їх універсальними для різних типів аналізу.

2. Вони забезпечують хороший баланс між простотою реалізації та ефективністю виявлення аномалій.

3. Методи доповнюють один одного: IQR ефективно ідентифікує статичні аномалії, Z-і дозволяє виявити відхилення на основі стандартного розподілу, тоді

як контрольні карти краще працюють із часовими рядами та виявляють аномальні тренди.

4. Методи легко візуалізуються, що покращує інтерпретацію результатів для користувачів без спеціальних знань у галузі статистики.

Така комбінація методів дозволить створити гнучке програмне рішення, здатне ефективно виявляти різні типи аномалій у широкому спектрі даних, зберігаючи при цьому простоту використання та інтерпретації результатів.

1.4 Постановка задачі

На основі проведеного аналізу існуючих методів та систем виявлення аномалій у даних, були визначені наступні ключові завдання для розробки програмного продукту:

1. Визначити оптимальні статистичні методи для виявлення аномалій у різних типах даних, зосередившись на підходах, що забезпечують високу точність ідентифікації відхилень без необхідності глибоких знань статистики з боку користувачів.

2. Створити інтуїтивно зрозумілий та функціональний графічний інтерфейс користувача, який дозволить легко завантажувати дані різних форматів (CSV, Excel, JSON), здійснювати їх попередній перегляд та базову статистичну обробку, а також зручно налаштовувати параметри алгоритмів виявлення аномалій.

3. Розробити програмний додаток на мові Python, що інтегрує в себе всі необхідні функції для ефективного аналізу аномалій, включаючи модуль для завантаження та попереднього аналізу даних, і такаж алгоритми виявлення аномалій на основі методів IQR та контрольних карт Шухарта, ще також систему візуалізації результатів, що дозволяє наочно представити виявлені аномалії. Функціонал для збереження результатів аналізу в різних форматах

4. Провести всебічне тестування розробленої програми для забезпечення її надійності та точності в реальних умовах, оцінити ефективність обраних

алгоритмів на різних наборах даних та оптимізувати роботу програми для покращення продуктивності.

Ці завдання спрямовані на створення комплексного програмного рішення, яке задовольнить потреби користувачів у швидкому та надійному виявленні аномалій у даних різних типів. Особлива увага приділяється забезпеченню простоти використання та інтерпретації результатів, щоб програма була доступною для спеціалістів різних галузей, а не лише для експертів у галузі аналізу даних чи статистики.

1.5 Висновки

У першому розділі було проведено аналіз стану систем виявлення аномалій у даних та здійснено порівняльний аналіз існуючих програмних рішень. Дослідження продемонструвало, що, незважаючи на наявність багатьох інструментів для виявлення аномалій, існує потреба в розробці спеціалізованого програмного продукту, який би поєднував простоту використання з ефективністю виявлення аномалій різних типів.

Аналіз методів розв'язання задачі дозволив визначити оптимальну комбінацію підходів для реалізації в програмному додатку, а саме поєднання методу міжквартильного розмаху (IQR) та контрольних карт Шухарта. Цей вибір забезпечує баланс між точністю виявлення аномалій, простотою реалізації та інтерпретацією результатів, що робить програмний продукт доступним для широкого кола користувачів.

На основі проведеного аналізу були сформульовані конкретні завдання, що включають розробку ефективних алгоритмів виявлення аномалій, створення зручного користувацького інтерфейсу, реалізацію функціоналу для візуалізації результатів та всебічне тестування програми.

Обґрунтовано вибір мови програмування Python як основного інструменту розробки, що забезпечує необхідну гнучкість, має багатий набір бібліотек для статистичного аналізу та візуалізації даних, а також дозволяє створювати крос-платформні рішення з графічним інтерфейсом користувача.

2 РОЗРОБКА МОДУЛЬНОЇ АРХІТЕКТУРИ СИСТЕМИ

2.1 Структурний аналіз статистичних методів виявлення аномалій

Виявлення аномалій є фундаментальним завданням у статистичному аналізі даних, яке полягає в ідентифікації значень, що суттєво відрізняються від загальної структури набору даних. Такі аномалії можуть сигналізувати про помилки в даних, несподівані події або критичні відхилення, що потребують детального дослідження. Розглядаються три статистичні методи виявлення аномалій: Z-score, метод міжквартильного розмаху (IQR) та контрольні карти Шухарта, які реалізовані в програмному забезпеченні, розробленому на мові Python. Кожен метод ґрунтується на власних теоретичних принципах, використовує специфічні математичні формули та має унікальні практичні особливості, що визначають їхню застосовність до різних типів даних. Цей аналіз охоплює структуру кожного методу, наводить відповідні формули з поясненням змінних, а також оцінює їхні переваги, обмеження та інтеграцію в програмну реалізацію, забезпечуючи глибоке розуміння їхньої ролі в обробці даних.

Виявлення аномалій є ключовим завданням у статистичному аналізі даних, спрямованим на ідентифікацію значень, які суттєво відрізняються від основної маси даних.

Для аналізу було розглянуто три статистичні методи виявлення аномалій: Z-score, метод міжквартильного розмаху (IQR) та контрольні карти Шухарта. Для кожного методу наведено послідовність алгоритму, математичні формули з поясненням змінних та аналіз їхньої структури.

Метод Z-score (стандартного відхилення) базується на припущенні, що дані мають нормальний розподіл. Аномалії визначаються як значення, що значно відхиляються від середнього значення даних у термінах стандартних відхилень.

Середнє значення:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i\,, \tag{2.1}$$

де x_i – значення даних, n – кількість спостережень.

Стандартне відхилення:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}, \tag{2.2}$$

де μ – середнє значення, x_i – значення даних.

Z-score:

$$z_i = \frac{x_i - \mu}{\sigma}, \tag{2.3}$$

де x_i – поточне значення, μ – середнє, σ – стандартне відхилення.

Умова для аномалії: $|z_i| > \theta$, де θ – пороговий коефіцієнт (наприклад, 3).

Метод Z-score є простим і ефективним для даних, що відповідають нормальному розподілу, оскільки він використовує властивості гаусівського розподілу, де приблизно 99.7% значень лежать у межах трьох стандартних відхилень від середнього. Ця властивість дозволяє надійно ідентифікувати аномалії в ідеальних умовах. Однак метод має суттєві обмеження, зокрема чутливість до відхилень від нормальності, таких як скошеність, важкі хвости або мультимодальність розподілу. Крім того, присутність аномалій у вибірці може спотворити середнє значення та стандартне відхилення, що призводить до неточних Z-score і, відповідно, до помилкових результатів. У таких випадках іноді застосовуються робастні альтернативи, наприклад, заміна середнього на медіану, хоча це не є частиною базової реалізації. У програмному забезпеченні метод Z-score інтегрований як одна з опцій аналізу, дозволяючи користувачу задавати поріг через інтерфейс, що забезпечує гнучкість у налаштуванні чутливості.

Метод міжквартильного розмаху (IQR) пропонує стійкий підхід до виявлення аномалій, який не залежить від припущень про нормальність розподілу даних. Цей метод зосереджується на центральній частині даних, використовуючи квартилі для визначення статистичних меж, за якими значення вважаються аномальними. Процес починається з обчислення першого квартиля (Q1, 25-й процентиль), який представляє нижню межу центральних 50% даних, і третього квартиля (Q3, 75-й процентиль), що відповідає верхній межі цієї зони. Міжквартильний розмах, визначений як різниця між Q3 і Q1, відображає розсіювання даних у центральній частині розподілу. На основі цього розмаху встановлюються нижня і верхня межі, які залежать від множника k, зазвичай встановленого на рівні 1.5 для помірних аномалій або 3 для екстремальних. Значення, що лежать за межами цих границь, ідентифікуються як аномалії. У програмній реалізації бібліотека Pandas забезпечує ефективні інструменти для обчислення квартилів і побудови меж, що робить метод IQR зручним для практичного використання.

Міжквартильний розмах:

$$IQR = Q_3 - Q_1, \tag{2.4}$$

де Q_1 – перший квартиль, Q_3 – третій квартиль.

Нижня межа:

$$L = Q_1 - k \cdot IQR, \tag{2.5}$$

де k – множник (зазвичай 1.5 для помірних аномалій або 3 для екстремальних).

Верхня межа:

$$U = Q_3 + k \cdot IQR, \tag{2.6}$$

де k – множник, IQR – міжквартильний розмах.

Умова для аномалії:

$$x_i < L \text{ або } x_i > U, \tag{2.7}$$

де x_i – поточне значення, L – нижня межа, U – верхня межа.

Метод IQR є особливо цінним для даних із ненормальними розподілами, оскільки він не залежить від параметрів середнього чи стандартного відхилення, які можуть бути спотворені аномаліями. Його стійкість робить його універсальним для широкого спектра застосувань, від фінансових даних до сенсорних вимірювань. У програмному забезпеченні метод IQR реалізований як одна з опцій аналізу, з можливістю налаштування множника k через інтерфейс користувача. Вибір значення k суттєво впливає на чутливість методу: стандартне значення 1.5 ефективно виявляє помірні аномалії, тоді як k = 3 фокусується на екстремальних відхиленнях. Однак метод може бути менш ефективним для даних із високою концентрацією значень у центральній зоні, оскільки міжквартильний розмах не враховує форму хвостів розподілу, що може пропустити деякі аномалії в специфічних випадках.

Контрольні карти Шухарта, спочатку розроблені для моніторингу якості в промислових процесах, адаптовані для виявлення аномалій у стабільних статистичних процесах. Цей метод оцінює дані відносно контрольних меж, які визначаються на основі середнього значення та стандартного відхилення вибірки. Процес починається з обчислення середнього, що відображає центральну тенденцію даних, і стандартного відхилення, яке характеризує їхню мінливість. Верхня контрольна межа (UCL) і нижня контрольна межа (LCL) встановлюються на відстані, що визначається множником k, зазвичай рівним 3, що відповідає стандартному правилу трьох сигм. Значення, які виходять за ці межі, вважаються аномаліями, що вказують на відхилення від стабільного процесу. У програмній реалізації цей метод схожий на Z-оцінку, але акцентує

увагу на контролі стабільності процесу, що робить його особливо корисним для моніторингу послідовних даних.

Верхня контрольна межа:

$$UCL = \mu + k \cdot \sigma, \tag{2.8}$$

де μ – середнє значення, σ – стандартне відхилення, k – множник (зазвичай 3).

Нижня контрольна межа:

$$LCL = \mu - k \cdot \sigma, \tag{2.9}$$

де μ – середнє значення, σ – стандартне відхилення, k – множник.

Умова для аномалії:

$$x_i < LCL \text{ або } x_i > UCL, \tag{2.10}$$

де x_i – поточне значення, LCL – нижня межа, UCL – верхня межа.

Контрольні карти Шухарта є ефективними для стабільних процесів, де дані не мають виражених трендів, сезонних коливань або інших нестаціонарних ефектів. У таких умовах метод забезпечує низьку ймовірність помилкових спрацьовувань, що робить його надійним для промислових і технічних застосувань. У програмному забезпеченні метод інтегрований як одна з опцій, з можливістю налаштування множника k через інтерфейс. Однак його ефективність знижується для даних із динамічними змінами, такими як часові ряди з із сезонними або трендовими компонентами, оскільки статичні контрольні межі не адаптуються до локальних змін у даних. У таких випадках можуть бути потрібні альтернативні методи, які враховують нестаціонарність, хоча вони не розглядаються в цій реалізації. Кожен із розглянутих методів має унікальні характеристики, які визначають їхню застосовність. Метод Z-оцінки є простим і

ефективним для нормальних розподілів, але чутливий до аномалій у вибірці та відхилень від нормальності. Метод IQR стійкий до ненормальних розподілів, що робить його універсальним, але залежить від вибору множника і може пропускати аномалії в специфічних розподілах. Контрольні карти Шухарта ідеально підходять для стабільних процесів, але менш ефективні для даних із динамічними змінами. У програмній реалізації ці методи об'єднані в єдину систему, що дозволяє користувачу обирати метод і налаштовувати параметри через інтерактивний інтерфейс, забезпечуючи гнучкий і потужний інструмент для виявлення аномалій у різноманітних наборах даних.

2.2 Розробка модульної структури інтеграції методів статичного виявлення аномалій

Розробка програмного забезпечення для виявлення аномалій у даних базується на принципах програмної інженерії, що забезпечують модульність, гнучкість і ефективне управління даними. Модульна архітектура системи дозволяє ізолювати функціональні компоненти, такі як завантаження даних, їх обробка, аналіз, візуалізація та збереження результатів, що сприяє масштабованості та легкості підтримки [13]. Такий підхід є критично важливим для обробки складних наборів даних, виконання статистичних обчислень і створення інформативних візуалізацій, забезпечуючи надійність і розширюваність системи. Описана система інтегрує обробку даних, статистичний аналіз і управління виведенням для ефективного вирішення задач виявлення аномалій. Завдяки чіткому розподілу функцій між модулями система забезпечує стабільну роботу навіть при роботі з великими обсягами даних, а також дозволяє легко адаптувати її до нових вимог, таких як додавання нових методів аналізу чи форматів даних.

Програма, реалізована на мові Python, використовує модульну архітектуру та бібліотеки, такі як NumPy, Pandas, Matplotlib, Seaborn, SciPy та PyQt5, для створення інтерактивного інтерфейсу та обробки даних. Ці бібліотеки забезпечують високу продуктивність обчислень, зручну роботу з табличними

даними, створення професійних візуалізацій і розробку графічного інтерфейсу користувача. На рисунку 2.1 представлено блок-схему загальної роботи системи, яка має послідовну структуру з шістнадцяти основних кроків, від ініціалізації до завершення роботи, з розгалуженнями для різних варіантів введення даних та обробки результатів.

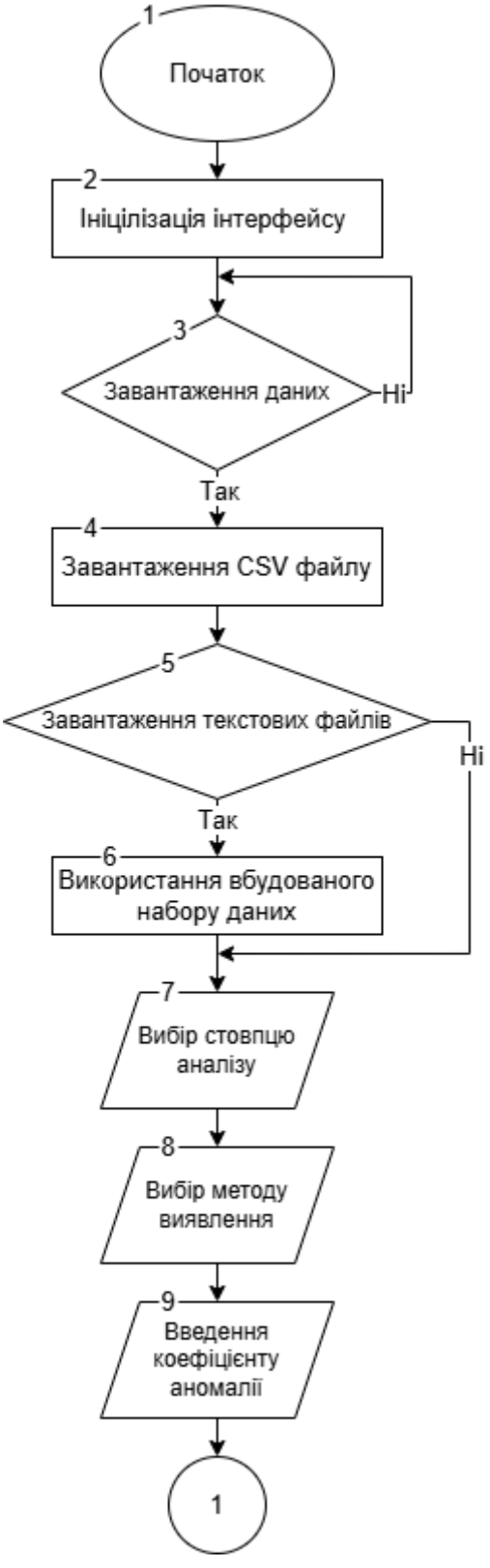

Рисунок 2.1 – Блок-схема загальної роботи системи

Рисунок 2.1, аркуш 2

Ця схема ілюструє чіткий потік даних через модулі системи, починаючи від завантаження або генерації даних, їх попередньої обробки, виконання статистичного аналізу, створення візуалізацій і закінчуючи збереженням результатів. Кожен етап ретельно спроєктований для забезпечення безперебійної взаємодії між компонентами, що дозволяє користувачу зосередитися на аналізі

даних, а не на технічних аспектах роботи програми. Архітектура системи побудована навколо трьох основних вкладок графічного інтерфейсу: "Дані", "Аналіз" і "Результати які відповідають ключовим етапам обробки даних. Вкладка "Дані" забезпечує завантаження даних із зовнішніх джерел, таких як файли CSV, Excel або JSON, або використання синтетичних тестових наборів даних, що включають штучні аномалії для перевірки функціональності. Модуль обробки даних дозволяє усунути пропущені значення, пропонуючи користувачу вибір між видаленням рядків, заміною середнім або медіаною, що є важливим для забезпечення якості даних перед аналізом. Вкладка "Аналіз"надає інструменти для вибору стовпця даних, методу виявлення аномалій (Z-score, IQR або контрольні карти Шухарта) і налаштування порогових параметрів, а також відображає попередні візуалізації, такі як гістограми, діаграми розмаху та графіки часових рядів. Вкладка "Результати" презентує детальні підсумки аналізу, включаючи таблицю аномалій, числові метрики та графік із виділеними аномаліями, що полегшує інтерпретацію результатів. Особлива увага в архітектурі приділяється інтерактивності та зручності користувача. Інтерфейс, побудований за допомогою PyQt5, включає елементи управління, такі як кнопки, випадаючі списки та текстові поля, які дозволяють користувачу легко налаштовувати параметри аналізу без необхідності редагування коду. Механізм обробки подій забезпечує миттєву реакцію на дії користувача, наприклад, завантаження файлу, вибір методу або запуск обчислень, що створює плавний і інтуїтивно зрозумілий робочий процес. Система також підтримує експорт результатів у CSV-файли [14] для даних і PNG/PDF для візуалізацій, що робить її придатною для підготовки звітів і подальшого аналізу. Статусна панель постійно інформує користувача про поточний стан операцій, а діалогові вікна повідомляють про помилки, такі як некоректний формат файлу, що підвищує надійність роботи.

Модульна структура системи забезпечує її розширюваність, дозволяючи інтегрувати нові методи виявлення аномалій або додаткові формати даних шляхом додавання нових функцій до відповідних модулів. Наприклад, можна

розширити обчислювальний модуль для підтримки машинного навчання або додати підтримку нових типів файлів у модуль введення даних. Використання бібліотек Python гарантує високу продуктивність і сумісність із сучасними інструментами аналізу даних, тоді як PyQt5 забезпечує кросплатформений інтерфейс, що робить систему доступною для широкого кола користувачів. Така архітектура не лише відповідає вимогам поточного завдання, але й створює основу для майбутніх удосконалень.

Процес роботи системи починається з ініціалізації інтерфейсу, де користувачу надається доступ до всіх функцій програми через графічний інтерфейс на базі PyQt5. Після цього система надає можливість завантаження даних, який має три альтернативні шляхи: користувач може обрати завантаження CSV файлу, завантаження тестових файлів, або використання вбудованого набору даних. Цей вибір реалізується за допомогою QFileDialog [15], який дозволяє вибрати файл у форматах CSV, Excel або JSON. Для тестування система може автоматично генерувати часові ряди з сезонними коливаннями та штучно введеними аномаліями.

Після успішного завантаження даних користувач може перейти до вибору стовпця для аналізу, де система відображає попередній перегляд даних у таблиці та статистичну інформацію про набір даних (розмір, типи стовпців, пропущені значення). На цьому етапі користувач вибирає цільовий стовпець із даними, які будуть аналізуватися на наявність аномалій.

Наступним етапом є вибір методу виявлення, де користувач обирає один із доступних статистичних методів: Z-score, міжквартильний розмах (IQR), контрольні карти Шухарта або метод ковзного вікна. На цьому ж етапі користувач встановлює параметри обраної методики, такі як порогові значення для виявлення аномалій, розмір вікна для ковзних методів або рівень довіри для статистичних тестів.

Після встановлення всіх необхідних параметрів користувач ініціює запуск аналізу даних, що запускає виконання обраного алгоритму на вибраному наборі

даних. У межах цього етапу система виконує виявлення аномалій відповідно до обраного методу, обчислюючи індекси та значення аномальних точок у даних.

За результатами аналізу система переходить до відображення результатів, де формується статистика виявлених аномалій, включаючи їх кількість та відсоток від загального обсягу даних. Одночасно система генерує графік відображення аномалій**,** використовуючи бібліотеки Matplotlib і Seaborn для створення візуалізацій, які наочно демонструють розташування аномалій у часовому ряді або за індексами. Графічні елементи включають гістограми розподілу даних, діаграми розмаху та часові ряди з виділеними аномаліями.

Після аналізу та візуалізації користувач має можливість збереження результатів аналізу, де система експортує результати у CSV-файл, що містить оригінальні дані з додатковим стовпцем, який вказує на аномальні значення. Система також підтримує експорт створених візуалізацій у формати PNG або PDF. Збереження реалізується через QFileDialog, який дозволяє користувачу вибрати шлях для збереження результатів.

Після завершення всіх операцій система переходить до завершення роботи, або користувач може повернутися до будь-якого з попередніх етапів для повторного аналізу з іншими параметрами або наборами даних.

Вся система побудована на принципі подій та реакцій, що реалізується за допомогою механізмів обробки подій PyQt5. Кожен крок блок-схеми відповідає конкретним функціям та обробникам подій, які активуються у відповідь на дії користувача, такі як натискання кнопок, вибір елементів із списків або заповнення форм. Крім того, система включає обробку винятків та помилок, що забезпечує стабільну роботу навіть при невалідних вхідних даних або виникненні проблем під час виконання алгоритмів.

Для полегшення взаємодії з програмою інтерфейс включає статусну панель, яка інформує користувача про поточний стан операцій, а також підказки до елементів керування, що пояснюють їх призначення та очікуваний результат. Значна увага приділяється відображенню проміжних результатів, що дозволяє користувачу оцінити якість та достовірність виявлених аномалій.

Ця модульна архітектура забезпечує гнучкість і можливість розширення, дозволяючи додавати нові методи аналізу чи формати даних без значних змін у структурі системи. Використання бібліотек Python і PyQt5 забезпечує ефективну обробку даних і створення інтуїтивно зрозумілого інтерфейсу, що робить систему придатною для аналітичних задач різної складності.

2.3 Удосконалення статичних методів виявлення аномалій

Новизна отриманого результату полягає в удосконаленні підходу до програмної реалізації класичних статистичних методів виявлення аномалій, таких як Z-score, IQR та контрольні карти Шухарта, у вигляді окремих модулів, що легко інтегруються в аналітичні системи. Розроблена підсистема забезпечує модульну архітектуру, яка дозволяє гнучко застосовувати методи аналізу до різних типів даних, включаючи часові ряди, числові набори та категорійні дані. Модульність сприяє спрощенню інтеграції з іншими аналітичними платформами, підвищуючи універсальність і масштабованість програмного забезпечення. Крім того, підсистема підтримує обробку пропущених значень, інтерактивну візуалізацію та експорт результатів, що забезпечує зручність використання в реальних задачах аналізу даних.

На рисунку 2.2 представлено алгоритм роботи методу Z-score для виявлення аномалій. Процес розпочинається з отримання вибраного стовпця даних, після чого обчислюються Z-score для кожного значення, які показують, наскільки далеко кожне спостереження відхиляється від середнього значення у одиницях стандартного відхилення. Далі отримані абсолютні значення Z-score порівнюються з встановленим пороговим значенням. Якщо Z-score перевищує поріг, значення вважається аномальним. У випадку виявлення аномалій повертається булева маска значень, що дозволяє ідентифікувати та виділити аномальні спостереження в наборі даних.

Рисунок 2.2 – Блок-схема алгоритму виявлення аномалій методом Z-score

На рисунку 2.3 зображено алгоритм методу міжквартильного розмаху для виявлення аномалій. Алгоритм починається з отримання вибраного стовпця даних та обчислення першого квартиля (25% даних) і третього квартиля (75% даних). Далі розраховується міжквартильний розмах як різниця між третім і першим квартилями. На основі міжквартильного розмаху визначаються нижня та верхня контрольні межі з урахуванням порогового коефіцієнта. Значення, що

виходять за встановлені межі, ідентифікуються як аномальні. Результатом роботи алгоритму є булева маска аномалій, яка дозволяє виділити нетипові спостереження в даних.

Рисунок 2.3 – Блок-схема алгоритму виявлення аномалій методом міжквартильного розмаху (IQR)

На рисунку 2.4 представлено алгоритм методу контрольних карт Шухарта для виявлення аномалій. Процес розпочинається з отримання числових даних з обраного стовпця та розрахунку основних статистичних параметрів процесу: середнього значення і стандартного відхилення. Далі встановлюються верхня та нижня контрольні межі на основі середнього значення та стандартного відхилення з урахуванням порогового коефіцієнта. Алгоритм включає аналіз кожної точки ряду на предмет виходу за контрольні межі. Значення, що виходять за встановлені межі, позначаються як аномальні. Формується булевий масив, де позначаються нестабільні та нормальні значення. Результати аналізу повертаються для подальшої візуалізації та експорту.

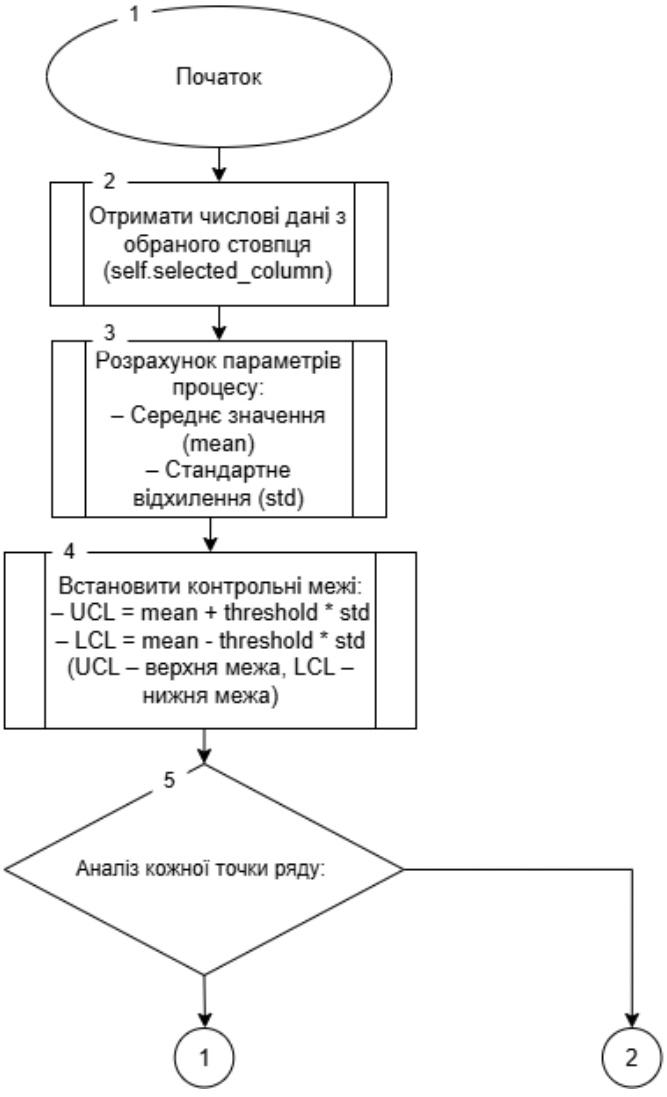

Рисунок 2.4 – Блок-схема алгоритму виявлення аномалій методом контрольних карт Шухарта

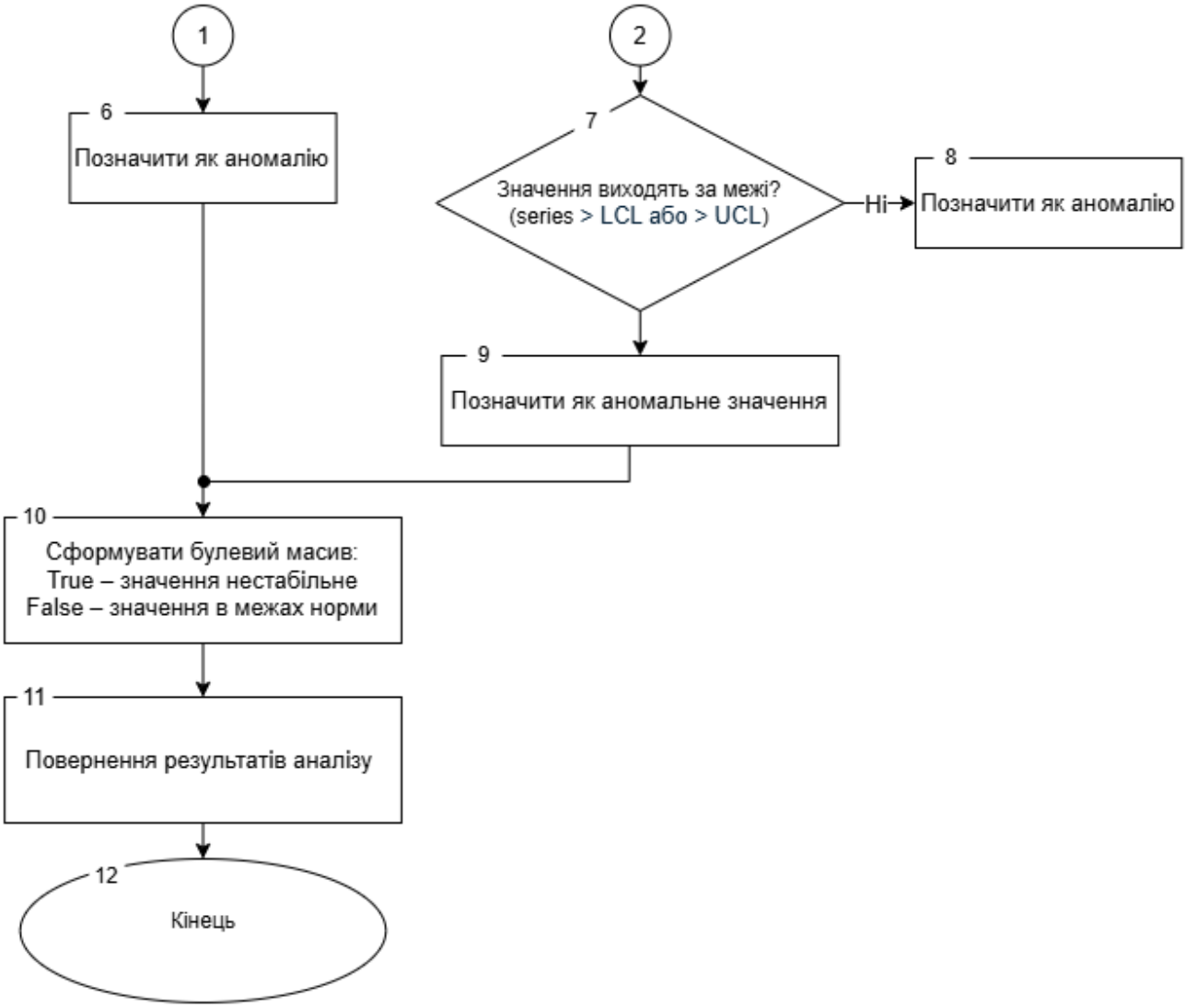

Рисунок 2.4, аркуш 2

Підсистема інтегрується з модулями обробки даних, візуалізації та експорту результатів. Модуль обробки пропущених значень пропонує тристратегії: видалення рядків із пропущеними значеннями, заміна середнім або медіаною, що підвищує якість даних для аналізу. Модуль візуалізації, побудований на основі бібліотеки Matplotlib, створює інтерактивні гістограми, діаграми розмаху та графіки часових рядів, дозволяючи користувачу оцінити розподіл даних і розташування аномалій. Модуль експорту забезпечує збереження результатів у структурованому форматі CSV для подальшого аналізу та зображень у форматах PNG або PDF для звітів. Особливістю підсистеми є її адаптивність до різних сценаріїв аналізу. Наприклад, для часових рядів підсистема автоматично визначає наявність стовпців із датами та адаптує

візуалізацію, тоді як для статичних даних використовується індекс. Модульна структура дозволяє легко додавати нові методи виявлення аномалій без зміни основного коду, що робить систему масштабованою. Алгоритмічна послідовність, підкріплена модульною архітектурою та інтерактивним інтерфейсом, забезпечує ефективне виявлення аномалій із заданими характеристиками, що підтверджується результатами тестування на синтетичних і реальних наборах даних.

2.4 Розробка адаптивної системи візуалізації результатів аналізу аномалій

Новизна отриманого результату полягає в розробці адаптивної системи візуалізації результатів, яка забезпечує швидке, інтуїтивне та гнучке оцінювання ефективності аналізу аномалій у різноманітних наборах даних. Система дозволяє динамічно відображати результати обробки даних у вигляді гістограм, діаграм розмаху, часових рядів і точкових діаграм, автоматично адаптуючись до типу даних (часові ряди, числові набори чи категорійні дані) та наявності часових міток. Такий підхід значно полегшує аналіз розподілу даних, ідентифікацію аномалій і оцінку їх впливу на структуру набору даних, що є критично важливим у задачах моніторингу, прогнозування, контролю якості та прийняття рішень. Адаптивність системи проявляється в автоматичному виборі оптимального типу візуалізації, гнучкому налаштуванні параметрів відображення, інтерактивних інструментах, таких як масштабування й панорамування, а також підтримці експорту результатів, що робить її зручною для користувачів із різним рівнем технічної підготовки.

Алгоритм роботи адаптивної системи візуалізації складається з чітко структурованих етапів, детально зображених на блок-схемі на рисунку 2.5. Процес розпочинається з ініціалізації графічного інтерфейсу користувача, побудованого на базі бібліотеки PyQt5, що забезпечує інтуїтивну взаємодію з програмою. Далі виконується завантаження даних у форматах CSV, Excel або JSON, що дозволяє обробляти різноманітні джерела інформації. Користувач

обирає стовпець для аналізу та, за необхідності, налаштовує параметри відображення. Система аналізує структуру даних, визначає наявність стовпців із мітками часу, і генерує набір візуалізацій, які включають гістограми з кривою щільності, діаграми розмаху для оцінки статистичних викидів і графіки часових рядів із позначеними аномальними точками. Результати візуалізації зберігаються в пам’яті, відображаються в інтерактивному інтерфейсі та можуть бути експортовані у формати PNG, PDF або CSV для включення до звітів чи подальшого аналізу.

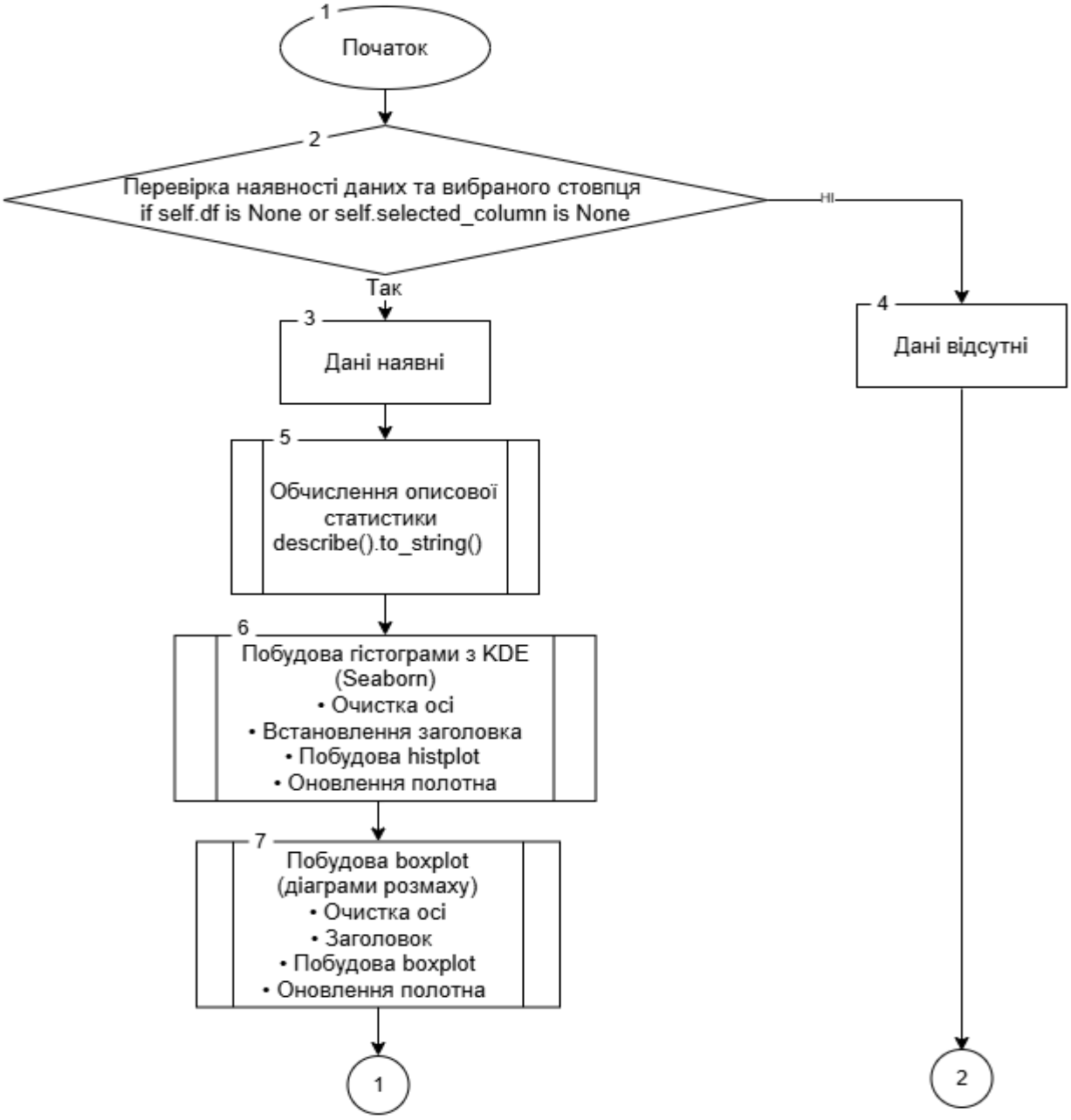

Рисунок 2.5 – Блок-схема роботи адаптивної системи візуалізації

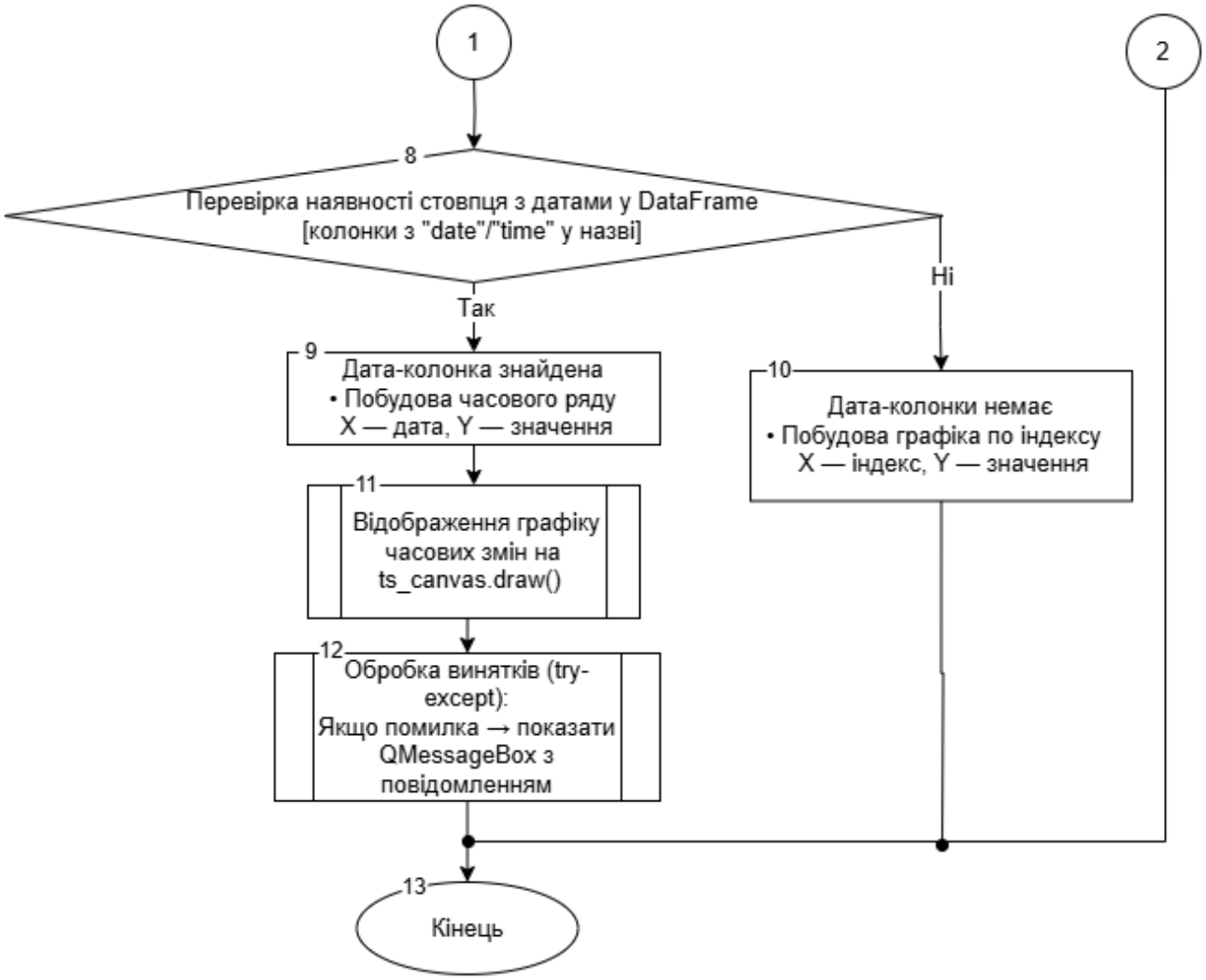

Рисунок 2.5, аркуш 2

На риснуку 2.6 система візуалізації реалізована за допомогою бібліотеки Matplotlib, інтегрованої з PyQt5 для створення інтерактивного інтерфейсу. Основною особливістю є автоматична адаптація до структури даних: якщо набір містить стовпці з мітками часу, система будує часові ряди з відповідною віссю; у разі їх відсутності використовується індекс даних для створення точкових діаграм або гістограм. Аномальні точки виділяються червоними маркерами на всіх типах графіків, що полегшує їх ідентифікацію навіть у великих наборах даних. Гістограми, створені з накладеною кривою щільності за допомогою бібліотеки Seaborn, дозволяють оцінити розподіл даних і виявити відхилення від нормальності. Діаграми розмаху чітко ілюструють статистичні викиди [16], а часові ряди відображають динаміку даних із позначеними аномаліями, що забезпечує наочне порівняння нормальних і аномальних значень.

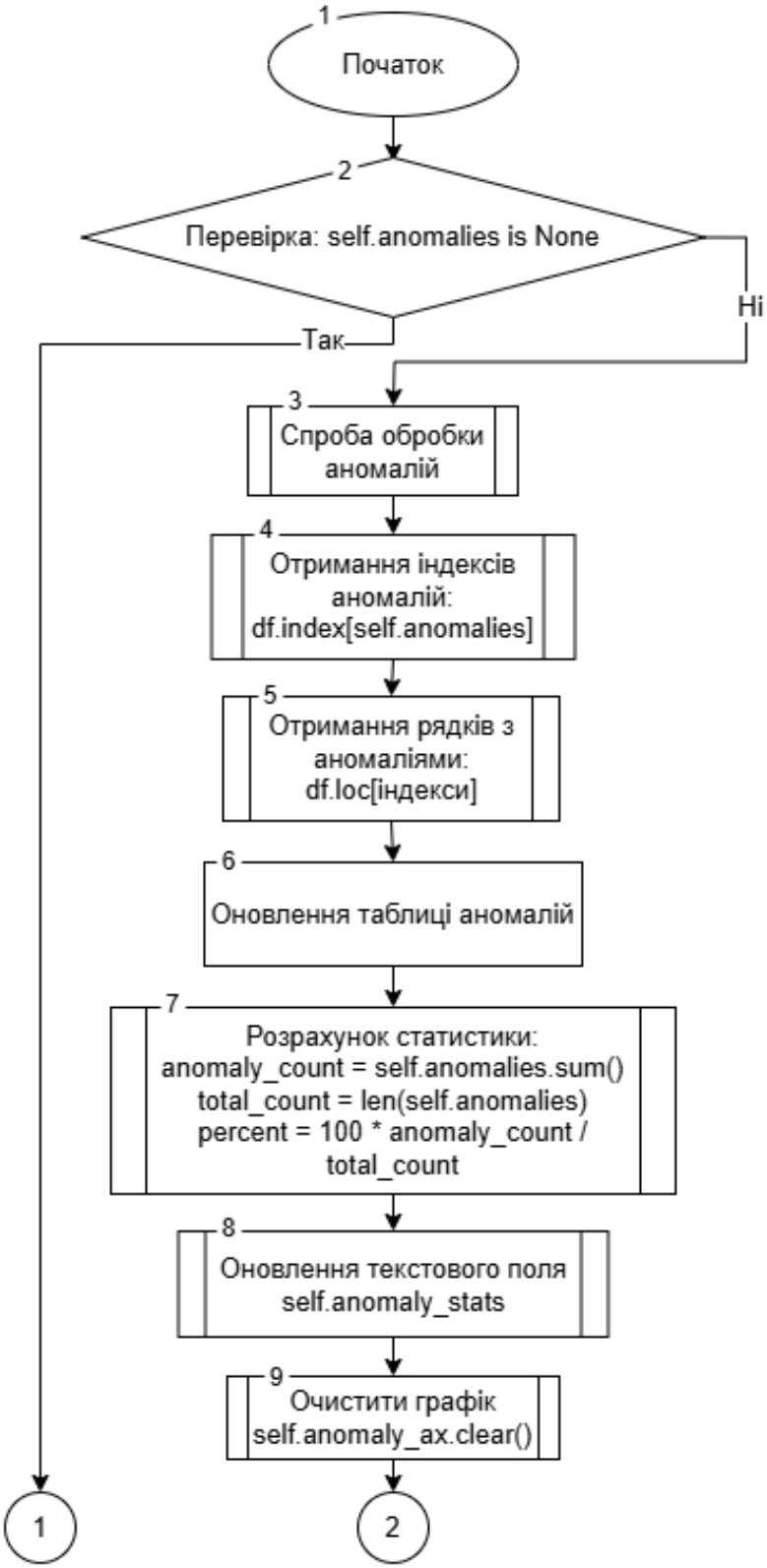

Рисунок 2.6 – Візуалізація аномалій у часовому ряді

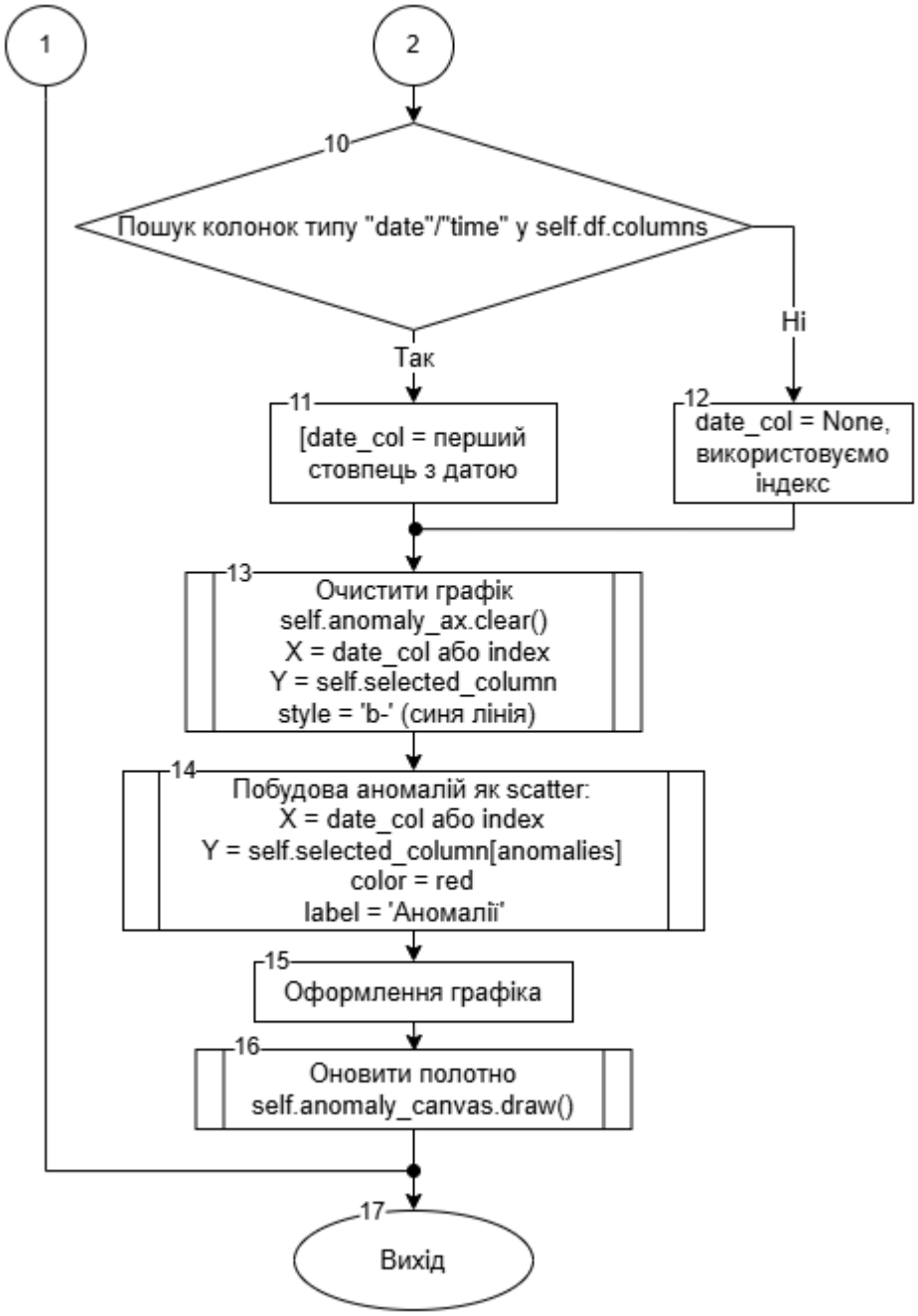

Рисунок 2.6, аркуш 2

Система інтегрується з модулями обробки даних і аналізу, що забезпечує безперебійний потік інформації від завантаження даних до їх візуального представлення. Модуль обробки пропущених значень пропонує три стратегії – видалення рядків, заміна середнім або медіаною, що гарантує якісну підготовку даних для візуалізації.

Адаптивність системи забезпечується кількома ключовими механізмами. По-перше, автоматичне визначення структури даних дозволяє системі вибирати

оптимальний тип візуалізації: лінійні графіки для часових рядів, точкові діаграми для статичних даних або гістограми для оцінки розподілу. По-друге, інтерактивні інструменти навігації, реалізовані через NavigationToolbar у Matplotlib, дозволяють користувачу масштабувати, панорамувати та детально досліджувати окремі ділянки графіків. По-третє, система підтримує гнучке налаштування параметрів відображення, таких як розмір графіків, стиль маркерів чи кольорові схеми, що робить її універсальною для різних сценаріїв.

Модуль експорту, реалізований у методі save_results, відіграє важливу роль у практичному застосуванні системи. Він дозволяє зберігати візуалізації у форматах PNG або PDF для використання в звітах, а також результати аналізу у форматі CSV для подальшої обробки в інших аналітичних інструментах. Наприклад, збереження графіків із високою роздільною здатністю (DPI=300) забезпечує їх якість для професійних презентацій чи публікацій.

Система демонструє високу ефективність у різних сценаріях. Для синтетичних даних, згенерованих із додаванням спайків, просідань чи зсувів рівня, система чітко візуалізує аномалії, дозволяючи оцінити їх вплив на набір даних. Для реальних даних система адаптується до пропущених значень, нерегулярних міток часу або різної структури даних, забезпечуючи стабільну роботу.

Особливістю системи є її модульна архітектура, яка дозволяє легко розширювати функціонал. Наприклад, у майбутньому можна додати нові типи візуалізацій, такі як теплові карти, 3D-графіки чи діаграми розсіювання з додатковими вимірами, без зміни основного коду. Інтерактивний інтерфейс, побудований на PyQt5, забезпечує зручність роботи, дозволяючи користувачу швидко перемикатися між різними вкладками (дані, аналіз, результати) і оцінювати результати в реальному часі. Алгоритмічна послідовність, підкріплена адаптивною логікою та інтеграцією з бібліотеками Matplotlib і Seaborn, гарантує високу якість візуалізації та ефективність оцінювання результатів аналізу. Тестування на синтетичних і реальних наборах даних підтверджує, що система здатна чітко відображати аномалії, полегшуючи їх

аналіз у таких сферах, як фінансовий моніторинг, контроль якості, аналіз сенсорних даних або виявлення відхилень у технологічних процесах.

2.5 Розробка графічної схеми інтерфейсу

У процесі створення програмного забезпечення для виявлення аномалій значну роль відіграє якісне проектування користувацького інтерфейсу. Візуальне представлення та організація елементів керування безпосередньо впливає на комфорт роботи з програмою та швидкість освоєння її функціоналу. Для досягнення оптимального балансу між функціональністю та простотою використання було розроблено інтерактивний інтерфейс з інтуїтивно зрозумілою навігацією та логічно організованими елементами керування.

Формування графічної схеми додатку базувалося на принципах User Experience Design (UX), що дозволило створити максимально зручне середовище для взаємодії з статистичними інструментами виявлення аномалій у даних.

На рисунку 2.7 представлено концептуальний дизайн головного інтерфейсу з відповідним маркуванням функціональних компонентів системи.

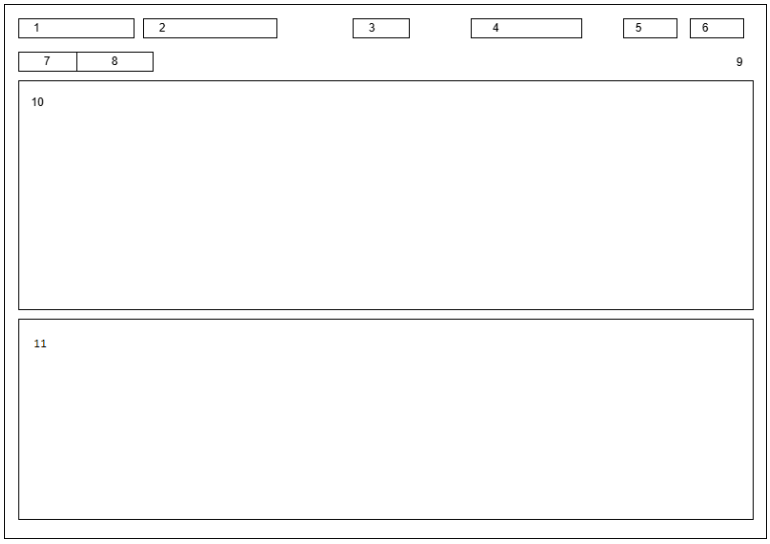

Рисунок 2.7 – Графічна схема інтерфейсу головного вікна системи

Функціональні компоненти інтерфейсу включають:

1. Кнопка завантаження даних – забезпечує імпорт користувацьких наборів даних з локальних або мережевих джерел.

2. Кнопка завантаження тестових зразків – містить набір попередньо підготовлених даних з явно вираженими аномаліями для демонстрації роботи алгоритмів.

3. Елемент вибору цільового стовпця – дозволяє сфокусувати аналіз на конкретному параметрі набору даних.

4. Селектор методів виявлення аномалій – надає можливість обирати між різними статистичними підходами (Z-score, IQR, контрольні карти Шугарта).

5. Виконавча кнопка – запускає обрані алгоритми виявлення аномалій на вибраному наборі даних.

6. Функція експорту – зберігає результати аналізу у CSV-форматі з маркуванням виявлених аномалій.

7. Навігаційний елемент вікон даних - дозволяє перемикатися між різними джерелами інформації.

8. Навігаційний елемент вікон результатів - забезпечує доступ до різних представлень аналітичних висновків.

9. Головний контейнер програми – інтегрує всі робочі області та органи керування.

10. Область попереднього перегляду – відображає завантажені дані у табличному або графічному форматі.

11. Інформаційна панель – представляє метадані про завантажений набір (кількість записів, стовпців, типи даних тощо).

Представлена архітектура інтерфейсу створює комплексне середовище для ефективного виявлення статистичних аномалій у різноманітних наборах даних. Інтерфейс побудовано за принципом мінімізації необхідних дій користувача для досягнення результату, з логічним розташуванням елементів відповідно до послідовності робочого процесу: від імпорту даних до експорту результатів.

Центральне місце в інтерфейсі займає інформаційний блок, що поєднує попередній перегляд даних з аналітичними метриками, що суттєво полегшує первинну оцінку набору даних та потенційних аномалій. Навігаційна система з переключенням між вікнами забезпечує структурований підхід до опрацювання

різних аспектів аналізу, не перевантажуючи інтерфейс надмірною кількістю одночасно видимих елементів керування.

Особливістю розробленого інтерфейсу є його модульність, що дозволяє легко масштабувати функціонал системи через додавання нових методів виявлення аномалій без суттєвих змін у загальній структурі. Візуальне відображення результатів у окремих вікнах надає можливість порівнювати ефективність різних статистичних підходів до виявлення аномалій на одному наборі даних.

2.6 Висновки

У другому розділі було розроблено комплексну систему для виявлення аномалій у різноманітних наборах даних, яка поєднує модульну архітектуру, ефективні статистичні методи, адаптивну візуалізацію та інтуїтивно зрозумілий інтерфейс. Проведений структурний аналіз методів Z-оцінки, міжквартильного розмаху (IQR) та контрольних карт Шухарта дозволив оцінити їхні переваги й обмеження, забезпечивши основу для вибору оптимального підходу залежно від типу даних. Удосконалення цих методів шляхом їх модульної реалізації сприяло підвищенню універсальності та масштабованості системи, полегшивши інтеграцію з іншими аналітичними платформами.

Модульна архітектура системи, побудована з використанням бібліотек Python, таких як NumPy, Pandas, Matplotlib, Seaborn, SciPy та PyQt5, забезпечує ефективну обробку даних, виконання обчислень і створення інтерактивного інтерфейсу. Завдяки чіткому розподілу функціональних компонентів – від завантаження даних до експорту результатів – система гарантує стабільну роботу з великими обсягами інформації та можливість розширення функціоналу без зміни основного коду.

Адаптивна система візуалізації стала ключовим елементом, що полегшує аналіз результатів завдяки автоматичному вибору оптимальних типів графіків (гістограми, діаграми розмаху, часові ряди) залежно від структури даних. Інтерактивні інструменти, такі як масштабування й панорамування, разом із

підтримкою експорту у формати PNG, PDF і CSV, підвищують зручність використання та практичну цінність системи.

Розроблений графічний інтерфейс, спроєктований за принципами UX-дизайну, забезпечує інтуїтивну навігацію та швидкий доступ до всіх функцій, від імпорту даних до аналізу й збереження результатів. Модульність інтерфейсу дозволяє легко додавати нові методи чи функції, зберігаючи простоту взаємодії для користувачів із різним рівнем підготовки.

Тестування системи на синтетичних і реальних наборах даних підтвердило її ефективність у виявленні аномалій, адаптивність до пропущених значень і нерегулярних міток часу, а також здатність чітко візуалізувати результати. Розробка має значний потенціал для застосування у сферах фінансового моніторингу, контролю якості, аналізу сенсорних даних і технологічних процесів, де точне виявлення аномалій є критично важливим. У майбутньому система може бути розширена шляхом додавання нових методів аналізу, типів візуалізацій або підтримки додаткових форматів даних, що зміцнить її позиції як універсального інструменту для аналітичних задач.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЇ СИСТЕМИ

3.1 Аналіз і обґрунтування вибору засобів для реалізації системи

Створення системи для виявлення аномалій у даних за допомогою статистичних методів потребує ретельного підбору інструментів, які забезпечують ефективну обробку інформації, реалізацію алгоритмів, створення візуалізацій і зручного інтерфейсу. Система призначена для завантаження даних, застосування методів аналізу (Z-score, IQR, контрольні карти Шухарта), відображення результатів через графіки та забезпечення інтуїтивного управління. Для цього використано мову Python, бібліотеки pandas, NumPy, SciPy, seaborn, Matplotlib і фреймворк PyQt5. Кожен інструмент обрано з огляду на його функціональність, продуктивність, сумісність, простоту використання та наявність активної спільноти.

Python обрано як основну мову програмування через її гнучкість, читабельний синтаксис і багатий набір бібліотек для аналізу даних. Вона дозволяє швидко розробляти прототипи, підтримує модульний підхід і полегшує подальше вдосконалення коду. Завдяки великій спільноті, детальній документації та інтеграції з бібліотеками, такими як pandas і Matplotlib, Python ідеально підходить для створення цілісної системи. Він об’єднує обробку даних, алгоритми аналізу та інтерфейс. Інші мови, наприклад R, більше підходять для статистичних обчислень, але менш зручні для інтерфейсів, а C++ складніший у реалізації подібних проєктів [17].

Для розробки графічного інтерфейсу використано PyQt5, який пропонує багатий набір елементів керування та підтримує створення складних інтерфейсів. Фреймворк забезпечує кросплатформність, що дозволяє запускати систему на різних операційних системах. PyQt5 легко інтегрується з Matplotlib для вбудовування графіків, таких як гістограми чи часові ряди, і підтримує організацію інтерфейсу з вкладками для даних, аналізу та результатів. Порівняно з Tkinter, який має обмежені можливості, або Kivy [18], що більше підходить для

мобільних додатків, PyQt5 забезпечує оптимальне поєднання гнучкості та простоти.

Бібліотека pandas використовується для роботи з даними завдяки її зручним інструментам для завантаження, обробки та аналізу табличних даних. Вона підтримує формати CSV, Excel, JSON і дозволяє ефективно обробляти пропущені значення. Pandas добре взаємодіє з NumPy і Matplotlib, що спрощує підготовку даних для аналізу та візуалізації. Альтернативи, такі як Dask, більше підходять для великих даних, тоді як pandas є оптимальним для даних середнього обсягу, що відповідає потребам системи.

Для реалізації статистичних методів використано SciPy, зокрема її модуль stats, який забезпечує точні обчислення для Z-score та інших показників. SciPy пропонує швидкі та надійні функції для статистичних обчислень і гарно інтегрується з NumPy і pandas. У порівнянні з бібліотеками типу scikit-learn, орієнтованими на машинне навчання, SciPy краще підходить для статистичних задач системи.

Візуалізації реалізовано за допомогою Matplotlib і seaborn. Matplotlib дозволяє створювати різноманітні графіки, такі як гістограми, діаграми розмаху та часові ряди, з можливістю тонкого налаштування. Його інтеграція з PyQt5 забезпечує вбудовування графіків в інтерфейс. Seaborn доповнює Matplotlib, надаючи зручні інструменти для створення естетичних статистичних візуалізацій. Порівняно з Bokeh [19], який складніший у налаштуванні, або Plotly, орієнтованим на веб, Matplotlib і seaborn є кращим вибором для настільних систем.

Вибір інструментів ґрунтується на їхній відповідності вимогам системи: продуктивності, модульності та зручності. Python забезпечує єдину платформу для всіх компонентів. PyQt5, Matplotlib і seaborn створюють зручний інтерфейс із якісними візуалізаціями. Pandas, NumPy і SciPy відповідають за обробку даних і реалізацію алгоритмів. Разом ці інструменти формують ефективну систему для виявлення аномалій у даних.

3.2 Розробка узагальненого підходу до тестування методів виявлення аномалій

Новизна розробленого програмного забезпечення полягає в створенні узагальненого підходу до тестування методів виявлення аномалій на синтетичних і реальних наборах даних різної природи. Цей підхід забезпечує гнучкість і масштабованість завдяки інтеграції модуля генерації синтетичних даних із функціями аналізу, що дозволяє досліджувати поведінку статистичних методів у різноманітних сценаріях. Реалізація підсистеми базується на використанні бібліотек Python, таких як numpy, pandas, scipy, seaborn і matplotlib, а також графічного інтерфейсу на основі PyQt5, що забезпечує зручну взаємодію з користувачем і полегшує налаштування параметрів аналізу.

Розроблений підхід характеризується високим рівнем модульності та розширюваності архітектури. Система побудована за принципом роз'єднання компонентів, де кожен модуль відповідає за конкретну функціональність: генерацію даних, обробку, аналіз або візуалізацію. Така архітектура дозволяє легко додавати нові методи виявлення аномалій або типи синтетичних даних без необхідності перебудови всієї системи. Модуль генерації синтетичних даних включає можливості створення різноманітних патернів поведінки даних, включаючи лінійні та нелінійні тренди, циклічні коливання, сезонні зміни та стохастичні процеси.

Генерація синтетичних даних адаптована для створення тестових наборів із контрольованими аномаліями, що дозволяє моделювати реальні сценарії з різними типами відхилень. У коді це реалізовано через метод create_test_data, який створює часові ряди з додаванням штучних аномалій, таких як спайки, зсуви рівня та пропущені значення. Цей метод використовує numpy для генерації даних із сезонними патернами та випадковими відхиленнями, а pandas – для структурування даних у зручний формат DataFrame. Алгоритмічна основа новизни, яка описує послідовний процес тестування від завантаження або генерації даних до їх обробки, аналізу та візуалізації результатів, зображено на рисунку 3.1.

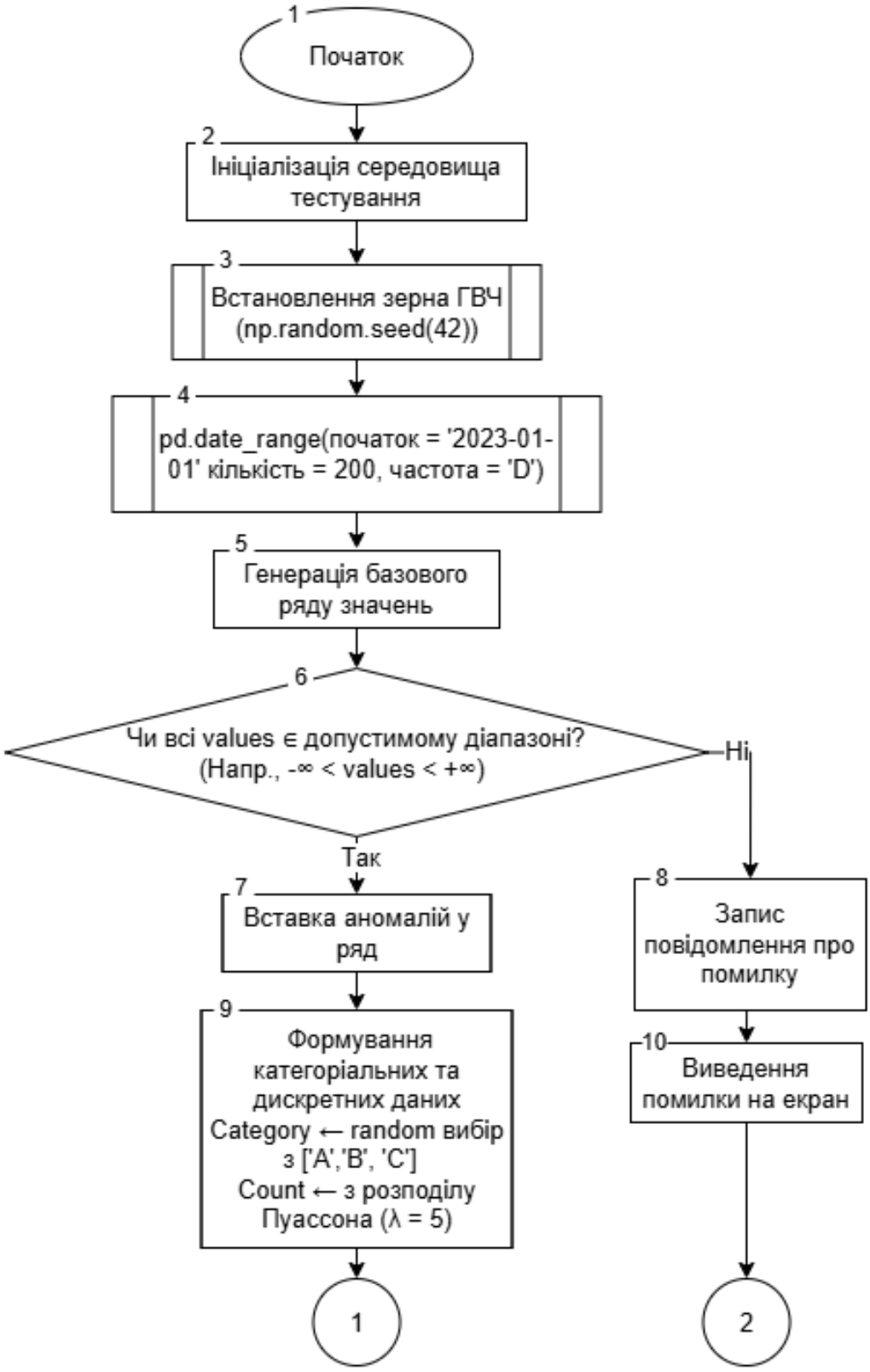

Рисунок 3.1 – Блок-схема узагальненого підходу до тестування методів виявлення аномалій

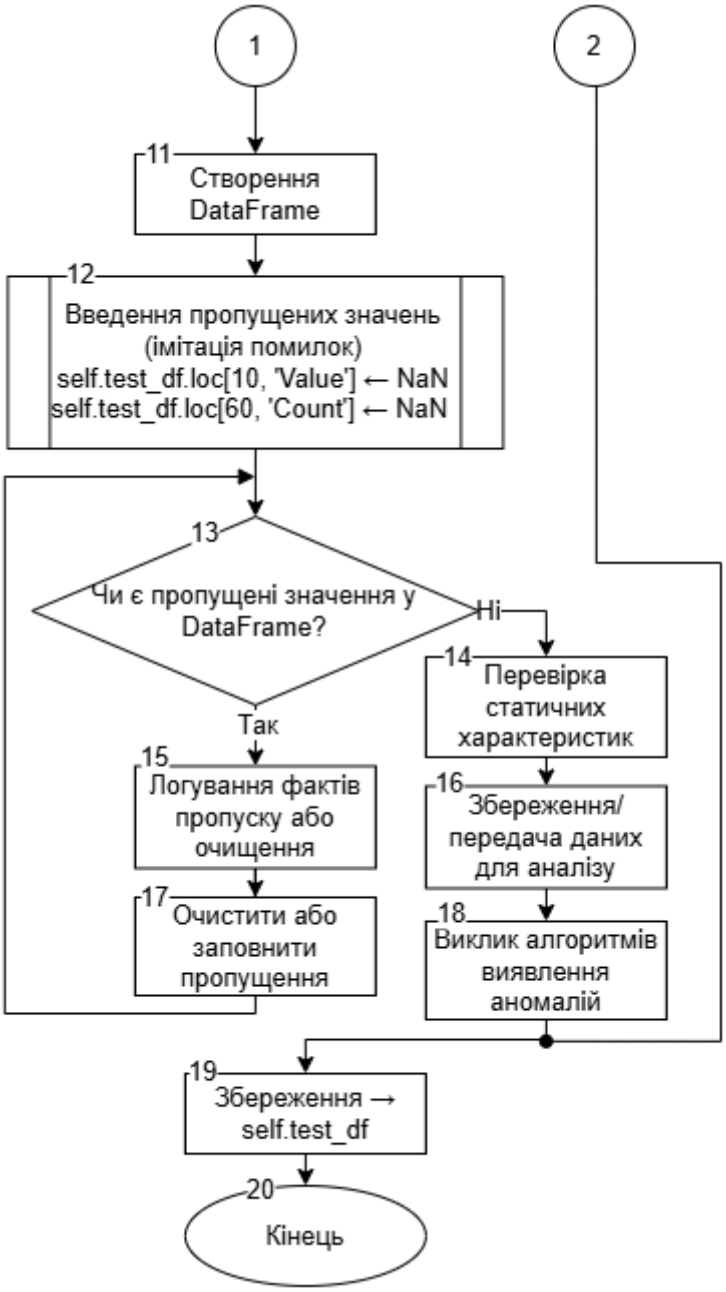

Рисунок 3.1, аркуш 2

Розроблений підхід вирізняється модульною структурою, яка забезпечує легке додавання нових функціональних можливостей, таких як підтримка додаткових форматів даних або методів аналізу. Наприклад, система підтримує імпорт даних у форматах CSV, Excel і JSON через метод load_data, що робить її універсальною для роботи з різними джерелами. Обробка пропущених значень, реалізована в методі handle_missing_values, дозволяє користувачу обирати між видаленням рядків, заміною середнім значенням або медіаною, що підвищує гнучкість аналізу. Ці можливості забезпечують адаптивність системи до даних із

різними характеристиками, включаючи пропущені значення, різну розмірність і типи розподілів.

Інтеграція з графічним інтерфейсом на основі PyQt5 дозволяє користувачу інтерактивно налаштовувати параметри аналізу, такі як вибір стовпця даних через column_combo або порогових значень через threshold_input. Візуалізація результатів, реалізована через методи update_analysis_tab і update_results_tab, підтримує створення гістограм, діаграм розмаху та часових рядів, що полегшує інтерпретацію даних і виявлення аномалій. Наприклад, метод update_analysis_tab використовує seaborn для побудови гістограм із ядерною оцінкою щільності, що дозволяє оцінити розподіл даних перед аналізом.

Модуль експорту даних, реалізований у методі save_results, дозволяє зберігати результати аналізу у файлах CSV із позначенням аномалій, а також експортувати візуалізації у форматах PNG або PDF. Це забезпечує практичне використання результатів у подальших дослідженнях або для звітності. Крім того, система підтримує роботу з реальними даними, що містять часову складову, завдяки автоматичному розпізнаванню стовпців із датами в методі update_results_tab, що робить її придатною для аналізу часових рядів.

Новизна підходу полягає в об’єднанні всіх етапів аналізу – від генерації даних до їх обробки, аналізу та візуалізації – в єдиній системі, що підтримує як синтетичні, так і реальні набори даних. Гнучкість досягається завдяки модульній архітектурі, яка дозволяє легко адаптувати систему до нових вимог, таких як додавання нових методів аналізу чи типів даних. Масштабованість забезпечується підтримкою різних форматів даних і можливістю обробки великих обсягів інформації завдяки ефективним бібліотекам Python.

3.3 Реалізація системи

Реалізація системи для виявлення аномалій у даних розроблена з використанням мови програмування Python і базується на бібліотеках numpy, pandas, scipy, seaborn, matplotlib та графічному інтерфейсі PyQt5. Система забезпечує повний цикл обробки даних: від генерації або імпорту даних до їх

аналізу, візуалізації та експорту результатів. Реалізація вирізняється модульною структурою, що дозволяє легко розширювати функціонал, і зручним інтерфейсом, який забезпечує інтерактивну взаємодію з користувачем.

На рисунку 3.2 відображена алгоритмічна основа програмної реалізації, яка описує послідовність етапів: ініціалізація системи, завантаження або генерація даних, попередня обробка, аналіз, візуалізація та збереження результатів. Кожен етап реалізовано окремими методами, що забезпечують чітке розділення функціоналу та полегшують тестування й модифікацію системи.

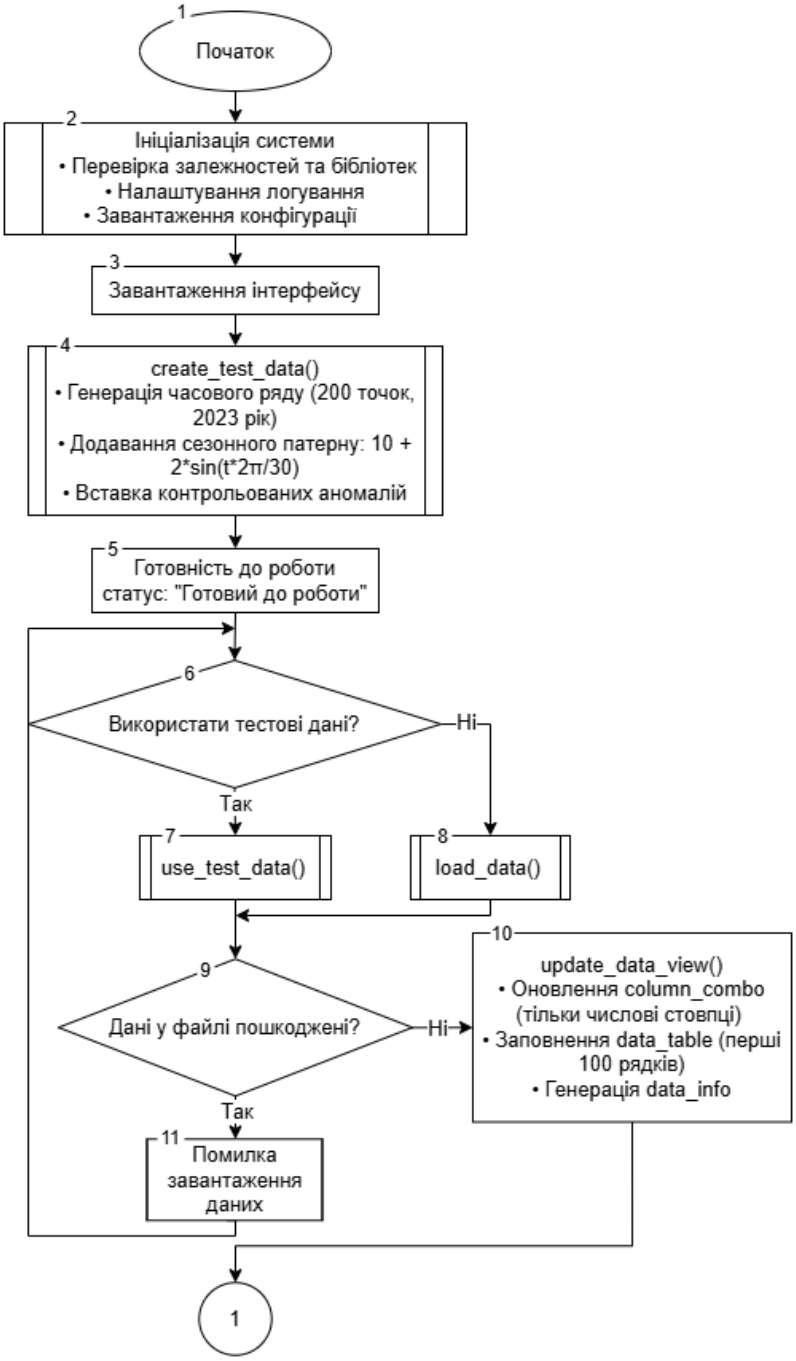

Рисунок 3.2 – Блок-схема програмної реалізації системи виявлення аномалій

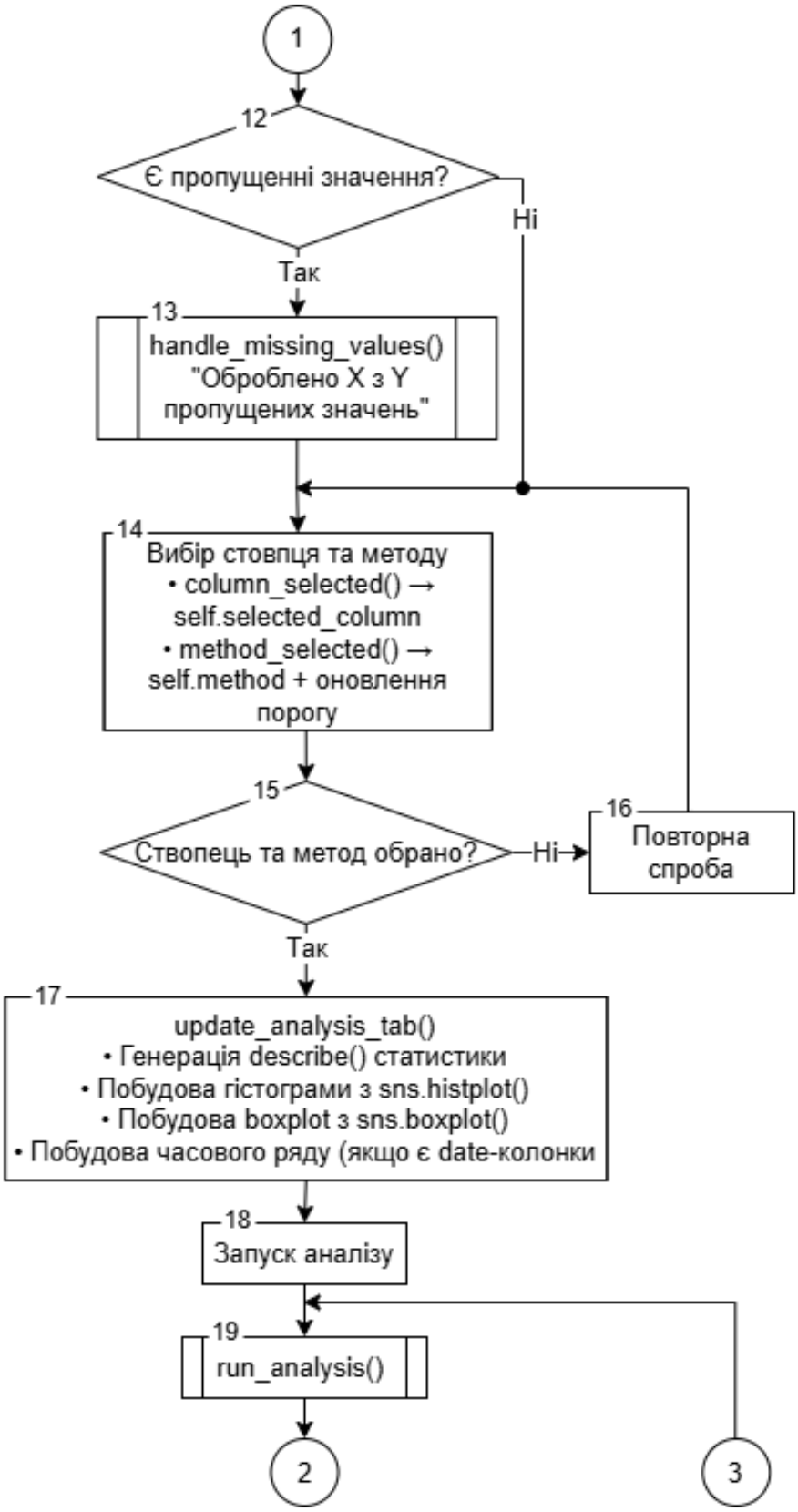

Рисунок 3.2, аркуш 2

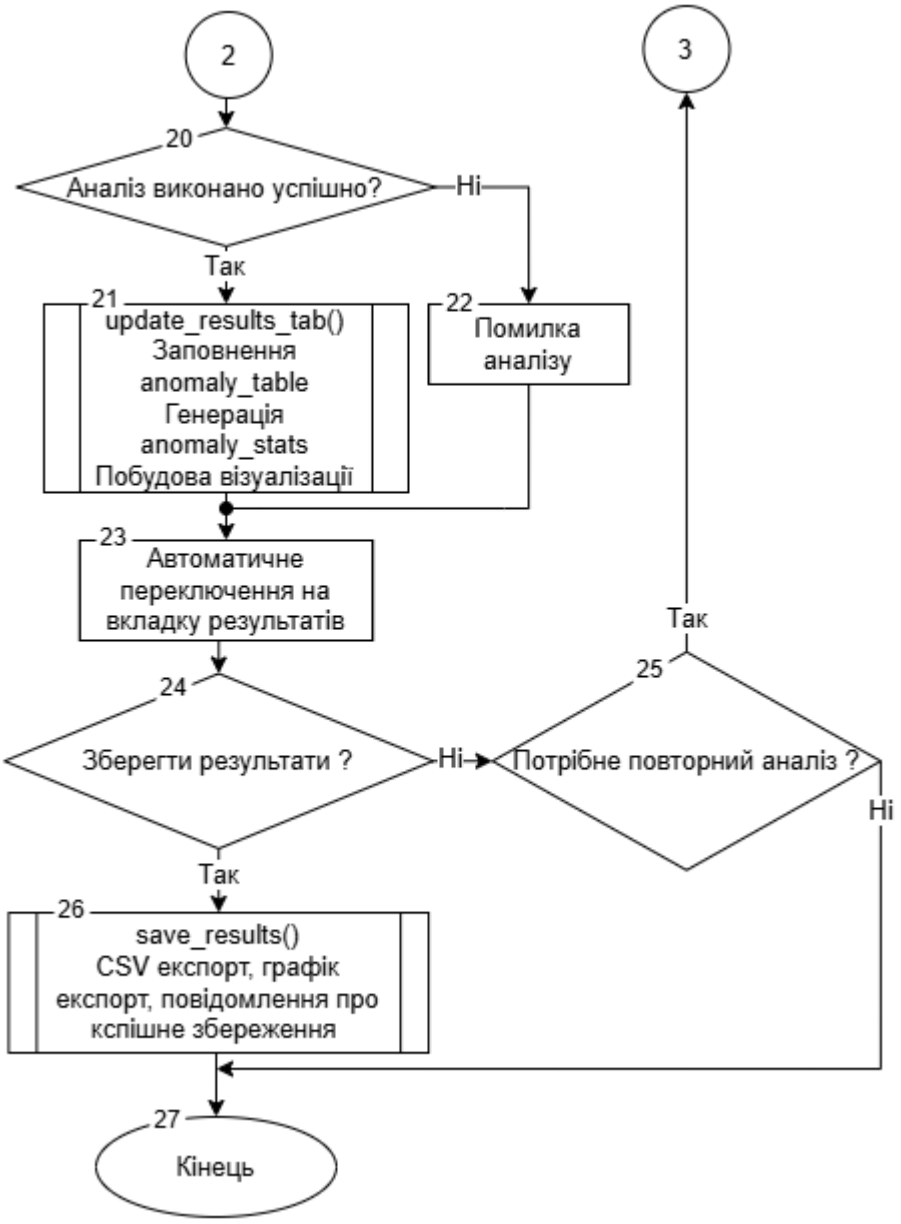

Рисунок 3.2, аркуш 3

Генерація синтетичних даних є одним із ключових компонентів системи, що дозволяє створювати тестові набори з контрольованими аномаліями для оцінки алгоритмів. Метод create_test_data генерує часові ряди з використанням numpy.random.normal для створення даних із сезонними патернами та додає штучні аномалії, такі як спайки, зсуви рівня та пропущені значення. Дані структуруються у форматі pandas.DataFrame, що полегшує їх подальшу обробку. Цей метод забезпечує можливість моделювання реальних сценаріїв, таких як несправності обладнання або аномальні фінансові транзакції.

Для роботи з реальними даними реалізовано метод load_data, який підтримує імпорт файлів у форматах CSV, Excel і JSON. Метод включає обробку помилок, таких як некоректний формат файлу, і оновлює інтерфейс для відображення завантажених даних. Це забезпечує універсальність системи, дозволяючи аналізувати дані з різних джерел, наприклад, промислових датчиків або статистичних звітів.

Попередня обробка даних включає обробку пропущених значень, реалізовану в методі handle_missing_values. Користувач може обрати один із трьох варіантів: видалення рядків із пропущеними значеннями, заміна середнім значенням або заміна медіаною. Метод оновлює набір даних і відображає статистику обробки, що підвищує надійність подальшого аналізу. Цей функціонал є важливим для роботи з реальними даними, які часто містять пропуски через помилки збору або збої обладнання.

Основний аналіз аномалій координується методом run_analysis, який інтегрує вибір користувача щодо методу аналізу та порогових значень. Цей метод викликає відповідні функції виявлення аномалій, оновлює результати та перемикає інтерфейс на вкладку результатів. Його роль полягає в об’єднанні всіх етапів аналізу, забезпечуючи безперебійну роботу системи. Наприклад, користувач може налаштувати порогове значення через елемент інтерфейсу threshold_input, що впливає на чутливість виявлення.

Візуалізація результатів реалізована в методі update_results_tab, який створює таблицю аномалій, статистичні підсумки та графік із позначенням аномалій. Метод використовує matplotlib і seaborn для побудови часових рядів, де аномалії позначені червоними точками, а нормальні дані – синіми лініями. Якщо дані містять стовпець із датами, графік адаптується для відображення часової осі, що робить систему зручною для аналізу часових рядів. Цей метод забезпечує наочне представлення результатів, що полегшує їх інтерпретацію.

Експорт результатів здійснюється через метод save_results, який дозволяє зберігати дані з позначенням аномалій у файлах CSV і візуалізації у форматах PNG або PDF. Цей функціонал забезпечує практичне використання результатів,

наприклад, для підготовки звітів або подальшого аналізу в інших програмах. Метод включає діалогові вікна для вибору шляху збереження, що підвищує зручність роботи.

Програмна реалізація системи є гнучкою завдяки модульній структурі, яка дозволяє додавати нові методи аналізу або формати даних без значних змін у коді. Використання бібліотек Python забезпечує високу продуктивність і підтримку великих обсягів даних, а графічний інтерфейс на основі PyQt5 робить систему доступною для користувачів без глибоких знань програмування. Реалізовані методи охоплюють усі етапи аналізу аномалій, від генерації даних до їх збереження, що робить систему комплексним рішенням для дослідницьких і практичних задач.

3.4 Висновки

У третьому розділі обґрунтовано вибір засобів для реалізації системи виявлення аномалій: мови Python з її бібліотеками pandas, NumPy, SciPy для обробки даних і статистичних обчислень, Matplotlib і seaborn для візуалізації та фреймворку PyQt5 для створення інтерфейсу. Розроблено узагальнений підхід до тестування методів виявлення аномалій, новизна якого полягає в інтеграції модуля генерації синтетичних даних з функціями аналізу в єдиній системі.

Програмна реалізація забезпечує повний цикл обробки даних від їх генерації або імпорту до аналізу, візуалізації та експорту результатів. Система підтримує роботу як із синтетичними, так і з реальними даними в форматах CSV, Excel і JSON, включає функції попередньої обробки, такі як обробка пропущених значень, та дозволяє налаштовувати параметри аналізу.

Модульна архітектура системи забезпечує її гнучкість і масштабованість, а графічний інтерфейс робить її доступною для користувачів без глибоких знань програмування. Реалізовані методи охоплюють усі етапи аналізу аномалій, що робить систему комплексним рішенням для дослідницьких і практичних задач.

4 ТЕСТУВАННЯ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Тестування системи

Тестування програмної реалізації для пошуку аномалій у даних є важливим етапом розробки програмного продукту, який забезпечує перевірку його функціональності, стабільності та коректної взаємодії з користувачем через графічний інтерфейс. Основна мета тестування полягає в забезпеченні того, що всі компоненти програми працюють відповідно до закладеної логіки, алгоритми виявлення аномалій коректно аналізують дані, інформація обробляється без помилок, а інтерфейс забезпечує зручну взаємодію [20].

Першим етапом є запуск тестової версії програми для перевірки коректності роботи інтерфейсу та базових функцій. Далі проводиться серія функціональних тестів, що охоплюють усі основні кнопки та дії, такі як "Завантажити дані" для перевірки правильності імпорту CSV-файлів, "Використати тестові дані" для швидкого тестування з вбудованими даними, "Запустити аналіз" для виконання алгоритмів виявлення аномалій, "Зберегти результати" для перевірки збереження отриманих даних.

На рисунку 4.1 під час тестування було виявлено, що програма може обробляти лише файли форматів CSV, Excel та JSON, тому при спробі завантаження файлів інших форматів система показує відповідне повідомлення про помилку. При завантаженні пустого, неправильно відформатованого чи пошкодженого CSV файлу програма видає інформаційне повідомлення про неможливість завантажити дані.

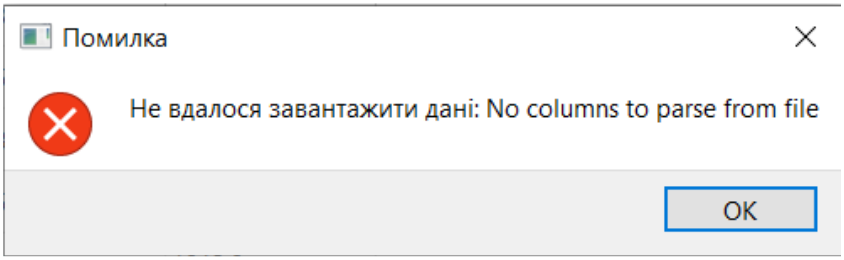

Рисунок 4.1 – Результат тестування при невдалому завантаженні CSV файлу

На рисунку 4.2 після тестування з використанням невалідного файлу було проведено повноцінне функціональне тестування з використанням коректного набору даних. При успішному завантаженні файлу система автоматично відображає попередній перегляд даних у таблиці та надає інформацію про структуру даних у відповідному розділі інтерфейсу.

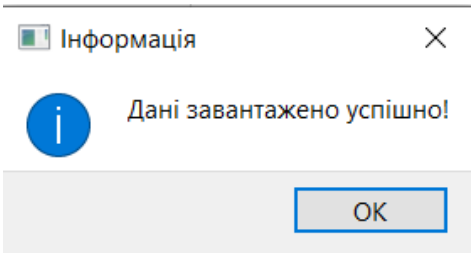

Рисунок 4.2 – Результат тестування при успішному завантаженні CSV файлу

На рисунку 4.3 зображено наступний етап тестування, яким стала перевірка правильності роботи алгоритмів виявлення аномалій. Для коректної роботи системи необхідно обрати числовий стовпець для аналізу та метод виявлення аномалій (IQR, Z-score або Контрольні карти Шухарта). При спробі запустити аналіз без вибору стовпця або методу система виводить повідомлення про помилку.

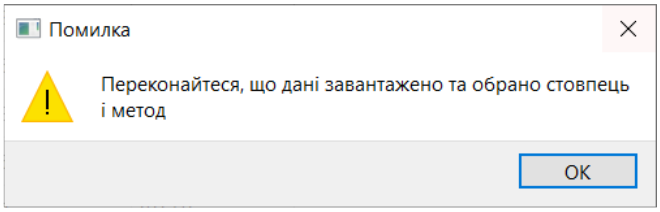

Рисунок 4.3 – Результат тестування при невибраному стовпці та методі

На рисунку 4.4 після вибору необхідних параметрів було перевірено правильність виконання аналізу даних. Система успішно визначає аномалії та відображає їх у вигляді графіка, де звичайні дані позначені синьою лінією, а аномалії проставлені червоними точками. Також програма генерує таблицю з

виявленими аномаліями, що дозволяє користувачу детально вивчити кожен аномальний запис.

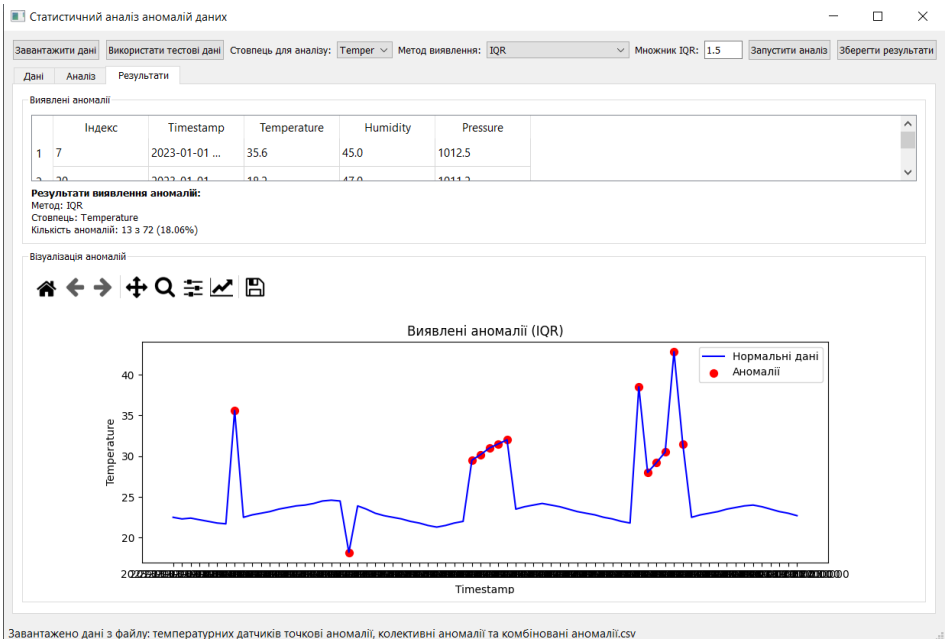

Рисунок 4.4 – Коректність побудови графіків при аналізі аномалій

На рисунку 4.5 під час тестування модуля завантаження даних було перевірено можливість системи коректно обробляти помилкові ситуації. При натисканні на кнопку «Видалити рядки з пропущеними значеннями» без попередньо завантаженого CSV-файлу система відображає інформаційне вікно з повідомленням про помилку. Це свідчить про те, що система належним чином обробляє ситуації, коли користувач намагається виконати операції без попереднього завантаження необхідних даних.

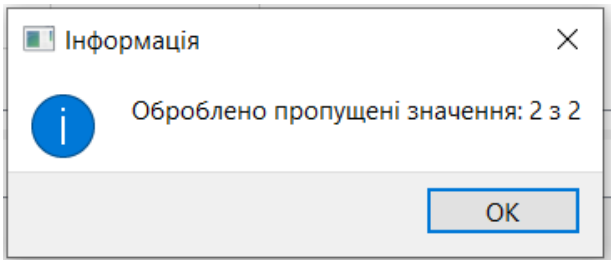

Рисунок 4.5 – Інформаційне вікно при спробі видалення рядків з пропущеними значеннями без завантаженого файлу

Тестування показало, що програма може обробляти лише файли форматів CSV, Excel та JSON, як і було передбачено в розробці. При спробі завантаження файлів інших форматів система показує відповідне повідомлення про помилку. При завантаженні пустого, неправильно відформатованого чи пошкодженого CSV файлу програма також видає інформаційне повідомлення про неможливість завантажити дані.

На рисунку 4.6 зображено інформаційне вікно про помилку, яка виникає при спробі використання функціональності для видалення рядків з пропущеними значеннями без попередньо завантаженого CSV-файлу. Це підтверджує, що система належним чином обробляє користувацькі помилки та інформує користувача про необхідність виконання попередніх кроків перед обробкою даних.

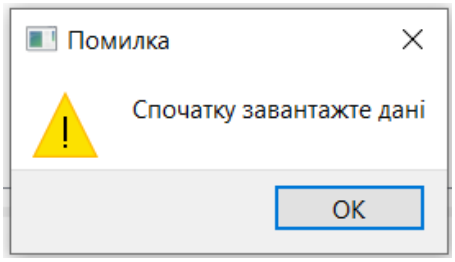

Рисунок 4.6 – Інформаційне вікно про помилку при натисканні кнопки «видалити рядки з пропущеними значеннями» без завантаженого файлу

При тестуванні з коректно завантаженим файлом функція видалення рядків з пропущеними значеннями працює належним чином – система успішно видаляє рядки та оновлює інформацію про дані, відображаючи зменшення загальної кількості рядків.

Тестування функції заміни пропущених значень середнім або медіаною також показало коректну роботу системи – пропущені значення успішно замінюються і оновлюється інформація про дані.

На рисунку 4.7 зображено повідомлення про помилку, яке з'являється при спробі заміни пропущених значень середнім значенням, коли у завантаженому

файлі немає пропущених даних. Це підтверджує, що система виконує належні перевірки перед застосуванням функцій обробки даних.

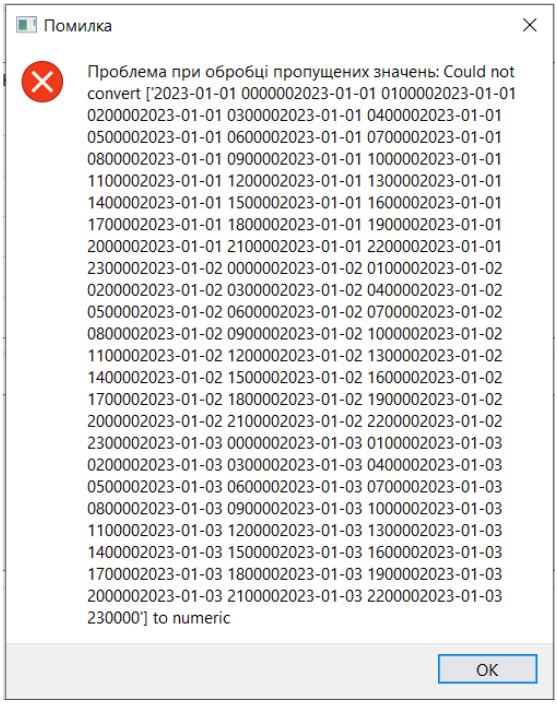

Рисунок 4.7 – Повідомлення про помилку при спробі заміни пропущених значень, коли їх немає

Важливим аспектом тестування було перевірка можливості збереження результатів. При натисканні на кнопку "Зберегти результати" користувач може вибрати місце збереження CSV-файлу з результатами аналізу та PNG або PDF файлу з візуалізацією. У разі вказання коректного шляху та імені файлу система успішно зберігає результати. Однак при вказанні неправильного імені файлу система видає повідомлення про помилку [21].

Додатково було перевірено роботу з тестовими даними. При натисканні на кнопку "Використати тестові дані" система генерує набір даних з вбудованими аномаліями та автоматично завантажує їх у програму. Це дозволяє швидко перевірити функціональність без необхідності завантаження зовнішніх файлів.

Було також проведено тестування на відображення даних у таблиці аномалій. Тестувалася здатність коректно відображати великі обсяги даних.

Проведене тестування програмного забезпечення для пошуку аномалій у даних підтвердило його функціональність, надійність і відповідність очікуваній логіці роботи. Усі основні компоненти інтерфейсу були перевірені на відповідність закладеній функціональності. Програма коректно реагує на дії користувача, забезпечуючи стабільну обробку вхідних даних, виконання алгоритмів виявлення аномалій та вивід результатів у зручній формі.

4.2 Розробка інструкції користувача

Керівництво користувача включає опис технічних умов, необхідних для стабільної роботи програмного продукту. У ньому зазначено параметри пристроїв, на яких застосунок може бути встановлений та ефективно функціонувати. Детальну інформацію щодо мінімальних та рекомендованих характеристик мобільного пристрою для запуску застосунку наведено у таблицях 4.1 та 4.2.

Таблиця 4.1 – Мінімальна конфігурація:

Тип процесора	Qualcomm Snapdragon 450
Об’єм оперативної пам’яті	1,5 ГБ для 64-разрядної системи
Місце на жорсткому диску	500 МБ
Операційна система	Android 8.0 або новіше

Таблиця 4.2 – Рекомендована конфігурація:

Тип процесора	2.4GHz Octa-core Qualcomm Snapdragon 765G
Об’єм оперативної пам’яті	4 ГБ для 64-разрядної системи
Місце на жорсткому диску	2-4 ГБ
Операційна система	Android 12 або новіше

Після встановлення програми користувач може запустити її через командний рядок або безпосередньо з IDE [22]. Основний інтерфейс програми представлений у вигляді графічного вікна з інтерактивними кнопками, випадаючими списками та полями для параметрів.

Перед початком роботи користувач має натиснути кнопку "Завантажити дані", щоб імпортувати файл з даними для аналізу, або кнопку "Використати тестові дані" для роботи з вбудованим набором даних. Після успішного завантаження стає доступним вибір стовпця для аналізу, який здійснюється через випадаючий список "Стовпець для аналізу".

Користувач також має вибрати метод виявлення аномалій: "IQR", "Z-score" та "Контрольні карти Шухарта" через відповідний випадаючий список. Залежно від обраного методу, користувач може налаштувати параметр "Поріг" для точного налаштування чутливості алгоритму.

Для запуску аналізу користувач натискає кнопку "Запустити аналіз", після чого система виконує обробку даних та відображає результати на вкладці "Результати". Візуалізація включає графік з позначеними аномаліями та таблицю з детальною інформацією про кожну виявлену аномалію.

Для збереження результатів аналізу передбачена кнопка "Зберегти результати", яка дозволяє експортувати дані у формат CSV та візуалізацію у форматі PNG або PDF.

Програма також надає інформацію про завантажені дані, включаючи розмір набору даних, типи даних у кожному стовпці та статистику щодо пропущених значень.

Таким чином, програмний продукт забезпечує зручну, інтуїтивно зрозумілу взаємодію з алгоритмами виявлення аномалій, дозволяючи проводити аналіз даних, візуалізувати результати та експортувати їх для подальшого використання.

4.3 Висновки

У четвертому розділі було проведено тестування програмного забезпечення, розробленого для пошуку аномалій у даних. Перевірено функціональність усіх основних елементів графічного інтерфейсу, включаючи завантаження даних, вибір методів аналізу, запуск алгоритмів виявлення аномалій та збереження результатів. Особливу увагу приділено перевірці стабільності обробки даних, коректності застосування методів виявлення аномалій та візуалізації результатів.

У ході тестування було підтверджено правильну роботу з різними форматами даних (CSV, Excel, JSON), а також забезпечення захисту від помилок при завантаженні порожніх або пошкоджених файлів. Система правильно реагує на відсутність вибору необхідних параметрів для аналізу, видаючи відповідні повідомлення про помилки.

Крім того, було проаналізовано реакцію системи на типові сценарії використання, зокрема роботу з тестовими даними, виявлення аномалій різними методами та збереження результатів у зовнішні файли. Усі протестовані компоненти показали високу стабільність і відповідність очікуваній логіці роботи, за винятком деяких проблем з відображенням великих обсягів даних у таблиці аномалій, що буде враховано в наступних версіях програми.

Завдяки ретельному тестуванню вдалося підтвердити ефективність реалізованого функціоналу, що дозволяє використовувати програму як зручний інструмент для виявлення та аналізу аномалій у різних наборах даних використовуючи методи статистичного аналізу.

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі вирішено актуальне науково-практичне завдання розробки програмної реалізації статистичних методів для виявлення аномалій у даних із застосуванням мови програмування Python. За результатами проведеного дослідження можна зробити наступні висновки.

Проведено аналіз існуючих статистичних методів виявлення аномалій та обґрунтовано вибір оптимальних підходів для різних типів даних. Визначено, що методи Z-score, міжквартильного розмаху (IQR) та контрольних карт Шухарта найкраще відповідають вимогам універсальності, точності та інтерпретованості результатів, що дозволяє ефективно застосовувати їх у різноманітних предметних областях. Розроблено програмний засіб для виявлення аномалій у даних з інтуїтивно зрозумілим графічним інтерфейсом користувача, що забезпечує зручну взаємодію з системою без необхідності глибоких знань програмування. Інтерфейс дозволяє завантажувати дані різних форматів, налаштовувати параметри аналізу та візуалізувати результати у зручному для інтерпретації вигляді.

Створено модульну архітектуру програмного рішення, що забезпечує гнучкість та масштабованість системи. Використання мови Python та бібліотек pandas, NumPy, SciPy, Matplotlib і seaborn дозволило реалізувати повний цикл обробки даних: від їх імпорту та попередньої обробки до аналізу, візуалізації та експорту результатів. Удосконалено підхід до програмної реалізації класичних статистичних методів виявлення аномалій через розробку окремих функціональних модулів, що легко інтегруються в аналітичні системи та можуть бути повторно використані в інших проєктах.

Розроблено узагальнений підхід до тестування методів на синтетичних і реальних наборах даних різної природи. Створено модуль генерації синтетичних даних, який дозволяє моделювати різні типи аномалій та оцінювати ефективність реалізованих алгоритмів у контрольованих умовах. Запропоновано адаптивну систему візуалізації результатів, що підтримує створення різних типів графіків

(гістограми, діаграми розмаху, часові ряди) з виділенням виявлених аномалій. Це значно спрощує інтерпретацію даних та прийняття рішень на їх основі.

Розроблений програмний засіб забезпечує можливість збереження результатів аналізу у різних форматах (CSV, PNG, PDF), що полегшує їх подальше використання в дослідницьких чи практичних цілях. Проведено всебічне тестування програмного рішення на різних наборах даних, що підтвердило його надійність, точність та ефективність у виявленні аномалій різних типів. Система успішно виявляє спайки, зсуви рівня та інші види відхилень у числових даних. Практична цінність розробленого програмного засобу полягає в можливості його застосування у різних галузях: фінансовому моніторингу, біоаналітиці, контролі якості технічних процесів та освітньому процесі. Система може використовуватись як самостійний інструмент для аналізу даних або інтегруватись в існуючі аналітичні платформи.

Результати дослідження демонструють, що поєднання класичних статистичних методів з сучасними технологіями програмування дозволяє створювати ефективні інструменти для виявлення аномалій у даних. Розроблена система не лише вирішує практичні завдання аналізу даних, але й може слугувати основою для подальших досліджень у сфері автоматизованого виявлення відхилень та прийняття рішень на основі даних.

аналізу даних, але й може слугувати основою для подальших досліджень у сфері автоматизованого виявлення відхилень та прийняття рішень на основі даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Slatkin B. Effective Python: 90 Specific Ways to Write Better Python. Addison-Wesley Professional, 2019. 480 p.

2. Rioux J. Data Analysis with Python and Pyspark. Manning Publications Co. LLC, 2022. 220 p.

3. Handbook of Anomaly Detection : with Python Outlier Detection: Build and Modernize Your Anomaly Detection Models with Examples. Independently Published, 2023. 237 p.

4. Ranga Suri N. N. R., Murty M N., Athithan G. Outlier Detection: Techniques and Applications. Cham : Springer International Publishing, 2019. 361 p.

5. Вараниця М.С. Статистичні методи для виявлення аномалій у даних із застосуванням мови програмування python. Матеріали LIV Всеукраїнської науково-технічної конференції професорсько-викладацького складу, науковців, аспірантів та студентів підрозділів університету з участю працівників підприємств (НТКП ВНТУ-2025). Вінниця, 2025. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/24516/20205 (дата звернення: 14.04.2025).

6. Вараниця М.С. Аналіз вимог до програмного забезпечення для виявлення аномалій у даних із застосуванням мови програмування python. Міжнародна науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)». Вінниця, 2025. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/24827 (дата звернення: 20.04.2025).

7. Глибоке навчання та комбіновані моделі URL: https://machinelearningmastery.com/ (дата звернення: 21.04.2025).

8. Anomaly Detection Toolkit (ADTK) – ADTK 0.6.2 documentation. URL: https://adtk.readthedocs.io/en/stable/ (date of access: 15.05.2025).

9. Machine Learning for Elasticsearch. Elastic – The Search AI Company | Elastic. URL: https://www.elastic.co/elasticsearch/machine-learning (date of access: 15.06.2025).

10. Anomaly Detection AI - Amazon Lookout for Metrics - AWS. Amazon Web Services, Inc. URL: https://aws.amazon.com/lookout-for-metrics/ (date of access: 15.06.2025).

11. NumPy. URL: https://numpy.org/ (date of access: 16.05.2025).

12. SciPy. URL: https://scipy.org/ (date of access: 17.05.2025).

13. Wolff E. Microservices: Flexible Software Architecture. Pearson Education, Limited, 2021. 186 p.

14. Hands-On Data Analysis with Pandas: Efficiently Perform Data Collection, Wrangling, Analysis, and Visualization Using Python. de Gruyter GmbH, Walter, 2019. 256 p.

15. Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming. Prentice Hall, 2015. 648 p.

16. Scott Berinato, Good Charts Workbook: Tips, Tools, and Exercises for Making Better Data Visualizations. ArtHuss, 2022. 288 p.

17. Wringe C. A. Effective Teaching of Modern Languages. Taylor & Francis Group, 2017. 287 p.

18. Kivy – Interactive Applications and Games in Python - Second Edition. Packt Publishing, 2015. 206 p.

19. Joshi P., Mahalle P. N. Data Storytelling and Visualization with Tableau: A Hands-On Approach. Taylor & Francis Group, 2022. 191 p.

20. Mehrotra K. G., Mohan C. K., Huang H. Anomaly Detection Principles and Algorithms. Springer, 2019. 239 p.

21. Arman. Mastering GraphQL Error Handling: Strategies, Best Practices, and Future Trends. Testfully. URL: https://testfully.io/blog/graphql-error-handling/ (date of access: 15.06.2025).

22. Effective Pycharm: Learn the Pycharm IDE with a Hands-On Approach. Independently Published, 2019. 221 p.

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., проф. О. Н. Романюк
"25" 03 2025 р.

Технічне завдання
на бакалаврську кваліфікаційну роботу «Програмна реалізація статистичних методів для виявлення аномалій у даних із застосуванням мови програмування Python»
за спеціальністю 121 Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
к.т.н., доц. каф. ПЗ Д.І. Катєльніков
" 24 " 03 2025 р.
Виконав:
студент гр.4ПІ-21б М. С. Вараниця
" 24 " 03 2025 р.

Вінниця – 2025 року

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Програмна реалізація статистичних методів для виявлення аномалій у даних із застосуванням мови програмування Python».

Галузь застосування – інформаційні технології.

2. Підстава для розробки.

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ від 20 березня 2025 р. №97 ректора по ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки.

Метою бакалаврської кваліфікаційної роботи є створення програмної реалізації для виявлення аномалій у даних з використанням статистичних методів, реалізованих мовою програмування Python.

Призначення роботи – розробка програмної реалізації статистичних методів для виявлення аномалій у даних із застосуванням мови програмування Python.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БКР:

1. Wolff E. Microservices: Flexible Software Architecture. Pearson Education, Limited, 2021. 186 p.

2. Ranga Suri N. N. R., Murty M N., Athithan G. Outlier Detection: Techniques and Applications. Cham : Springer International Publishing, 2019. 361 p.

3. Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming. Prentice Hall, 2015. 648 p.

4. Mehrotra K. G., Mohan C. K., Huang H. Anomaly Detection Principles and Algorithms. Springer, 2019. 239 p.

5. Технічні вимоги

Дії програмної реалізації – завантаження та обробка табличних даних у форматах CSV, Excel, JSON; застосування статистичних методів виявлення аномалій; візуалізація даних та результатів аналізу через гістограми, діаграми розмаху, часові ряди; експорт результатів у табличні файли та графічні зображення; статистичні методи – Z-score аналіз; міжквартильний розмах (IQR); контрольні карти Шухарта; Обробка даних – завантаження через pandas для роботи з різними форматами файлів; статистична обробка через NumPy та SciPy для математичних розрахунків; контроль якості – перевірка типів даних та їх відповідності для статистичного аналізу; валідація порогових значень та параметрів методів; кінцевий результат – інтерактивна програма з графічним інтерфейсом для статистичного аналізу аномалій у даних.

6. Конструктивні вимоги.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до БКР;
– технічне завдання;
– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз розвитку систем виявлення аномалій у даних та формування завдань дослідження	25.03.25 – 31.03.2025
2	Розробка модульної архітектурної системи	31.03.25 – 05.04.2025
3	Удосконалення статичних методів виявлення аномалій	05.03.2025 – 07.04.2025
4	Розробка адаптивної системи візуалізації результатів аналізу аномалій	07.04.2025 – 12.04.2025
5	Розробка узагальненого підходу до тестування методів виявлення аномалій	12.04.2025 – 23.04.2025
6	Розробка програмної реалізації статистичних методів для виявлення аномалій у даних	23.04.2025 – 01.05.2025
7	Тестування програмної реалізації	01.05.2025 – 10.05.2025
8	Оформлення матеріалів до захисту БКР	10.05.2025 – 30.05.25

10. Порядок контролю та прийняття.

Виконання етапів бакалаврської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Захист бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком**.**

Додаток Б – Протокол перевірки БКР на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Програмна реалізація статистичних методів для виявлення аномалій у даних із застосуванням мови програмування Python.

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра програмного забезпечення, ФІТКІ, 4ПІ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 6,8 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Катєльніков Д.І., к.т.н., доц. каф. ПЗ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Вараниця М.С.
(підпис) (прізвище, ініціали)

Додаток В – Лістинг програмної реалізації

```
import os
import sys

import numpy as np
import pandas as pd
import seaborn as sns
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import (QApplication, QMainWindow, QWidget,
QVBoxLayout, QHBoxLayout,
                             QPushButton, QFileDialog, QLabel, QComboBox, QTabWidget,
                             QLineEdit, QMessageBox, QTableWidget, QTableWidgetItem,
                             QGroupBox, QRadioButton, QGridLayout,
                             QScrollArea, QSplitter, QFrame)
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as
FigureCanvas
from matplotlib.backends.backend_qt5agg import NavigationToolbar2QT as
NavigationToolbar
from matplotlib.figure import Figure
from scipy import stats


class AnomalyDetectionApp(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Статистичний аналіз аномалій даних")
        self.setGeometry(100, 100, 1200, 800)

        # Initialize data attributes
```

```
self.df = None
self.selected_column = None
self.anomalies = None
self.method = None

# Create main layout
self.central_widget = QWidget()
self.setCentralWidget(self.central_widget)
self.main_layout = QVBoxLayout(self.central_widget)

# Create toolbar
self.create_toolbar()

# Create tabs
self.tabs = QTabWidget()
self.main_layout.addWidget(self.tabs)

# Create individual tabs
self.data_tab = QWidget()
self.analysis_tab = QWidget()
self.results_tab = QWidget()

self.tabs.addTab(self.data_tab, "Дані")
self.tabs.addTab(self.analysis_tab, "Аналіз")
self.tabs.addTab(self.results_tab, "Результати")

# Set up individual tabs
self.setup_data_tab()
self.setup_analysis_tab()
self.setup_results_tab()
```

```
    # Status bar
    self.statusBar().showMessage("Готовий до роботи")

    # Create test data
    self.create_test_data()

def create_toolbar(self):
    toolbar_layout = QHBoxLayout()

    # Load data button
    self.load_btn = QPushButton("Завантажити дані")
    self.load_btn.clicked.connect(self.load_data)
    toolbar_layout.addWidget(self.load_btn)

    # Test data button
    self.test_data_btn = QPushButton("Використати тестові дані")
    self.test_data_btn.clicked.connect(self.use_test_data)
    toolbar_layout.addWidget(self.test_data_btn)

    # Column selection
    self.column_label = QLabel("Стовпець для аналізу:")
    toolbar_layout.addWidget(self.column_label)

    self.column_combo = QComboBox()
    self.column_combo.currentTextChanged.connect(self.column_selected)
    toolbar_layout.addWidget(self.column_combo)

    # Method selection
    self.method_label = QLabel("Метод виявлення:")
```

```
        toolbar_layout.addWidget(self.method_label)

        self.method_combo = QComboBox()
        self.method_combo.addItems(["Z-score", "IQR", "Контрольні карти
Шухарта", "Ковзне вікно"])
        self.method_combo.currentTextChanged.connect(self.method_selected)
        toolbar_layout.addWidget(self.method_combo)

        # Threshold input
        self.threshold_label = QLabel("Поріг:")
        toolbar_layout.addWidget(self.threshold_label)

        self.threshold_input = QLineEdit("3.0")
        self.threshold_input.setFixedWidth(50)
        toolbar_layout.addWidget(self.threshold_input)

        # Run analysis button
        self.run_btn = QPushButton("Запустити аналіз")
        self.run_btn.clicked.connect(self.run_analysis)
        toolbar_layout.addWidget(self.run_btn)

        # Save results button
        self.save_btn = QPushButton("Зберегти результати")
        self.save_btn.clicked.connect(self.save_results)
        toolbar_layout.addWidget(self.save_btn)

        # Add toolbar to main layout
        self.main_layout.addLayout(toolbar_layout)

    def setup_data_tab(self):
```

```
layout = QVBoxLayout(self.data_tab)

# Create a splitter for resizable sections
splitter = QSplitter(Qt.Vertical)
layout.addWidget(splitter)

# Data preview table
table_frame = QFrame()
table_layout = QVBoxLayout(table_frame)
table_layout.addWidget(QLabel("<b>Попередній перегляд даних</b>"))

self.data_table = QTableWidget()
table_layout.addWidget(self.data_table)
splitter.addWidget(table_frame)

# Data info section
info_frame = QFrame()
info_layout = QVBoxLayout(info_frame)
info_layout.addWidget(QLabel("<b>Інформація про дані</b>"))

self.data_info = QLabel("Дані не завантажено")
info_scroll = QScrollArea()
info_scroll.setWidgetResizable(True)
info_scroll.setWidget(self.data_info)
info_layout.addWidget(info_scroll)
splitter.addWidget(info_frame)

# Missing values handling
missing_frame = QGroupBox("Обробка пропущених значень")
missing_layout = QVBoxLayout(missing_frame)
```

```
        self.missing_drop = QRadioButton("Видалити рядки з пропущеними значеннями")
        self.missing_drop.setChecked(True)
        missing_layout.addWidget(self.missing_drop)

        self.missing_mean = QRadioButton("Замінити середнім значенням")
        missing_layout.addWidget(self.missing_mean)

        self.missing_median = QRadioButton("Замінити медіаною")
        missing_layout.addWidget(self.missing_median)

        self.handle_missing_btn = QPushButton("Обробити пропущені значення")
        self.handle_missing_btn.clicked.connect(self.handle_missing_values)
        missing_layout.addWidget(self.handle_missing_btn)

        info_layout.addWidget(missing_frame)

    def setup_analysis_tab(self):
        layout = QVBoxLayout(self.analysis_tab)

        # Statistics section
        stats_group = QGroupBox("Описова статистика")
        stats_layout = QVBoxLayout(stats_group)
        self.stats_label = QLabel("Оберіть стовпець для відображення статистики")
        stats_layout.addWidget(self.stats_label)
        layout.addWidget(stats_group)

        # Visualization section
        viz_group = QGroupBox("Візуалізація")
```

```
    viz_layout = QGridLayout(viz_group)

    # Histogram
    self.hist_canvas = FigureCanvas(Figure(figsize=(5, 4)))
    self.hist_ax = self.hist_canvas.figure.add_subplot(111)
    self.hist_toolbar = NavigationToolbar(self.hist_canvas, self)
    viz_layout.addWidget(self.hist_toolbar, 0, 0)
    viz_layout.addWidget(self.hist_canvas, 1, 0)

    # Box plot
    self.box_canvas = FigureCanvas(Figure(figsize=(5, 4)))
    self.box_ax = self.box_canvas.figure.add_subplot(111)
    self.box_toolbar = NavigationToolbar(self.box_canvas, self)
    viz_layout.addWidget(self.box_toolbar, 0, 1)
    viz_layout.addWidget(self.box_canvas, 1, 1)

    # Time series
    self.ts_canvas = FigureCanvas(Figure(figsize=(10, 4)))
    self.ts_ax = self.ts_canvas.figure.add_subplot(111)
    self.ts_toolbar = NavigationToolbar(self.ts_canvas, self)
    viz_layout.addWidget(self.ts_toolbar, 2, 0, 1, 2)
    viz_layout.addWidget(self.ts_canvas, 3, 0, 1, 2)

    layout.addWidget(viz_group)

def setup_results_tab(self):
    layout = QVBoxLayout(self.results_tab)

    # Results section
    results_group = QGroupBox("Виявлені аномалії")
```

```
        results_layout = QVBoxLayout(results_group)

        # Table for anomalies
        self.anomaly_table = QTableWidget()
        results_layout.addWidget(self.anomaly_table)

        # Summary statistics
        self.anomaly_stats = QLabel("Запустіть аналіз для відображення
результатів")
        results_layout.addWidget(self.anomaly_stats)

        layout.addWidget(results_group)

        # Visualization of anomalies
        anomaly_viz_group = QGroupBox("Візуалізація аномалій")
        anomaly_viz_layout = QVBoxLayout(anomaly_viz_group)

        self.anomaly_canvas = FigureCanvas(Figure(figsize=(10, 6)))
        self.anomaly_ax = self.anomaly_canvas.figure.add_subplot(111)
        self.anomaly_toolbar = NavigationToolbar(self.anomaly_canvas, self)

        anomaly_viz_layout.addWidget(self.anomaly_toolbar)
        anomaly_viz_layout.addWidget(self.anomaly_canvas)

        layout.addWidget(anomaly_viz_group)

    def create_test_data(self):
        # Create time series with anomalies
        np.random.seed(42)
        dates = pd.date_range(start='2023-01-01', periods=200, freq='D')
```

```
        # Normal data with seasonal pattern
        values = 10 + 2 * np.sin(np.arange(200) * 2 * np.pi / 30) + np.random.normal(0, 0.5, 200)

        # Insert anomalies
        values[20] = 20  # Spike
        values[50] = 0  # Drop
        values[80:85] = values[80:85] + 5  # Level shift
        values[150] = 18  # Another spike

        # Create a DataFrame
        self.test_df = pd.DataFrame({
            'Date': dates,
            'Value': values,
            'Category': np.random.choice(['A', 'B', 'C'], size=200),
            'Count': np.random.poisson(lam=5, size=200)
        })

        # Add a few missing values
        self.test_df.loc[10, 'Value'] = np.nan
        self.test_df.loc[60, 'Count'] = np.nan

    def use_test_data(self):
        self.df = self.test_df.copy()
        self.update_data_view()
        QMessageBox.information(self, "Інформація", "Тестові дані завантажено успішно!")
        self.statusBar().showMessage("Тестові дані завантажено")
```

```
    def load_data(self):
        file_dialog = QFileDialog()
        file_path, _ = file_dialog.getOpenFileName(
            self, "Відкрити файл з даними", "",
            "CSV Files (*.csv);;Excel Files (*.xlsx *.xls);;JSON Files (*.json);;All Files (*)"
        )

        if not file_path:
            return

        try:
            if file_path.endswith('.csv'):
                self.df = pd.read_csv(file_path)
            elif file_path.endswith(('.xlsx', '.xls')):
                self.df = pd.read_excel(file_path)
            elif file_path.endswith('.json'):
                self.df = pd.read_json(file_path)
            else:
                QMessageBox.warning(self, "Помилка", "Непідтримуваний формат файлу")
                return

            self.update_data_view()
            QMessageBox.information(self, "Інформація", "Дані завантажено успішно!")
            self.statusBar().showMessage(f"Завантажено дані з файлу: {os.path.basename(file_path)}")

        except Exception as e:
```

```
            QMessageBox.critical(self, "Помилка", f"Не вдалося завантажити дані: {str(e)}")

    def update_data_view(self):
        if self.df is None:
            return

        # Update column selector
        self.column_combo.clear()
        numeric_columns = self.df.select_dtypes(include=[np.number]).columns.tolist()
        self.column_combo.addItems(numeric_columns)

        # Update data table
        self.data_table.setRowCount(min(100, len(self.df)))
        self.data_table.setColumnCount(len(self.df.columns))
        self.data_table.setHorizontalHeaderLabels(self.df.columns)

        for i in range(min(100, len(self.df))):
            for j, col in enumerate(self.df.columns):
                value = str(self.df.iloc[i, j])
                self.data_table.setItem(i, j, QTableWidgetItem(value))

        # Update data info
        missing_data = self.df.isna().sum()
        missing_percent = 100 * missing_data / len(self.df)

        info_text = f"<b>Розмір даних:</b> {self.df.shape[0]} рядків × {self.df.shape[1]} стовпців<br>"
        info_text += f"<b>Типи даних:</b><br>"
        for col, dtype in self.df.dtypes.items():
```

```
            info_text += f"  - {col}: {dtype}<br>"

        info_text += f"<b>Пропущені значення:</b><br>"
        for col in self.df.columns:
            if missing_data[col] > 0:
                info_text += f"  - {col}: {missing_data[col]}
({missing_percent[col]:.2f}%)<br>"

        self.data_info.setText(info_text)

    def handle_missing_values(self):
        if self.df is None:
            QMessageBox.warning(self, "Помилка", "Спочатку завантажте дані")
            return

        try:
            before_count = self.df.isna().sum().sum()

            if self.missing_drop.isChecked():
                self.df = self.df.dropna()
            elif self.missing_mean.isChecked():
                self.df = self.df.fillna(self.df.mean())
            elif self.missing_median.isChecked():
                self.df = self.df.fillna(self.df.median())

            after_count = self.df.isna().sum().sum()

            QMessageBox.information(self, "Інформація",
                                    f"Оброблено пропущені значення: {before_count -
after_count} з {before_count}")
```

```
            self.update_data_view()

        except Exception as e:
            QMessageBox.critical(self, "Помилка", f"Проблема при обробці
пропущених значень: {str(e)}")

    def column_selected(self, column_name):
        if not column_name or self.df is None:
            return

        self.selected_column = column_name
        self.update_analysis_tab()

    def method_selected(self, method_name):
        self.method = method_name
        # Update threshold label based on method
        if method_name == "Z-score":
            self.threshold_label.setText("Поріг (σ):")
            self.threshold_input.setText("3.0")
        elif method_name == "IQR":
            self.threshold_label.setText("Множник IQR:")
            self.threshold_input.setText("1.5")
        elif method_name == "Контрольні карти Шухарта":
            self.threshold_label.setText("Сигма:")
            self.threshold_input.setText("3.0")
        elif method_name == "Ковзне вікно":
            self.threshold_label.setText("Вікно:")
            self.threshold_input.setText("10")
```

```
    def update_analysis_tab(self):
        if self.df is None or self.selected_column is None:
            return

        try:
            # Update stats
            stats_text = self.df[self.selected_column].describe().to_string()
            self.stats_label.setText(f"<pre>{stats_text}</pre>")

            # Update histogram
            self.hist_ax.clear()
            self.hist_ax.set_title(f'Гістограма: {self.selected_column}')
            sns.histplot(self.df[self.selected_column], ax=self.hist_ax, kde=True)
            self.hist_canvas.draw()

            # Update box plot
            self.box_ax.clear()
            self.box_ax.set_title(f'Діаграма розмаху: {self.selected_column}')
            sns.boxplot(y=self.df[self.selected_column], ax=self.box_ax)
            self.box_canvas.draw()

            # Update time series if possible
            self.ts_ax.clear()
            date_cols = [col for col in self.df.columns if 'date' in col.lower() or 'time' in
col.lower()]

            if date_cols:
                date_col = date_cols[0]
                self.ts_ax.set_title(f'Часовий ряд: {self.selected_column}')
                self.ts_ax.plot(self.df[date_col], self.df[self.selected_column])
```

```
            self.ts_ax.set_xlabel(date_col)
            self.ts_ax.set_ylabel(self.selected_column)
        else:
            # Use index as time
            self.ts_ax.set_title(f'Значення: {self.selected_column}')
            self.ts_ax.plot(self.df.index, self.df[self.selected_column])
            self.ts_ax.set_xlabel('Індекс')
            self.ts_ax.set_ylabel(self.selected_column)

        self.ts_canvas.draw()

    except Exception as e:
        QMessageBox.warning(self, "Помилка", f"Помилка під час аналізу: {str(e)}")

def run_analysis(self):
    if self.df is None or self.selected_column is None or self.method is None:
        QMessageBox.warning(self, "Помилка", "Переконайтеся, що дані завантажено та обрано стовпець і метод")
        return

    try:
        # Get threshold from input
        threshold = float(self.threshold_input.text())

        # Detect anomalies based on selected method
        if self.method == "Z-score":
            self.anomalies = self.detect_zscore_anomalies(threshold)
        elif self.method == "IQR":
            self.anomalies = self.detect_iqr_anomalies(threshold)
```

```
            elif self.method == "Контрольні карти Шухарта":
                self.anomalies = self.detect_control_chart_anomalies(threshold)
            elif self.method == "Ковзне вікно":
                self.anomalies = self.detect_moving_window_anomalies(int(threshold))

            # Update results tab
            self.update_results_tab()

            # Switch to results tab
            self.tabs.setCurrentIndex(2)

        except Exception as e:
            QMessageBox.critical(self, "Помилка", f"Помилка під час аналізу:
{str(e)}")

    def detect_zscore_anomalies(self, threshold):
        series = self.df[self.selected_column]
        z_scores = np.abs(stats.zscore(series))
        return z_scores > threshold

    def detect_iqr_anomalies(self, threshold):
        series = self.df[self.selected_column]
        q1 = series.quantile(0.25)
        q3 = series.quantile(0.75)
        iqr = q3 - q1
        lower_bound = q1 - threshold * iqr
        upper_bound = q3 + threshold * iqr
        return (series < lower_bound) | (series > upper_bound)

    def detect_control_chart_anomalies(self, threshold):
```

```
    series = self.df[self.selected_column]
    mean = series.mean()
    std = series.std()
    upper_control_limit = mean + threshold * std
    lower_control_limit = mean - threshold * std
    return (series < lower_control_limit) | (series > upper_control_limit)

def detect_moving_window_anomalies(self, window_size):
    series = self.df[self.selected_column]
    if len(series) <= window_size:
        return pd.Series(False, index=series.index)

    rolling_mean = series.rolling(window=window_size, center=True).mean()
    rolling_std = series.rolling(window=window_size, center=True).std()

    # For the first and last window/2 points, use the closest valid value
    rolling_mean = rolling_mean.fillna(method='bfill').fillna(method='ffill')
    rolling_std = rolling_std.fillna(method='bfill').fillna(method='ffill')

    z_scores = np.abs((series - rolling_mean) / rolling_std)
    return z_scores > 3.0  # Using 3 sigma rule for the moving window

def update_results_tab(self):
    if self.anomalies is None:
        return

    # Update anomaly table
    anomaly_indices = self.df.index[self.anomalies]
    anomaly_data = self.df.loc[anomaly_indices]
```

```
        self.anomaly_table.setRowCount(len(anomaly_data))
        self.anomaly_table.setColumnCount(len(self.df.columns) + 1)  # +1 for index

        header_labels = ['Індекс'] + self.df.columns.tolist()
        self.anomaly_table.setHorizontalHeaderLabels(header_labels)

        for i, (idx, row) in enumerate(anomaly_data.iterrows()):
            self.anomaly_table.setItem(i, 0, QTableWidgetItem(str(idx)))
            for j, col in enumerate(self.df.columns):
                self.anomaly_table.setItem(i, j + 1, QTableWidgetItem(str(row[col])))

        # Update anomaly stats
        anomaly_count = self.anomalies.sum()
        total_count = len(self.anomalies)
        percent = 100 * anomaly_count / total_count if total_count > 0 else 0

        stats_text = f"<b>Результати виявлення аномалій:</b><br>"
        stats_text += f"Метод: {self.method}<br>"
        stats_text += f"Стовпець: {self.selected_column}<br>"
        stats_text += f"Кількість аномалій: {anomaly_count} з {total_count}
({percent:.2f}%)"

        self.anomaly_stats.setText(stats_text)

        # Update visualization
        self.anomaly_ax.clear()

        # Check if there are date columns for time series visualization
        date_cols = [col for col in self.df.columns if 'date' in col.lower() or 'time' in
col.lower()]
```

```
        if date_cols:
            date_col = date_cols[0]
            # Plot normal data
            self.anomaly_ax.plot(self.df[date_col], self.df[self.selected_column], 'b-',
label='Нормальні дані')
            # Plot anomalies
            anomaly_data = self.df[self.anomalies]
            self.anomaly_ax.scatter(anomaly_data[date_col],
anomaly_data[self.selected_column], color='red',
                                    label='Аномалії', s=50)
            self.anomaly_ax.set_xlabel(date_col)
        else:
            # Use index for visualization
            self.anomaly_ax.plot(self.df.index, self.df[self.selected_column], 'b-',
label='Нормальні дані')
            # Plot anomalies
            anomaly_indices = self.df.index[self.anomalies]
            anomaly_values = self.df.loc[anomaly_indices, self.selected_column]
            self.anomaly_ax.scatter(anomaly_indices, anomaly_values, color='red',
                                    label='Аномалії', s=50)
            self.anomaly_ax.set_xlabel('Індекс')

        self.anomaly_ax.set_ylabel(self.selected_column)
        self.anomaly_ax.set_title(f'Виявлені аномалії ({self.method})')
        self.anomaly_ax.legend()
        self.anomaly_canvas.draw()

    def save_results(self):
        if self.df is None or self.anomalies is None:
```

```
            QMessageBox.warning(self, "Помилка", "Спочатку необхідно виконати аналіз")
            return

        try:
            # Save data with anomaly flags
            file_dialog = QFileDialog()
            csv_path, _ = file_dialog.getSaveFileName(
                self, "Зберегти результати", "", "CSV Files (*.csv)"
            )

            if csv_path:
                result_df = self.df.copy()
                result_df['is_anomaly'] = self.anomalies
                result_df.to_csv(csv_path, index=True)
                QMessageBox.information(self, "Інформація", f"Результати збережено у {csv_path}")

            # Save visualization
            img_path, _ = file_dialog.getSaveFileName(
                self, "Зберегти візуалізацію", "", "PNG Files (*.png);;PDF Files (*.pdf)"
            )

            if img_path:
                self.anomaly_canvas.figure.savefig(img_path, dpi=300, bbox_inches='tight')
                QMessageBox.information(self, "Інформація", f"Візуалізацію збережено у {img_path}")

        except Exception as e:
```

```
            QMessageBox.critical(self, "Помилка", f"Помилка при збереженні результатів: {str(e)}")


if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = AnomalyDetectionApp()
    window.show()
    sys.exit(app.exec_())
```

Додаток Г – Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

ПРОГРАМНА РЕАЛІЗАЦІЯ СТАТИСТИЧНИХ МЕТОДІВ ДЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ У ДАНИХ ІЗ ЗАСТОСУВАННЯМ МОВИ ПРОГРАМУВАННЯ PYTHON

Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Актуальність теми

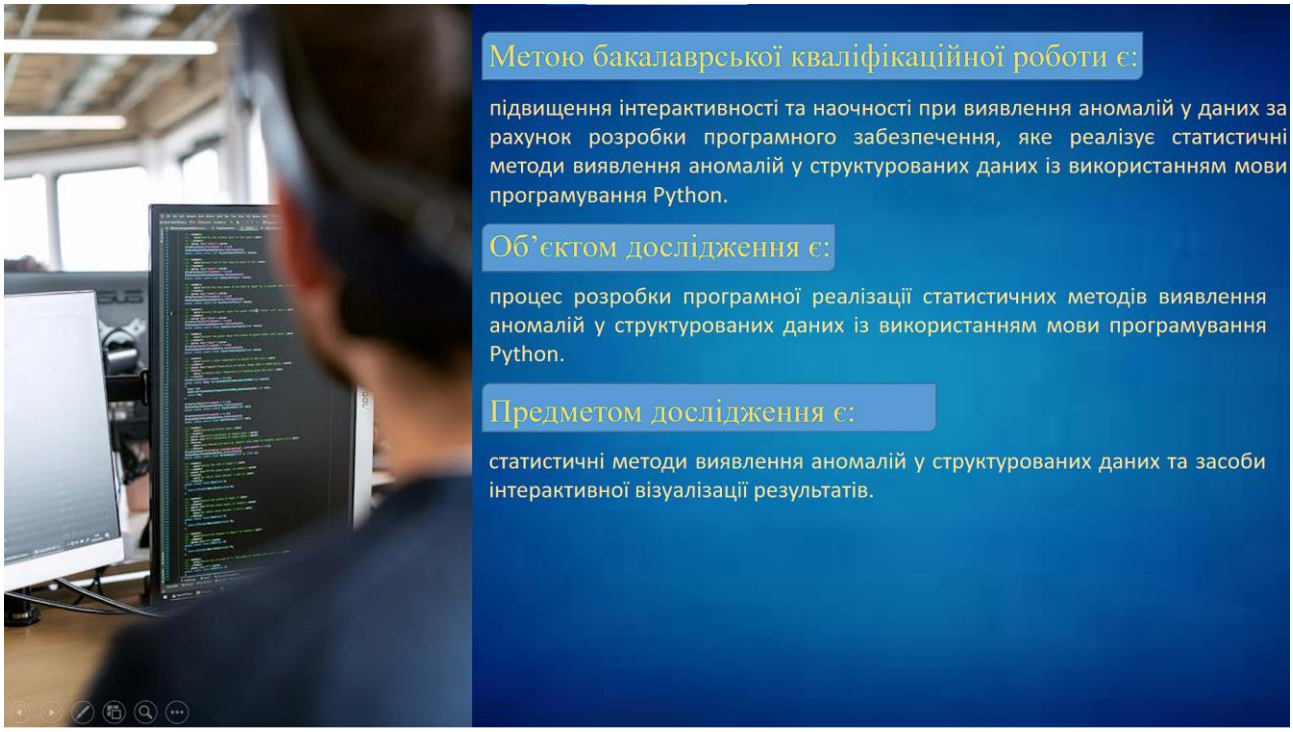

Рисунок Г.3 – Мета, об'єкт дослідження та предмет дослідження

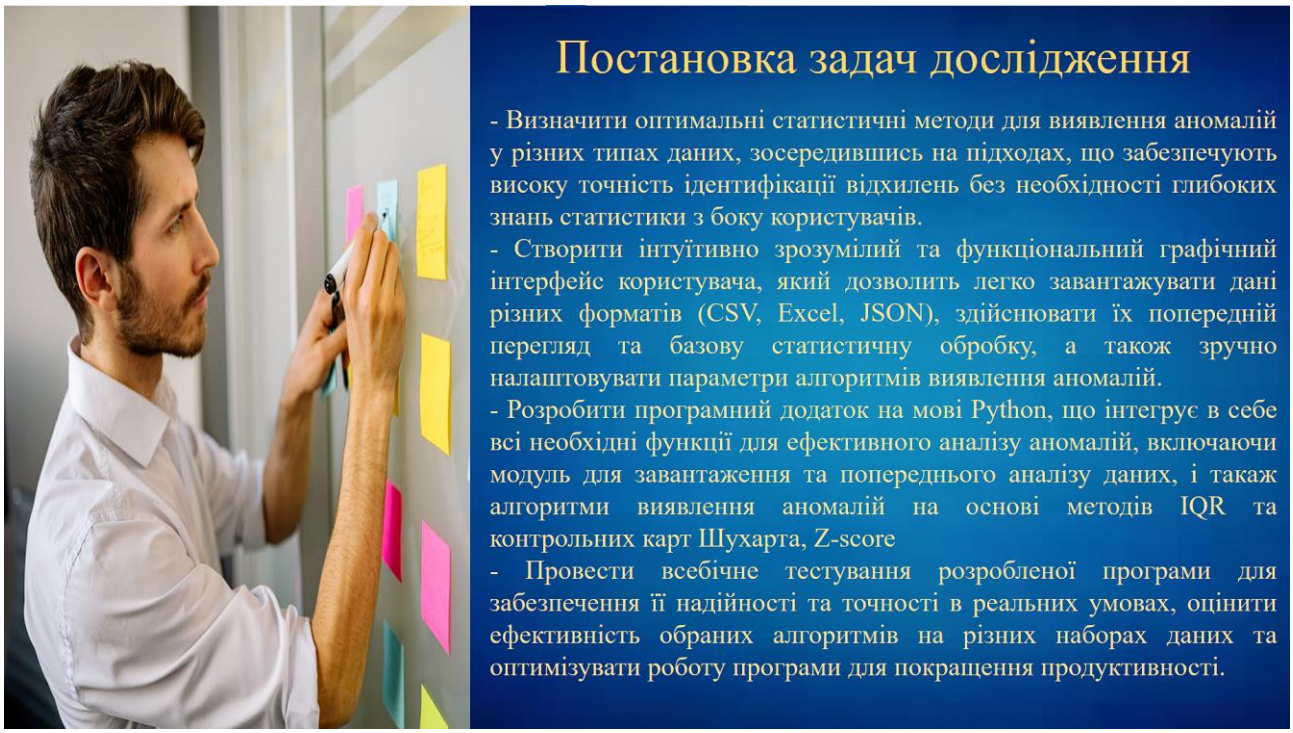

Рисунок Г.4 – Постановка задач дослідження

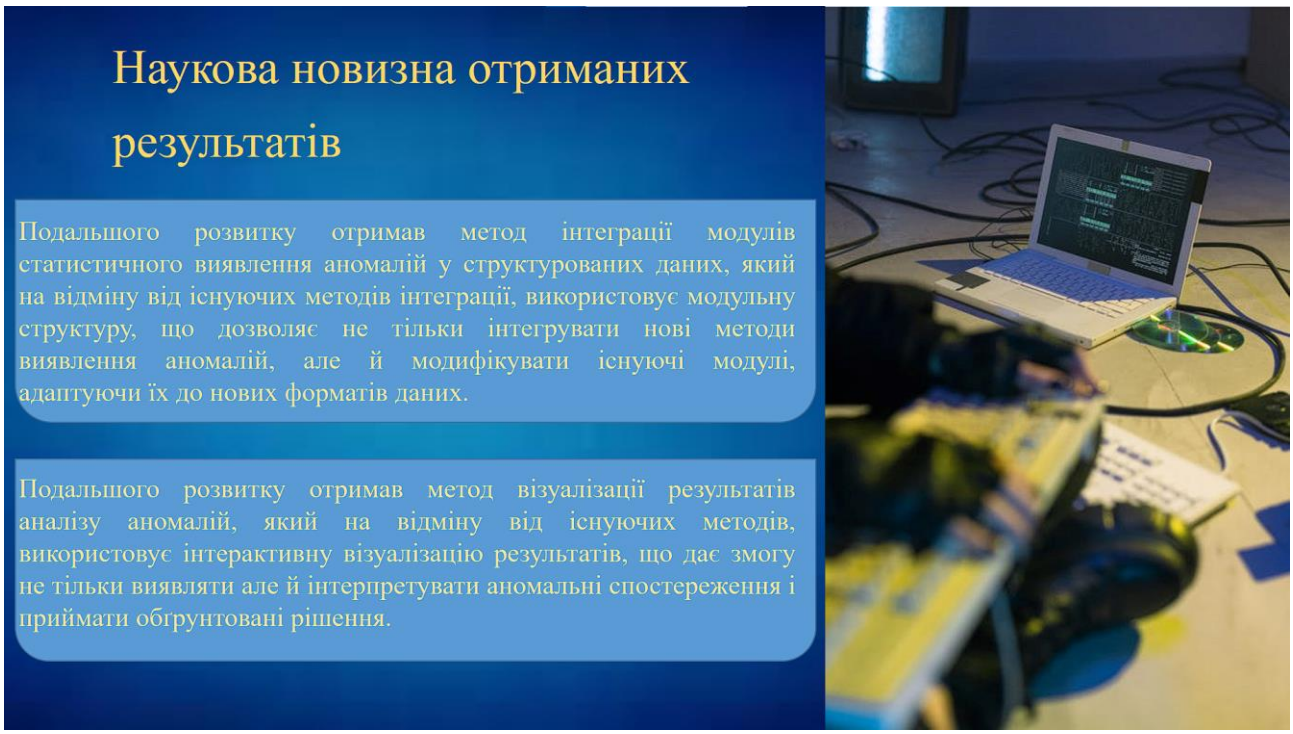

Рисунок Г.5 – Наукова новизна отриманих результатів

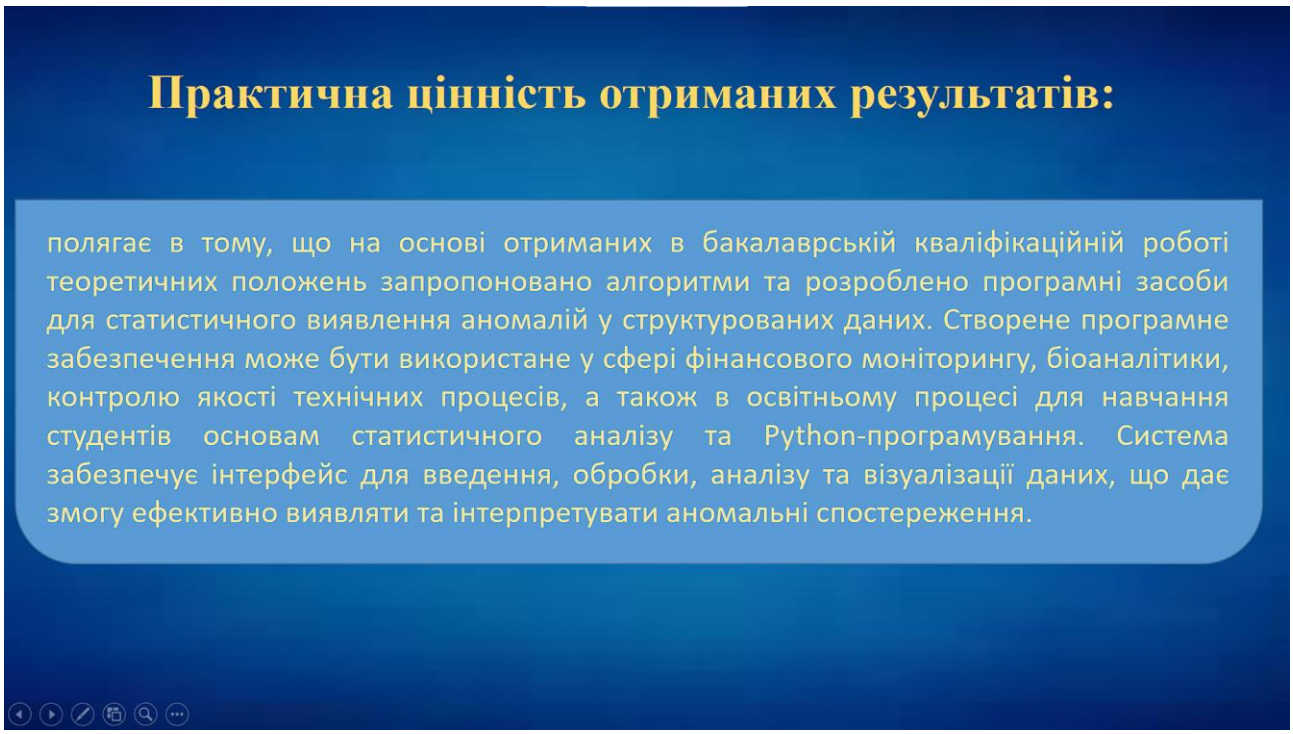

Рисунок Г.6 – Практична цінність отриманих результатів:

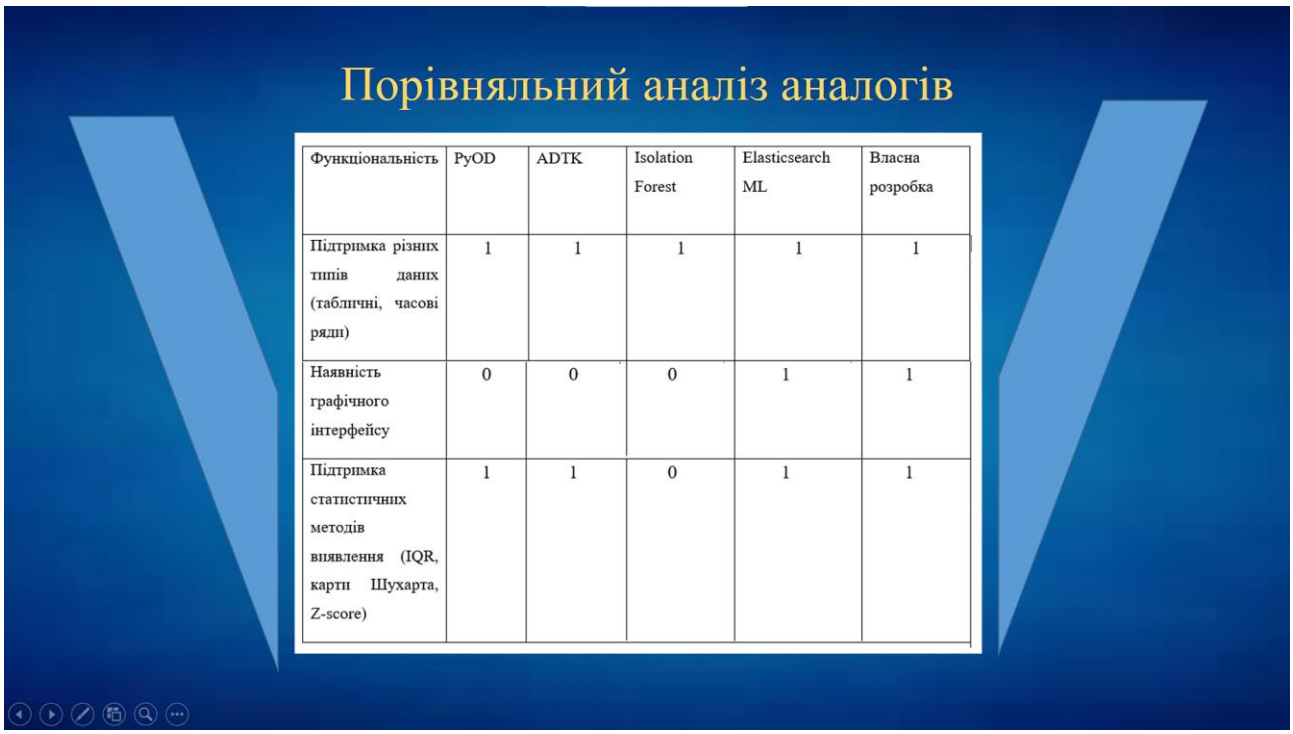

Рисунок Г.7 – Порівняльний аналіз аналогів

Рисунок Г.8 – Порівняльний аналіз аналогів

Рисунок Г.9 – Блок-схема загальної роботи системи

Рисунок Г.10 – Блок-схема виявлення аномалій методом Z-score

Рисунок Г.11 – Блок-схема виявлення аномалій методом IQR

Рисунок Г.12 – Блок-схема виявлення аномалій методом карти Шухарта

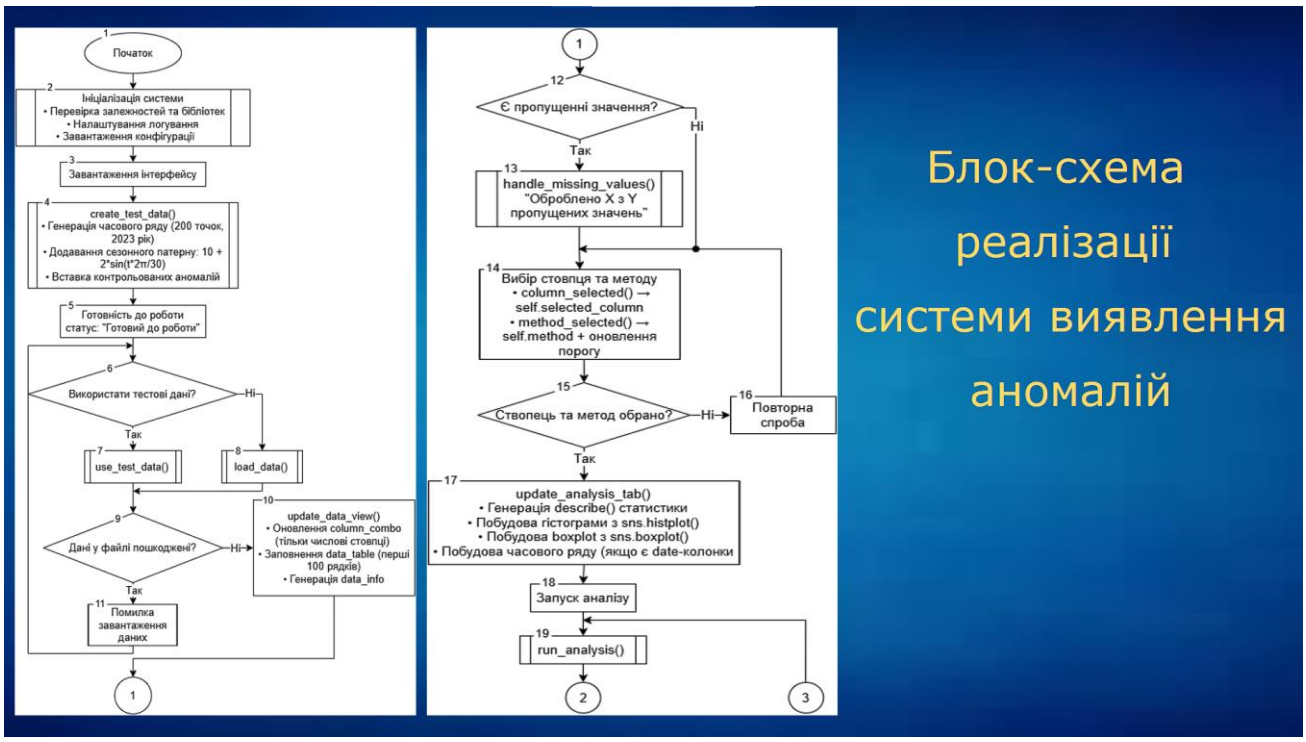

Рисунок Г.13 – Блок-схема реалізації системи виявлення аномалій

Рисунок Г.14 – Блок-схема реалізації системи виявлення аномалій

Рисунок Г.15 – Блок-схема адаптивної системи візуалізації

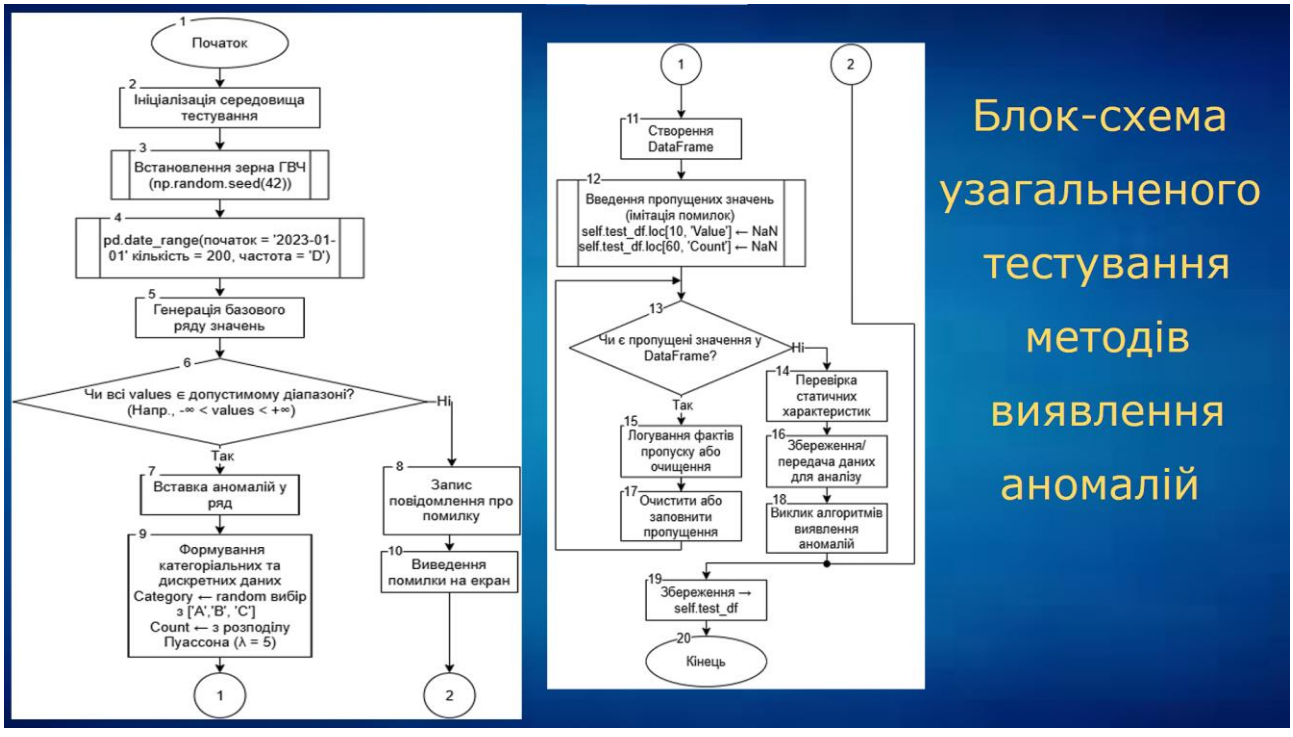

Рисунок Г.16 – Блок-схема узагальненого тестування методів виявлення аномалій

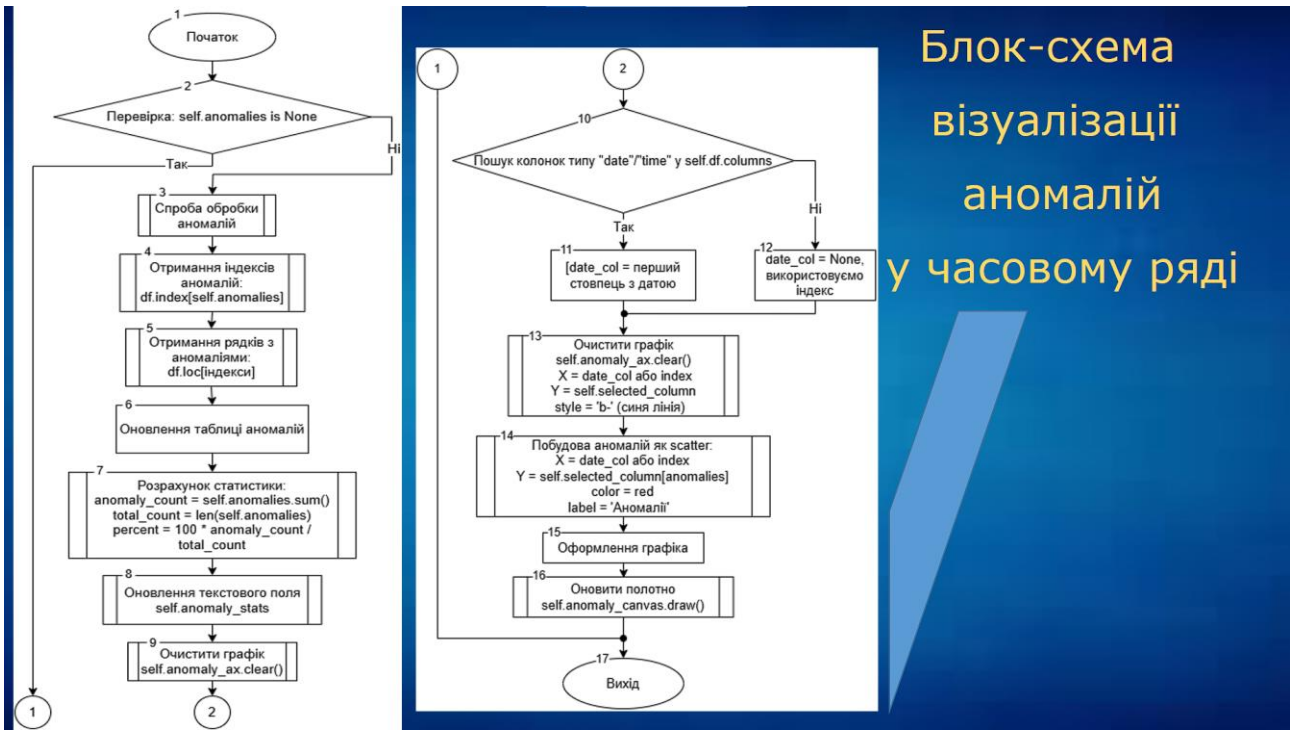

Рисунок Г.17 – Блок-схема візуалізації аномалій у часовому ряді

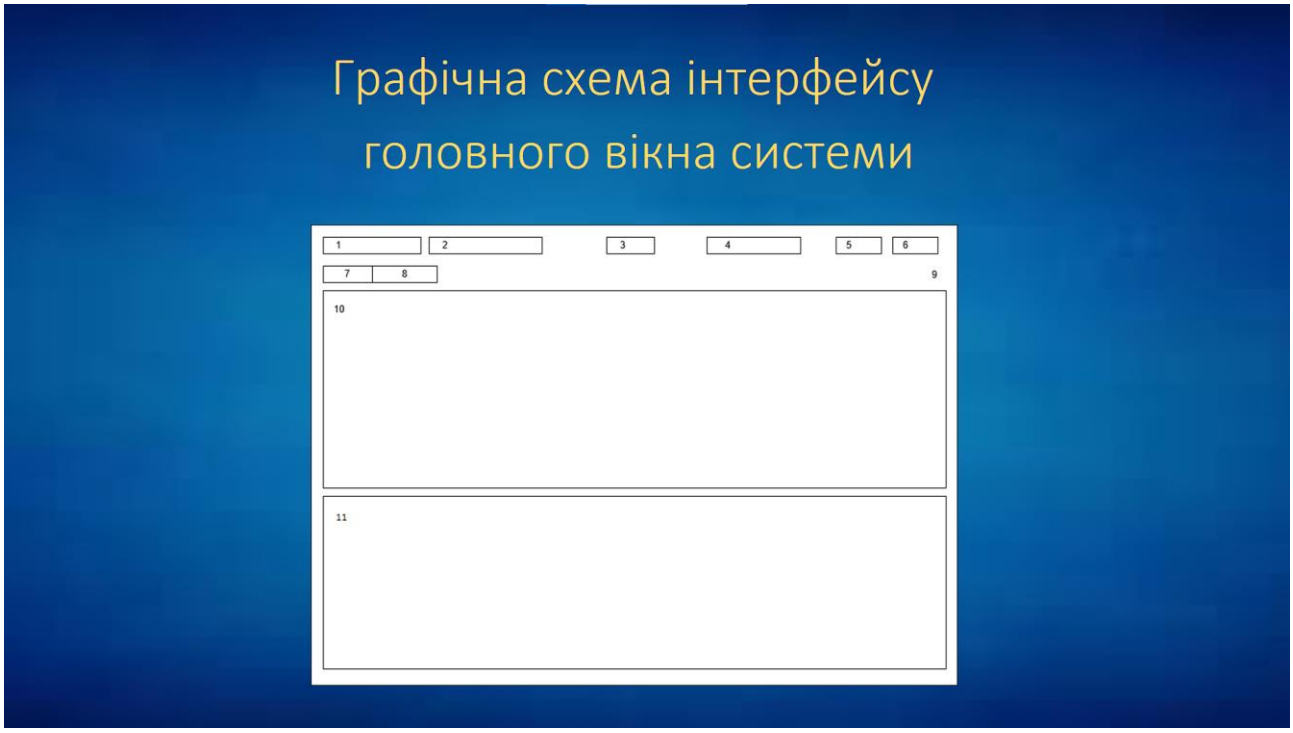

Рисунок Г.18 – Графічна схема інтерфейсу головного вікна системи

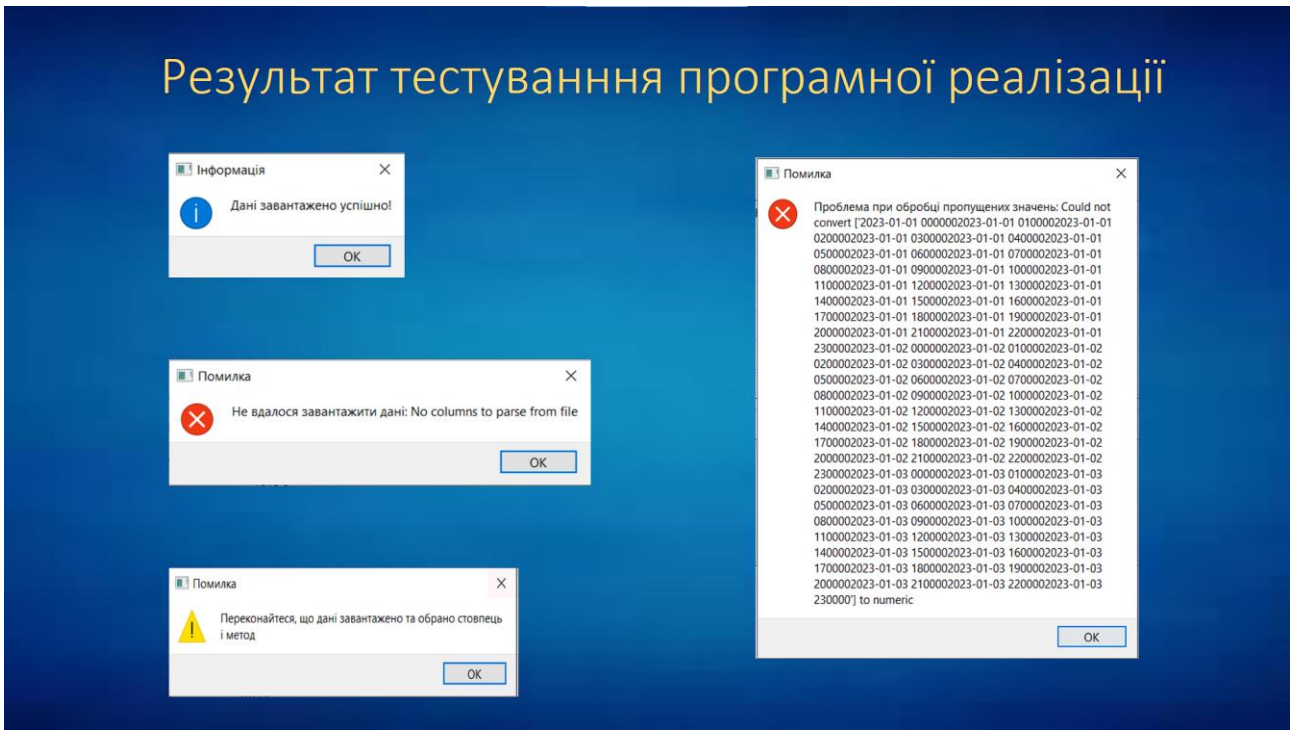

Рисунок Г.19 – Результат тестування програмної реалізації

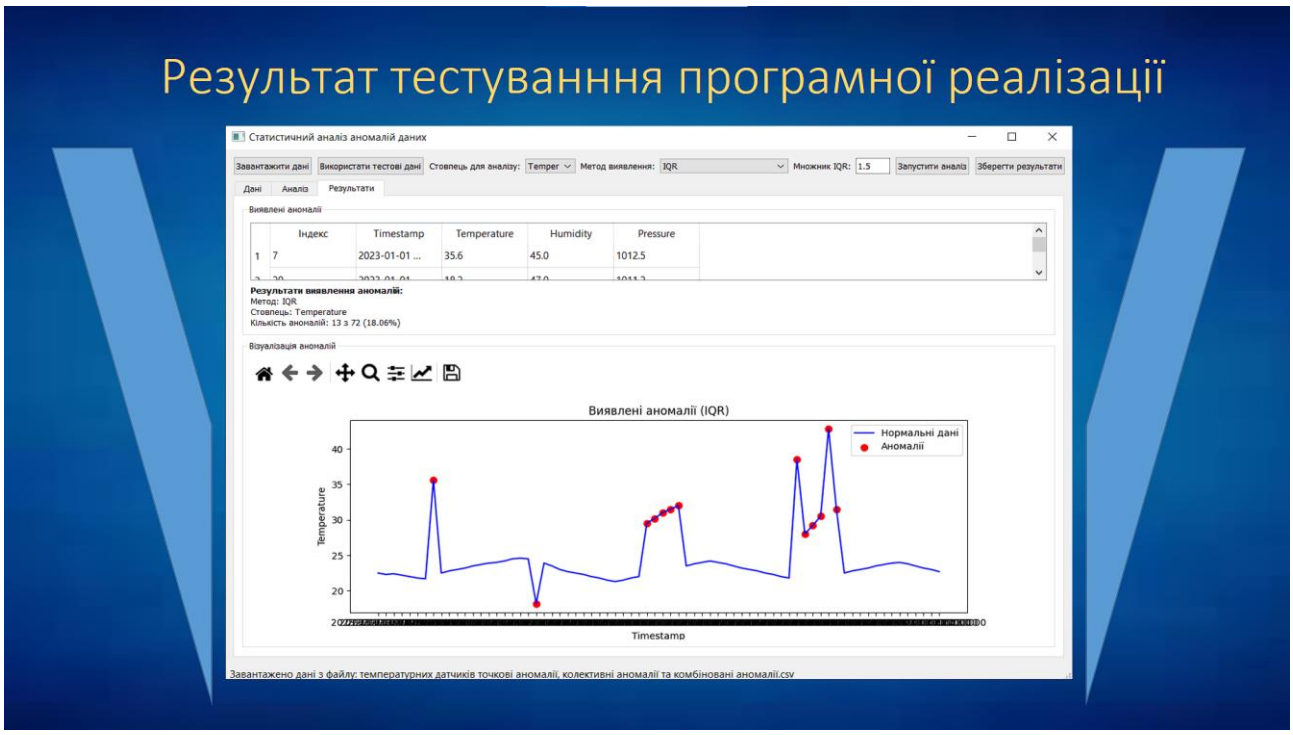

Рисунок Г.20 – Результат тестування програмної реалізації

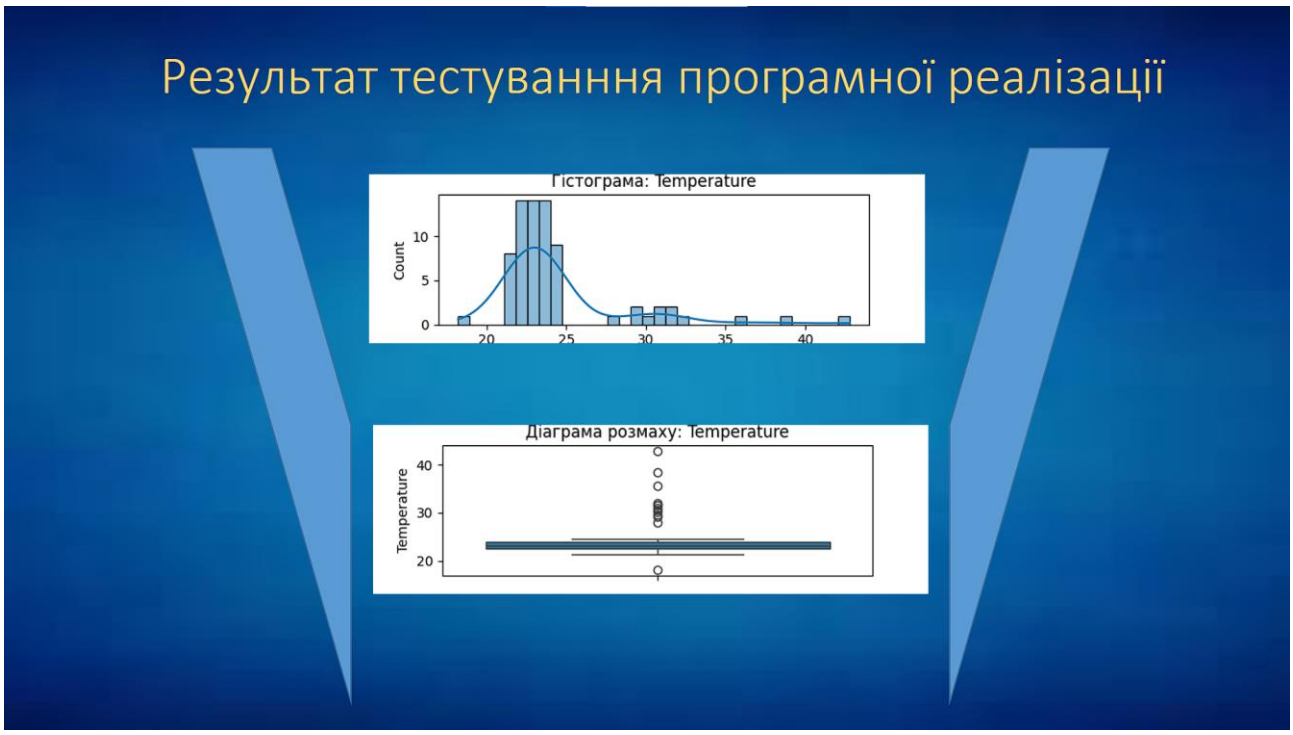

Рисунок Г.21 – Результат тестування програмної реалізації

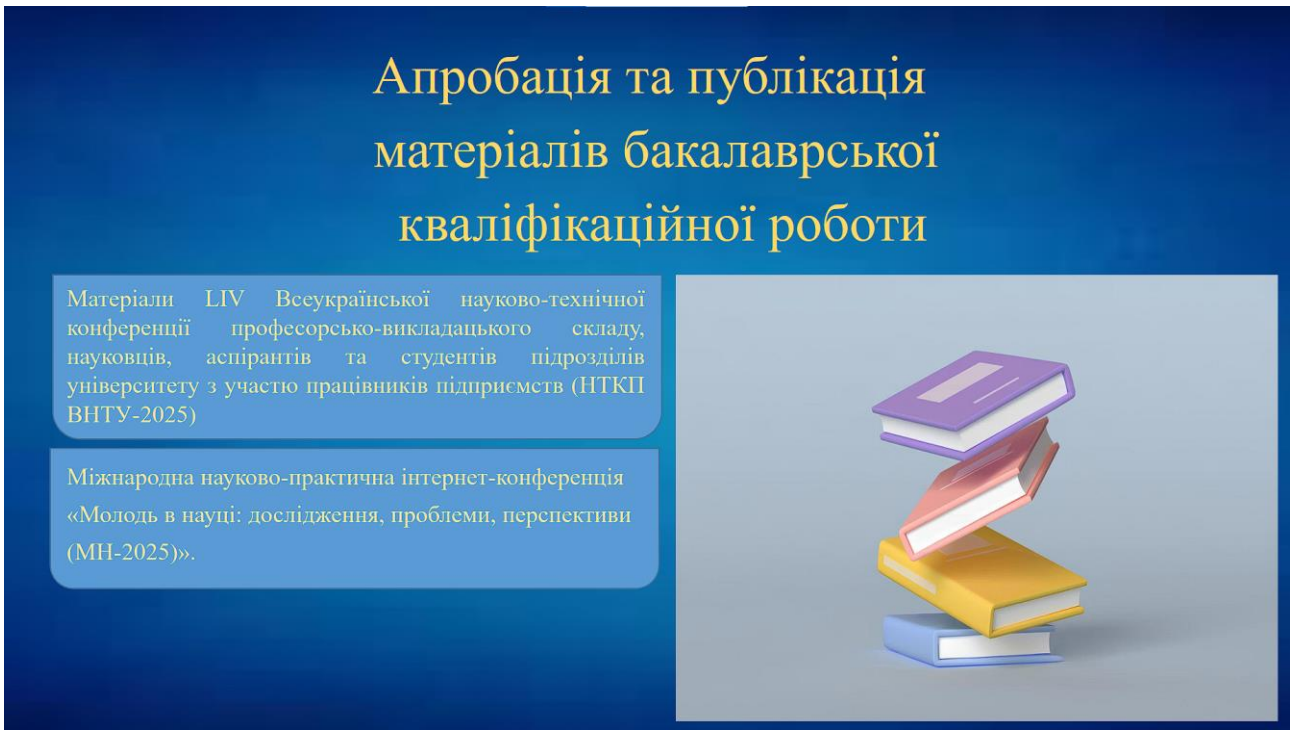

Рисунок Г.22 – Апробація та публікація матеріалів бакалаврської кваліфікаційної роботи

Рисунок Г.23 – Фінальний слайд