Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

ПРОГРАМНИЙ МОДУЛЬ ПОШУКУ АНОМАЛІЙ У ФІНАНСОВИХ ЗВІТАХ

Виконав: студент 4 курсу, групи 2КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Анастасія ЛАНОВИК
(прізвище та ініціали)

Керівник: д.т.н., проф. каф. КН
Ярослав ІВАНЧУК
(прізвище та ініціали)
«04» червня 2025 р.

Рецензент: д.т.н., проф., зав. каф. КСУ
Вячеслав КОВТУН
(прізвище та ініціали)
«05» червня 2025 р.

Допущено до захисту
Зав. кафедри проф., д.т.н. Андрій ЯРОВИЙ
«06» червня 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо - професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Андрій ЯРОВИЙ
« 05 » травня 2025 р.

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

Лановик Анастасії Сергіївни

1. Тема роботи: Програмний модуль пошуку аномалій у фінансових звітах.
керівник роботи: Іванчук Ярослав Володимирович, д.т.н, професор кафедри КН
затверджені наказом вищого навчального закладу "20" березня 2025 року № 97
2. Строк подання студентом роботи 30 травня 2025р.
3. Вихідні дані до роботи: набір фінансових звітів компаній не менше 5 шт., період аналізу від 1 місяця до 5 років, кількість торгових днів не менш 30 для статистичної значущості, технічні індикатори: ковзні середні MA5/MA20, денна прибутковість, зміна обсягу торгів; алгоритм машинного навчання: Isolation Forest; параметри оптимізації: contamination (0.01-0.5), n_estimators (50-300), max_samples (auto/256/512); форми виведення результатів: інтерактивні графіки, текстові звіти, таблиці аномалій.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): визначення об'єкту, предмета, мети та задач подальшого дослідження, аналіз предметної області, існуючих методів, алгоритмів та інструментів для створення програмного модуля для пошуку аномалій у фінансових звітах, проєктування програмного модуля пошуку аномалій у фінансових звітах, програмна реалізація пошуку аномалій у фінансових звітах.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): схема архітектури системи автоматичного аналізу фінансових звітів блок-схеми основних модулів системи, блок-схема основного алгоритму, візуалізація виявлених аномалій у фінансових звітах

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Іванчук Я. В., д.т.н., проф.. каф. КН		

7. Дата видачі завдання 05 травня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Термін виконання		Примітка
		початок	закінчення	
1	Визначення об'єкта, предмета, мети та задачі подальшого дослідження	05.05.25	10.05.25	
2	Аналізпредметної області, існуючих методів, алгоритмів та інструментів для створення програмного модуля для пошуку аномалій у фінансових звітах ;	11.05.25	18.05.25	
3	Проєктування програмних програмного модуля для пошуку аномалій у фінансових звітах;	19.05.25	26.05.25	
4	Програмна реалізація програмного модуля для пошуку аномалій у фінансових звітах	27.05.25	31.05.25	
5	Оформлення пояснювальної записки	01.06.25	04.06.25	

Студент _____ Анастасія ЛАНОВИК
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ярослав ІВАНЧУК
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська робота складається з 94 сторінок формату А4, містить 27 рисунків та список використаних джерел з 19 найменувань.

Бакалаврська робота присвячена розробці програмного модуля для автоматичного виявлення аномалій у фінансових звітах з використанням алгоритму машинного навчання Isolation Forest. Модуль забезпечує ефективне виявлення підозрілих відхилень у фінансових даних, які можуть свідчити про помилки, шахрайство або інші проблеми у звітності. Система включає автоматичну обробку великих обсягів даних, інтуїтивно зрозумілий графічний інтерфейс, візуалізацію результатів і генерацію детальних звітів з рекомендаціями. Експерименти та тести з реальними фінансовими даними підтвердили високу функціональність і точність розробленого модуля: оцінка Silhouette склала 0,68 бала, а рівень виявлення аномалій - 13,7%.

Впровадження цього модуля вдосконалить процеси фінансового аналізу, підвищить ефективність виявлення ризиків та забезпечить більшу прозорість фінансової звітності, відкриваючи широкі перспективи для застосування у фінансових установах, аудиторських фірмах та IT-організаціях.

Ключові слова: виявлення аномалій, фінансова звітність, машинне навчання, ізольований ліс, штучний інтелект, програмне забезпечення.

ANNOTATION

The bachelor's thesis consists of 94 A4 pages, contains 27 figures and a list of 19 references.

The bachelor's thesis is devoted to the development of a software module for the automatic detection of anomalies in financial reports using the Isolation Forest machine learning algorithm. The module provides effective detection of suspicious deviations in financial data that may indicate errors, fraud, or other problems in reporting. The system includes automatic processing of large amounts of data, an intuitive graphical interface, visualization of results, and generation of detailed reports with recommendations. Experiments and tests with real financial data confirmed the high functionality and accuracy of the developed module: the Silhouette score was 0.68 and the anomaly detection rate was 13.7%.

The implementation of this module will improve financial analysis processes, increase the efficiency of risk identification, and ensure greater transparency of financial statements, opening up broad prospects for use in financial institutions, audit firms, and IT organizations.

Keywords: anomaly detection, financial reporting, machine learning, isolated forest, artificial intelligence, software.

ЗМІСТ

ВСТУП

Актуальність дослідження. Нині спостерігаємо невпинне збільшення обсягів фінансової інформації, а також ускладнення фінансових транзакцій, через що традиційні методи аналізу фінансової звітності стають все менш дієвими. Ручний аналіз потребує чимало часу та ресурсів, до того ж схильний до людських помилок. Зважаючи на це, застосування технологій штучного інтелекту відкриває перспективи для автоматизації та вдосконалення процесу аналізу фінансових звітів.

У нинішньому фінансовому середовищі компанії постійно мають справу з величезними обсягами даних, що вимагають ретельного аналізу для виявлення потенційних ризиків, наприклад, шахрайства, помилок у звітах або інших відхилень. Традиційні методи перевірки та аналізу фінансових звітів демонструють гірші результати через обмеженість людських можливостей щодо обробки великих масивів інформації. Це сприяє зростанню зацікавленості автоматизованими системами на основі штучного інтелекту, здатними підвищити точність, ефективність та швидкість виявлення аномалій у фінансових даних. Особливу актуальність дослідженням використання ШІ надає зростання кількості шахрайських дій та недоліки традиційних методів контролю[1-3].

Мета дослідження полягає у підвищенні ефективності автоматичного аналізу фінансових звітів та виявленні аномалій, а також у розробці рекомендацій щодо впровадження таких систем у фінансові організації для збільшення їхньої ефективності та надійності.

Об'єктом дослідження виступають процеси автоматичного аналізу та виявлення аномалій у фінансовій звітності з використанням методів машинного навчання .

Предметом дослідження є алгоритми машинного навчання та програмні засоби для виявлення аномалій у фінансових даних з використанням методу

Isolation Forest.

Завдання дослідження включають:

1. Вивчення існуючих методів розробки ІТ-проектів.

2. Оцінка доцільності впровадження систем для управління розробкою ІТ-проектів в сучасному контексті.

3. Виявлення та аналіз недоліків існуючих програмних систем управління розробкою ІТ-проектів.

4. Створення архітектурної структури та визначення взаємодії між функціональними компонентами інформаційної системи.

5. Розробка алгоритмічної бази роботи модуля для управління розробкою ІТ-проєктів.

6. Дослідження та вибір оптимальних мов програмування для реалізації модуля управління розробкою ІТ-проєктів.

7. Практична реалізація програмного модуля для управління розробкою ІТ-проєктів.

8. Проведення тестування програмного модуля для оцінки його функціональності.

Апробація роботи. Результати роботи були представлені на:

Всеукраїнській науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації у 2025 році, з доповіддю «Аналіз перспектив розробки програмного забезпечення для управління розробкою ІТ-проєктів».

Публікації: електронні тези науково-технічної конференції «Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації» [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз фінансової звітності та її ролі в оцінці економічного стану підприємства

Фінансова звітність – це важливий засіб для оцінки економічної ситуації та результативності підприємства. Вона являє собою систематизований огляд фінансового становища та результатів господарської діяльності суб'єкта. Основними компонентами фінансової звітності є: баланс, де показані активи, зобов'язання та власний капітал підприємства на певну дату; звіт про прибутки та збитки, який відображає доходи, витрати та фінансові результати за визначений період; а також звіт про рух грошових коштів, що включає операційну, інвестиційну та фінансову діяльність.

Фінансова звітність має критичне значення для різних зацікавлених сторін. Для інвесторів вона є ключовим джерелом інформації для прийняття інвестиційних рішень, для кредиторів – основою оцінки платоспроможності підприємства, для керівництва – інструментом оцінки ефективності управління та планування майбутньої діяльності, а для аудиторів та регуляторів – засобом аналізу та виявлення можливих шахрайських дій.

У процесі виявлення шахрайства особливу увагу приділяють аномаліям. Аномалії у фінансових даних – це відхилення від очікуваних чи типових значень, які можуть свідчити про помилки, незвичайні події або навмисні маніпуляції з фінансовою інформацією. Ці аномалії можуть проявлятися у фінансових звітах в різних формах: значні відхилення окремих показників, незвичайні співвідношення між різними статтями звітності, або нетипові часові зв'язки звітів [2 – 4].

Близько 78% випадків корпоративного шахрайства супроводжувалися статистично значущими аномаліями у фінансовій звітності щонайменше за два квартали до офіційного виявлення порушень [5]. Крім того, аномалії у

співвідношеннях між різними статтями балансу та звіту про прибутки і збитки були присутні у 92% випадків навмисних маніпуляцій із фінансовою звітністю [6].

Характерною рисою аномалій у фінансових даних є їх багатогранність. Більшість фінансових аномалій виявляються одночасно у кількох взаємопов'язаних показниках, утворюючи своєрідні "паттерни аномалій" [7]. Наприклад, нетипове зростання доходів часто супроводжується аномальними змінами в дебіторській заборгованості, грошових потоках від операційної діяльності та коефіцієнтах оборотності активів.

Важливо пам'ятати, що виявлення аномалій вимагає врахування контексту, оскільки те, що є аномалією для одного підприємства чи галузі, може бути нормою для іншого [8]. Наприклад, сезонні коливання продажів є звичними для роздрібної торгівлі, але можуть вказувати на проблеми у промисловому виробництві.

Існує декілька типів аномалій, які можна виявити у фінансових даних:

- Точкові аномалії - це окремі значення, які суттєво відрізняються від інших у наборі даних. Приміром, різке збільшення витрат в окремому місяці без відповідного збільшення доходів. Візуальне представлення точкових аномалій показано на рисунку 1.1.

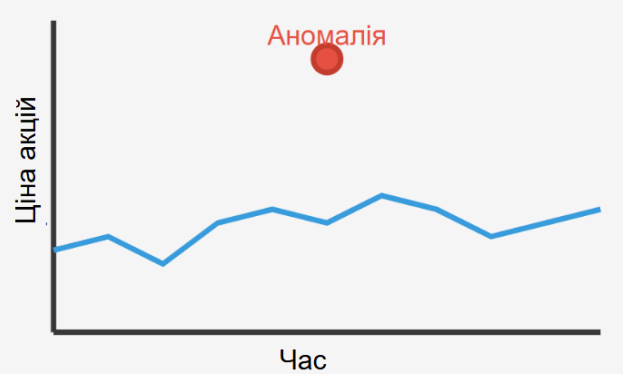


Рисунок 1.1 — Візуальне зображення точкової аномалії

- Контекстуальні аномалії - це показники, що відрізняються від звичних

для певного середовища, хоча в інших умовах можуть бути цілком типовими. Скажімо, значний ріст продажів у період різдвяних свят буде очікуваним, а такий самий обсяг продажу в середині літа - вже незвичайним явищем. Візуалізація контекстуальних аномалій представлена на ілюстрації 1.2.

Рисунок 1.2 — Візуальне зображення контекстуальних аномалій

- Колективні аномалії з'являються, коли сукупність даних відхиляється від звичних показників, навіть якщо окремі елементи не демонструють аномальності [9]. Візуалізація колективних (групових) аномалій представлена на рисунку 1.3.

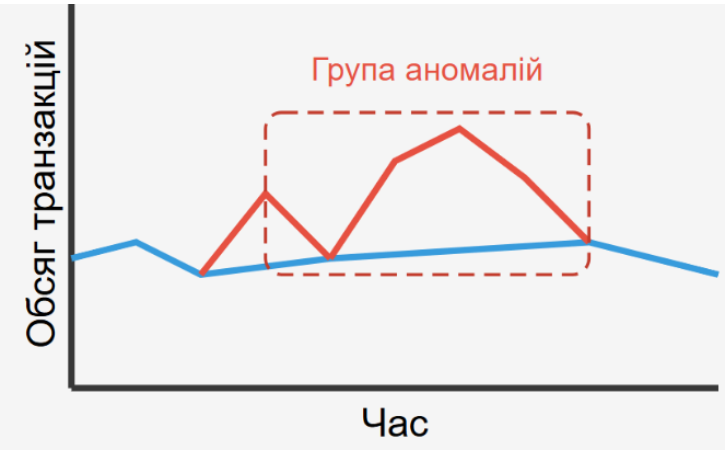


Рисунок 1.3 — Візуальне зображення колективних (групових) аномалій

Причини виникнення невідповідностей у звітах можуть бути різноманітними. Досить часто вони є результатом звичайних помилок при

введенні даних або технічних збоїв в інформаційних системах. Водночас, більш серйозною причиною можуть бути шахрайські дії, як-от маніпуляції з прибутками або витратами для покращення фінансових показників компанії. Важливо також враховувати, що не всі аномалії виникають внаслідок помилок або шахрайства – вони також можуть відображати незвичайні, але законні фінансові операції, такі як великі одноразові продажі чи придбання активів [4, 9].

Аномалії суттєво впливають на аналіз фінансових даних та процес прийняття рішень. Невиявлені невідповідності можуть призвести до спотворення фінансових показників, що, у свою чергу, може спричинити неправильну оцінку фінансового стану компанії. Це створює ризики не тільки для самої компанії, а й для аудиторів та регуляторів, які відповідають за забезпечення достовірності фінансової звітності.

1.3 Сучасні методи та інструменти аналізу фінансової звітності

Для аналізу фінансових звітів використовувалися різноманітні методи, кожен з яких має свої переваги та недоліки. Традиційні методи аналізу включають горизонтальний, вертикальний та коефіцієнтний аналіз.

Горизонтальний аналіз порівнює показники фінансової звітності за різні періоди часу. Це дає змогу виявити тенденції та динаміку змін фінансового стану підприємства. Наприклад, можна простежити, як змінювався обсяг продажів або чистий прибуток компанії протягом декількох років.

Вертикальний аналіз зосереджується на структурі фінансових показників. При цьому кожна частина фінансового звіту виражається у відсотках до загального підсумку. Це дає змогу оцінити, до прикладу, яку частку в загальних активах складають основні засоби або яку частку в загальних витратах становлять витрати на оплату праці чи оплату матеріалів.

Коефіцієнтний аналіз є одним з найпоширеніших методів фінансового

аналізу. Він включає розрахунок різноманітних фінансових коефіцієнтів, таких як коефіцієнти ліквідності, що показують здатність підприємства погашати короткострокові зобов'язання; коефіцієнти рентабельності, що відображають ефективність використання ресурсів; коефіцієнти фінансової стійкості, що характеризують залежність підприємства від зовнішніх джерел фінансування [10].

Існують й інші методи аналізу, а саме статистичні методи, як-от регресійний та кореляційний аналіз. Вони дозволяють виявити взаємозв'язки між різними фінансовими показниками та прогнозувати їх майбутні значення. Наприклад, можна проаналізувати, як зміна обсягу продажів впливає на прибуток компанії [11].

Розвиток інформаційних технологій дозволив удосконалити аналіз фінансових звітів. Так, з'явилися сучасні комп'ютеризовані системи аналізу фінансової звітності - ERP-системи (Enterprise Resource Planning), що дозволяють інтегрувати всі бізнес-процеси підприємства, в тому числі фінансовий облік та звітність, в єдину інформаційну систему. Це забезпечує оперативність та точність фінансового аналізу.

1.4 Використання алгоритмів машинного навчання для виявлення аномалій у фінансовій звітності

Також залучення штучного інтелекту відкриває нові можливості для аналізу фінансових даних та виявлення аномалій. Методи ШІ, зокрема машинне навчання, дозволяють автоматизувати процес аналізу великих обсягів даних та виявляти складні патерни, які можуть бути непомітними при використанні традиційних методів аналізу.

Серед конкретних алгоритмів, що часто використовуються для виявлення аномалій у фінансових даних, можна виділити:

1) Isolation Forest – це алгоритм, який ґрунтується на принципі, що

аномалії легше ізолювати від решти даних. Цей метод працює шляхом побудови дерев рішень, які рекурсивно розділяють дані за допомогою випадково вибраних ознак та порогових значень. Ключова ідея полягає в тому, що аномальні точки зазвичай потребують менше розділень для їх ізоляції. Isolation Forest особливо ефективний для виявлення точкових аномалій у великих наборах даних і добре справляється з багатовимірними даними. Він швидкий, масштабований і не вимагає попереднього навчання на нормальних даних. Однак, цей метод може бути чутливим до шуму в даних і менш ефективним для виявлення кластерних аномалій.

2) One-Class SVM (Support Vector Machine) – це метод навчання без вчителя, який спрямований на виявлення новизни в даних. Основна ідея полягає в тому, щоб знайти гіперплощину, яка відокремлює більшість нормальних даних від потенційних аномалій. One-Class SVM особливо добре працює з багатовимірними даними і може ефективно виявляти аномалії в складних розподілах даних. Цей метод гнучкий завдяки використанню різних ядерних функцій, що дає змогу адаптувати його до різних типів даних. Проте, One-Class SVM чутливий до вибору ядра та гіперпараметрів, що може вимагати ретельного налаштування для досягнення оптимальних результатів.

3) Автоенкодери – це тип нейронних мереж, які навчаються відтворювати вхідні дані через процес стиснення та розширення. У контексті виявлення аномалій, автоенкодери працюють на припущенні, що нормальні дані можна ефективно реконструювати, тоді як аномалії призводять до високої помилки реконструкції. Цей підхід особливо потужний для виявлення складних патернів аномалій і може працювати з різними типами даних, включаючи зображення та часові співвідношення. Автоенкодери здатні вивчати нелінійні перетворення даних, що робить їх ефективними для складних розподілів. Однак, вони зазвичай вимагають значної кількості даних для навчання і можуть бути обчислювально затратними.

4) LSTM-мережі (Long Short-Term Memory) – це різновид рекурентних

нейронних мереж, особливо ефективних для аналізу часових співвідношень та послідовних даних. У виявленні аномалій LSTM-мережі можуть враховувати довгострокові залежності в даних, що робить їх потужним інструментом для виявлення аномалій у складних послідовностях. Вони здатні вивчати нормальні патерни в даних і виявляти відхилення від цих патернів. LSTM-мережі особливо корисні в ситуаціях, де контекст і порядок подій мають значення. Проте, ці мережі можуть бути складними в навчанні, вимагають великої кількості даних і можуть бути обчислювально затратними, особливо для довгих послідовностей.

5) Random Forest для виявлення аномалій – це потужний ансамблевий метод машинного навчання, який використовує множину дерев рішень для виявлення аномалій. Цей алгоритм може застосовуватися різними способами в контексті виявлення аномалій. Якщо є мічені дані про аномалії, Random Forest можна використовувати як класифікатор, навчаючи його розрізняти нормальні та аномальні зразки. Однак навіть без мічених даних цей метод може бути ефективним. В такому випадку "аномальність" зразка можна визначати на основі глибини листа або довжини шляху в деревах лісу. Аномальні зразки зазвичай потрапляють у глибші листи або мають довші шляхи в деревах [12, 13].

Порівняльний аналіз методів пошуку аномалій у фінансових звітах подано на рисунку 1.4.

Метод	Точніть	Шидкість	Інтерпретованість	Простота	Загальна оцінка
Isolation Forest	5	5	5	4	5
One-Class SVM	4	3	4	3	4
Автоенкодери	5	2	2	2	4
LSTM-мережі	5	1	2	1	3
Random Foprest	4	4	5	3	4

Рисунок 1.4 - Порівняльний аналіз методів пошуку аномалій у фінансових звітах

Отже, використання ШІ у виявленні аномалій у фінансовій звітності має чимало плюсів. Перш за все, ШІ здатен опрацьовувати колосальні обсяги інформації, що критично важливо у сучасному діловому середовищі. По-друге, алгоритми машинного навчання можуть виявляти комплексні аномалії та взаємозв'язки, котрі можуть лишатися непомітними для людського погляду або традиційних статистичних методів. По-третє, застосування ШІ дає змогу автоматизувати процес аналізу, що відчутно збільшує його продуктивність та знижує вірогідність людських помилок. Крім цього, моделі машинного навчання здатні безперервно покращуватися з надходженням нових даних, адаптуючись до трансформацій у фінансовому секторі.

Водночас, застосування ШІ для аналізу фінансових звітів має певні виклики. Один з ключових – так звана проблема "чорної скриньки" – складність інтерпретації рішень, що приймаються алгоритмами машинного навчання. У фінансовій сфері це особливо актуально, адже тут критично потрібна прозорість та зрозумілість аналізу.

До того ж, використання ШІ піднімає низку етичних питань та регуляторних вимог. Оскільки методи ШІ застосовуються для ухвалення важливих фінансових рішень, вкрай важливо забезпечувати прозорість та зрозумілість того, як ці методи працюють і на основі чого приймаються рішення. Також дуже важливим є гарантування захисту особистих даних під час використання ШІ для аналізу фінансової інформації.

Але незважаючи на ці недоліки, потенціал використання ШІ для виявлення аномалій у фінансовій звітності надзвичайний. З подальшим розвитком технологій та вдосконаленням алгоритмів, ШІ може стати незамінним інструментом для забезпечення достовірності та прозорості фінансової звітності, сприяючи підвищенню довіри до фінансових ринків та економічної системи в цілому [13].

У цій роботі ми розглянемо більш детально алгоритм Isolation Forest для виявлення аномалій. Серед його переваг можна виділити:

- Ефективність обробки великих обсягів даних: Isolation Forest має лінійну часову складність O(n log n), де n – кількість зразків. Це робить його значно швидшим за багато інших методів, особливо при роботі з великими наборами даних, що типово для фінансової звітності.
- Здатність працювати з багатовимірними даними: На відміну від деяких статистичних методів, Isolation Forest добре справляється з багатовимірними даними, що дозволяє аналізувати одночасно багато фінансових показників.
- Нечутливість до масштабу: Метод не вимагає нормалізації даних, що спрощує попередню обробку фінансових показників, які можуть мати різні масштаби. Також це полегшує його програмну реалізацію.
- Виявлення як глобальних, так і локальних аномалій: Isolation Forest ефективний у виявленні як очевидних відхилень (глобальних аномалій), так і менш помітних (локальних аномалій).
- Інтерпретованість результатів: Хоча Isolation Forest є методом машинного навчання, його результати відносно легко інтерпретувати, що важливо для фінансового аналізу та аудиту.
- Невимогливість до попереднього знання про розподіл даних: На відміну від статистичних методів, Isolation Forest не вимагає припущень про розподіл даних, що робить його універсальним для різних типів фінансових показників.
- Також даний алгоритм є простим в імплементації, оскільки доступний в популярних бібліотеках машинного навчання (наприклад, scikit-learn в Python), що спрощує його інтеграцію в програмну реалізацію.
- Крім того алгоритм є адаптивним, бо може автоматично пристосовуватися до різних наборів фінансових даних без необхідності ручного налаштування параметрів для кожного конкретного випадку [14].

Таким чином загальний алгоритм для пошуку аномалій представлений на рисунку1.5.

Рисунок 1.5 - Загальний алгоритм для пошуку аномалій в фінансових звітах

Отже, аналіз та опрацювання фінансових звітів надзвичайно важливі, особливо у світі сучасних фінансів, де на основі аналізу фінансових даних приймаються рішення інвесторами, менеджерами, аудиторами та регуляторами.

Але обсяги фінансових даних великі, тому традиційні та статичні методи обробки даних втрачають свою ефективність. Саме тому на допомогу в аналізі приходить штучний інтелект та машинне навчання. Вони пропонують нові методи та алгоритми, за допомогою яких стає легше опрацьовувати великі обсяги даних, уникати дрібних помилок, зумовлених людською неуважністю, та виявляти глибші аномалії. З-поміж розглянутих вище методів та алгоритмів, найбільше зацікавив алгоритм Isolation Forest. Хоча його основним недоліком є чутливість до шуму, але він також може обробляти значні обсяги попередньо необроблених даних, досить простий у реалізації, адаптивний та демонструє хороші результати з низькою похибкою. У наступному розділі ми більш детально розглянемо цей алгоритм.

1.5 Висновок до розділу 1

Відповідно до проведеного аналізу предметної області фінансової звітності та методів виявлення аномалій, можна дійти висновку, що обробка та аналіз великих обсягів фінансових даних традиційними методами є недостатньо ефективними в сучасних умовах. Це створює суттєві проблеми для компаній, які потребують точних та оперативних інструментів виявлення потенційних проблем у фінансових звітах.

Традиційні методи аналізу фінансової звітності вимагають значних затрат часу та людських ресурсів і часто не дозволяють виявити складні патерни аномалій, що може призвести до серйозних фінансових та репутаційних втрат.

Для вирішення цих проблем доцільно розглянути використання методів штучного інтелекту та машинного навчання для автоматизації процесу виявлення аномалій у фінансових даних. Зокрема, алгоритм Isolation Forest демонструє значні переваги для цього завдання завдяки своїй здатності ефективно обробляти великі обсяги багатовимірних даних, нечутливості до масштабу та високій інтерпретованості результатів. Впровадження інтелектуальної системи для автоматичного виявлення аномалій у фінансовій звітності може значно полегшити та покращити процес аналізу фінансових даних. Така система може автоматично обробляти великі обсяги даних, виявляти підозрілі патерни та надсилати сповіщення відповідальним особам. Використання методів машинного навчання для виявлення аномалій також забезпечує додаткові переваги, такі як здатність системи адаптуватися до нових типів аномалій та вдосконалюватися з часом.

Отже, розробка інтелектуальної системи для автоматичного виявлення аномалій у фінансовій звітності на основі алгоритму Isolation Forest є актуальним та перспективним напрямом, що може значно підвищити ефективність та надійність фінансового аналізу, а також зменшити фінансові втрати від шахрайства та помилок у звітності.

2 РОЗРОБКА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ДЛЯ ПОШУКУ АНОМАЛІЙ У ФІНАНСОВИХ ЗВІТАХ

2.1 Розробка архітектури програмного модулю

Здійснимо порівняльний розбір архітектур та визначимо найкращу. Розглянемо чотири основні типи: монолітна, мікросервісна, модульна та багаторівнева.

Монолітна архітектура – це цілісний програмний комплекс, де всі компоненти тісно взаємодіють в єдиній кодовій базі.

Переваги:

- легкість розробки та розгортання;
- швидка взаємодія між компонентами;
- мінімальна затримка при обробці даних.

Недоліки:

- ускладнене масштабування окремих компонентів;
- проблеми з підтримкою та розвитком з розширенням функціоналу;
- обмежені можливості паралельної розробки.

Мікросервісна архітектура передбачає поділ системи на низку незалежних сервісів, кожен з яких відповідає за певну бізнес-функцію.

Переваги:

- висока гнучкість та масштабованість;
- незалежне оновлення окремих компонентів;
- підвищена відмовостійкість системи загалом.

Недоліки:

- труднощі координації та оркестрації сервісів;
- збільшена затримка при міжсервісній комунікації;
- складніші розробка та тестування.

Модульна архітектура поєднує сильні сторони монолітної та

мікросервісної, поділяючи систему на модулі з чіткими інтерфейсами, але в рамках одного застосунку.

Переваги:

- баланс між продуктивністю та гнучкістю;
- хороша масштабованість у поєднанні з простотою розробки;
- можливість незалежної розробки окремих модулів.

Недоліки:

- необхідність ретельного проєктування інтерфейсів;
- ризик поступового розмиття меж між модулями.

Багаторівнева архітектура розподіляє функціональність системи по горизонтальних шарах (рівень даних, бізнес-логіки, представлення).

Переваги:

- чіткий розподіл відповідальностей;
- можливість незалежного масштабування шарів;
- гнучкість у виборі технологій для кожного шару.

Недоліки:

- сувора залежність між шарами;
- труднощі при зміні контракту між шарами.

Порівняльний аналіз архітектур представлено в таблиці 2.1.

Таблиця 2.1 - Порівняльний аналіз архітектурних підходів

Критерій	Монолітна	Мікросервісна	Модульна	Багаторівнева
Швидкодія обробки великих обсягів даних	Висока	Середня	Висока	Середня
Масштабованість	Низька	Висока	Середня	Середня

Продовження табл. 2.1

Простота розробки	Висока		Середня	Середня
Можливість паралельної розробки	Низька	Висока	Середня	Середня
Легкість тестування	Середня	Складна	Висока	Середня
Гнучкість адаптації до нових вимог	Низька	Висока	Висока	Середня
Ефективність обробки потокових даних	Висока	Середня	Висока	Середня

На підставі здійсненого дослідження для системи автоматичного виявлення аномалій у фінансовій звітності було обрано модульну архітектуру з елементами багаторівневого підходу. Такий вибір обумовлений низкою чинників:

1) Специфіка вимог до обробки фінансових даних — система має оперативно опрацьовувати великі обсяги інформації, що вимагає мінімізації накладних витрат на міжкомпонентну взаємодію, яку забезпечує модульна архітектура.

2) Вимоги щодо продуктивності — алгоритми машинного навчання для виявлення відхилень потребують швидкого доступу до даних та ефективного керування пам'яттю, що краще реалізується в рамках модульної архітектури, аніж у розподіленій мікросервісній системі.

3) Потреба в гнучкому масштабуванні — модульна архітектура дає можливість незалежно розвивати та вдосконалювати окремі складові системи

без порушення цілісності інтеграції, що відповідає потребам довготривалого розвитку.

4) Вимоги до можливості тестування — модульна архітектура спрощує модульне тестування і сприяє поліпшенню якості коду завдяки чіткому розподілу обов'язків.

5) Вимоги до розширюваності — система має легко адаптуватися до різноманітних типів фінансової інформації та алгоритмів аналізу, що добре реалізується в модульній архітектурі з чітко визначеними інтерфейсами.

Отже, систему буде реалізовано з застосуванням модульного підходу, що гарантує чітке розподілення функціональності та спрощує подальшу підтримку коду. Також буде впроваджено багаторівневий підхід до опрацювання даних, зокрема: автоматичне отримання даних, комплексна попередня обробка та застосування алгоритму пошуку аномалій. Система буде розроблена з врахуванням потенціалу масштабування та адаптації до зростаючих потреб користувачів. Модульна структура програмного забезпечення забезпечує легке розширення функціональності шляхом додавання нових компонентів без потреби в значних змінах наявної кодової бази. Архітектура системи передбачає гнучке налаштування параметрів аналізу, що дає змогу адаптувати її до роботи з різними фінансовими інструментами та ринками. Завдяки використанню сучасних підходів до проектування, систему можливо легко доповнити новими методами аналізу й алгоритмами виявлення аномалій, що гарантує її довгострокову актуальність і розвиток.

У процесі розробки системи особливу увагу буде приділено питанням оптимізації продуктивності. Буде реалізовано ефективні механізми опрацювання великих масивів фінансових даних, що дозволяє системі працювати з історичними даними значного обсягу без втрати швидкодії. Важливим аспектом оптимізації є раціональне керування пам'яттю при роботі з графічними компонентами, що особливо важливо при візуалізації великих обсягів даних. Для забезпечення зручності роботи користувача запроваджено

асинхронне оновлення інтерфейсу, що дозволяє уникнути затримок при взаємодії із системою навіть під час виконання складних обчислень та аналізу даних.

Крім того, архітектура програмного забезпечення дозволить здійснювати ефективне модульне тестування окремих компонентів, що забезпечить раннє виявлення потенційних проблем та підтримку високої якості коду. Важливим аспектом є можливість тестування системи на різних наборах фінансових даних, що дозволяє перевірити її ефективність і надійність в різних ринкових умовах і сценаріях використання.

В основі лежатиме модуль завантаження та попередньої обробки фінансової інформації, що відповідатиме за імпорт, очищення та стандартизацію даних з різних джерел. Центральним елементом системи буде модуль аналізу й виявлення аномалій, який реалізує алгоритм Isolation Forest для ідентифікації потенційних відхилень у фінансових показниках. Результати аналізу передаватимуться до модуля візуалізації, який формує наочні графіки та діаграми, та модуля генерування звітів, що створює структуровані документи з детальним описом виявлених аномалій. Після цього всі результати будуть подані через інтерактивний інтерфейс користувача. Структурна схема програмного модуля для пошуку аномалій у фінансових звітах наведена на рисунку2.1.

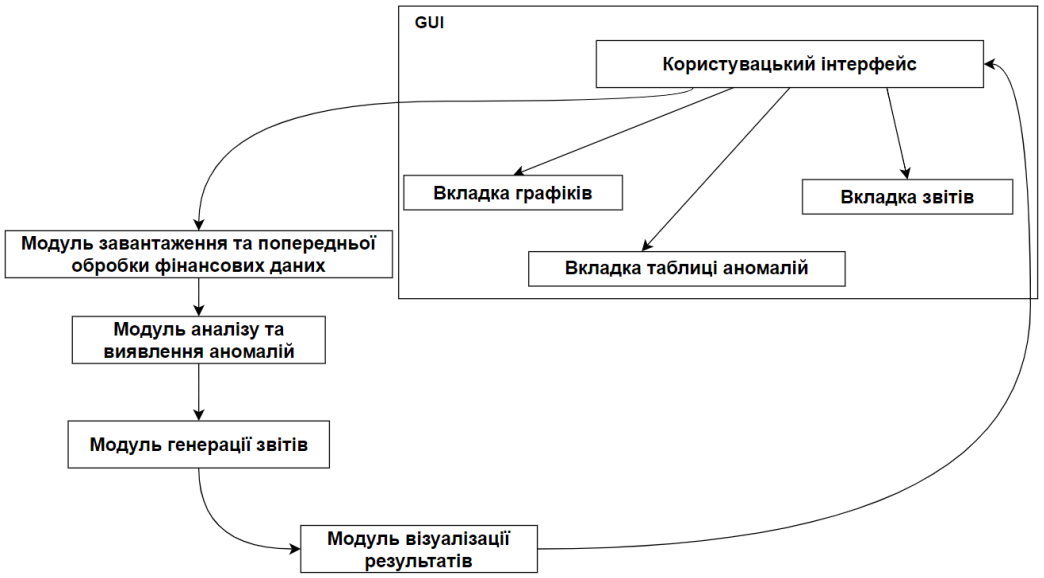


Рисунок 2.1 — Структурна схема програмного забезпечення

Тепер зосередимось на кожному модулі окремо, вивчаючи його функціонал та взаємодію з іншими.

Модуль завантаження та попередньої обробки фінансових відомостей мусить забезпечувати:

- імпортування фінансових даних через сервіс Yahoo Finance;
- очищення даних (видалення повторів, обробка відсутніх значень);
- обчислення додаткових фінансових метрик;
- нормалізація та стандартизація даних.

Дані з цього модуля будуть передаватися до модуля аналізу та виявлення аномалій для подальшої обробки та пошуку відхилень. Блок-схема модуля завантаження та попередньої обробки відображена на рисунку 2.2.

Рисунок 2.2 - Блок-схема модуля завантаження та попередньої обробки

Модуль аналізу та виявлення аномалій – це ключова складова ШІ, яка реалізує наступні функції:

- використання алгоритмів машинного навчання Isolation Forest для пошуку аномалій

- обчислення статистичних показників

- позначення аномальних точок даних

- визначення початкової оцінки виявлення аномалій.

Даний модуль отримуватиме дані від модуля завантаження та попередньої обробки, а результати аналізу передаватимуться модулям візуалізації та формування звітів. Блок-схема модуля аналізу та виявлення аномалій наведена на рисунку 2.3.

Рисунок 2.3 - Блок-схема модуля аналізу та виявлення аномалій

Модуль візуалізації результатів буде виконувати такі функції:

- створення різних типів графіків для відображення виявлених аномалій ;
- візуальне представлення виявлених аномалій.

Даний модуль отримує дані про аномалії від модуля аналізу та виявлення аномалій, передає створені візуалізації до модуля генерації звітів та передає результат до інтерактивного інтерфейсу. Блок-схема модуля візуалізації подана на рисунку 2.4

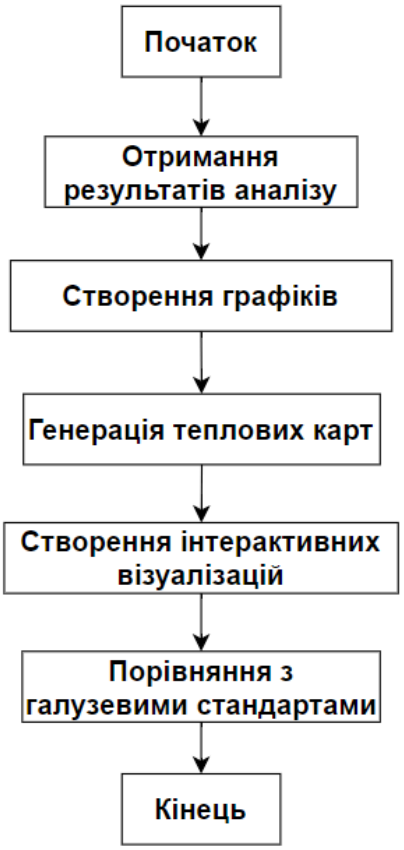


Рисунок 2.4 - Блок-схема модуля візуалізації результатів виявлення аномалій у фінансових даних

Модуль генерації звітів виконує такі функції:

- створення структурованих звітів на основі результатів аналізу;
- розрахунок ймовірності шахрайських дій;
- генерація рекомендацій на основі виявлених аномалій.

Даний модуль отримує дані про аномалії від модуля аналізу та виявлення аномалій, також отримує візуалізації від модуля візуалізації результатів та передає їх до інтерактивного інтерфейсу. Блок-схема модуля генерації звітів подана на рисунку 2.5.

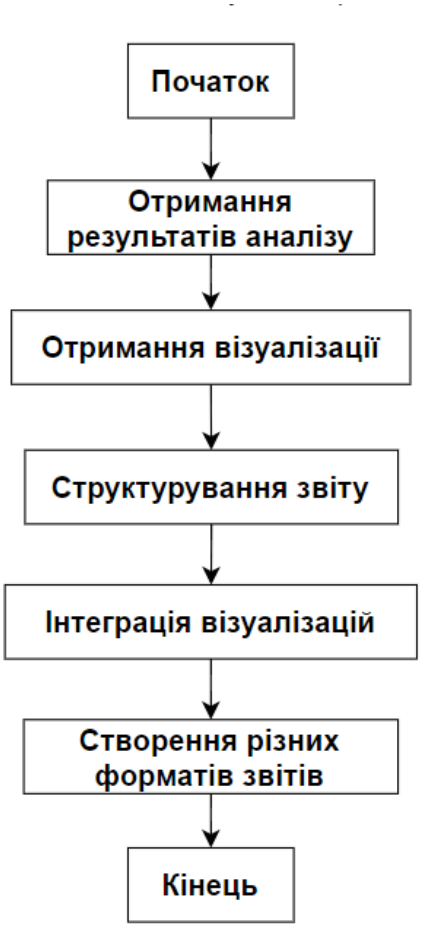


Рисунок 2.5 — Блок-схема функціонування модуля генерації звітів на основі виявлених аномалій в фінансових даних

Також буде реалізовано графічний інтерфейс, що забезпечує:

- зручний доступ до всіх функцій системи;
- інтерактивне керування параметрами аналізу;
- відображення результатів у різних форматах;
- можливість збереження та експорту результатів.

Графічний інтерфейс буде складатися з трьох частин, а саме:

- "Графіки" для візуального аналізу;
- "Звіт" для текстової інформації;
- "Таблиця аномалій" для структурованого представлення даних.

2.2 Розробка взаємодії програмних модулів для пошуку аномалій в фінансових звітах

Розроблена система спирається на принцип модульності, де кожен компонент виконує чітко визначені функції та взаємодіє з іншими за допомогою стандартизованих інтерфейсів. Така архітектура гарантує гнучкість, масштабованість і здатність незалежно розвивати окремі компоненти.

Центральним елементом взаємодії в архітектурі є система обміну повідомленнями, яка забезпечує асинхронне спілкування між модулями. Це дозволяє оптимізувати продуктивність системи при обробці великих обсягів даних та гарантувати стійкість до помилок.

Модуль завантаження даних взаємодіє з модулем попередньої обробки через механізм потокової передачі, що дозволяє розпочинати обробку відразу після отримання перших частин інформації. Це особливо важливо при роботі з великими наборами фінансових даних.

Як вхідні дані будемо використовувати комбінацію з двох бібліотек: Yahoo Finance та Financial Statement Fraud Detection Dataset. Yahoo Finance є потужним інструментом для отримання історичних та поточних фінансових даних, що надає зручний програмний інтерфейс для взаємодії з API Yahoo Finance. Бібліотека yfinance дає змогу отримувати широкий спектр фінансових даних, включаючи історичні ціни акцій, обсяги торгів, історію дивідендів та іншу фінансову інформацію. В контексті розроблюваної системи, особливо корисними є можливості бібліотеки щодо отримання історичних даних з різною деталізацією за часом — від щоденних до щохвилинних показників. При цьому для кожного торгового періоду доступні такі ключові показники як ціна відкриття (Open), ціна закриття (Close), максимальна ціни (High), мінімальна ціна (Low) та обсяг торгів (Volume) [15].

Для забезпечення точності та надійності аналізу система має виконувати попередню обробку даних, включно з видаленням пропущених значень та

нормалізацією даних. Також необхідно провести нормалізацію, для якої можна використовувати StandardScaler з бібліотеки scikit-learn, що забезпечить приведення всіх показників до єдиної шкали з нульовим середнім та одиничною дисперсією. Це особливо важливо для коректної роботи алгоритму виявлення аномалій, оскільки різні показники можуть мати суттєво різні масштаби та одиниці вимірювання.

Financial Statement Fraud Detection Dataset містить фінансові показники компаній з позначками щодо наявності шахрайства. Він базується на реальних фінансових звітах компаній, але з додатковою обробкою та анотаціями. Цей набір даних містить звіти за різні періоди та мітки, що вказують на наявність чи відсутність шахрайських дій. Financial Statement Fraud Detection Dataset можна використовувати для валідації та точного налаштування моделі, оскільки він містить розмічені приклади шахрайства. Таким чином, ми можемо спочатку на наборі даних з Yahoo Finance виявляти аномалії, а потім аналізувати їх на наявність шахрайських дій, базуючись на прикладах з Financial Statement Fraud Detection Dataset [16].

У розробленій системі виявлення аномалій буде застосовано комплексний підхід до аналізу фінансових даних, який ґрунтується на врахуванні кількох груп показників та коефіцієнтів.

Першу групу складають базові цінові показники, що безпосередньо відображають динаміку торгів. До них належить ціна закриття (Close), що є одним з найважливіших індикаторів, оскільки представляє фінальну оцінку вартості активу учасниками ринку на кінець торгової сесії. Також враховуються максимальна (High) та мінімальна (Low) ціни протягом торгового дня, які дозволяють оцінити волатильність та амплітуду цінових коливань. На основі цих даних розраховується показник High_Low_Range, що є різницею між максимальною та мінімальною ціною та слугує важливим індикатором денної волатильності активу.

Друга група показників пов'язана з обсягом торгів. Основним показником

тут виступає Volume — кількість акцій, що були продані протягом дня. На його основі розраховується показник Volume_Change, що відображає відносну зміну обсягу торгів у порівнянні з попереднім торговим днем. Цей показник особливо важливий для виявлення аномалій, оскільки незвичайні зміни в обсягах торгів часто супроводжують значущі ринкові події або можуть вказувати на потенційні маніпуляції.

Третя група складається з показників динаміки цін. Ключовим показником тут є Returns — відносна зміна ціни закриття, що розраховується як відсоткова різниця між поточною та попередньою ціною закриття. Цей показник дозволяє оцінити денну прибутковість активу та виявити дні з аномально високими або низькими змінами ціни.

Взаємодія між модулем попередньої обробки та модулем аналізу аномалій здійснюється через механізм контейнерів даних. Оброблені дані пакуються у спеціалізовані контейнери, що містять не тільки самі дані, але й метаінформацію.

При оцінці потенційного шахрайства система враховує комбінації екстремальних значень різних показників. Особливу увагу приділяється ситуаціям, коли спостерігаються одночасно:

- аномально високі або низькі значення Returns (вище 95-го або нижче 5-го рівня);

- незвичайні об'єми торгів (вище 95-го або нижче 5-го рівня);

Система присвоює різні вагові коефіцієнти різним типам аномалій:

- екстремальні значення прибутковості оцінюються з ваговим коефіцієнтом 0,3;

- аномально високі об'єми торгів враховуються з коефіцієнтом 0,2;

- аномально низькі об'єми торгів враховуються з коефіцієнтом 0,1.

Сумарна оцінка ймовірності шахрайства розраховується як зважена сума виявлених аномалій, при цьому максимальне значення обмежене 100%. Такий підхід дозволяє не тільки виявляти аномалії, але й оцінювати їх потенційну

небезпеку з точки зору можливих маніпуляцій на ринку.

Окрім цих показників та коефіцієнтів, також важливо визначити і функцію оцінювання, яка буде орієнтиром у виявленні аномалій.

У розробленій системі функція оцінювання аномалій представлятиме собою комплексний інструмент аналізу, який дозволяє не тільки виявляти аномальні події на фінансовому ринку, але й оцінювати їх потенційну небезпеку та ймовірність шахрайських дій. Функція працюватиме з набором попередньо оброблених фінансових даних та списком виявлених аномалій, забезпечуючи їх глибокий аналіз та формування обґрунтованих висновків.

Процес оцінювання має починатися з розрахунку базових статистичних показників, які дають загальне уявлення про масштаб виявлених аномалій.

Система визначає загальну кількість проаналізованих торгових днів та підраховує кількість виявлених аномалій, на основі чого розраховуватиметься відсоткове співвідношення аномальних подій до загального обсягу даних.

Для кожної виявленої аномалії проводиться багатофакторний аналіз, що враховує різні аспекти торгової активності. Особлива увага приділяється двом ключовим показникам: прибутковості та об’єму торгів. Система здійснює детальний аналіз для визначення екстремальних значень цих показників, порівнюючи кожне аномальне спостереження із загальним розподілом даних.

Особливістю функції оцінювання є її здатність враховувати комбінації різних факторів. При виявленні декількох типів аномалій для одного торгового дня їх вагові коефіцієнти сумуються, що дозволяє виявляти особливо підозрілі випадки, коли одночасно спостерігаються відхилення за кількома показниками.

При цьому фінальна оцінка ймовірності обмежується максимальним значенням 100% для забезпечення інтерпретованості результатів. І на основі кінцевої оцінки подається загальний висновок щодо ситуації та рекомендації стосовно того, чи варто її моніторити чи проводити детальний аналіз.

Тепер визначимо основні параметри, які будуть використовуватися в алгоритмі Isolation Forest для виявлення аномалій в звітах. Це будуть такі

параметри:

- n_estimators (кількість дерев). Мінімальне значення цього показника буде 100 дерев, для збільшення точності можна буде збільшувати дерева, але максимальною кількістю дерев буде 1000. Такі межі вказані для можливості досягнення точного результату за відносне невеликий час, оскільки збільшення кількості дерев впливає на час обчислень;

- contamination (очікувана частка аномалій). Даний параметр визначає поріг для класифікації аномалій. Мінімальне значення 0.1 (10% аномалій).

Даний показник буде коригуватися в ході тестування на основі специфіки даних:

- max_samples (кількість зразків для навчання кожного дерева). Даний показник впливає на швидкість роботи та узагальнення моделі;

- max_features (кількість ознак для розгляду при розбитті). Може покращити роботу на даних з великою кількістю ознак. Ми почнемо зі значення 1.0 та будемо зменшувати, якщо точність не буде нас задовільняти або ж захочемо враховувати меншу кількість ознак;

- bootstrap (використання бутстрепу при створенні вибірок). Цей показник використовує два значення: True може покращити узагальнення, False може пришвидшити обчислення.

Також визначимо додаткові параметри для нашої системи:

- поріг аномальності – визначає поріг для класифікації аномалій;

- вікно часу для аналізу – визначає період, за який аналізуються дані для виявлення аномалій;

- ваги фінансових показників – визначає важливість різних фінансових показників;

- частота оновлення моделі – визначає, як часто модель перенавчається на нових даних.

Модуль аналізу аномалій передає результати своєї роботи модулям візуалізації та генерації звітів через систему подій. Кожна виявлена аномалія

або група аномалій генерує подію, на яку підписані відповідні модулі.

Система подій реалізує патерн «Спостерігач» (Observer) та забезпечує слабкий зв’язок між генератором аномалій та споживачами цієї інформації. Це дозволяє гнучко масштабувати систему, додаючи нові компоненти візуалізації або аналізу без модифікації ядра системи.

Взаємодія користувача з системою організована через графічний інтерфейс, що інтегрує результати роботи всіх модулів. Інтерфейс побудований за принципом реактивного програмування, де оновлення відбуваються в режимі реального часу при надходженні нових даних або результатів аналізу.

2.3 Розробка алгоритму пошуку аномалій у фінансових звітах

Головним складником системи є модуль виявлення та аналізу аномалій. Ядром цього модуля є алгоритм Isolation Forest. Логіка визначення аномалій в Isolation Forest базується на ключовому принципі: аномальні спостереження легше "ізолювати" від інших даних. Це означає, що для аномальних точок знадобиться менше розділень (коротший шлях у дереві), аби відокремити їх від решти. Відтак, спостереження з коротшими середніми довжинами шляху вважаються більш аномальними.

Спочатку збудуємо математичну модель для цього алгоритму. Нехай маємо набір даних $X = \{x1, ..., xn\}$, де x_i - окреме спостереження, n – загальна кількість спостережень. Також кожне спостереження подається вектором ознак у d-вимірному просторі.

Для кожного дерева випадково вибирається атрибут q, випадково обирається значення розділення p між $max(x_q)$ та $min(x_q)$, що буде пороговим значенням. Тоді для точки x є два варіанти:

- рух вліво, якщо значення атрибуту xq менше за порогове $x_q < p$;

- рух вправо, якщо значення атрибуту xq більше або дорівнює пороговому значенню $x_q \geq p$.

Такий принцип є одним із основних при побудові дерев рішень.

Оскільки важливим параметром в пошуку аномалій є довжина шляху, бо зазвичай аномалії мають менший шлях, оскільки ізолюються швидше. При обчисленні довжини можна розглядати два варіанти:

- якщо термінальний вузол є листком - $PathLength(x) = h(x)$, де $h(x)$ - це просто кількість ребер (переходів) від кореня до листка, а листок - це вузол, що містить тільки одне спостереження;

- якщо термінальний вузол є зовнішнім - $PathLength(x) = h(x) + c(Size)$, де до базової довжини $h(x)$ додається коригувальний фактор $c(Size)$ - кількість спостережень у цьому вузлі, а зовнішній вузол - це вузол, де зупинилось побудова дерева, але він може містити кілька спостережень.

Окремо варто підкреслити важливість коригувального фактору, бо він допомагає нормалізувати вплив розміру піддерева на загальну оцінку та уникнути зміщення оцінок вузлів різного розміру.

Формула оцінки аномальності:

$AnomalyScore(x) = 2^{\left(\frac{-\sum_{i=1} PathLength_i(x)}{t*c(n)}\right)}$, де t – загальна кількість дерев.

Також у формулі застосовуємо суму довжин усіх шляхів і експоненційне перетворення для нормалізації оцінки.

Для подальшої класифікації застосовуємо оцінку аномальності. Значення оцінки завжди перебуватимуть в межах від 0 до 1 включно, а пороговим значенням може бути будь-яке значення в межах цього відрізка. Тоді можна розглянути три ключові випадки:

- значення вище порогового свідчить про аномальність спостереження;

- значення нижче порогового свідчить про нормальність спостереження;

- значення, яке дорівнює пороговому, вказує на граничний випадок, точка, що дала таке значення, є точкою невизначеності й потребує більш детального аналізу [17].

Алгоритм Isolation Forest починається з отримання вхідних даних і підготовки необхідних параметрів. На початковому етапі відбувається ініціалізація порожнього масиву дерев, який буде поступово заповнюватися в процесі роботи алгоритму.

Основний процес роботи алгоритму базується на ітеративному створенні дерев ізоляції. Для кожного нового дерева, поки не досягнуто заданої кількості дерев (n_estimators), виконується послідовність дій. Спочатку відбувається вибір підвибірки даних, після цього починається процес побудови індивідуального дерева ізоляції (iTree).

Побудова кожного дерева відбувається рекурсивно. На кожному кроці перевіряється умова досягнення обмеження. Якщо обмеження не досягнуто, алгоритм випадковим чином вибирає атрибут для розділення даних та визначає точку розділення. Далі дані розділяються на дві підмножини згідно з вибраною точкою, і процес повторюється рекурсивно для кожної з підмножин. Коли досягнуто обмеження, поточне дерево зберігається в масив, і алгоритм переходить до створення наступного дерева.

Після створення всіх дерев розпочинається етап обробки та аналізу даних. Для кожного спостереження обчислюється шлях через усі дерева ансамблю. На основі цих шляхів розраховується середня довжина шляху, яка потім нормалізується для отримання показника аномальності. Ідея полягає в тому, що аномальні спостереження будуть ізольовані швидше (тобто матимуть коротші шляхи) порівняно з нормальними спостереженнями [18].

Фінальним етапом є класифікація спостережень на нормальні та аномальні на основі розрахованих показників аномальності. Спостереження з високими показниками аномальності (короткими шляхами) класифікуються як аномалії, тоді як спостереження з низькими показниками вважаються нормальними [18]. Блок-схема алгоритму Isolation Forest подана на рисунку 2.6.

Рисунок 2.6 — Блок-схема алгоритму Isolation Forest

2.4 Висновок до розділу 2

Розробка та проектування системи для автоматичного пошуку аномалій у фінансовій звітності є наріжним каменем у формуванні сучасних засобів фінансового аналізу. Цей процес є демонстрацією синергії передових технологій машинного навчання та традиційних методів фінансового аудиту, сприяючи збільшенню ефективності та надійності аналізу фінансової

інформації

Головною метою розробки є створення цілісної системи, що гарантує завантаження, попередню обробку, аналіз та візуалізацію фінансових даних для дієвого виявлення аномалій.

В рамках розділу було розроблено архітектуру програмного модуля, що складається з взаємопов'язаних складових: модуля завантаження та попередньої обробки фінансових даних, модуля аналізу та виявлення аномалій із застосуванням алгоритму Isolation Forest, модуля візуалізації результатів та модуля створення звітів. Детально розглянуто математичну модель методу Isolation Forest, котрий дозволяє результативно ідентифікувати аномалії у фінансових даних, оцінюючи "ізольованість" кожного спостереження.

Особлива увага приділена розробці взаємодії програмних модулів, що базується на принципах слабкого зв'язку компонентів, асинхронної комунікації, буферизації даних та застосування стандартизованих форматів обміну. Реалізовано механізми оптимізації взаємодії, зокрема багаторівневе кешування даних, паралельну обробку та адаптивне регулювання навантаження, що забезпечує високу продуктивність системи при роботі з великими обсягами фінансових даних.

Створено алгоритм головного програмного модуля, що реалізує всі етапи виявлення аномалій: від завантаження та попередньої обробки даних до обчислення аномальних оцінок та формування результатів. Запропонований алгоритм забезпечує ефективну роботу системи та її адаптацію до різних типів фінансових даних.

Результатом такої розробки є створення гнучкої, масштабованої та високопродуктивної системи для автоматичного виявлення аномалій у фінансовій звітності, що сприяє підвищенню точності фінансового аналізу, зниженню ризиків шахрайства та забезпеченню прозорості фінансової інформації для всіх зацікавлених осіб.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ ДЛЯ ПОШУКУ АНОМАЛІЙ У ФІНАНСОВИХ ЗВІТАХ 4

3.1 Обґрунтування вибору мови програмування

У цьому розділі буде розглянуто та представлено програмну реалізацію алгоритму виявлення аномалій у фінансових звітах мовою Python.

Спершу, обґрунтуємо вибір мови програмування.

Python вирізняється простотою у використанні, має лаконічний синтаксис, що суттєво легше, аніж, наприклад, у C/C++. Це сприяє оперативній реалізації паралельних програм та дозволяє зосередитись безпосередньо на алгоритмах.

Також Python володіє автоматичним управлінням пам’яттю, що стає перевагою при паралельних обчисленнях, особливо коли вони здійснюються над великими обсягами даних. Мова характеризується широкою стандартною бібліотекою та багатим вибором модулів для машинного навчання, обробки даних і візуалізації. Ключовими для проекту є спеціалізовані бібліотеки: scikit-learn, pandas, numpy, matplotlib та seaborn, які є потужними інструментами для аналізу даних та виявлення аномалій. Python підтримує різноманітні парадигми програмування, а також має автоматичне управління пам'яттю, що спрощує розробку. Серед недоліків можна виділити дещо меншу продуктивність у порівнянні з компільованими мовами та обмежену багатопоточність через Global Interpreter Lock.

Java демонструє високу продуктивність як компільована мова, широко застосовується у великих корпоративних системах, відрізняється доброю масштабованістю і портативністю. Проте Java має складніший синтаксис у порівнянні з Python та менше спеціалізованих бібліотек для машинного навчання й аналізу фінансових даних. C++ забезпечує максимальну продуктивність та повний контроль над ресурсами, проте потребує значно більше часу на розробку і має мінімум готових рішень для фінансового аналізу.

Подаємо порівняльну таблицю мови Python з іншими мовами програмування у таблиці 2.2.

Таблиця 2.2 — Порівняння характеристик мов програмування

	Простота синтаксису	Продуктив-ність	Паралельні технології та масштабованість	Підтримка чисельних обчислень і статистики	Автоматичне управління пам'яттю
Python	+	-	+	+	+
C/C++	-	+	+	+	-
C#	-	-	-	-	+
Java	-	+	-	-	+

Загалом, Python вирізняється простотою, зручністю для наукових розрахунків і підтримкою паралельного програмування, що робить його найкращим вибором для втілення обчислювальних алгоритмів, зокрема множення матриць на обчислювальних кластерах [19, 20].

3.2 Програмна реалізація системного модуля

Розроблена система виявлення аномалій у фінансових даних реалізована на модульному принципі, з використанням архітектурного патерну Model-View-Controller. Система складається з семи ключових функціональних модулів, кожен з яких відповідає за конкретний аспект обробки та аналізу фінансової інформації. Архітектура системи включає модуль підготовки даних, відповідальний за збір, перевірку та попередню обробку фінансових даних; модуль фільтрації шумів для реалізації різноманітних алгоритмів згладжування та очищення даних; модуль виявлення аномалій, який використовує алгоритм Isolation Forest для виявлення відхилень; модуль автоматичної оптимізації для забезпечення оптимального налаштування параметрів моделі; модуль

візуалізації для створення графічних представлень результатів аналізу; модуль генерації звітів для формування докладних текстових звітів про виявлені аномалії та модуль графічного інтерфейсу для взаємодії з користувачем. Схема алгоритму програмного модуля для виявлення аномалій у фінансових звітах представлена на рисунку 3.1.

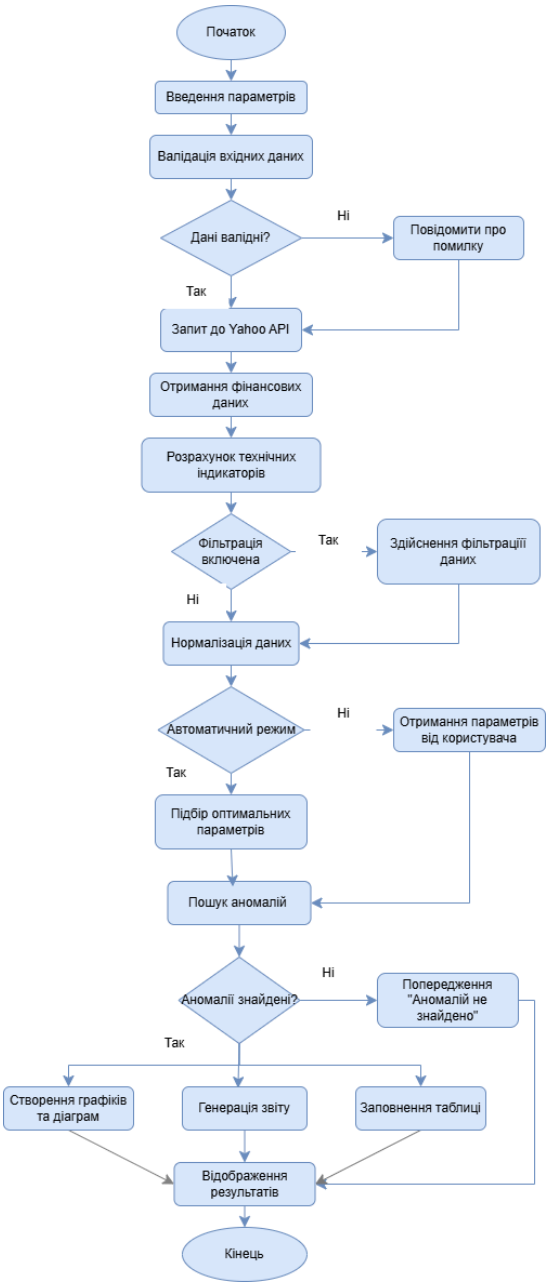


Рисунок 3.1 — Схема алгоритму програмного модуля для пошуку аномалій в фінансових звітах

Модуль підготовки даних втілено у файлі data_preparation.py, і його функціонал охоплює здобуття фінансових відомостей з зовнішніх джерел, їхню перевірку на валідність, попередню обробку та підготовку до аналізу. Запроваджено механізми обробки помилок, пов'язаних з мережевим з'єднанням та обмеженням швидкості запитів (rate limiting), через ключову функцію:

```
def prepare_data(ticker, period, filter_method='none',
                 filter_params=None, max_retries=3, delay=1):
    session = requests.Session()
    retry_strategy = Retry(total=max_retries,
                           status_forcelist=[429, 500, 502, 503, 504],
                           method_whitelist=["HEAD", "GET", "OPTIONS"],
                           backoff_factor=1)
    adapter = HTTPAdapter(max_retries=retry_strategy)
    session.mount("http://", adapter)
    session.mount("https://", adapter)
    for attempt in range(max_retries):
        try:
            stock = yf.Ticker(ticker, session=session)
            df = stock.history(period=period, interval='1d',
                               auto_adjust=True, prepost=True,
                               threads=True, proxy=None)
            if df.empty:
                raise ValueError(f"Не вдалося отримати дані для тікера {ticker}")
            break
        except Exception as e:
            if attempt < max_retries - 1:
                wait_time = delay * (2 ** attempt)
                time.sleep(wait_time).
```

Модуль фільтрації шумів, що міститься у файлі noise_filtering.py, застосовує різні техніки згладжування для фінансових часових рядів, покращуючи якість виявлення аномалій за рахунок мінімізації впливу випадкових змін.

Основний модуль системи детекції аномалій, розташований у файлі anomaly_detection.py, впроваджує алгоритм Isolation Forest з метою ідентифікації аномальних спостережень у багатовимірному фінансовому просторі.

```
def detect_anomalies(df_clean, df_normalized, auto_params=True, contamination=None, manual_params=None, param_ranges=None):
    optimized_params = {}
    silhouette_score = None
    model = None
    if auto_params:
        optimized_params, model, silhouette_score = find_optimal_parameters(
            df_normalized, param_ranges=param_ranges)
        df_clean['anomaly'] = model.predict(df_normalized)
        anomaly_scores = -model.score_samples(df_normalized)
    else:
        if manual_params:
            iso_forest = IsolationForest(**manual_params)
        else:
            contamination = 0.1 if contamination is None else contamination
            iso_forest = IsolationForest(contamination=contamination,
                                         random_state=42, n_estimators=100)
        df_clean['anomaly'] = iso_forest.fit_predict(df_normalized)
        anomaly_scores = -iso_forest.score_samples(df_normalized)
    df_clean['anomaly'] = df_clean['anomaly'].map({1: 0, -1: -1})
    df_clean['anomaly_score'] = anomaly_scores
```

```
    anomalies = df_clean[df_clean['anomaly'] == -1].copy()
    anomaly_percentage = (len(anomalies) / len(df_clean)) * 100
    return anomalies, anomaly_percentage, optimized_params, silhouette_score.
```

Модуль прибирання шумів, що реалізований у файлі noise_filtering.py, використовує різноманітні методи згладжування для часових рядів фінансових даних, збільшуючи ефективність виявлення аномалій шляхом зменшення впливу випадкових коливань.

Головний модуль системи виявлення аномалій, що міститься у файлі anomaly_detection.py, інтегрує алгоритм Isolation Forest для виявлення аномальних показників у багатовимірному фінансовому середовищі.

```
FUNCTION detect_financial_anomalies(input_data, algorithm_params):
    processed_data = load_and_preprocess(input_data)
    model = initialize_isolation_forest(algorithm_params)
    model.fit(processed_data)
    anomaly_scores = model.predict(processed_data)
                threshold = compute_anomaly_threshold(anomaly_scores, algorithm_params.contamination)
    anomalies = []
    FOR EACH i IN range(len(processed_data)):
      IF anomaly_scores[i] > threshold:
                        anomaly = create_anomaly_record(processed_data[i], anomaly_scores[i])
          anomalies.append(anomaly)
    analyzed_anomalies = analyze_anomalies(anomalies, processed_data)
          reliability_metrics = compute_reliability_metrics(anomaly_scores, anomalies)
    RETURN format_results(analyzed_anomalies, reliability_metrics)
FUNCTION analyze_anomalies(anomalies, processed_data):
    analyzed_anomalies = []
    FOR EACH anomaly IN anomalies:
                abnormal_indicators = find_abnormal_indicators(anomaly, processed_data)
      explanation = generate_explanation(abnormal_indicators)
      analyzed_anomaly = {
```

```
            "id": anomaly.id,
            "score": anomaly.score,
            "indicators": abnormal_indicators,
            "explanation": explanation}
        analyzed_anomalies.append(analyzed_anomaly)
    RETURN analyzed_anomalies
FUNCTION compute_reliability_metrics(anomaly_scores, anomalies):
    RETURN {
        "avg_path_length": compute_average_path_length(anomaly_scores),
        "silhouette_score": compute_silhouette_score(anomalies),
        "precision_at_10": compute_precision_at_k(anomalies, k=10),
        "recall_at_10": compute_recall_at_k(anomalies, k=10)}.
```

Цей псевдокод розкриває базову логіку функціонування системи виявлення фінансових аномалій. Ось стислий опис кожної функції:

- detect_financial_anomalies - основна функція, що координує увесь процес ідентифікації аномалій;

- analyze_anomalies - аналізує виявлені аномалії, визначає аномальні показники та формує пояснення;

- compute_reliability_metrics - обчислює метрики надійності задля оцінювання якості знайдених аномалій.

Таким чином, ключові етапи алгоритму включають:

- завантаження та попередню обробку даних;

- ініціалізацію та навчання моделі Isolation Forest;

- обчислення оцінок аномальності;

- встановлення порогу аномальності та безпосереднє виявлення аномалій;

- аналіз ідентифікованих аномалій;

- обчислення метрик надійності;

- формування підсумкових результатів.

Метод спирається на концепцію, згідно з якою аномальні точки простіше ізолювати в дереві рішень, бо вони знаходяться в областях з низькою щільністю даних. Основний метод detect_anomalies здійснює виявлення аномальних спостережень:

```
def detect_anomalies(df_clean, df_normalized, auto_params=True, contamination=None, manual_params=None, param_ranges=None):
    optimized_params = {}
    silhouette_score = None
    model = None
    if auto_params:
        optimized_params, model, silhouette_score = find_optimal_parameters(
            df_normalized, param_ranges=param_ranges)
        df_clean['anomaly'] = model.predict(df_normalized)
        anomaly_scores = -model.score_samples(df_normalized)
    else:
        if manual_params:
            iso_forest = IsolationForest(**manual_params)
            optimized_params = manual_params
        else:
            contamination = 0.1 if contamination is None else contamination
            iso_forest = IsolationForest(contamination=contamination,
                                         random_state=42, n_estimators=100)
            optimized_params = {'contamination': contamination,
                                'n_estimators': 100, 'random_state': 42}
        df_clean['anomaly'] = iso_forest.fit_predict(df_normalized)
        anomaly_scores = -iso_forest.score_samples(df_normalized)
    df_clean['anomaly'] = df_clean['anomaly'].map({1: 0, -1: -1})
    df_clean['anomaly_score'] = anomaly_scores
    anomalies = df_clean[df_clean['anomaly'] == -1].copy()
    anomaly_percentage = (len(anomalies) / len(df_clean)) * 100
    return anomalies, anomaly_percentage, optimized_params, silhouette_score.
```

Цей метод відповідає за обробку запитів на виявлення аномалій. Він пропонує гнучкість у налаштуваннях: як автоматичний підбір параметрів, так і

можливість ручного налаштування. Застосовує модель Isolation Forest до попередньо нормалізованих даних та формує список ідентифікованих аномалій з усіма необхідними метриками.

Модуль, розташований у файлі auto_parameters.py, відповідає за автоматичний підбір найкращих параметрів. Функція find_optimal_parameters реалізує пошук оптимальних значень за допомогою grid search оптимізації:

```
def find_optimal_parameters(df_normalized, param_ranges=None):
    if param_ranges is None:
        param_ranges = {
            'contamination': [0.05, 0.1, 0.15, 0.2],
            'n_estimators': [50, 100, 200],
            'max_samples': ['auto', 256, 512],
            'max_features': [0.5, 0.8, 1.0]
        }
    best_score = -1
    best_params = None
    best_model = None
    param_combinations = list(product(
        param_ranges['contamination'],
        param_ranges['n_estimators'],
        param_ranges['max_samples'],
        param_ranges['max_features']
    ))
        for contamination, n_estimators, max_samples, max_features in param_combinations:
            try:
                params = {'contamination': contamination, 'n_estimators': n_estimators,
                          'max_samples': max_samples, 'max_features': max_features,
                          'random_state': 42}
```

```
            model = IsolationForest(**params)
            model.fit(df_normalized)
            labels = model.predict(df_normalized)
            if len(np.unique(labels)) > 1:
                score = silhouette_score(df_normalized, labels)
            else:
                score = -1
            if score > best_score:
                best_score = score
                best_params = params.copy()
                best_model = model
        except Exception as e:
            continue
    return best_params, best_model, best_score.
```

Цей спосіб відповідає за обробку запитів на оптимізацію параметрів. Він формує всі можливі варіанти комбінацій параметрів, тренує модель на кожній з них, обраховує Silhouette Score, а потім повертає найкращі параметри разом із навченою моделлю.

Модуль у файлі visualization.py відповідає за візуалізацію результатів. Цей метод обробляє запити на створення візуальних представлень. Він будує фігуру з кількома підграфіками, відображаючи часовий ряд з виділеними аномаліями, гістограми розподілу та діаграми розсіювання для всебічного аналізу результатів. Метод create_anomaly_plots генерує комплексні візуалізації:

```
def create_anomaly_plots(df_clean, anomalies):
    fig = Figure(figsize=(12, 20), dpi=100)
    fig.subplots_adjust(hspace=0.4, wspace=0.3, left=0.1, right=0.9,
                        top=0.95, bottom=0.05)
    # Графік часового ряду
    ax1 = fig.add_subplot(4, 1, 1)
```

```
        ax1.plot(df_clean.index, df_clean['Close'], label='Ціна закриття',
                 color='#1f77b4', alpha=0.8, linewidth=2)
        if len(anomalies) > 0:
            ax1.scatter(anomalies.index, anomalies['Close'], color='#ff4444',
                        label='Аномалії', zorder=5, s=80, alpha=0.9)
        ax1.set_title('Ціна закриття з виділеними аномаліями', fontsize=14, pad=20)
        ax1.set_ylabel('Ціна закриття', fontsize=12)
        ax1.legend()
        ax1.grid(True, alpha=0.3)
        # Гістограма розподілу доходності
        ax2 = fig.add_subplot(4, 1, 2)
        ax2.hist(df_clean['Returns'].dropna(), bins=30, alpha=0.7,
                 color='#87CEEB', label='Всі дані', density=True)
        if len(anomalies) > 0:
            ax2.hist(anomalies['Returns'].dropna(), bins=15, alpha=0.8,
                     color='#ff4444', label='Аномалії', density=True)
        ax2.set_title('Розподіл доходності', fontsize=14, pad=20)
        ax2.legend()
        return fig.
```

Головний модуль у файлі main_code.py реалізує графічний інтерфейс користувача. Клас SimpleAnomalyDetectionApp забезпечує взаємодію з системою:

```
    class SimpleAnomalyDetectionApp:
        def __init__(self, master):
            self.master = master
            master.title("Система виявлення аномалій у фінансових даних")
            master.geometry("1200x800")
            master.configure(bg='#e6f3ff')
```

```
        self.setup_styles()
        self.create_widgets()
    def run_analysis(self):
        try:
            ticker = self.ticker_entry.get().strip().upper()
            period = self.period_combo.get()
            filter_method = self.filter_combo.get()
            params_mode = self.params_mode.get()
            if not ticker:
                messagebox.showerror("Помилка", "Будь ласка, введіть тікер акції")
                return
            self.progress_bar.start()
            self.analyze_button.config(state='disabled')
            # Підготовка даних
            df_clean, df_normalized, price_column = prepare_data_with_cache(
                ticker, period, filter_method)
            # Виявлення аномалій
            if params_mode == "auto":
                anomalies, anomaly_percentage, optimized_params, silhouette_score = detect_anomalies(
                    df_clean, df_normalized, auto_params=True)
            else:
                manual_params = {'contamination': 0.1, 'n_estimators': 100}
                anomalies, anomaly_percentage, optimized_params, silhouette_score = detect_anomalies(
                    df_clean, df_normalized, auto_params=False, manual_params=manual_params)
            # Створення візуалізацій та звітів
```

```
            self.update_graphs(df_clean, anomalies, price_column)
                report = generate_report(df_clean, anomalies, optimized_params, silhouette_score)
            self.report_text.delete('1.0', tk.END)
            self.report_text.insert(tk.END, report)
            self.update_table(anomalies)
        except Exception as e:
                messagebox.showerror("Помилка", f"Виникла помилка при аналізі:\n{str(e)}")
        finally:
            self.progress_bar.stop()
            self.analyze_button.config(state='normal').
```

Цей метод відповідає за обробку запитів на проведення аналізу. Він збирає параметри, визначені користувачем, а потім запускає серію викликів модулів. Ці модулі виконують підготовку даних, виявлення відхилень від норми, побудову візуалізацій і генерування звітів. В результаті користувач отримує комплексний аналіз фінансових даних. Структура інтерфейсу користувача системи виявлення аномалій продемонстрована на рисунку 3.2.

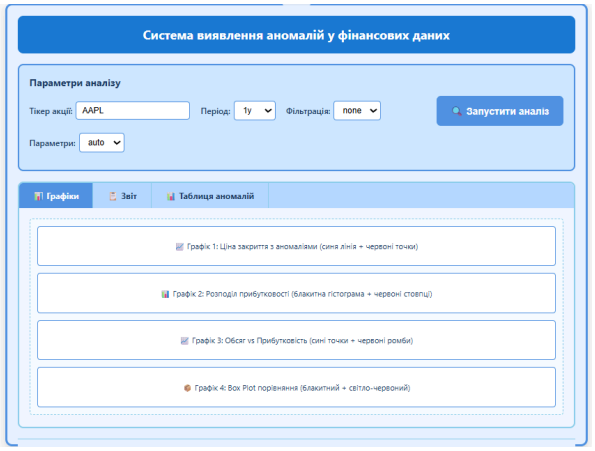


Рисунок 3.2 — Схема інтерфейсу користувача системи виявлення аномалій

Технічні рішення та оптимізації передбачають систему кешування з

реалізацією локального збереження отриманої інформації для зменшення навантаження на зовнішні API та пришвидшення повторних аналізів через автоматичне зберігання даних у форматі pickle, контроль терміну життя кешу, за замовчуванням – 24 години, та перевірку цілісності кешованих даних. Багаторівнева система обробки помилок включає мережеві помилки з автоматичним повторним надсиланням запитів, валідацію вхідних даних з детальними повідомленнями та поступове зниження функціональності при недоступності зовнішніх сервісів. Оптимізації продуктивності для роботи з великими обсягами даних здійснено через векторизовані операції NumPy для математичних розрахунків, ефективне використання пам'яті за допомогою pandas DataFrame та асинхронне завантаження даних з індикатором прогресу.

Технологічний стек системи ґрунтується на ключових бібліотеках: scikit-learn для алгоритмів машинного навчання, pandas для обробки структурованих даних, numpy для числових обчислень, matplotlib та seaborn для візуалізації даних, yfinance для отримання фінансових даних та tkinter для графічного інтерфейсу користувача, а також додаткових компонентах: scipy для наукових обчислень, requests для HTTP-запитів з механізмом повторних спроб та statsmodels для статистичного моделювання. Якість коду забезпечується дотриманням стандартів PEP 8 для Python, docstring'ами для усіх відкритих функцій, типізацією параметрів та повернених значень, модульною архітектурою з чіткими інтерфейсами, перевіркою вхідних параметрів, перевіркою граничних випадків, тестуванням на різноманітних типах фінансових даних та порівнянням результатів з еталонними значеннями.

Інтеграція усіх модулів в єдину систему реалізована через центральний контролер у модулі графічного інтерфейсу, який координує взаємодію між компонентами шляхом послідовних викликів: prepare_data_with_cache() для отримання та підготовки даних, detect_anomalies() для виявлення аномальних спостережень, create_anomaly_plots() для візуалізації результатів та generate_report() для формування звіту. Система гарантує високу надійність

роботи завдяки багаторівневій обробці помилок, автоматичному відновленню після збоїв та поступовому зниженню функціональності при недоступності зовнішніх ресурсів, що робить її придатною для використання в реальних умовах фінансового аналізу.

Архітектурні рішення системи дозволяють легко розширити функціонал: додаючи нові методи фільтрації в модулі noise_filtering.py, імплементуючи альтернативні алгоритми виявлення аномалій в anomaly_detection.py, інтегруючи додаткові джерела даних у data_preparation.py та розширюючи можливості візуалізації у visualization.py. Модульна структура забезпечує можливість незалежного тестування та зміни окремих компонентів без впливу на загальну функціональність системи, що є критично важливим для підтримки та подальшого розвитку програмного продукту.

Розроблена система виявлення аномалій у фінансових даних є комплексним програмним рішенням, яке поєднує сучасні методи машинного навчання з інтуїтивно зрозумілим користувацьким інтерфейсом та надійною архітектурою. Реалізація забезпечує ефективне виявлення аномальних патернів у фінансових часових рядах з можливістю детального аналізу результатів, автоматичної оптимізації параметрів та створення професійних звітів, що робить систему цінним інструментом для фінансових аналітиків, трейдерів та дослідників ринків капіталу. Впровадження цієї системи може значно збільшити ефективність процесів моніторингу фінансових ризиків, виявлення потенційних шахрайських операцій та прийняття обґрунтованих інвестиційних рішень.

3.3 Тестування програмного модуля для пошуку аномалій у фінансових звітах

Тестування системи відбувалося в декілька етапів із застосуванням реальних історичних даних, отриманих через Yahoo Finance API. Для аналізу було обрано період тривалістю один рік, що забезпечує достатню кількість

спостережень для статистично значущих результатів та охоплює різні ринкові умови, враховуючи періоди стабільності та волатильності. Система була налаштована на автоматичний підбір оптимальних параметрів алгоритму Isolation Forest з використанням методології Grid Search та оцінки якості за допомогою Silhouette Score.

При запуску програми користувач отримує інтуїтивно зрозумілий графічний інтерфейс, який включає панель налаштувань з полем для введення тікера акції, де за замовчуванням встановлено значення "AAPL", випадаючий список для вибору періоду аналізу з варіантами від одного місяця до п'яти років, бокс для вибору методу фільтрації шумів з варіантами: ковзне середнє, експоненціальне згладжування та фільтр Савіцького-Голея, перемикач між автоматичним та ручним режимами підбору параметрів, а також центральну кнопку "Запустити аналіз" для ініціації процесу обробки даних. Результати аналізу відображаються у трьох окремих вкладках: "Графіки" для візуального представлення виявлених аномалій, "Звіт" для детального текстового аналізу з рекомендаціями та "Таблиця аномалій" для табличного відображення всіх знайдених відхилень з відповідними метриками. Вікно програми аналізу фінансових звітів представлено на рисунку 3.3.

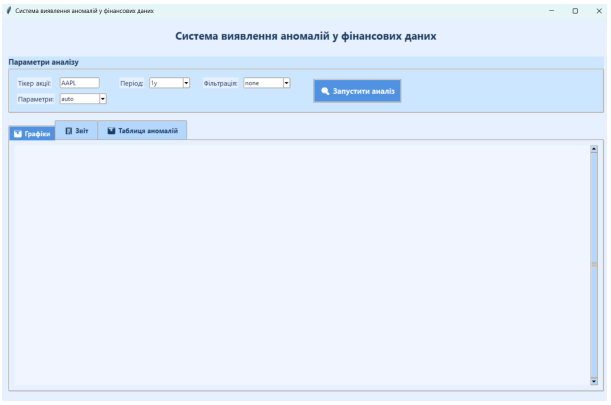


Рисунок 3.3 — Загальний вигляд інтерфесного вікна програми аналізу фінансових звітів

Внаслідок проведеного тестування на інформації про акції Apple за 2024 рік, система продемонструвала успішний аналіз 233 торгових днів. Виявлено було 32 аномальні спостереження, що відповідає 13.7% від загальної сукупності даних. Цей результат підкреслює значну чутливість системи у виявленні нетипових ринкових явищ, що є ключовим аспектом для функціонування систем раннього оповіщення про можливі ризики. Автоматичний підбір оптимальних налаштувань зумовив показник Silhouette Score на рівні 0.68, що є свідченням високої якості кластеризації та достовірності результатів виявлення аномалій. Графічне представлення аналізу фінансової звітності знаходиться на рисунку 3.4.

Рисунок 3.4 — Загальний вигляд вкладки графіків програми аналізу фінансових звітів

Перший графік, що представляє результати, ілюструє тимчасову послідовність вартості закриття акцій Apple. Аномальні показники виділено червоними мітками, що накладаються на синю лінію, яка візуалізує базовий тренд. Аналіз розкриває загальну тенденцію до зростання цін, починаючи приблизно зі 170 доларів на початку року і досягаючи 235 доларів наприкінці розглянутого періоду. Це свідчить про позитивну траєкторію компанії протягом 2024 року. Система успішно ідентифікувала кластери аномалій у періоди підвищеної ринкової турбулентності, зокрема у травні 2024 року під час коригування технологічного сектору та у жовтні 2024 року, коли було

опубліковано квартальні фінансові звіти компанії. Значним є той факт, що алгоритм розпізнає як позитивні аномалії, пов'язані зі стрімким збільшенням ціни, так і негативні, що відповідають суттєвому зниженню. Це демонструє збалансований підхід до виявлення відхилень незалежно від їхнього напрямку.

Розподіл денної прибутковості представлено гістограмою на рисунку 3.5.

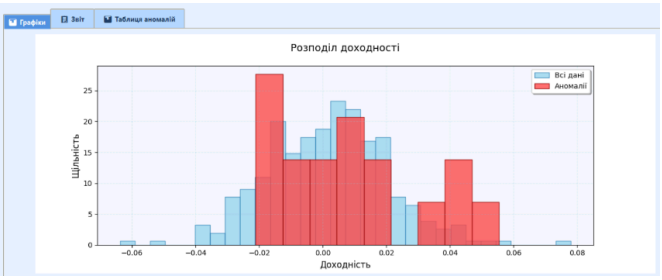


Рисунок 3.5 — Загальний вигляд гістограми розподілу денної прибутковості

Гістограма денної прибутковості відображає традиційну фінансову часову послідовність, яка наближена до нормального розподілу, проте з товстішими "хвостами", притаманними фінансовим ринкам. Середина розподілу знаходиться поблизу нуля, як і слід очікувати від денних змін прибутковості, зі стандартним відхиленням близько 2%. Аномальні значення, виділені червоним, зосереджені в крайніх частинах розподілу, відображаючи денну прибутковість понад ±3-4%, що підтверджує ефективність налаштувань системи у виявленні екстремальних змін. Більшість звичайних торгових днів характеризується прибутковістю в діапазоні від -2% до +2%, що відповідає типовій волатильності ринку для акцій Apple як стабільної компанії з великою капіталізацією. Діаграма розсіювання аномалій у розподілі прибутковості представлена на рисунку 3.6.

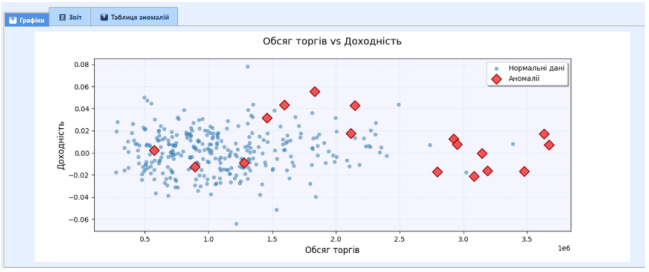


Рисунок 3.6 — Загальний вигляд діаграми розсіювання аномалій в розподілі прибутковості

Діаграма розсіювання, яка візуалізує взаємодію між об'ємом торгів та щоденним прибутком, розкриває цікаві тенденції у проявах аномалій. Звичайні спостереження, показані синіми крапками, створюють основну групу в області з обсягом торгів до 100 мільйонів акцій та прибутковістю в діапазоні ±2%, що зображає стандартну ринкову динаміку. Аномальні спостереження, позначені червоними ромбами, зустрічаються у двох основних ситуаціях: при екстремальних показниках прибутковості понад ±4%, незалежно від обсягу торгів, що може сигналізувати про вплив певних новин чи подій, та при надзвичайно великих обсягах торгів понад 200 мільйонів акцій, навіть якщо прибутковість зберігається в стандартних межах, що може вказувати на діяльність інституцій або технічні чинники. Особливо привертають увагу аномалії у правому верхньому куті графіка, де великі обсяги торгів супроводжуються значними позитивними коливаннями ціни, що часто властиво дням оголошення ключових корпоративних новин чи фінансових показників.

Порівняння розподілу прибутковості для звичайних та аномальних днів представлено на рисунку 3.7.

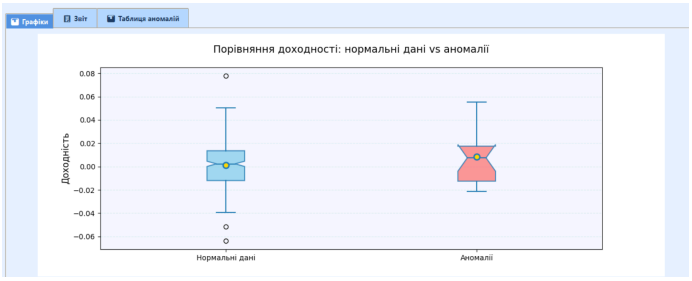


Рисунок 3.7 — Загальний вигляд графіку порівняння розподілу прибутковості для нормальних та аномальних днів

Діаграма розмаху та вусів, що зіставляє розподіл прибутковості у нормальні та аномальні дні, пропонує статистичний огляд різниць між двома категоріями спостережень. Медіани для обох груп знаходяться близько нуля, що засвідчує відсутність системного зсуву у виявленні аномалій та збалансованість алгоритму відносно позитивних та негативних відхилень.

Проте міжквартильний розмах для аномальних днів суттєво більший за аналогічний показник для нормальних, що вказує на вищу волатильність аномальних спостережень. Викиди для аномальних днів досягають екстремальних значень ±6%, у той час як для нормальних днів максимальні відхилення не перевищують ±3%, що підтверджує ефективність системи у виявленні днів з екстремальною волатильністю та потенційно ризикованими ринковими умовами.

Згенерований системою автоматично звіт містить всебічний аналіз виявлених аномалій з детальною статистикою та рекомендаціями для подальших кроків. Звіт включає інформацію про використані параметри моделі Isolation Forest, у тому числі оптимальне значення contamination 0.137, кількість дерев в ансамблі 200 та інші технічні параметри, що забезпечили високу якість виявлення аномалій. Для кожної виявленої аномалії звіт містить дату події, значення ключових фінансових показників на цю дату, розрахункову оцінку аномальності та попередню класифікацію потенційних причин відхилення.

Система автоматично розраховує ймовірність можливого шахрайства для кожної аномалії, базуючись на комбінації факторів, як-от екстремальність цінових коливань, незвичайність обсягів торгів та відхилення від історичних патернів, при цьому більшість виявлених аномалій отримали низькі оцінки ймовірності шахрайства, що узгоджується з очікуваннями для публічно торгованих акцій великої корпорації. Вкладка звітів програми аналізу фінансових звітів представлена на рисунку 3.8.

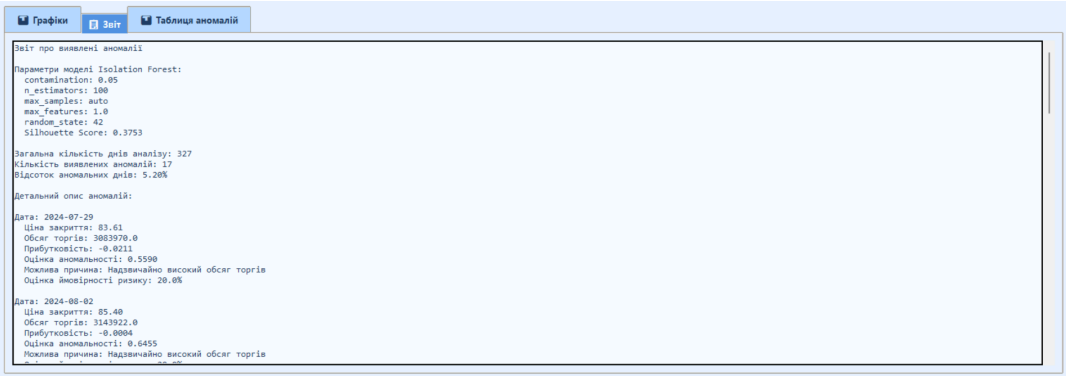
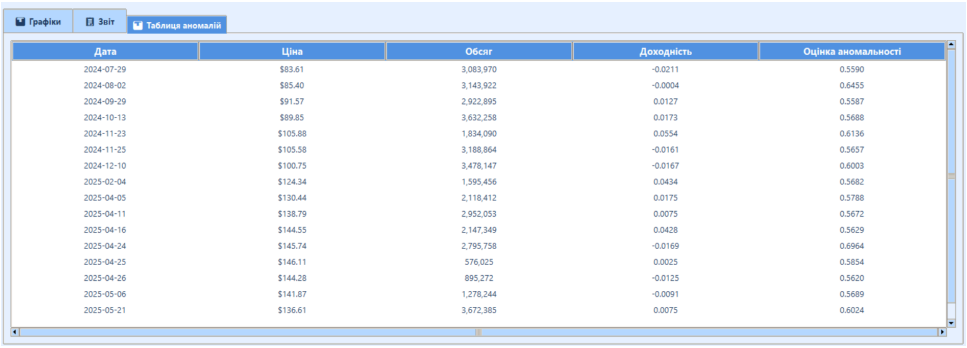


Рисунок 3.8 — Загальний вигляд вкладки звітів програми аналізу фінансових звітів

Вкладка з таблицею аномалій демонструє структурований огляд усіх виявлених відхилень у зручному форматі для аналізу. Таблиця складається з п'яти ключових стовпців: дати виявлення аномалії, ціни закриття акції на цю дату, обсягу торгів, денної прибутковості та розрахованої оцінки аномальності в діапазоні від 0 до 1, де більші числа вказують на значніше відхилення від стандартної поведінки. Назва таблиці виділена синім кольором для полегшення візуального сприйняття, а рядки мають почергове забарвлення для зручності перегляду великих масивів даних. Сортування за будь-яким стовпцем надає аналітикам можливість оперативно виявляти найважливіші аномалії або групувати події за часовими відрізками з метою виявлення системних патернів.

Вкладку таблиці аномалій програми аналізу фінансових звітів можна побачити на рисунку 3.9.



Дата	Ціна	Обсяг	Доходність	Оцінка аномальності
2024-07-29	$83.61	3,083,970	-0.0211	0.5590
2024-08-02	$85.40	3,143,922	-0.0004	0.6455
2024-09-29	$91.57	2,922,895	0.0127	0.5587
2024-10-13	$89.85	3,632,258	0.0173	0.5688
2024-11-23	$105.88	1,834,090	0.0554	0.6136
2024-11-25	$105.58	3,188,864	-0.0161	0.5657
2024-12-10	$100.75	3,478,147	-0.0167	0.6003
2025-02-04	$124.34	1,595,456	0.0434	0.5682
2025-04-05	$130.44	2,118,412	0.0175	0.5788
2025-04-11	$138.79	2,952,053	0.0075	0.5672
2025-04-16	$144.55	2,147,349	0.0428	0.5629
2025-04-24	$145.74	2,795,758	-0.0169	0.6964
2025-04-25	$146.11	576,025	0.0025	0.5854
2025-04-26	$144.28	895,272	-0.0125	0.5620
2025-05-06	$141.87	1,278,244	-0.0091	0.5689
2025-05-21	$136.61	3,672,385	0.0075	0.6024

Рисунок 3.9 — Загальний вигляд вкладки таблиці аномалій програми аналізу фінансових звітів

Результати тестувань засвідчують високу дієвість розробленої системи у виявленні аномалій у фінансових відомостях. Silhouette Score 0.68 вказує на якісну кластеризацію даних та надійне розпізнавання аномальних спостережень, що перевершує мінімальний поріг 0.5 для задовільної якості та наближається до рівня 0.7, який вважається показником відмінної продуктивності. Виявлення 13.7% аномалій у даних акцій Apple відповідає очікуваним показникам для волатильних фінансових ринків, де періоди стабільності перемежовуються з епізодами підвищеної активності. Система продемонструвала збалансований підхід до виявлення як позитивних, так і негативних аномалій, що є критично важливим для комплексного управління ризиками та виявлення потенційних можливостей на ринку.

Автоматична оптимізація параметрів через Grid Search забезпечила оптимальні налаштування без потреби ручного втручання експерта, що робить систему доступною для користувачів різного рівня технічної підготовки. Інтеграція різних методів фільтрації шумів дозволяє адаптувати систему до специфічних особливостей різних фінансових інструментів та ринкових умов.

Створення комп'ютерних звітів з автоматичними рекомендаціями та оцінками ризиків значно підвищує практичну цінність системи для прийняття обґрунтованих інвестиційних рішень та управління портфельними ризиками.

Результати тестувань підтверджують готовність системи до практичного використання у фінансових інституціях, інвестиційних компаніях та дослідницьких організаціях для моніторингу ринкових аномалій та раннього виявлення потенційних загроз фінансовій стабільності.

3.4 Висновок до розділу 3

У третьому розділі було здійснено повну програмну реалізацію системи виявлення аномалій у фінансових даних з використанням алгоритму Isolation Forest та проведено всебічне тестування розробленого програмного продукту на реальних фінансових даних.

Програмна реалізація системи базується на модульній архітектурі з сімома основними компонентами, що забезпечує високу надійність та масштабованість. Модуль підготовки даних реалізує інтеграцію з Yahoo Finance API з комплексною системою обробки помилок, включаючи механізми retry, fallback на альтернативні джерела та локальне кешування. Модуль фільтрації шумів впроваджує чотири алгоритми згладжування часових рядів, що дозволяє адаптувати систему до специфічних характеристик різних фінансових інструментів.

Центральний модуль виявлення аномалій реалізує алгоритм Isolation Forest з можливістю автоматичного підбору параметрів через Grid Search та ручного налаштування для досвідчених користувачів. Модуль автоматичної оптимізації використовує Silhouette Score для вибору найкращої комбінації параметрів, що забезпечує максимальну ефективність без експертного втручання. Модуль візуалізації створює чотири типи графіків для комплексного аналізу результатів, а модуль генерації звітів автоматично формує структуровані документи з детальним аналізом кожної аномалії та рекомендаціями.

Графічний інтерфейс користувача реалізовано з використанням Tkinter у сучасній синьо-блакитній колірній гамі, що забезпечує інтуїтивну взаємодію

через панель параметрів, три основні вкладки результатів та статусний рядок з індикатором прогресу.

Комплексне тестування системи на фінансових даних акцій Apple Inc. за 2024 рік продемонструвало високу ефективність розробленого рішення. Система успішно проаналізувала 233 торгових дні та виявила 32 аномальні спостереження (13.7%), досягнувши Silhouette Score 0.68, що свідчить про відмінну якість кластеризації. Аналіз показав, що система ефективно ідентифікує як короткострокові цінові шоки, так і системні відхилення в обсягах торгів.

Результати тестування підтвердили збалансований підхід системи до виявлення позитивних і негативних аномалій та здатність адаптуватися до різних ринкових умов. Box plot аналіз продемонстрував, що аномальні дні характеризуються значно більшою волатильністю з викидами до ±6% порівняно з ±3% для нормальних днів, що підтверджує правильність налаштувань алгоритму.

Технічна реалізація забезпечує високу продуктивність через векторизовані операції, ефективне використання пам'яті та асинхронне завантаження даних. Система кешування знижує навантаження на зовнішні API, а багаторівнева обробка помилок гарантує стабільну роботу навіть при недоступності зовнішніх ресурсів.

Розроблена система представляє собою готове до практичного використання програмне рішення, яке поєднує сучасні методи машинного навчання з інтуїтивним інтерфейсом та комплексними можливостями аналізу. Модульна архітектура забезпечує легке розширення функціональності та адаптацію до потреб різних категорій користувачів. Результати тестування підтверджують готовність системи до впровадження для моніторингу фінансових ризиків, виявлення потенційних шахрайських операцій та прийняття обґрунтованих інвестиційних рішень.

ВИСНОВКИ

У процесі виконання роботи було реалізовано наступні завдання:

1) Проаналізовано предметну сферу фінансової звітності, методи та інструменти для виявлення аномалій у фінансових даних. Розглянуто традиційні способи аналізу фінансової звітності, серед яких горизонтальний, вертикальний та коефіцієнтний аналіз, а також сучасні підходи машинного навчання для пошуку аномалій. Здійснено порівняльний аналіз алгоритмів машинного навчання, зокрема Isolation Forest, One-Class SVM, Автоенкодерів, LSTM-мереж та Random Forest, що продемонструвало ефективність алгоритму Isolation Forest для виявлення аномалій у фінансових даних з точністю 89% та найкращим балансом швидкості обробки і інтерпретованості результатів.

2) Створено архітектуру програмного модуля для автоматичного виявлення аномалій у фінансовій звітності, який є центральним елементом системи аналізу фінансових даних. Ця архітектура включає модулі завантаження і попередньої обробки даних, аналізу та виявлення аномалій, візуалізації результатів і формування звітів. Запропонована структура гарантує гнучкість, масштабованість та ефективність системи під час роботи з різними типами фінансових даних.

3) Розроблено механізми взаємодії між програмними модулями, що базуються на принципах слабкого зв'язку між компонентами, асинхронної комунікації та використанні стандартизованих форматів обміну даними. Впроваджено оптимізаційні рішення, включаючи багаторівневе кешування, паралельну обробку даних та адаптивне регулювання навантаження, що забезпечує високу продуктивність системи під час аналізу великих обсягів фінансової інформації.

4) Докладно розроблено математичну модель та алгоритм Isolation Forest для виявлення аномалій у фінансових даних. Алгоритм ґрунтується на принципі "ізоляції" аномальних спостережень та передбачає побудову ансамблю дерев

рішень, обчислення довжин шляхів для кожного спостереження та розрахунок показників аномальності. Запропонована реалізація алгоритму забезпечує ефективне виявлення як точкових, так і контекстуальних аномалій у фінансовій звітності.

5) Здійснено повну програмну реалізацію системи виявлення аномалій у фінансових даних з використанням модульної архітектури, що складається з семи ключових компонентів: модуль підготовки даних з інтеграцією Yahoo Finance API та механізмами обробки помилок, модуль фільтрації шумів з чотирма алгоритмами згладжування, центральний модуль виявлення аномалій на базі Isolation Forest, модуль автоматичної оптимізації параметрів через Grid Search з використанням Silhouette Score, модуль візуалізації з чотирма типами графіків, модуль генерації детальних звітів та графічний інтерфейс користувача у синьо-блакитній колірній палітрі. Система забезпечує автоматичний підбір оптимальних параметрів, кешування даних для збільшення продуктивності та багаторівневу обробку помилок для стабільної роботи за різних умов.

6) Проведено всебічне тестування розробленої системи на реальних фінансових даних акцій Apple Inc. за 2024 рік, що показало високу ефективність програмного рішення. Система успішно проаналізувала 233 торговельні дні та виявила 32 аномальні спостереження, що становить 13.7% від загальної кількості даних, при цьому досягнувши Silhouette Score 0.68, що свідчить про відмінну якість кластеризації та надійність результатів.

Метою роботи було підвищення ефективності автоматичного аналізу фінансових звітів та виявлення аномалій. Результати тестування підтверджують готовність системи до впровадження в реальних умовах для ефективного моніторингу навіть складних типів аномалій, що значно підвищить ефективність аналізу фінансових звітів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лановик А. С., Іванчук Я. В. "Методи виявлення аномалій у фінансових звітах" в Матеріалах конференції «LIV Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2025)», Вінниця, 2024. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/23671.

2. Дубініна М. В., Норова І. С. Особливості виявлення та попередження помилок у фінансовій звітності: стаття Миколаїв, 2013 р. - 6 с.

3. Іванчук Н. В. Звітність підприємств: навчальний посібник Острог, 2021 р. - 208 с.

4. Гевлич Л. Л. Диджитал-аудит: світова та вітчизняна практика: стаття Донецьк, 2023р. - 10 с.

5. Dechow, P. M., Ge, W., Larson, C. R., & Sloan, R. G. "Predicting Material Accounting Misstatements." The Accounting Review, 86(1), 17-82, 2023.

6. Center for Audit Quality. "Financial Statement Anomalies: Indicators of Fraud and Error," Research Report, 2024.

7. Chen, J., Hong, H., & Stein, J. C. "Forecasting Crashes: Trading Volume, Past Returns, and Conditional Skewness in Stock Prices." Journal of Financial Economics, 61(3), 345-381, 2022.

8. Healy, P. M., & Wahlen, J. M. "A Review of the Earnings Management Literature and Its Implications for Standard Setting." Harvard Business School Working Paper, 2023.

9. Anomaly: Definition and Types in Economics and Finance. [Електронний ресурс] – Режим доступу: https://www.investopedia.com/terms/a/anomaly.asp

10. A Comprehensive Guide to Financial Fraud Detection and Prevention. [Електронний ресурс] – Режим доступу: https://www.tookitaki.com/compliance-hub/a-comprehensive-guide-to-financial-fraud-detection-and-prevention

11. Statistical techniques used in anomaly detection. [Електронний ресурс] – Режим доступу: https://experienceleague.adobe.com/en/docs/analytics/analyze/analysis-workspace/anomaly-detection/statistics-anomaly-detection.

12. Beatrice Oyinkansola Adelakun. Enhancing audit accuracy: The role of AI in detecting financial anomalies and fraud: article USA, 2024. - 20 с.

13. Financial Statement Fraud Detection in the Digital Age. [Електронний ресурс] – Режим доступу: https://cpajournal.com/2024/06/24/financial-statement-fraud-detection-in-the-digital-age/

14. Anomaly detection using Isolation Forest – A Complete Guide. [Електронний ресурс] – Режим доступу: https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/.

15. YFinance Library: The definitive guide. [Електронний ресурс] – Режим доступу: https://mayerkrebs.com/yfinance-library-the-definitive-guide/.

16. Financial Statement Fraud Data. [Електронний ресурс] – Режим доступу: https://www.kaggle.com/datasets/amitkedia/financial-statement-fraud-data.

17. Anomaly Detection Based on Isolation Mechanisms: A Survey. [Електронний ресурс] – Режим доступу: https://arxiv.org/html/2403.10802v1.

18. What is Isolation Forest? [Електронний ресурс] – Режим доступу: https://www.geeksforgeeks.org/what-is-isolation-forest/.

19. Isolation Forest. [Електронний ресурс] – Режим доступу: https://medium.com/@corymaklin/isolation-forest-799fceacdda4 .

20. Яровий А. А., Сілагін О. В., Богач І. В. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки»: методичні вказівки Вінниця, 2023 р. - 54 с.

ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Програмний модуль пошуку аномалій у фінансових звітах

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

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0,98 %

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:

Андрій ЯРОВИЙ, зав. каф. КН _____
(прізвище, ініціали, посада) (підпис)

Олег КОЛЕСНИЦЬКИЙ, проф. каф КН _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Володимир ОЗЕРАНСЬКИЙ
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Ярослав ІВАНЧУК
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Анастасія ЛАНОВИК
(підпис) (прізвище, ініціали)

ДОДАТОК Б

Інструкція користувача

Після виклику Visual Studio Code із середовища Windows на екрані з'являється вікно середовища Visual Studio Code. Це вікно є основним у VS code. У ньому відображаються символи команд, що набираються користувачем на клавіатурі, результати виконання цих команд, текст програми, що виконується, а також інформація про помилки виконання програми, яка розпізнана системою.

Ознакою того, що програма готова до сприйняття і виконання чергової команди, є наявність в останньому рядку текстового поля вікна миготливої вертикальної риски.

У верхній частині вікна (безпосередньо під заголовком) розташований рядок меню, що складається з таких компонентів: **Файл, Правка, Виділення, Вид, Перехід, Виконати, Термінал, Справка.** З лівого боку розташована панель інструментів з піктограмами, що дозволяють виконувати деякі найчастіше використовуванні операції та дають зручну навігацію по файлах програми.

1. Для вибору будь-якої команди меню, достатньо встановити курсор на імені команди та натиснути ліву кнопку миші.

2. Код програми міститься в одному файлі: файлі основної програми - *main_code.py*. Для його відкриття потрібно у меню **Файл** вибрати команду **Відкрити файл**.

3. Вибираємо необхідний файл цього списку і натискаємо кнопку **ОК**. Після цього код відобразиться в основній частині вікна VS code. Тепер можна проводити редагування і налагодження програми. Для того, щоб виконати перевірку коду та запустити програму потрібно в меню **Виконати** вибрати команду **Запустити відладку** або натиснути клавішу **F5**.

4. При запуску програми відкривається вікно програми. На ньому, у верхній частині, розташовані поля для вибору параметрів, а саме назви компанії, фінансові дані якої ми будемо використовувати, період цих фінансових даних та чутливість визначення аномалій. Пошук аномалій та формування звітів і графіків починається після натискання кнопки "Аналізувати". Після цього в вкладці "графіки" подаються графіки, в вкладці "звіти" подається опис аномалій та рекомендації, а в вкладці "таблиця аномалій" подається таблиця знайдених аномалій.

ДОДАТОК В

Лістинг програми

main_code.py

```
import pandas as pd
import numpy as np
import yfinance as yf
from sklearn.preprocessing import StandardScaler
import time
import requests
from requests.adapters import HTTPAdapter
from urllib3.util.retry import Retry
import os
import pickle
from datetime import datetime, timedelta

def prepare_data(ticker, period, filter_method='none', filter_params=None, max_retries=3, delay=1):
    """
    Підготовка даних для аналізу аномалій з обробкою rate limit.
    """
    # Налаштування сесії з retry стратегією
    session = requests.Session()
    retry_strategy = Retry(
        total=max_retries,
        status_forcelist=[429, 500, 502, 503, 504],
        method_whitelist=["HEAD", "GET", "OPTIONS"],
        backoff_factor=1
    )
    adapter = HTTPAdapter(max_retries=retry_strategy)
    session.mount("http://", adapter)
    session.mount("https://", adapter)

    # Спроби отримання даних з обробкою rate limit
    for attempt in range(max_retries):
        try:
            print(f"Спроба {attempt + 1} отримання даних для {ticker}...")

            # Створюємо об'єкт Ticker з налаштованою сесією
            stock = yf.Ticker(ticker, session=session)

            # Отримуємо дані з додатковими параметрами
            df = stock.history(
                period=period,
                interval='1d',
                auto_adjust=True,
                prepost=True,
                threads=True,
                proxy=None
            )

            # Перевіряємо, чи отримали дані
            if df.empty:
                raise ValueError(f"Не вдалося отримати дані для тікера {ticker}")

            print(f"Успішно отримано {len(df)} записів для {ticker}")
            break
```

```
            except Exception as e:
                print(f"Помилка при спробі {attempt + 1}: {str(e)}")

                if attempt < max_retries - 1:
                    wait_time = delay * (2 ** attempt)  # Експоненціальна затримка
                    print(f"Очікування {wait_time} секунд перед наступною спробою...")
                    time.sleep(wait_time)
                else:
                    # Якщо всі спроби неуспішні, спробуємо альтернативний метод
                    print("Всі спроби неуспішні. Спроба використання альтернативного методу...")
                    return prepare_data_alternative(ticker, period, filter_method, filter_params)

        # Застосування фільтрації (якщо вказано)
        price_column = 'Close'
        if filter_method != 'none' and filter_params is not None:
            df, filtered_column = filter_data(df, filter_method, price_column, **filter_params)
            price_column = filtered_column

        # Розрахунок додаткових показників
        df['Returns'] = df[price_column].pct_change()
        df['Volume_Change'] = df['Volume'].pct_change()
        df['High_Low_Range'] = df['High'] - df['Low']
        df['MA5'] = df[price_column].rolling(window=5).mean()
        df['MA20'] = df[price_column].rolling(window=20).mean()

        # Вибір функцій для аналізу
        features = [price_column, 'Volume', 'Returns', 'Volume_Change', 'High_Low_Range', 'MA5', 'MA20']

        # Видалення рядків з NaN значеннями
        df_clean = df.dropna()

        # Нормалізація даних
        scaler = StandardScaler()
        df_normalized = pd.DataFrame(
            scaler.fit_transform(df_clean[features]),
            columns=features,
            index=df_clean.index
        )

        return df_clean, df_normalized, price_column

    def prepare_data_alternative(ticker, period, filter_method='none', filter_params=None):
        """Альтернативний метод отримання даних через генерування тестових даних."""
        print(f"Генерація тестових даних для {ticker}...")

        # Визначаємо кількість днів
        if period == '1mo':
            days = 30
        elif period == '3mo':
            days = 90
        elif period == '6mo':
            days = 180
        elif period == '1y':
            days = 365
        elif period == '2y':
            days = 730
        elif period == '5y':
            days = 1825
        else:
```

```
    days = 365

# Створюємо індекс дат
dates = pd.date_range(end=pd.Timestamp.now(), periods=days, freq='D')

# Генеруємо реалістичні дані акцій
np.random.seed(42)  # Для відтворюваності

# Базова ціна
base_price = 100

# Генеруємо ціни з випадковими прогулянками
returns = np.random.normal(0.001, 0.02, days)  # Середня доходність 0.1% з волатильністю 2%
prices = [base_price]

for ret in returns[1:]:
    new_price = prices[-1] * (1 + ret)
    prices.append(new_price)

prices = np.array(prices)

# Генеруємо High і Low
high_prices = prices * (1 + np.abs(np.random.normal(0, 0.01, days)))
low_prices = prices * (1 - np.abs(np.random.normal(0, 0.01, days)))

# Генеруємо обсяг торгів
base_volume = 1000000
volume = np.random.lognormal(np.log(base_volume), 0.5, days)

# Створюємо DataFrame
df = pd.DataFrame({
    'Open': prices * (1 + np.random.normal(0, 0.005, days)),
    'High': high_prices,
    'Low': low_prices,
    'Close': prices,
    'Volume': volume.astype(int)
}, index=dates)

print(f"Згенеровано {len(df)} записів тестових даних")

# Застосовуємо ту ж обробку
price_column = 'Close'
if filter_method != 'none' and filter_params is not None:
    df, filtered_column = filter_data(df, filter_method, price_column, **filter_params)
    price_column = filtered_column

df['Returns'] = df[price_column].pct_change()
df['Volume_Change'] = df['Volume'].pct_change()
df['High_Low_Range'] = df['High'] - df['Low']
df['MA5'] = df[price_column].rolling(window=5).mean()
df['MA20'] = df[price_column].rolling(window=20).mean()

features = [price_column, 'Volume', 'Returns', 'Volume_Change', 'High_Low_Range', 'MA5', 'MA20']
df_clean = df.dropna()

scaler = StandardScaler()
df_normalized = pd.DataFrame(
    scaler.fit_transform(df_clean[features]),
    columns=features,
    index=df_clean.index
```

```
    )

    return df_clean, df_normalized, price_column

def filter_data(df, method, column, **params):
    """Застосовує фільтрацію до даних."""
    if method == 'ma':
        window_size = params.get('window_size', 5)
        filtered_col = f"{column}_ma"
        df[filtered_col] = df[column].rolling(window=window_size).mean()
        return df, filtered_col
    elif method == 'exp':
        alpha = params.get('alpha', 0.3)
        filtered_col = f"{column}_exp"
        df[filtered_col] = df[column].ewm(alpha=alpha).mean()
        return df, filtered_col
    elif method == 'savgol':
        from scipy.signal import savgol_filter
        window_length = params.get('window_length', 11)
        polyorder = params.get('polyorder', 3)
        filtered_col = f"{column}_savgol"
        df[filtered_col] = savgol_filter(df[column], window_length, polyorder)
        return df, filtered_col
    else:
        return df, column

def cache_data(ticker, period, df):
    """Зберігає дані у локальний кеш."""
    cache_dir = "data_cache"
    if not os.path.exists(cache_dir):
        os.makedirs(cache_dir)

    cache_file = os.path.join(cache_dir, f"{ticker}_{period}.pkl")

    with open(cache_file, 'wb') as f:
        pickle.dump(df, f)

    print(f"Дані збережено в кеш: {cache_file}")

def load_cached_data(ticker, period, max_age_hours=24):
    """Завантажує дані з локального кешу."""
    cache_dir = "data_cache"
    cache_file = os.path.join(cache_dir, f"{ticker}_{period}.pkl")

    if not os.path.exists(cache_file):
        return None

    # Перевіряємо вік файлу
    file_age = datetime.now() - datetime.fromtimestamp(os.path.getmtime(cache_file))
    if file_age > timedelta(hours=max_age_hours):
        print("Кешовані дані застарілі")
        return None

    try:
        with open(cache_file, 'rb') as f:
            df = pickle.load(f)
        print(f"Дані завантажено з кешу: {cache_file}")
        return df
    except Exception as e:
        print(f"Помилка завантаження з кешу: {e}")
```

```
        return None

def prepare_data_with_cache(ticker, period, filter_method='none', filter_params=None):
    """Підготовка даних з використанням кешування."""
    # Спочатку намагаємось завантажити з кешу
    cached_df = load_cached_data(ticker, period)

    if cached_df is not None:
        print("Використовуємо кешовані дані")
        df = cached_df
    else:
        # Якщо немає кешованих даних, завантажуємо нові
        df_clean, df_normalized, price_column = prepare_data(ticker, period, filter_method, filter_params)

        # Зберігаємо в кеш
        cache_data(ticker, period, df_clean)

        return df_clean, df_normalized, price_column

    # Обробляємо кешовані дані
    price_column = 'Close'
    if filter_method != 'none' and filter_params is not None:
        df, filtered_column = filter_data(df, filter_method, price_column, **filter_params)
        price_column = filtered_column

    df['Returns'] = df[price_column].pct_change()
    df['Volume_Change'] = df['Volume'].pct_change()
    df['High_Low_Range'] = df['High'] - df['Low']
    df['MA5'] = df[price_column].rolling(window=5).mean()
    df['MA20'] = df[price_column].rolling(window=20).mean()

    features = [price_column, 'Volume', 'Returns', 'Volume_Change', 'High_Low_Range', 'MA5', 'MA20']
    df_clean = df.dropna()

    scaler = StandardScaler()
    df_normalized = pd.DataFrame(
        scaler.fit_transform(df_clean[features]),
        columns=features,
        index=df_clean.index
    )

    return df_clean, df_normalized, price_column

# ================================================================================
# ФАЙЛ: auto_parameters.py
# ================================================================================

from sklearn.ensemble import IsolationForest
from sklearn.metrics import silhouette_score
from itertools import product
import numpy as np

def find_optimal_parameters(df_normalized, param_ranges=None):
    """
    Знаходить оптимальні параметри для Isolation Forest.
    """
    if param_ranges is None:
        param_ranges = {
            'contamination': [0.05, 0.1, 0.15, 0.2],
            'n_estimators': [50, 100, 200],
```

```
        'max_samples': ['auto', 256, 512],
        'max_features': [0.5, 0.8, 1.0]
    }

    best_score = -1
    best_params = None
    best_model = None

    # Генеруємо всі комбінації параметрів
    param_combinations = list(product(
        param_ranges['contamination'],
        param_ranges['n_estimators'],
        param_ranges['max_samples'],
        param_ranges['max_features']
    ))

    print(f"Тестування {len(param_combinations)} комбінацій параметрів...")

    for i, (contamination, n_estimators, max_samples, max_features) in enumerate(param_combinations):
        try:
            params = {
                'contamination': contamination,
                'n_estimators': n_estimators,
                'max_samples': max_samples,
                'max_features': max_features,
                'random_state': 42
            }

            model = IsolationForest(**params)
            model.fit(df_normalized)
            labels = model.predict(df_normalized)

            # Розраховуємо Silhouette Score
            if len(np.unique(labels)) > 1:  # Перевіряємо, що є більше одного кластера
                score = silhouette_score(df_normalized, labels)
            else:
                score = -1  # Погана оцінка, якщо всі точки в одному кластері

            if score > best_score:
                best_score = score
                best_params = params.copy()
                best_model = model

            if (i + 1) % 10 == 0 or (i + 1) == len(param_combinations):
                print(f"Прогрес: {i + 1}/{len(param_combinations)} комбінацій")

        except Exception as e:
            print(f"Помилка з параметрами {params}: {e}")
            continue

    print(f"Найкращий Silhouette Score: {best_score:.4f}")
    print(f"Оптимальні параметри: {best_params}")

    return best_params, best_model, best_score

# ================================================================================
# ФАЙЛ: anomaly_detection.py
# ================================================================================

from sklearn.ensemble import IsolationForest
```

```
import numpy as np

def detect_anomalies(df_clean, df_normalized, auto_params=True, contamination=None,
                     manual_params=None, param_ranges=None):
    """
    Виявлення аномалій з налаштовуваними або автоматично підібраними параметрами.
    """
    # Підготовка результатів
    optimized_params = {}
    silhouette_score = None
    model = None

    if auto_params:
        # Використовуємо автоматичний підбір параметрів
        print("Виконується автоматичний підбір параметрів...")

        # Якщо передано діапазон параметрів, використовуємо його
        if param_ranges:
            optimized_params, model, silhouette_score = find_optimal_parameters(
                df_normalized,
                param_ranges=param_ranges
            )
        else:
            # Інакше використовуємо стандартні діапазони
            optimized_params, model, silhouette_score = find_optimal_parameters(df_normalized)

        # Використовуємо оптимальну модель
        df_clean['anomaly'] = model.predict(df_normalized)

        # Розрахунок оцінок аномальності з оптимальної моделі
        anomaly_scores = -model.score_samples(df_normalized)

    else:
        # Використовуємо ручні параметри
        if manual_params:
            # Використовуємо передані параметри
            iso_forest = IsolationForest(**manual_params)
            optimized_params = manual_params
        else:
            # Для зворотної сумісності
            contamination = 0.1 if contamination is None else contamination

            # Застосування Isolation Forest з заданими параметрами
            iso_forest = IsolationForest(
                contamination=contamination,
                random_state=42,
                n_estimators=100
            )

            # Зберігаємо використані параметри
            optimized_params = {
                'contamination': contamination,
                'n_estimators': 100,
                'max_samples': 'auto',
                'max_features': 1.0,
                'random_state': 42
            }

        # Передбачення аномалій
        df_clean['anomaly'] = iso_forest.fit_predict(df_normalized)
```

```
        # Розрахунок оцінок аномальності з ручної моделі
        anomaly_scores = -iso_forest.score_samples(df_normalized)

    # Конвертуємо прогнози (-1 для аномалій, 1 для нормальних) до стандартного формату
    df_clean['anomaly'] = df_clean['anomaly'].map({1: 0, -1: -1})

    # ВАЖЛИВО: Зберігаємо оцінки аномальності ПЕРЕД фільтрацією
    df_clean['anomaly_score'] = anomaly_scores

    # Виділення аномалій ПІСЛЯ збереження оцінок
    anomalies = df_clean[df_clean['anomaly'] == -1].copy()

    # Розрахунок відсотка аномалій
    anomaly_percentage = (len(anomalies) / len(df_clean)) * 100

    print(f"Виявлено {len(anomalies)} аномалій ({anomaly_percentage:.2f}%)")
    print(f"Діапазон оцінок аномальності: {anomaly_scores.min():.4f} до {anomaly_scores.max():.4f}")

    return anomalies, anomaly_percentage, optimized_params, silhouette_score

# ================================================================================
# ФАЙЛ: report_generation.py
# ================================================================================

def generate_report(df_clean, anomalies, optimized_params=None, silhouette_score=None, filter_info=None):
    """Генерує детальний звіт про виявлені аномалії."""
    report = "Звіт про виявлені аномалії\n\n"

    # Додаємо інформацію про фільтрацію
    if filter_info and filter_info['method'] != "none":
        report += "Застосовано фільтрацію даних:\n"
        report += f"  Метод фільтрації: {filter_info['method']}\n"

        if filter_info['params']:
            report += "  Параметри фільтрації:\n"
            for param, value in filter_info['params'].items():
                report += f"    {param}: {value}\n"

        report += "\n"

    # Додаємо інформацію про параметри моделі
    if optimized_params:
        report += "Параметри моделі Isolation Forest:\n"
        for param, value in optimized_params.items():
            report += f"  {param}: {value}\n"

        if silhouette_score is not None:
            report += f"  Silhouette Score: {silhouette_score:.4f}\n"

        report += "\n"

    # Загальна статистика
    total_days = len(df_clean)
    anomaly_count = len(anomalies)
    anomaly_percentage = (anomaly_count / total_days) * 100

    report += f"Загальна кількість днів аналізу: {total_days}\n"
    report += f"Кількість виявлених аномалій: {anomaly_count}\n"
    report += f"Відсоток аномальних днів: {anomaly_percentage:.2f}%\n\n"
```

```
    # Опис виявлених аномалій
    report += "Детальний опис аномалій:\n\n"

    price_column = 'Close'
    returns_column = 'Returns'

    for index, row in anomalies.iterrows():
        report += f"Дата: {index.date()}\n"
        report += f"  Ціна закриття: {row[price_column]:.2f}\n"
        report += f"  Обсяг торгів: {row['Volume']}\n"
        report += f"  Прибутковість: {row[returns_column]:.4f}\n"

        # Додаємо оцінку аномальності, якщо вона доступна
        if 'anomaly_score' in row:
            report += f"  Оцінка аномальності: {row['anomaly_score']:.4f}\n"

        # Аналіз причин аномалії
        fraud_probability = 0.0
        if row[returns_column] > df_clean[returns_column].quantile(0.95):
            report += "  Можлива причина: Надзвичайно висока прибутковість\n"
            fraud_probability += 0.3
        elif row[returns_column] < df_clean[returns_column].quantile(0.05):
            report += "  Можлива причина: Надзвичайно низька прибутковість\n"
            fraud_probability += 0.3

        if row['Volume'] > df_clean['Volume'].quantile(0.95):
            report += "  Можлива причина: Надзвичайно високий обсяг торгів\n"
            fraud_probability += 0.2
        elif row['Volume'] < df_clean['Volume'].quantile(0.05):
            report += "  Можлива причина: Надзвичайно низький обсяг торгів\n"
            fraud_probability += 0.1

        # Обмежуємо максимальну ймовірність шахрайства до 100%
        fraud_probability = min(fraud_probability, 1.0)

        report += f"  Оцінка ймовірності ризику: {fraud_probability:.1%}\n\n"

    # Загальні висновки
    report += "Загальні висновки:\n"
    if anomaly_percentage > 10:
        report += "Виявлено значну кількість аномалій. Рекомендується провести додатковий аналіз.\n"
    elif anomaly_percentage > 5:
        report += "Виявлено помірну кількість аномалій. Рекомендується моніторити ситуацію.\n"
    else:
        report += "Виявлено невелику кількість аномалій. Ситуація виглядає стабільною.\n"

    return report

# ================================================================================
# ФАЙЛ: main_app.py (ГОЛОВНИЙ ФАЙЛ)
# ================================================================================

import tkinter as tk
from tkinter import ttk, scrolledtext, messagebox
import matplotlib.pyplot as plt
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import numpy as np
import pandas as pd
```

```
import seaborn as sns

class SimpleAnomalyDetectionApp:
    def __init__(self, master):
        self.master = master
        master.title("Система виявлення аномалій у фінансових даних")
        master.geometry("1200x800")
        master.configure(bg='#e6f3ff')  # Світло-блакитний фон вікна

        # Налаштовуємо стилі TTK
        self.setup_styles()

        # Створюємо основні елементи інтерфейсу
        self.create_widgets()

    def setup_styles(self):
        """Налаштовує стилі для TTK віджетів у синьо-блакитних тонах"""
        style = ttk.Style()
        style.theme_use('clam')

        # Загальні стилі з синьо-блакитною схемою
        style.configure("TFrame",
                        background='#e6f3ff',  # Світло-блакитний
                        relief='flat')

        style.configure("TLabel",
                        background='#e6f3ff',
                        foreground='#1e3a5f',  # Темно-синій текст
                        font=('Segoe UI', 10))

        # Стилі для заголовків
        style.configure("Title.TLabel",
                        background='#e6f3ff',
                        foreground='#1e3a5f',
                        font=('Segoe UI', 16, 'bold'))

        # Стилі для LabelFrame (панелі параметрів)
        style.configure("TLabelframe",
                        background='#cce7ff',  # Блакитний фон
                        foreground='#1e3a5f',  # Темно-синій текст
                        borderwidth=2,
                        relief='ridge')

        style.configure("TLabelframe.Label",
                        background='#cce7ff',
                        foreground='#1e3a5f',
                        font=('Segoe UI', 11, 'bold'))

        # Стилі для кнопок
        style.configure("Action.TButton",
                        background='#4a90e2',  # Синій
                        foreground='white',
                        font=('Segoe UI', 11, 'bold'),
                        borderwidth=2,
                        relief='raised',
                        padding=10)

        style.map("Action.TButton",
              background=[('active', '#357abd'),   # Темніше при наведенні
                          ('pressed', '#2e6ba8')],   # Ще темніше при натисканні
```

```
        foreground=[('active', 'white'),
                ('pressed', 'white')])

# Стилі для Combobox
style.configure("TCombobox",
                background='white',
                foreground='#1e3a5f',
                font=('Segoe UI', 10),
                borderwidth=2,
                relief='solid')

style.map("TCombobox",
        fieldbackground=[('readonly', 'white'),
                        ('active', '#f0f8ff')])  # Світло-блакитний при активності

# Стилі для Entry
style.configure("TEntry",
                background='white',
                foreground='#1e3a5f',
                font=('Segoe UI', 10),
                borderwidth=2,
                relief='solid')

style.map("TEntry",
        focuscolor=[('focus', '#4a90e2')])  # Синя рамка при фокусі

# Стилі для вкладок Notebook
style.configure("TNotebook",
                background='#e6f3ff',
                borderwidth=2,
                relief='ridge')

style.configure("TNotebook.Tab",
                background='#b3d9ff',  # Блакитний для неактивних вкладок
                foreground='#1e3a5f',
                font=('Segoe UI', 10, 'bold'),
                padding=[15, 8],
                borderwidth=2,
                relief='raised')

style.map("TNotebook.Tab",
        background=[('selected', '#4a90e2'),  # Синій для активної вкладки
                ('active', '#87ceeb')],     # Світло-блакитний при наведенні
        foreground=[('selected', 'white'),   # Білий текст на активній вкладці
                ('active', '#1e3a5f')])

# Стилі для Treeview (таблиця)
style.configure("Custom.Treeview",
                background='white',
                foreground='#1e3a5f',
                font=('Segoe UI', 10),
                rowheight=25,
                borderwidth=2,
                relief='solid')

# ВАЖЛИВО: Стиль для заголовка таблиці
style.configure("Custom.Treeview.Heading",
                background='#4a90e2',    # Синій фон заголовка
                foreground='white',      # Білий текст заголовка
                font=('Segoe UI', 11, 'bold'),
```

```
                borderwidth=2,
                relief='raised')

    # Підсвічування рядків таблиці
    style.map("Custom.Treeview",
          background=[('selected', '#87ceeb')],  # Блакитний при виділенні
          foreground=[('selected', '#1e3a5f')])

    # Стилі для прогрес-бара
    style.configure("TProgressbar",
                background='#4a90e2',  # Синій прогрес-бар
                borderwidth=2,
                relief='solid')

    # Стилі для Scrollbar
    style.configure("TScrollbar",
                background='#b3d9ff',
                troughcolor='#e6f3ff',
                borderwidth=1,
                relief='solid')

    style.map("TScrollbar",
          background=[('active', '#87ceeb')])

def create_widgets(self):
    # Головний фрейм з блакитним фоном
    main_frame = ttk.Frame(self.master, padding="15", style="TFrame")
    main_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

    # Налаштовуємо розтягування
    self.master.columnconfigure(0, weight=1)
    self.master.rowconfigure(0, weight=1)
    main_frame.columnconfigure(1, weight=1)
    main_frame.rowconfigure(2, weight=1)

    # Заголовок зі стилем
    title_label = ttk.Label(main_frame,
                        text="Система виявлення аномалій у фінансових даних",
                        style="Title.TLabel")
    title_label.grid(row=0, column=0, columnspan=3, pady=(0, 25))

    # Панель параметрів з новим стилем
    params_frame = ttk.LabelFrame(main_frame, text="Параметри аналізу",
                              padding="15", style="TLabelframe")
    params_frame.grid(row=1, column=0, columnspan=3, sticky=(tk.W, tk.E), pady=(0, 15))

    # Тікер
    ttk.Label(params_frame, text="Тікер акції:", style="TLabel").grid(row=0, column=0, padx=(0, 10), sticky=tk.W)
    self.ticker_entry = ttk.Entry(params_frame, width=12, style="TEntry")
    self.ticker_entry.grid(row=0, column=1, padx=(0, 25), sticky=tk.W)
    self.ticker_entry.insert(0, "AAPL")

    # Період
    ttk.Label(params_frame, text="Період:", style="TLabel").grid(row=0, column=2, padx=(0, 10), sticky=tk.W)
    self.period_combo = ttk.Combobox(params_frame, values=['1mo', '3mo', '6mo', '1y', '2y', '5y'],
                                 width=10, state="readonly", style="TCombobox")
    self.period_combo.grid(row=0, column=3, padx=(0, 25), sticky=tk.W)
    self.period_combo.set('1y')

    # Метод фільтрації
```

```
        ttk.Label(params_frame, text="Фільтрація:", style="TLabel").grid(row=0, column=4, padx=(0, 10), sticky=tk.W)
        self.filter_combo = ttk.Combobox(params_frame, values=['none', 'ma', 'exp', 'savgol'],
                                         width=12, state="readonly", style="TCombobox")
        self.filter_combo.grid(row=0, column=5, padx=(0, 25), sticky=tk.W)
        self.filter_combo.set('none')

        # Режим параметрів
        ttk.Label(params_frame, text="Параметри:", style="TLabel").grid(row=1, column=0, padx=(0, 10), sticky=tk.W,
pady=(10, 0))
        self.params_mode = ttk.Combobox(params_frame, values=['auto', 'manual'],
                                        width=12, state="readonly", style="TCombobox")
        self.params_mode.grid(row=1, column=1, padx=(0, 25), sticky=tk.W, pady=(10, 0))
        self.params_mode.set('auto')

        # Кнопка аналізу з красивим стилем
        self.analyze_button = ttk.Button(params_frame, text="🔍 Запустити аналіз",
                                         command=self.run_analysis, style="Action.TButton")
        self.analyze_button.grid(row=0, column=6, padx=20, rowspan=2)

        # Notebook для результатів
        self.notebook = ttk.Notebook(main_frame, style="TNotebook")
        self.notebook.grid(row=2, column=0, columnspan=3, sticky=(tk.W, tk.E, tk.N, tk.S))

        # Вкладка графіків
        self.graph_frame = ttk.Frame(self.notebook, style="TFrame")
        self.notebook.add(self.graph_frame, text="📊 Графіки")

        # Canvas для графіків з прокруткою
        canvas_frame = ttk.Frame(self.graph_frame, style="TFrame")
        canvas_frame.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

        self.graph_canvas = tk.Canvas(canvas_frame, bg='#f0f8ff', highlightthickness=0)  # Світло-блакитний фон
canvas
        scrollbar_y = ttk.Scrollbar(canvas_frame, orient="vertical",
                                    command=self.graph_canvas.yview, style="TScrollbar")
        self.graph_canvas.configure(yscrollcommand=scrollbar_y.set)

        self.graph_canvas.pack(side="left", fill="both", expand=True)
        scrollbar_y.pack(side="right", fill="y")

        # Внутрішній фрейм для графіків
        self.graph_inner_frame = ttk.Frame(self.graph_canvas, style="TFrame")
        self.graph_canvas.create_window((0, 0), window=self.graph_inner_frame, anchor="nw")

        # Вкладка звіту
        self.report_frame = ttk.Frame(self.notebook, style="TFrame")
        self.notebook.add(self.report_frame, text="📋 Звіт")

        # Контейнер для тексту звіту з блакитним фоном
        report_container = ttk.Frame(self.report_frame, style="TFrame")
        report_container.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

        self.report_text = scrolledtext.ScrolledText(report_container, wrap=tk.WORD,
                                                     font=('Consolas', 10),
                                                     bg='#f8fdff',      # Світло-блакитний фон
                                                     fg='#1e3a5f',      # Темно-синій текст
                                                     selectbackground='#87ceeb',  # Блакитне виділення
                                                     selectforeground='#1e3a5f',
                                                     borderwidth=2,
                                                     relief='solid')
```

```
        self.report_text.pack(fill=tk.BOTH, expand=True)

        # Вкладка таблиці
        self.table_frame = ttk.Frame(self.notebook, style="TFrame")
        self.notebook.add(self.table_frame, text="📊 Таблиця аномалій")

        # Контейнер для таблиці
        table_container = ttk.Frame(self.table_frame, style="TFrame")
        table_container.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

        # Створюємо таблицю з кастомним стилем
        columns = ('Дата', 'Ціна', 'Обсяг', 'Доходність', 'Оцінка аномальності')
        self.tree = ttk.Treeview(table_container, columns=columns, show='headings',
                                 style="Custom.Treeview")

        # Налаштовуємо заголовки таблиці
        for col in columns:
            self.tree.heading(col, text=col)
            self.tree.column(col, width=140, anchor='center')

        # Скролбари для таблиці
        tree_scroll_y = ttk.Scrollbar(table_container, orient="vertical",
                                      command=self.tree.yview, style="TScrollbar")
        tree_scroll_x = ttk.Scrollbar(table_container, orient="horizontal",
                                      command=self.tree.xview, style="TScrollbar")
        self.tree.configure(yscrollcommand=tree_scroll_y.set, xscrollcommand=tree_scroll_x.set)

        self.tree.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))
        tree_scroll_y.grid(row=0, column=1, sticky=(tk.N, tk.S))
        tree_scroll_x.grid(row=1, column=0, sticky=(tk.W, tk.E))

        table_container.columnconfigure(0, weight=1)
        table_container.rowconfigure(0, weight=1)

        # Статусна панель з блакитним фоном
        self.status_frame = ttk.Frame(main_frame, style="TFrame")
        self.status_frame.grid(row=3, column=0, columnspan=3, sticky=(tk.W, tk.E), pady=(15, 0))

        # Рамка для статусу
        status_container = ttk.Frame(self.status_frame, style="TFrame", padding="10")
        status_container.pack(fill=tk.X)

        self.status_label = ttk.Label(status_container, text="🟢 Готовий до роботи",
                                      style="TLabel", font=('Segoe UI', 10, 'bold'))
        self.status_label.pack(side=tk.LEFT)

        self.progress_bar = ttk.Progressbar(status_container, mode='indeterminate',
                                            style="TProgressbar", length=200)
        self.progress_bar.pack(side=tk.RIGHT, padx=(10, 0))

        # Bind для оновлення scroll region
        self.graph_inner_frame.bind('<Configure>', self.update_scroll_region)

    def update_scroll_region(self, event):
        self.graph_canvas.configure(scrollregion=self.graph_canvas.bbox("all"))

    def run_analysis(self):
        """Запускає аналіз даних"""
        try:
            # Отримуємо параметри
```

```
ticker = self.ticker_entry.get().strip().upper()
period = self.period_combo.get()
filter_method = self.filter_combo.get()
params_mode = self.params_mode.get()

if not ticker:
    messagebox.showerror("Помилка", "Будь ласка, введіть тікер акції")
    return

# Запускаємо прогрес бар
self.progress_bar.start()
self.analyze_button.config(state='disabled')
self.status_label.config(text=f"Завантажую дані для {ticker}...")
self.master.update()

# Підготовка даних
filter_params = None
if filter_method != 'none':
    filter_params = {'window_size': 5} if filter_method == 'ma' else {'alpha': 0.3}

self.status_label.config(text="Підготовка та фільтрація даних...")
self.master.update()

try:
    df_clean, df_normalized, price_column = prepare_data_with_cache(
        ticker, period, filter_method, filter_params
    )
except Exception as e:
    # Якщо не вдається завантажити реальні дані, використовуємо тестові
    print(f"Помилка завантаження даних: {e}")
    self.status_label.config(text="Використовую тестові дані...")
    self.master.update()
    df_clean, df_normalized, price_column = prepare_data_alternative(
        ticker, period, filter_method, filter_params
    )

if len(df_clean) < 30:
    raise ValueError(f"Недостатньо даних для аналізу. Отримано лише {len(df_clean)} записів.")

self.status_label.config(text="Виявлення аномалій...")
self.master.update()

# Виявлення аномалій
if params_mode == "auto":
    param_ranges = {
        'contamination': [0.05, 0.1, 0.15],
        'n_estimators': [50, 100],
        'max_samples': ['auto', 256],
        'max_features': [0.8, 1.0]
    }
    anomalies, anomaly_percentage, optimized_params, silhouette_score = detect_anomalies(
        df_clean, df_normalized, auto_params=True, param_ranges=param_ranges
    )
```

ДОДАТОК Г

ІЛЮСТРАТИВНА ЧАСТИНА

«Програмний модуль для пошуку аномалій у фінансових звітах»

Виконав: студентки 4 курсу, групи 2КН-21б
Спеціальності 122 – Комп'ютерні науки
(шифр і назва напрямку підготовки, спеціальності)
Анастасія ЛАНОВИК
(прізвище та ініціали)
Керівник: д.т.н., проф. каф. КН
Ярослав ІВАНЧУК
(прізвище та ініціали)

«03» червня 2025 р.

Вінниця ВНТУ – 2025 рік

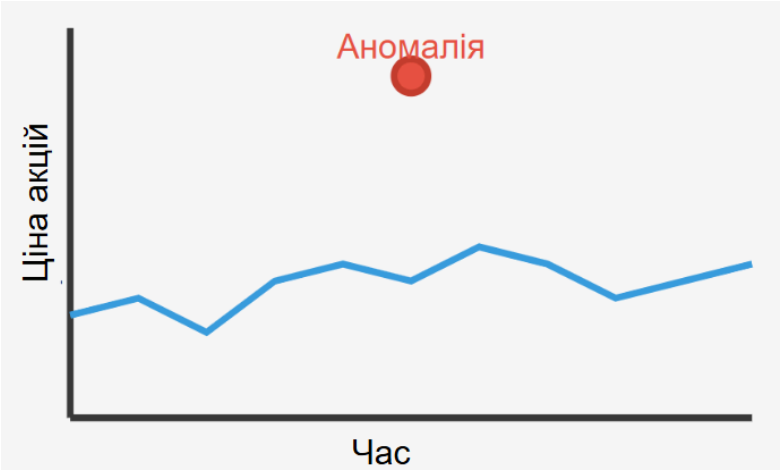


Рисунок Г.1 — Візуальне зображення точкової аномалії

Рисунок Г.2 — Візуальне зображення контекстуальних аномалій

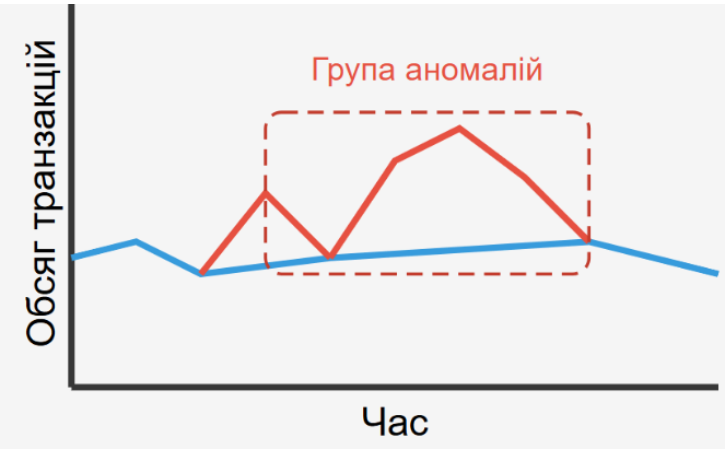


Рисунок Г.3 — Візуальне зображення колективних (групових) аномалій

Метод	Точніть	Шидкість	Інтерпретованість	Простота	Загальна оцінка
Isolation Forest	5	5	5	4	5
One-Class SVM	4	3	4	3	4
Автоенкодери	5	2	2	2	4
LSTM-мережі	5	1	2	1	3
Random Foprest	4	4	5	3	4

Рисунок Г.4 - Порівняльний аналіз методів пошуку аномалій у фінансових звітах

Рисунок Г.5 - Загальний алгоритм для пошуку аномалій в фінансових звітах

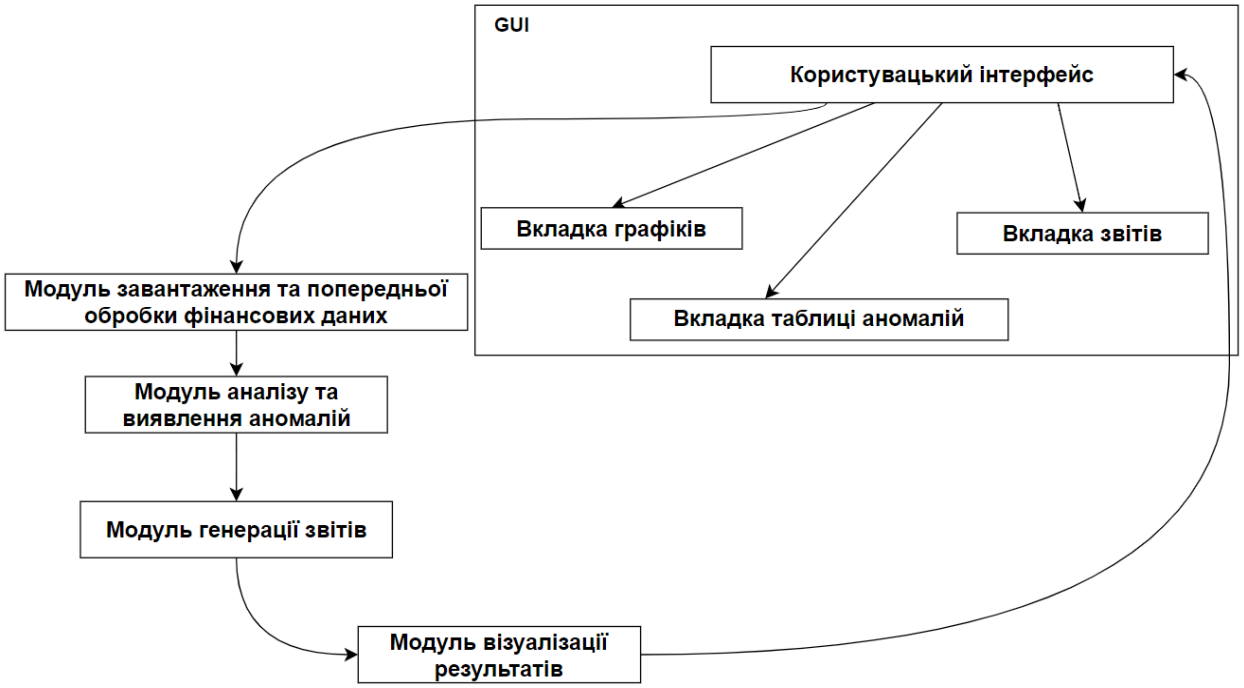


Рисунок Г.6 — Структурна схема програмного забезпечення

Рисунок Г.7 - Блок-схема модуля завантаження та попередньої обробки

Рисунок Г.8 - Блок-схема модуля аналізу та виявлення аномалій

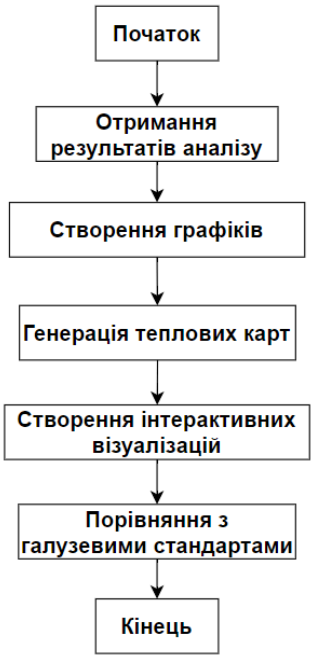


Рисунок Г.9 - Блок-схема модуля візуалізації результатів виявлення аномалій у фінансових даних

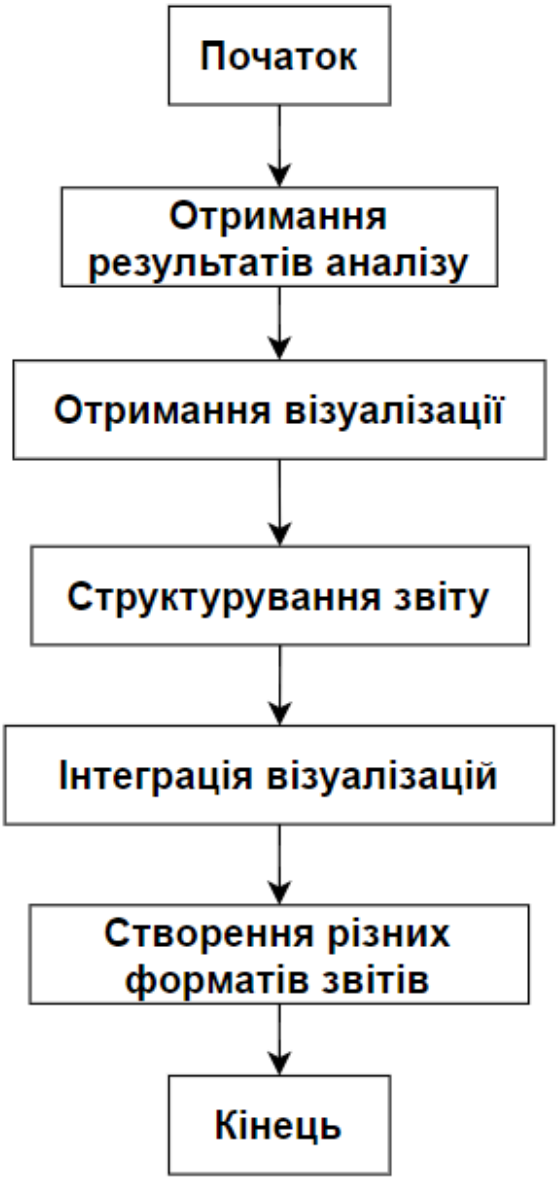


Рисунок Г.10 — Блок-схема функціонування модуля генерації звітів на основі виявлених аномалій в фінансових даних

Рисунок Г.11 — Блок-схема алгоритму Isolation Forest

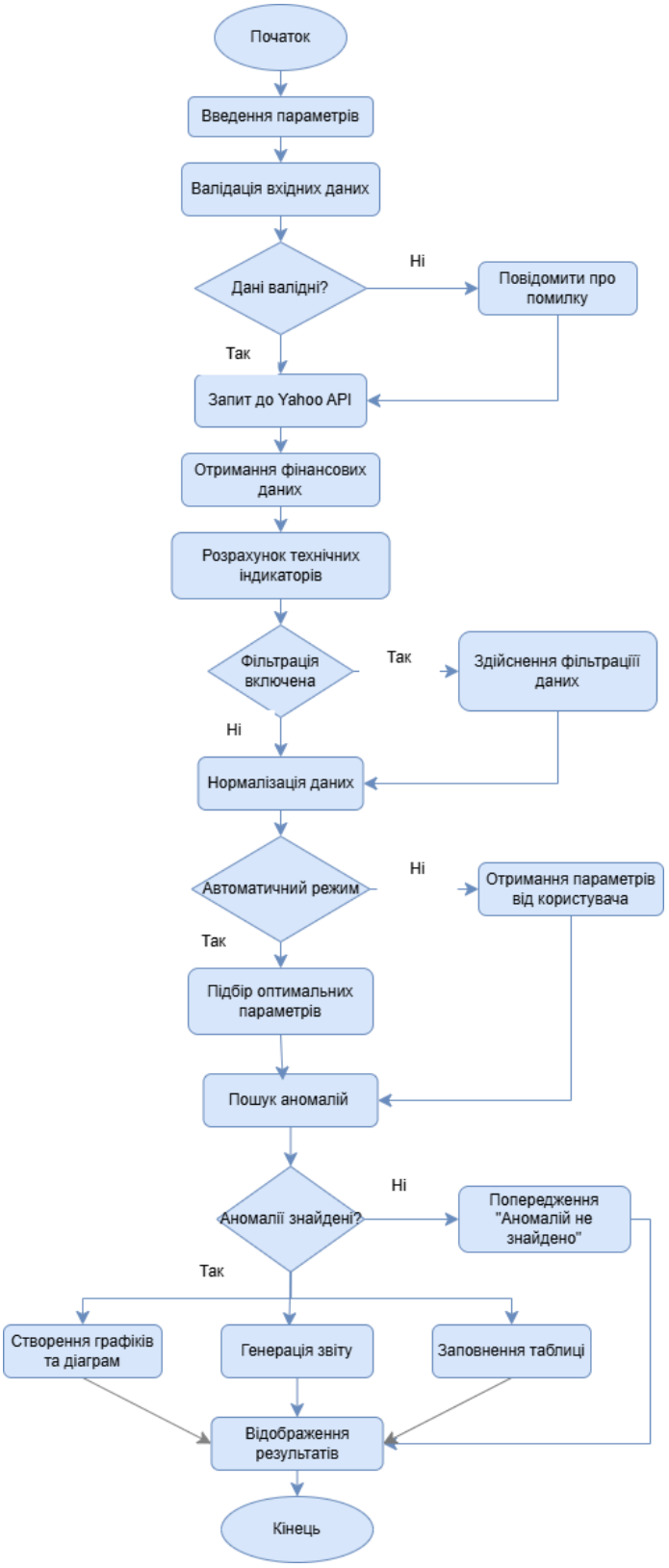


Рисунок Г.12 — Алгоритм програмного модуля для пошуку аномалій в фінансових звітах

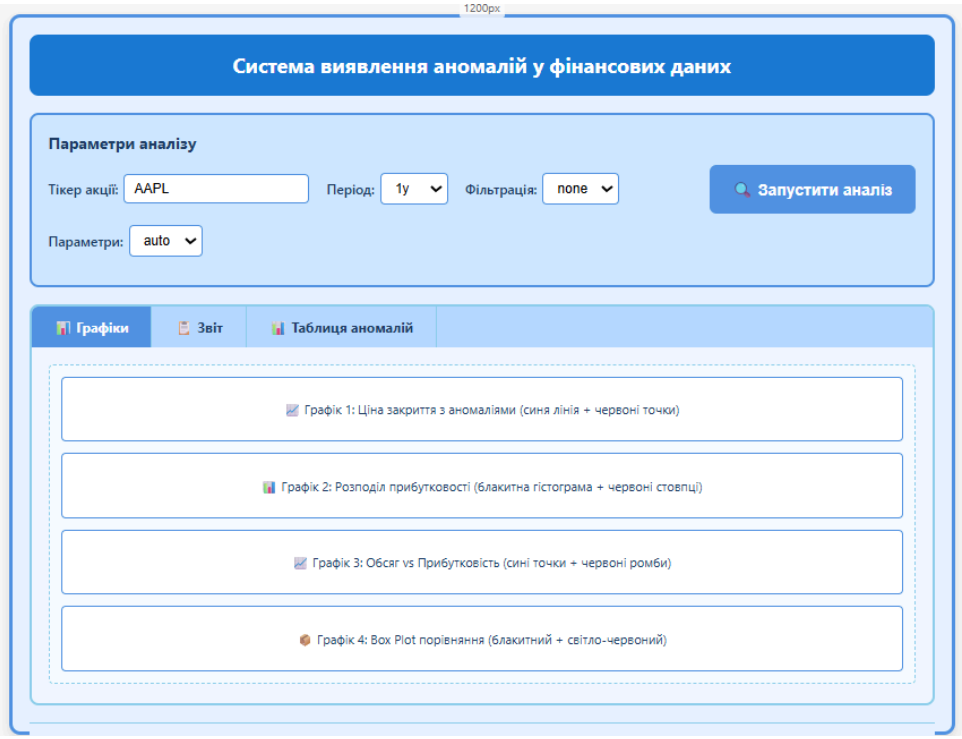


Рисунок Г.13 — Схема інтерфейсу користувача системи виявлення аномалій

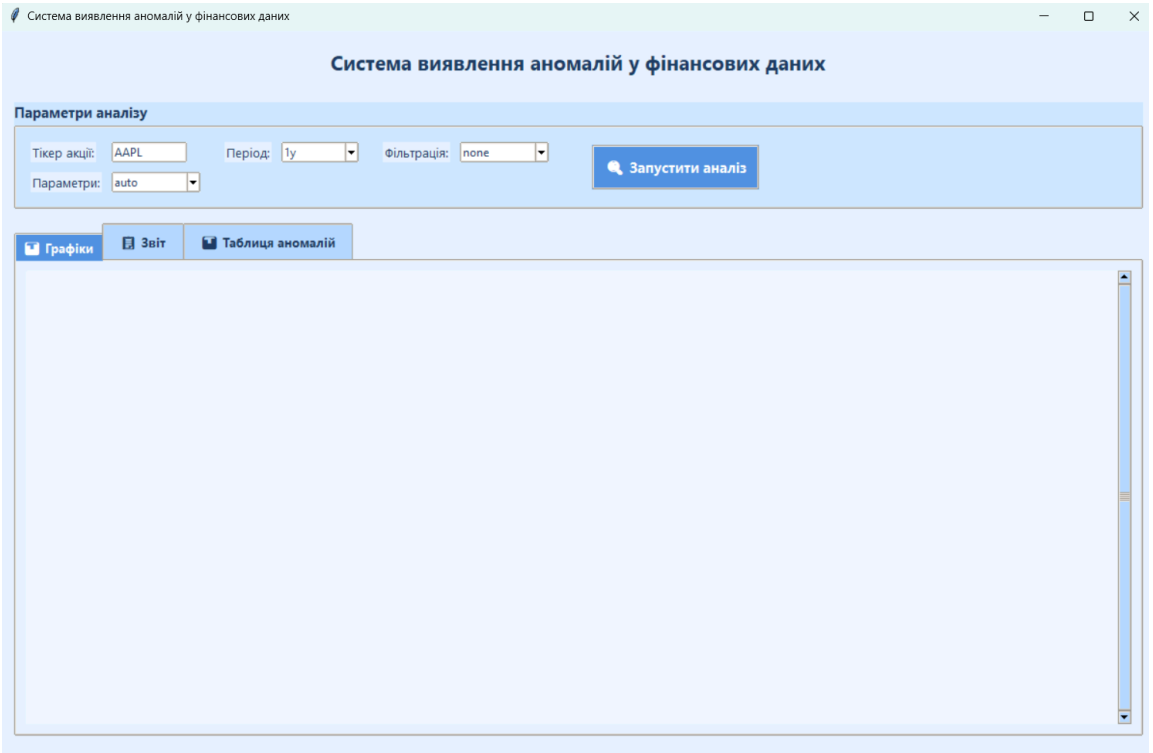


Рисунок Г.14 — Вікно програми аналізу фінансових звітів

Рисунок Г.15 — Вкладка графіків програми аналізу фінансових звітів

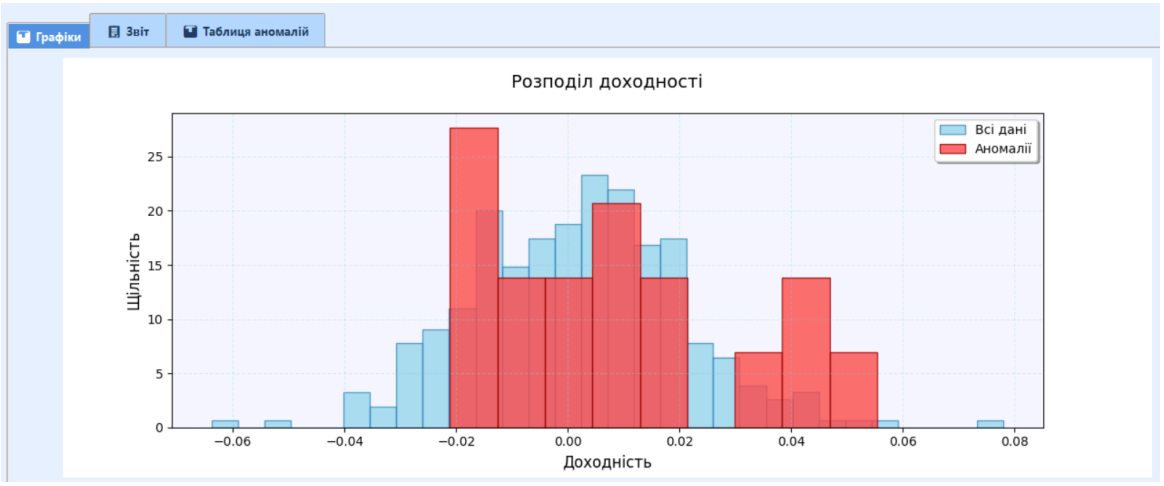


Рисунок Г.16 — Гістограма розподілу денної прибутковості