УДК 004.41
DOI: 10.18523/2617-3808.2025.8.158-166

Постніков М. А., Гороховський С. С.

АВТОМАТИЗОВАНА СИСТЕМА ВИЯВЛЕННЯ АНОМАЛІЙ У БІЗНЕС-ДАНИХ

У статті описано проведений аналіз процесу виявлення аномалій у бізнес-даних, відомі програмні рішення, сформульовано вимоги до системи та описано розроблену автоматизовану програмну систему виявлення аномалій. Ця система складається з програмних модулів, має високу адаптивність, є легкою до модифікації і зручною у використанні.

Розроблена система повністю відповідає поставленим раніше вимогам: легкість у налаштуванні забезпечується інтерфейсом користувача і інтерактивним процесом, гнучкість і легкість кастомізації — обраними технологіями та архітектурними абстракціями, надійність — розділенням компонентів через чергу задач, функціональні вимоги — розробленими складовими модулями. Система виконує поставлену задачу автоматизованого виявлення аномалій у бізнес-даних і відповідає сучасним стандартам у галузі даних.

Ключові слова: виявлення аномалій, програмна система, вимоги до системи, архітектура, технології, вебдодаток, FastAPI, Celery, REST API, Pandas.

Вступ

«Дані — нафта XXI століття» — ці слова британського математика Кліва Гамбі у 2006 р. [1] (тут і далі переклад наш) ознаменували головну характеристику сучасного цифрового світу. Дані сьогодні є найціннішим і затребуваним ресурсом. Однак дані не мають прямої користі в тому вигляді, в якому вони є, — для застосування вони потребують оброблення і інтерпретації [1]. Великий обсяг даних потребує автоматизованого оброблення.

Таким чином, для забезпечення своєї діяльності компанії мусять базувати свою аналітику на даних, що стає дедалі важчим завданням через постійне зростання їхньої варіативності та об'єму. Головними проблемами тут є надійність даних (data reliability) та швидка ідентифікація зміни трендів. Проблема надійності даних є наріжною в індустрії, оскільки сучасні бізнеси оперують даними, що походять із різноманітних джерел (бухгалтерські системи, системи звітності, продажів, CRM системи, дані активності соціальних мереж, дані систем обліку співробітників тощо.). Такі дані часто непридатні до використання одразу, оскільки містять помилки та неточності. Їх знаходження є класичною проблемою аналізу даних. Також бізнес стикається з потребою швидко ідентифікувати незвичну поведінку даних, що є індикатором певних суттєвих змін, які вносять корективи до бізнес-процесів. Вирізняти такі випадки серед великих обсягів даних швидко — складна задача, що важко розв'язується за допомогою звичних практик аналізу даних.

Тому багато бізнесів звертаються до техніки автоматизації виявлення аномалій (outlier detection), що допомагає визначати одиниці даних, які не дотримуються патерну поведінки інших даних. Ці методи широко застосовують для вирішення проблем на кшталт оптимізації витрат, виявлення випадків шахрайства тощо.

1. Аналіз досліджень про виявлення аномалій

Станом на сьогодні можна сказати, що виявлення аномалій (outlier detection, також anomaly detection) — це усталена галузь науки про дані (data science), що досліджує підходи і методи вирішення проблеми визначення аномалій у даних. Однією з перших відомих наукових робіт на цю тему вважають статтю викладача Королівської школи Лондона Ф. Еджворс (Francis Edgeworth) «On discordant observations» [4], де автор визначає дискордантні спостереження як ті, що «...справляють враження відмінності від інших спостережень відносно до закону, за яким розподілені їхні частоти». З плином часу дослідженням цієї проблеми приділяли дедалі більше уваги.

Перші ґрунтовні напрацювання щодо визначення аномалій зробили дослідники в галузі статистики. Так, класичне визначення аномалії наводить Ф. Ґрабс (Frank Grubbs): «Аномальним спостереженням, або "аномалією", є те, що помітно виділяється серед інших членів вибірки, де воно зустрічається» [9, p. 1]. Автор виділяє два типи аномальних спостережень: ті, «...що є серйозним свідченням про варіативність даних випадкового характеру...» [9, p. 1], і ті, «...що є результатом серйозних відхилень від процедури [статистичного] експерименту або помилки в підрахунках чи записах числового значення» [9, p. 1]. Дослідник наводить опис розроблених ним критеріїв визначення аномалій у статистичних даних і зазначає, що «...майже всі [розроблені] критерії для аномалій ґрунтуються на передбаченні про нормальний (Ґаусовий) розподіл або популяцію, що лежить в основі [даних]» [9, p. 3]. Додатково автор підкреслює, що знаходження аномалій є насамперед індикатором потреби подальшого дослідження і виявлення причини їхньої появи, оскільки наявність аномалій не обов'язково означає, що такі спостереження потребують особливого поводження чи мають бути відкинуті [9, pp. 20–21].

Ґрунтовним дослідженням визначення аномалій у статистичних даних стала монографія Д. Гокінза (Douglas Hawkins) «Identification of Outliers» [10]. За Гокінзом, аномалія — це «...спостереження, що настільки сильно відхиляється від інших спостережень, що виникає підозра, що воно було згенероване іншим механізмом» (тут і далі переклад наш) [10, p. 1]. Автор виділяє два основні механізми походження аномалій: коли дані походять від розподілу з «важкими хвостами» (heavy-tailed distributions) і коли дані походять від двох розподілів, один з яких генерує нормальні дані, а інший — аномалії [10, pp. 1–2].

Найбільш значущою роботою останніх років у сфері стало дослідження науковців з Університету Міннесоти В. Чандола (Varun Chandola), А. Банерджі (Arindam Banerjee) і В. Кумара (Vipin Kumar), опубліковане в 2009 р. [3], де було проаналізовано понад сотню досліджень і підсумовано наявні звершення в цій галузі. Так, дослідники дають визначення аномаліям як «...патернам в даних, що не підкорюються поняттю про нормальну поведінку» [3, p. 2].

Через те, що аномалії є широким поняттям, техніки виявлення аномалій мають різноманітне практичне застосування: *виявлення шкідливої поведінки, виявлення дефектів, виявлення новизни (novelty detection).* Загалом, виявлення аномалій може бути застосоване у будь-якій галузі, де є потреба відстежувати зміни в даних. Особливо корисною вона буде в тих випадках, де на аналіз усіх потрібних даних бракує ресурсів.

Проведене нами дослідження показало, що серед відкритого програмного забезпечення на сьогодні немає системи, яка могла б задовольнити потребу бізнесу у швидкому виявленні аномалій у найбільш поширеній формі бізнес-даних, а саме — в часових рядах. Ця проблема і стала поштовхом для нашого дослідження. Потрібно було на основі досвіду наявних розробок сформулювати вимоги до програмної системи (ПС), провести проєктування архітектури і вибір технологій і розробити систему автоматизованого виявлення аномалій у бізнес-даних (часових рядах).

Виявлення аномалій може бути «на сторожі» певних важливих параметрів, сповіщаючи лише тоді, коли справді потрібна додаткова увага. Ключовим є те, що виявлення аномалій не є інтерпретацією даних: аномальні дані не завжди є чимось добрим або поганим, вони не завжди означають, що щось пішло не так, як планувалось. Вони дають знати про критичну зміну в поведінці певних показників. Як і з будь-яким моделюванням, аномалії не є стовідсотково точними: так, можливість хибнопозитивних (false-positive) чи хибнонегативних (false-negative) результатів залежить від якості вхідних даних, їхньої попередньої обробки, вибраної моделі і її параметрів, варіативності і обсягу вибірки.

Нині відомо багато різних технік виявлення аномалій. Вони різняться передусім за рахунок предметної сфери застосування, характеру вхідних даних, бажаної точності передбачень та об'єму зусиль на впровадження. Основними методами серед них є [3, p. 3] статистичні, гістограмні, класифікаційні, кластеризаційні, інтервальні. Наприклад, гістограмний метод полягає в такому: за допомогою гістограми будується частотний профіль даних (тільки аномальних або тільки нормальних), його використовують для порівняння з гістограмою тих даних, які ми хочемо аналізувати, і оцінка аномальності випливає з різниці частотних профілів даних [3, p. 37; 8, p. 4]. Ядерна оцінка щільності (kernel density) використовується дуже схожим чином; можна сказати, що цей метод є гістограмним, який використовує згладжування, за рахунок чого вирішується проблема впливу кількості стовпчиків гістограми на дані [3, p. 38; 8, p. 4].

3. Вимоги до програмної системи

Для бізнесу сьогодні ефективне використання даних розглядається не тільки як конкурентна перевага, а і як необхідність для виживання [2]. Автори дослідження говорять про те, що інтерпретація даних для отримання з них користі є основною задачею бізнесу, і це зробити зовсім непросто. Зокрема, одною з проблем тут є якість даних (поняття «якість даних» тут і далі використовується як для позначення власне якості даних, англ. data quality, так і цілісності даних, англ. data integrity, та інших термінів, що мають під собою здатність даних бути використаними для отримання правильних висновків). За даними авторів дослідження, 71 % бізнесу стикається з проблемами якості даних, і вони називають це «бар'єром для досягнення цілі (*отримання користі з*)» [2].

Велика частина цих даних належить до *часових рядів* (time series data). За нашим досвідом, більшість потреб бізнесу у даних задовольняють саме такі дані: вони представляють зміну певних показників із часом, наприклад, це може бути кількість продажів, дохід, кількість активних користувачів, маркетингові показники, такі як retention rate (частка користувачів, які продовжують користуватись продуктом після певного часу). Цих даних на сьогодні набагато більше, ніж можливостей у більшості бізнесів для їхнього аналізу. Дослідники з SRM Institute of Science and Technology та VIT Bhopal University виділяють такі характеристики часових рядів: тренд, сезонність, циклічність і випадкову компоненту (шум), і наголошують, що аналіз часових рядів дозволяє бізнесу покращити процес прийняття рішень за рахунок кращого розуміння трендів на ринку [11].

Отже, особливості бізнес-даних можна сформулювати так: *необхідність визначати незвичну інформацію серед великої кількості даних, представлених у вигляді часових рядів.* Тобто метою ПС буде *надання можливості легкого визначення аномалій у часових рядах*. Система має бути інструментом, за допомогою якого можна *швидко* і *легко* налаштувати відслідковування аномалій у таких даних.

Для аналізу були відібрані ті ПС, які декларують про вирішення задачі відстежування аномалій: *ChaosGenius* [6], *CueObserve* [7], *Elementary Data* [8].

Серед особливостей усіх цих систем можна виділити такі: підтримка багатьох баз і сховищ даних (Postgres, Google BigQuery, Amazon Redshift); підтримка інтеграцій із різними системами для відправки сповіщень (Slack, електронна пошта); автоматизоване виявлення аномалій, яке запускається за розкладом; наявність вебінтерфейсу для перегляду інформації і налаштувань; підтримка різних моделей визначення аномалій; наявність налаштувань для моделей визначення аномалій.

Серед недоліків цих систем називають такі. Жодна з основних систем більше не підтримується станом на час проведення дослідження. Налаштування систем є заплутаним і неочевидним, зокрема в процесі налаштування бракує інтерактивності і можливості подивитись, як налаштування впливають на результат. У системах мала кількість підтримуваних моделей визначення аномалій.

Однак головними недоліками цих систем є їхня непристосованість до потреб бізнесу. Ці інструменти не дають можливості швидко і інтерактивно налаштувати відстеження аномалій, їхні налаштування не інтуїтивні і не дозволяють швидко розібратись в системі або в тому, як найкраще налаштувати параметри визначення, враховуючи особливості тих чи інших даних. Бізнес потребує інструменту, який дозволяв би за короткий час налаштувати відстеження аномалій у часових рядах, які періодично оновлюються. Це дозволило б бізнесу без особливих витрат часу і грошей стежити за багатьма показниками.

Тому ми визначили основні вимоги до нашої ПС: легке і швидке визначення аномалій у часових рядах за допомогою інтерактивних налаштувань; легкість у налаштуванні з використанням вебінтерфейсу; гнучкість і легкість кастомізації системи; надійність. Оскільки бізнес буде покладатись на таку систему у своїй діяльності, система має мати високі показники стійкості до помилок і відмов і мати змогу обробляти великі масиви даних. Передусім це має бути забезпечено архітектурними рішеннями: наприклад, використання черг задач.

4. Програмна реалізація системи

Як мову програмування ми обрали мову Python. В індустрії оброблення даних ця мова є найпопулярнішою і стандартом де-факто галузі, що дозволить користувачам за потреби легко модифікувати систему.

Для реалізації серверної частини обрали бібліотеку FastAPI: це популярний асинхронний вебфреймворк, який почали розробляти в 2018 р. Фреймворк повністю асинхронний і базується на

typehints — оголошеннях про типи в Python [5]. Підтримка typehints реалізована через бібліотеку Pydantic, що є найпоширенішою бібліотекою для валідації даних у Python. Асинхронність FastAPI дозволила підвищити продуктивність роботи вебсервера за рахунок зменшення процесорного часу, що в синхронному режимі витрачається на затратні I/O операції.

FastAPI часто називають мікрофреймворком (micro framework) за аналогією з Flask через те, що він надає «з коробки» небагато можливостей, а весь додатковий функціонал потрібно реалізовувати окремо через під'єднання сторонніх бібліотек або власноруч написану логіку. Разом з фреймворком були використані такі бібліотеки: Psycopg3 як рушій для спілкування з БД Postgres, SQLAlchemy як ORM для СКБД Postgres, Jinja2 як рушій серверного рендерингу, Alembic для міграцій БД.

Для маніпуляції даними ми обрали бібліотеку Pandas, вона є найбільш популярною такою бібліотекою на Python і також є стандартом де-факто у галузі даних, що також дозволить легко робити зміни в системі. Дуже багато людей критикують Pandas за неінтуїтивність і нелогічність API, відсутність розподіленості і порівняно низьку швидкість роботи, але для роботи з помірним розміром даних цього цілком достатньо, тому вибір було зроблено на користь цієї бібліотеки, а не її конкурентів на кшталт Polars, які ще не набули такої популярності.

Правильним є відокремити роботу вебсервера від роботи з даними (запуску перевірок), тому ці компоненти мають якимось чином спілкуватись між собою. Для цього ми обрали Celery — чергу задач на Python, яка дозволяє легко передавати задачі на виконання іншим компонентам, зручно стежити за їхнім виконанням, станом, успішністю і забирати результат. Оскільки перевірки мають виконуватись за розкладом, потрібен компонент, який би це робив: для цього обрано Celery-Redbeat. Як чергу повідомлень обрано Redis як надшвидку базу даних, що зберігається в оперативній пам'яті: між компонентами системи будуть передаватись повідомлення малого розміру, тому зберігання даних в оперативній пам'яті є допустимим. Для адміністрування Celery було обрано інструмент Flower: він надає можливість через вебінтерфейс дивитись за станом черг, задач, їхнім виконанням і можливими помилками.

Замість застосування планувальника Redbeat для Celery також розглядали вибір іншого планувальника або оркестратора задач, наприклад Airflow або Dagster, але через складність цих інструментів, трудоємність інтеграції та малу утилізацію їхнього потенціалу для поставленої задачі ці опції були визначені як надлишкові та відкинуті.

Для демонстрації алгоритмів визначення аномалій було обрано такі бібліотеки: Statsmodels для виявлення за допомогою STL декомпозиції, Prophet як бібліотеку аналізу часових рядів за допомогою неконтрольованого навчання.

Для відправки електронних листів було вирішено обрати Gmail як найбільш популярну систему електронної пошти. Аутентифікація в Gmail проводиться через протокол OAuth 2.0. Відправка повідомлень в Slack відбувається за допомогою Slack API. Запити до обидвох систем відправляються через бібліотеку httpx.

Як спосіб реалізації вебчастини системи було вирішено вибрати стратегію змішаного рендерингу: серверний рендеринг відбувається за допомогою Jinja2, а клієнтський рендеринг за допомогою JavaScript бібліотек. Для комунікації між сервером і клієнтом при клієнтському рендерингу обрано концепцію RESTful API. Стисло кажучи, ця концепція полягає в атомарності і незалежності один від одного запитів між клієнтом та сервером: головним є те, що запит до сервера має мати всю необхідну інформацію, щоб його опрацювати, тобто запит не має контексту.

На боці клієнта реалізацію вебсторінок було зроблено за допомогою мови розмітки HTML 5 та мови стилів CSS 3, з використанням JavaScript для динамічності сторінок. Інтерфейс зроблено на базі бібліотеки компонентів Tabler: вона постачається у вигляді CSS та JS файлів, які під'єднуються до потрібного вебсайту, і надає широкий набір заготовлених компонентів, зовнішній вигляд яких зроблений в одному стилі. Tabler дуже популярний через те, що побудований на базі Bootstrap — CSS-бібліотеки стилів, яка повсюдно використовується в вебсторінках, оскільки дозволяє максимально швидко розробляти красиві вебсторінки, адаптовані до різних пристроїв і форматів екранів. Для клієнтського рендерингу було обрано бібліотеку Alpine.js, яка надає можливість декларативно описувати залежності компонентів, і при зміні цих залежностей компоненти будуть перебудовуватись з новими даними, що позбавляє потреби робити оновлення вмісту і вигляду компонентів вручну.

Керування кодовою базою відбувається через систему контролю версій Git, а як віддалене сховище для Git репозиторію обрано GitHub як стандарт де-факто в індустрії. Опис до Git комітів зроблений за допомогою концепції Conventional Commits.

Для виконання поставлених вимог на базі обраних технологій було розроблено ПС автоматизованого виявлення аномалій. Архітектурним патерном виконання системи є мікросервісна архітектура.

Система складається з трьох компонентів: вебсервер (web server) — відповідає за роботу вебінтерфейсу системи; планувальник (scheduler) — планує виконання перевірок аномалій за розкладом; виконавець (worker) — виконує задачі, що приходять від планувальника та вебсервера. Схему взаємодії компонентів системи подано на рис. 1.

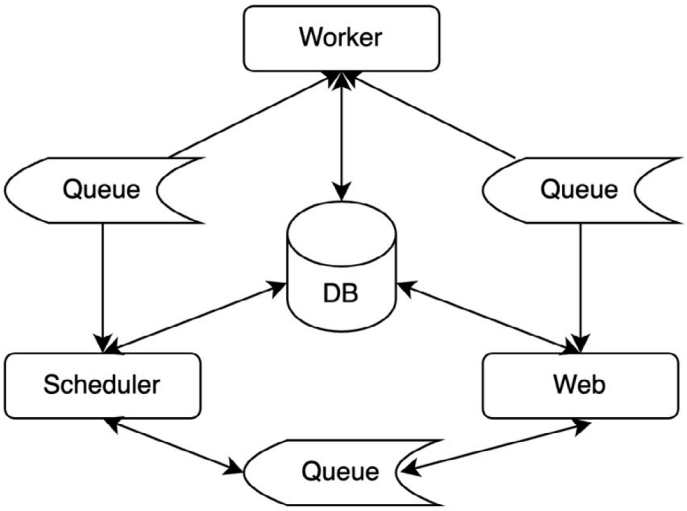

Рис. 1. Схема взаємодії компонентів системи

Компоненти взаємодіють один з одним за допомогою черги задач (task queue): вебсервер і планувальник надсилають у цю чергу задачі, які потрібно виконати, виконавець виконує їх і сигналізує про виконання результатом, який надсилає в цю саму чергу. Черга відповідає лише за комунікацію про завдання між компонентами системи; за зберігання даних відповідає БД, яка доступна всім компонентам.

Система складається з п'яти функціональних модулів: конфігурації виявлення, запуски перевірок, під'єднання до БД, контакти та повідомлення. Для того, щоб налаштувати систему, користувачу потрібно зробити такі кроки: додати під'єднання до бази даних, додати контакт для надсилання сповіщень, створити конфігурацію виявлення.

Головний елемент системи — менеджер налаштування конфігурацій виявлення. Ключовою особливістю системи є її інтерактивність: користувач має в інтерактивному режимі налаштовувати виявлення аномалій. Цю особливість було реалізовано в менеджері налаштування.

Налаштування складається з чотирьох кроків, які користувач проходить послідовно:

1. *Вибір під'єднання до БД*. Користувач обирає попередньо додане ним під'єднання до бази даних.
2. *Вибір даних для відстеження*. Менеджер сканує обрану базу даних і пропонує користувачу обрати потрібну таблицю та колонки, з яких система братиме дані. Система сповіщає користувача, якщо певну таблицю чи колонку неможливо обрати.
3. *Вибір і налаштування моделі*. На цьому етапі користувач має можливість вибрати модель, налаштувати часові інтервали і параметри моделі та одразу перевірити вибрану конфігурацію. Таким чином можна швидко налаштувати конфігурацію так, як це потрібно користувачу. Користувач бачить, що визначає модель і про що вона сповіщає, на графіку.
4. *Налаштування розкладу і сповіщень*. Користувач вибирає розклад запуску перевірки за допомогою Cron Expression (нотація для запису періодичності), та вибирає канал, в який потрібно буде надіслати сповіщення. Після проходження всіх цих кроків створюється конфігурація запуску визначення аномалій, яка зберігається в базі даних і додається до розкладу планувальника. При цьому перезавантажень планувальника для додавання чи видалення конфігурації не потребується.

Планувальник працює постійно і порівнює поточний час з розкладом виконання моделі, за збігу — планує виконання визначень, конфігурує виконання і надсилає в чергу повідомлення з задачею на виявлення. Ці повідомлення отримує виконавець та робить саму перевірку. Таким чином, за

допомогою черги задач процес оброблення даних, який є дуже вартісним із погляду ресурсів, відділений від інших функцій системи, і можна окремо керувати його ресурсами.

Виконання перевірки складається з чотирьох етапів: вибір даних з бази, оброблення даних моделлю, запис результатів до БД, надсилання сповіщення (за потреби). Якщо при виконанні перевірки відбулась програмна помилка, користувач буде сповіщений про це в обраному каналі комунікації, також це буде відображено на сторінці запуску. Якщо при надсиланні сповіщення виникла проблема, через яку це неможливо зробити, користувач побачить це тільки на сторінці запуску.

Для надсилання сповіщень про помилку при виконанні перевірки або про знайдені аномалії в системі підтримується два механізми — Slack і Gmail.

Для запуску перевірок розроблено таку систему часових інтервалів: інтервал спостереження, інтервал виявлення та інтервал сповіщення, які обраховують з урахуванням зсуву. Інтервал спостереження — це проміжок часу, який використовують для тренування моделі аномалій. Його початком є різниця моменту запуску аномалій, зсуву в часі та тривалість інтервалу спостереження, його кінцем — різниця моменту запуску аномалій і зсуву в часі. Для знайдених на цьому проміжку аномалій сповіщення не надсилаються. Інтервал виявлення — це проміжок часу, який використовують для оброблення моделлю аномалій. Його початком є різниця моменту запуску аномалій, зсуву в часі та тривалість інтервалу виявлення, його кінцем — різниця моменту запуску аномалій та зсуву в часі. Для знайдених у цьому проміжку аномалій сповіщення надсилаються (навіть з урахуванням того, що точки в цьому інтервалі також належать інтервалу спостереження). Зсув у часі — це різниця між часом виконання перевірки та кінцем інтервалів спостереження і виявлення. Аномалії, які лежать поза межами інтервалу виявлення, називають *аномаліями без сповіщень*, а ті, що лежать в межах інтервалу, — *аномаліями зі сповіщеннями*.

На сторінці конфігурації можна побачити статистику виконання перевірок для цієї конфігурації: відсоток виконаних перевірок зі сповіщеннями, без сповіщень і без аномалій. Також можна побачити статус виконання 30 останніх перевірок, вибрані налаштування перевірки, вибрані під'єднання до БД і канал сповіщення, і внизу сторінки доданий список останніх запусків для моделі.

На сторінці запуску (рис. 2) можна побачити статуси запуску, графік, побудований для даних, список аномалій, які викликали сповіщення, список аномалій без сповіщень, конфігурацію запуску перевірки, розміри часових інтервалів та інші параметри запуску.

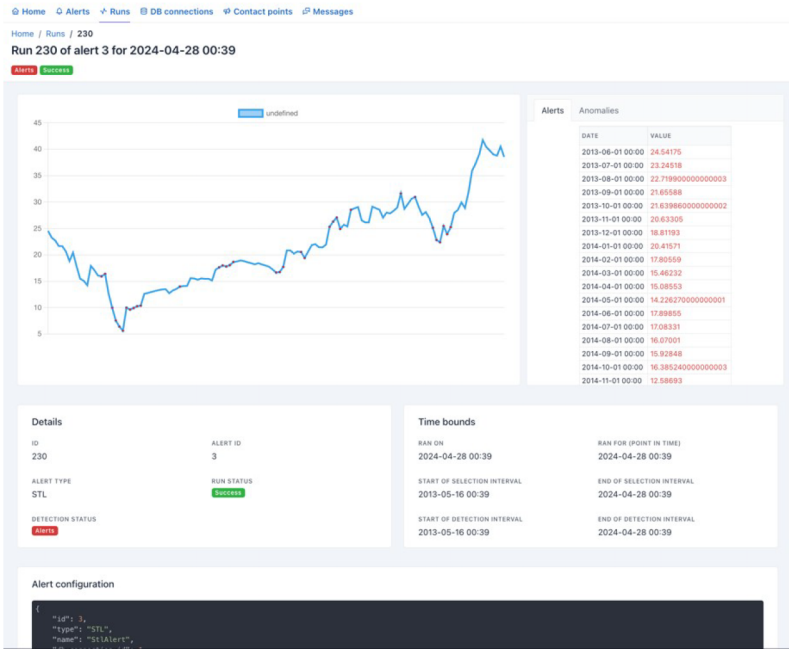

Рис. 2. Сторінка запуску

Модулі під'єднань до БД та контакти схожі між собою: обидва модулі містять сторінки для додавання і перегляду таких під'єднань. Кожне під'єднання користувачу обов'язково треба перевірити перед додаванням за допомогою кнопки «Перевірити», некоректні під'єднання неможливо додати в систему.

Важливо зазначити, що ключовим моментом реалізації цих модулів є архітектурно і інтерфейсно передбачена можливість додавання нових механізмів (провайдерів). У коді системи створені абстракції, для додавання нових провайдерів потрібно унаслідуватись від цих абстракцій у новому класі і перевизначити методи абстракції, додавши логіку, що стосується саме цієї системи.

При цьому інтерфейс користувача автоматично підлаштовується під кожного провайдера: для цього достатньо лише визначити нову модель даних, і системою буде створений інтерфейс, що матиме всі потрібні поля без потреби в додаткових налаштуваннях.

Подібний механізм створений і для моделей визначення аномалій: для додавання нових достатньо унаслідуватись від абстракцій і перевизначити логіку, що відмінна саме для цієї моделі, так само тут зроблене автоматичне генерування налаштувань. Підтримка різних моделей і провайдерів, легкість додавання нових імплементацій цих компонентів, і тим самим висока адаптивність є ключовою особливістю системи, що особливо важлива в індустрії даних, де використовують багато різних технологій, які часто змінюються. Це дає можливість використовувати систему в різних умовах.

Розроблена система повністю задовольняє поставлені раніше вимоги: легкість у налаштуванні забезпечується інтерфейсом користувача і інтерактивним процесом, гнучкість і легкість кастомізації — обраними технологіями та архітектурними абстракціями, надійність — розділенням компонентів через чергу задач, функціональні вимоги — розробленими складовими модулями. Система виконує поставлену задачу автоматизованого виявлення аномалій в бізнес-даних і відповідає сучасним стандартам у галузі даних.

5. Перспективи розвитку системи

Одним із обмежень на етапі формулювання вимог була мінімальна достатня функціональність продукту (MVP), оскільки реалізація додаткового функціоналу потребувала б більше часу і ресурсів. Саме тому система має великий потенціал до розвитку і може бути допрацьована в багатьох моментах.

Передусім, до системи може бути додано підтримку інших баз і сховищ даних (наприклад, AWS Athena, Amazon Redshift), підтримку API для отримання даних та інтеграцію з іншими системами комунікації (наприклад, Microsoft Teams). Це забезпечить можливість використання системи в інших умовах і компаніях, оскільки в галузі одночасно використовується велика кількість різних технологій. Велику практичну користь матиме і додавання інших моделей виявлення аномалій, наприклад, тих, що базуються на машинному навчанні.

Одним із потрібних доопрацювань є додавання вимірів для даних (data dimensions) і аналіз аномалій усередині цих вимірів, як індивідуально, так і в комбінації (наприклад, відстежуються продажі певного продукту для 10 різних платформ, у цьому випадку ці платформи будуть вимірами даних, і можна буде аналізувати аномалії для кожної з таких платформ).

Також до системи можна буде додати попереднє оброблення вхідних даних, наприклад фільтрацію, агрегацію, або написання власного SQL-запиту для вибору даних із бази. Для надсилання сповіщень можливо додати шаблони, за допомогою яких задаватиметься текст таких сповіщень, що підвищить їхню інформативність і цінність для певного користувача.

Функціональність статистики в системі може бути розширено для перегляду перевірок і їхніх результатів, профілювання даних на основі різних перевірок, або й таким чином, який би дав можливість визначати стан даних на основі всіх перевірок.

Висновки

У дослідженні проведено огляд і аналіз наукової літератури за темою, що дозволив систематизувати таксономію аномалій і технік їхнього визначення. Було наведено основні види і класифікації для аномалій і технік визначення аномалій, що дає уявлення про практичну цінність виявлення аномалій в обробленні даних.

Ми дали визначення поняття бізнес-даних, розглянули застосування цього підходу в цій галузі і перелічили проблеми, з якими сьогодні стикається галузь при аналізі даних. На основі проведеного дослідження підходу виявлення аномалій і поставленої проблеми було зроблене припущення про можливість використання автоматизованої ПС для вирішення знайдених проблем аналізу даних. Результатом дослідження стала побудова автоматизованої системи виявлення аномалій.

Для ефективної реалізації ПС проведено дослідження наявних інструментів, що близькі до тематики роботи. Було знайдено подібні рішення і проведений їхній аналіз із погляду відповідності їх потребам бізнесу і якості вирішення цих проблем. Аналіз показав, що наявні системи неактуальні, не підтримуються і не виконують задачі, потрібні для розв'язання проблеми. Таким чином було підтверджено потребу в системі, що вирішувала б розглянуту в роботі проблему.

На основі зробленого дослідження підходу виявлення аномалій, знайдених при аналізі рішень переваг і недоліків наявних систем, і власного досвіду авторів у галузі було розроблено вимоги, яким має відповідати розроблена система. На виконання цих вимог було обрано і спроєктовано архітектуру системи та проаналізовано і обрано технології для її програмної імплементації.

Практичним результатом роботи є автоматизована ПС виявлення аномалій. Система складається з кількох програмних модулів, що разом забезпечують виконання задач аналізу бізнес-даних. Ця ПС має високу адаптивність, є легкою до модифікації і зручною у використанні. Легкість налаштування забезпечується інтерактивним вебінтерфейсом, а надійність використання — розмежованою архітектурою з обмеженою відповідальністю компонентів.

Отже, в результаті роботи вдалося підтвердити поставлену раніше гіпотезу про можливість ефективного використання програмних систем вирішення проблеми аналітики бізнес-даних.

Список літератури

1. Arthur C. Tech giants may be huge, but nothing matches big data [Electronic resource] / C. Arthur // The Guardian. — Mode of access: https://www.theguardian.com/technology/2013/aug/23/tech-giants-data (date of access: 15.05.2025).
2. Brownlow J. Data-Driven Business Models: A Blueprint for Innovation [Electronic resource] / J. Brownlow et al. — 2015. — Mode of access: https://doi.org/10.13140/RG.2.1.2233.2320 (date of access: 15.05.2025).
3. Chandola V. Anomaly Detection: A Survey [Electronic resource] / V. Chandola, A. Banerjee, V. Kumar // ACM Computing Surveys. — 2009. — Vol. 41, no. 3. — Pp. 1–58. — Mode of access: https://doi.org/10.1145/1541880.1541882 (date of access: 15.05.2025).
4. Edgeworth F. Y. XLI. On discordant observations [Electronic resource] / F. Y. Edgeworth // The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science. — 1887. — Vol. 23, no. 143. — Pp. 364–375. — Mode of access: https://doi.org/10.1080/14786448708628471 (date of access: 15.05.2025).
5. Features — FastAPI [Electronic resource] // FastAPI. — Mode of access: https://fastapi.tiangolo.com/features/ (date of access: 15.05.2025).
6. GitHub — chaos-genius/chaos_genius: ML powered analytics engine for outlier detection and root cause analysis [Electronic resource] // GitHub. — Mode of access : https://github.com/chaosgenius/chaos_genius (date of access: 15.05.2025).
7. GitHub — cuebook/CueObserve: Timeseries Anomaly detection and Root Cause Analysis on data in SQL data warehouses and databases [Electronic resource] // GitHub. — Mode of access: https://github.com/cuebook/CueObserve (date of access: 15.05.2025).
8. GitHub — elementary-data/elementary: The dbt-native data observability solution for data & analytics engineers. Monitor your data pipelines in minutes. Available as self-hosted or cloud service with premium features [Electronic resource] // GitHub. — Mode of access: https://github.com/elementary-data/elementary (date of access: 15.05.2025).
9. Grubbs F. E. Procedures for Detecting Outlying Observations in Samples / F. E. Grubbs // Technometrics. — 1969. — Vol. 11, no. 1. — Pp. 1–21.
10. Hawkins D. M. Identification of Outliers [Electronic resource] / D. M. Hawkins. — Dordrecht : Springer Netherlands, 1980. — Mode of access: https://doi.org/10.1007/978-94-015-3994-4 (date of access: 15.05.2025).
11. Ravishankar D. T. N. Application of Time Series Analysis for Better Decision Making in Business [Electronic resource] / D. T. N. Ravishankar // TechnoareteTransactions on Intelligent Data Mining and Knowledge Discovery. — 2022. — Vol. 2, no. 4. — Mode of access: https://doi.org/10.36647/ttidmkd/02.04.a005 (date of access: 15.05.2025).

References

Arthur, C. (2013). Tech giants may be huge, but nothing matches big data. *The Guardian*. https://www.theguardian.com/technology/2013/aug/23/tech-giants-data.

Brownlow, J., et al. (2015). *Data-driven business models: A blueprint for innovation*. https://doi.org/10.13140/RG.2.1.2233.2320.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, *41* (3), 1–58. https://doi.org/10.1145/1541880.1541882.

Edgeworth, F. Y. (1887). On discordant observations. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *23* (143), 364–375. https://doi.org/10.1080/14786448708628471.

Features — FastAPI. *FastAPI*. https://fastapi.tiangolo.com/features.

GitHub — chaos-genius/chaos_genius: ML powered analytics engine for outlier detection and root cause analysis. *GitHub*. https://github.com/chaosgenius/chaos_genius.

GitHub — cuebook/CueObserve: Timeseries anomaly detection and root cause analysis on data in SQL data warehouses and databases. *GitHub*. https://github.com/cuebook/CueObserve.

GitHub — elementary-data/elementary: The dbt-native data observability solution for data & analytics engineers. Monitor your data pipelines in minutes. Available as self-hosted or cloud service with premium features. *GitHub*. https://github.com/elementary-data/elementary.

Grubbs, F. E. (1969). Procedures for detecting outlying observations in samples. *Technometrics*, *11* (1), 1–21.

Hawkins, D. M. (1980). *Identification of outliers*. Springer Netherlands. https://doi.org/10.1007/978-94-015-3994-4.

Ravishankar, D. T. N. (2022). Application of time series analysis for better decision making in business. *Technoarete Transactions on Intelligent Data Mining and Knowledge Discovery*, *2* (4). https://doi.org/10.36647/ttidmkd/02.04.a005.

M. Postnikov, S. Gorokhovsky

AUTOMATED SYSTEM FOR DETECTION OF ANOMALIES IN BUSINESS DATA

The article describes the analysis of the anomaly-detecting process in business data, known software solutions, formulated requirements for the system and developed an automated software system for detecting anomalies. The developed system consists of software modules, has high adaptability, ease of modification, and ease of use. The system fully satisfies the previously set requirements: ease of configuration is provided by the user interface and interactive process, flexibility and ease of customization – by selected technologies and architectural abstractions, reliability – by separation of components through a queue of tasks, functional requirements – by developed component modules. The system performs the task of automated detection of anomalies in business data and meets modern standards in the field of data. The main types and classifications for anomalies and anomaly detection techniques were given, which give an idea of the practical value of anomaly detection in data processing. The analysis of the problem showed that existing systems are irrelevant, not supported, and do not perform the tasks required to solve the problem. Based on the research, the requirements that the system must meet were developed. To meet these requirements, the system architecture was selected and designed, and technologies for its software implementation were analyzed and selected. The practical result of the work is an automated software system for anomaly detection. The system consists of several software modules that together provide the performance of business data analysis tasks. The developed software system has high adaptability, ease of modification, and ease of use. Ease of configuration is ensured by an interactive web interface, and reliability of use is ensured by a delimited architecture with limited component liability. Thus, because of the work, it was possible to confirm the previously formulated hypothesis about the possibility of effective use of software systems to solve the problem of business data analysis. The system has great potential for development and can be developed in many directions, for example, support for other databases and data warehouses, as well as integration with other communication systems.

Keywords: anomaly detection, software system, system requirements, architecture, technologies, web application, FastAPI, Celery, REST API, Pandas.

Матеріал надійшов 25.06.2025