МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

“До захисту допущена”
Зав. кафедри комп’ютерних наук та прикладної математики
д.т.н., професор Турбал Ю. В.
«___» _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

«Проєктування та розробка модуля виявлення аномалій в даних IoT пристроїв системи моніторингу водних резервуарів»

Виконав: Демчук Юрій Миколайович
студент навчально-наукового інституту автоматики, кібернетики та обчислювальної техніки
група ІПЗ-41

_____
(підпис)

Керівник: доц., к.т.н. Жуковський Віктор Володимирович

_____
(підпис)

Рівне – 2023

ЗМІСТ

РЕФЕРАТ

Кваліфікаційна робота: 60 сторінок, 34 рисунків, 18 наукових джерел. Автором опубліковано 1 тези за результатами доповіді на Науково-практичній конференції навчально-наукового інституту автоматики, кібернетики та обчислювальної техніки 12–19 травня 2022 року та 1 статтю, що індексується в наукометричній базі Scopus.

Мета роботи: проєктування та розробка модуля виявлення аномалій в даних IoT пристроїв системи моніторингу водних резервуарів.

Засоби розробки: Python, ASP.NET Core, AngularTs, Angular Material, Chart.js, Node.js, HTML5, SCSS, CSS, JavaScript, TypeScript, SQL Server Management Studio.

Актуальність роботи заключається в тому, щоб автоматизувати процес виявлення підозрілих даних, які виникають в роботі пристроїв інтернету речей. Автоматизувати збір щоденної статистики для підтримки актуальної інформації про стан пристроїв.

Ключові слова: *Інтернет речей, моніторинг, автоматизація, обробка інформації, технології, Python, Big Data, Anomaly detection, Angular, TypeScript, C#, SQL Management Studio.*

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СУБД – система управління базами даних (в роботі використовувався MS SQL Server);

Frontend – клієнтська частина проекту;

Backend – серверна частина проекту;

Anomaly detection – виявлення аномалій;

AngularTS – Angular TypeScript;

Дрейфи (Drifts) – повільна, ненаправлена, довгострокова зміна даних;

Викиди (Outliers) – невеликі аномальні шаблони, які нерегулярно з’являються під час збору даних;

Зміна подій (Change in Events) – систематична або раптова зміна попередньої нормальної поведінки джерела генерування інформації;

SSMS – SQL Server Management Studio.

ВСТУП

Одним з найважливіших факторів, що впливають на різноманітні процеси у всьому світі є інформація. Станом на сьогодні будь-які обсяги інформації мають свою як фінансову, так і організаційну цінність. До інформації відносять різного роду дані, для прикладу: повідомлення різних видів (письмові, графічні та ін), відомості про рутинні життєві процеси, дані від пристроїв інтернету речей та автоматизованих систем і тому подібне.

В сучасному світі існує безліч джерел, які на протязі певного проміжку часу можуть генерувати великі обсяги даних і відправляти їх в хмару для подальшої обробки. Процес подальшої обробки, як правило, включає в себе субпроцеси, які в свою чергу маніпулюють вхідною інформацією так як задумав розробник. Вхідна інформація може зберігатися на сервері або у хмарі, що дозволить в майбутньому проводити дослідження, класифікацію, фільтрування, сортування масиву даних, або ж сформувати новий масив даних, який базуватиметься на початкових даних, але матиме суттєві або не суттєві відмінності від нього. На основі великих обсягів даних можуть бути побудовані та реалізовані різні алгоритми. До них можна віднести алгоритми нормалізації, обробки, візуалізації даних та ін. Реалізовані алгоритми, як правило, направлені на те, щоб полегшити взаємодію користувача з великими обсягами інформації.

На основі великих обсягів даних є сенс будувати проект, який дозволив би забезпечити:

- комфортний моніторинг вхідних даних;
- комунікацію з користувачами;
- формування додаткових статистичних даних;
- виявлення аномальних даних.

При виконанні великого збору даних постає потреба в виявленні аномальних даних. Процес виявлення аномальних даних – це пошук “унікальних” даних, що не відповідають очікуваній, або заздалегідь прогнозованій поведінці в конкретному масиві даних.

Методи виявлення аномальних даних дозволяють виконати автоматизацію рутинних людських процесів, таких як: надсилання сповіщень користувачам у разі виникнення негараздів (збоїв) в роботі джерела надсилання інформації, або ж виконання прогнозування майбутніх негараздів в роботі цілісної системи та ін.

В сучасному світі процеси, які відповідають за виявлення аномальних даних відіграють важливу роль в різних галузях (особливо технологічних), тому що вони спроможні виявляти несанкціонований доступ до певного об’єкту чи ресурсу, знаходити помилки в роботі проекту або іншої системи, класифікувати типи помилок і багато іншого.

РОЗДІЛ І. АНОМАЛЬНІ ДАНІ ТА СПОСОБИ ЇХ ВИЯВЛЕННЯ

1.1. Аномальні екземпляри даних та їх властивості

На сьогоднішній день інформація відіграє невід'ємно-важливу роль у нашому житті. Інформація (дані) є результатом вимірювання, спостережень, логічних та/або арифметичних операцій, яка перетворена, як правило, на форму, що може бути постійно збережена, передана і застосована в автоматизованій обробці.

При збільшенні деякого масиву інформації, який, припустимо, містить в собі критично важливі дані про користувачів, життєвий цикл програмного забезпечення чи діагностичні дані від пристрою – постає потреба у виявленні унікальних екземплярів даних (аномалій). Виявлення цих точкових даних дозволяє виконувати різного роду прогнозування джерела генерування інформації.

Виявлення аномалій – це процес аналізу даних проекту, метою якого є знаходження підозрілих даних, які не відповідають стандартній поведінці конкретного масиву значень (даних).

Виявлені аномальні дані можна розподілити на різні категорії та класифікувати за такими ознаками:

- **Викиди (Outliers)**
- **Зміна подій (Change in Events)**
- **Дрейфи (Drifts)**

Прості статистичні методи, такі як середнє значення, медіана, квантилі, можна використовувати для виявлення аномалій у наборі даних. Для виявлення аномалій також можна використовувати різні методи візуалізації даних і дослідницького аналізу даних.

Багато методів виявлення викидів, особливо неконтрольованих методів, не спроможні виявити раптові стрибки активності. Однак ці типи мікрокластерів часто можна легше ідентифікувати за допомогою алгоритмів кластерного аналізу.

Методи контрольованого виявлення аномалій вимагають масиву даних із повним набором «нормальних» і «ненормальних» значень для роботи алгоритму. Ця техніка також передбачає навчання нейромережі. Можна сказати, що подібне виявлення аномалій еквівалентно знаходженню шаблонів даних, що не відповідають заздалегідь спрогнозованій поведінці Не всі алгоритми статистичної класифікації добре підходять для виявлення аномалій [10].

Існує три основні класи методів виявлення аномалій: неконтрольовані, напівконтрольовані та контрольовані. Зазвичай, при виборі одного або декількох з вищевказаних методів виявлення аномалій особливу увагу приділяють інформації, яка буде відігравати ключову роль у детекції рідкісних екземплярів даних. Іншими словами – правильний метод виявлення аномалії залежить від доступних значень у наборі даних [10].

Як правило, алгоритми виявлення аномальних даних впроваджуються та використовуються з метою аналізу конкретного пристрою або іншого джерела генерування інформації, що в подальшому дозволяє робити прогнозування помилок, автоматично повідомляти власників пристроїв про негаразди, або ж для дослідження суперечливих даних (побудови статистики та ін.).

Зазвичай до аномальних даних можна віднести ті дані, що не відповідають очікуваній, або заздалегідь прогнозованій поведінці в деякому масиві даних. Масиви даних можуть базуватися на будь-яких подіях з усього світу. В даній роботі за основу досліджень було взято проект, в якому дані формуються за допомогою пристроїв інтернету речей (IoT). Ці пристрої надсилають дані про рівень заряду батареї пристрою, рівень води та кількість відправлених та отриманих байтів в хмару. Такий масив даних дозволяє виконати автоматизацію процесів і провести коректний аналіз.

Станом на сьогодення виявлення аномалій також знаходить своє застосування в різних областях, таких як виявлення шахрайських банківських транзакцій, виявлення мережевих вторгнень, раптове зростання/падіння продажів, зміна поведінки клієнтів, тощо. Алгоритми виявлення аномалій також часто використовують у попередній обробці для видалення аномальних даних із

набору даних. Це робиться з кількох причин. Статистичні дані, такі як середнє значення та стандартне відхилення, стають більш точними після видалення аномалій. У контрольованому навчанні видалення аномальних даних із набору даних часто призводить до статистично значущого підвищення точності.

Успішне виявлення аномалій залежить від можливості точно аналізувати дані часових рядів у режимі реального часу. Дані часових рядів складаються з послідовності значень у часі. Іншими словами – кожен отриманий запис від джерела інформації повинен бути прив’язаним до одиниці часу, таким чином утворюється пара елементів дата-значення, за допомогою яких можна провести аналіз масиву [4].

Для того, щоб коректно виконати виявлення аномальних даних – потрібно провести процеси дослідження, аналізу та реалізації відповідних алгоритмів. В результаті чого алгоритми виявлення аномальних даних повинні правильно обробляти масив даних та виконувати всі подальші маніпуляції над ним (знаходити підозрілі дані, сформувати масив з підозрілими даними та ін).

Завдяки правильній реалізації програмного забезпечення для виявлення викидів деякому користувачу буде легше вимірювати кожен окремий аспект бізнес активності. Це включає в себе ефективність всього проекту в цілому, а також ключові показники ефективності. Методи виявлення аномалій можна використовувати для створення більш надійних джерел генерування даних.

1.2. Контрольовані, неконтрольовані та напівконтрольовані класи виявлення аномалій

Процес виявлення аномалій можна поділити на три категорії методів виявлення аномалій:

- **Контрольовані.** Ці методи виявлення аномалій вимагають масиву даних, який був класифікований за однією з ознак виявлення аномалій і передбачає навчання класифікатора. Як правило, цей підхід рідко

використовується для виявлення аномалій через притаманну незбалансованість класів [11].

- **Неконтрольовані.** Неконтрольовані методи виявлення аномалій виявляють аномалії в тестовому наборі даних виключно на основі внутрішніх властивостей цих даних. Вони грунтуються на припущенні, що, як правило, більшість екземплярів у наборі даних є нормальними. Алгоритми виявлення аномалій, які базуються саме на цьому методі виконують ідентифікацію екземплярів, які найменше відповідають решті даних з того чи іншого масиву [10].
- **Напівконтрольовані.** Цей метод використовує звичайний навчальний набір даних для побудови моделі, що представляє нормальну поведінку джерела генерування інформації. Далі ця ж модель використовується для виявлення аномалій, перевіряючи, наскільки ймовірно, що модель генерує будь-який один випадковий викид [10].

Кожен з вищевказаних методів обирається відповідно до конкретного масиву значень, тому що дані, які потенційно можуть бути проаналізовані алгоритмами мають різну структуру, значення та ключовий індексуючий елемент. Зазвичай індексуючим елементом виступає одиниця часу. Відповідно до цих ознак виконується відбір найкращих класів виявлення аномалій, для того, щоб в подальшому максимізувати точність виявлення аномалій.

Для певних наборів даних можна застосовувати одразу декілька класів виявлення аномалій, але за умови, що буде проведено нормалізацію та належне тестування для кожного з цих класів.

1.3. Алгоритми виявлення аномальних даних

На сьогоднішній день існує багато алгоритмів для виявлення аномальних даних, але одними з найбільш популярних алгоритмів є DBSCAN, K-means та IForest. Кожен з цих алгоритмів працює по-різному, тому вони мають як плюси так і мінуси.

DBSCAN – алгоритм просторової кластеризації з присутністю шуму. Даний алгоритм виконує операції з щільністю (частотою) даних. Вхідними параметрами цього алгоритму є e-epsilon (радіус) та кількість сусідніх точок. Іншими словами ці обов'язкові параметри є матрицею близькості. Його можна використовувати для кластеризації точок даних на основі щільності, тобто шляхом групування областей із багатьма масивами. На *рис. 1.1* відображено принцип роботи даного алгоритму [1].

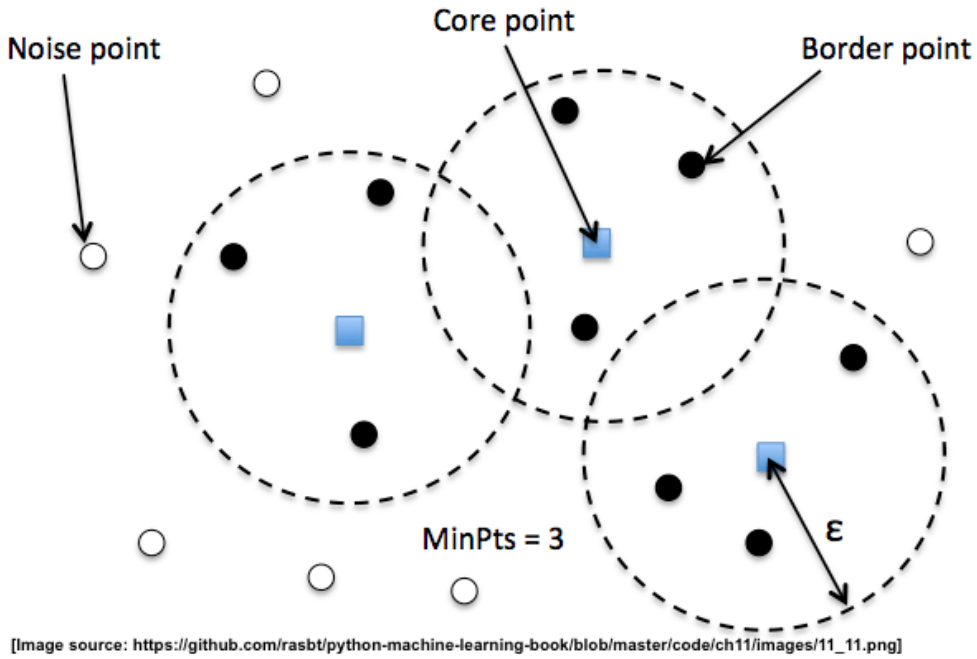
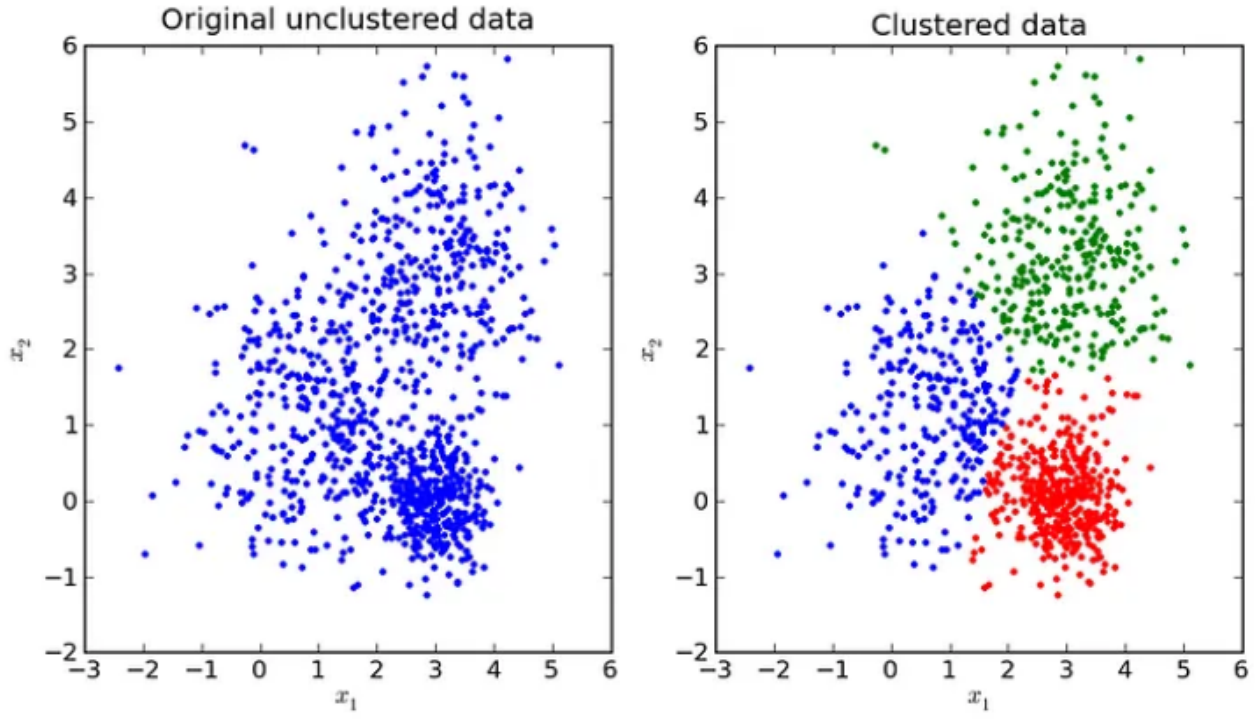


Рис. 1.1. Концепція роботи алгоритму DBSCAN

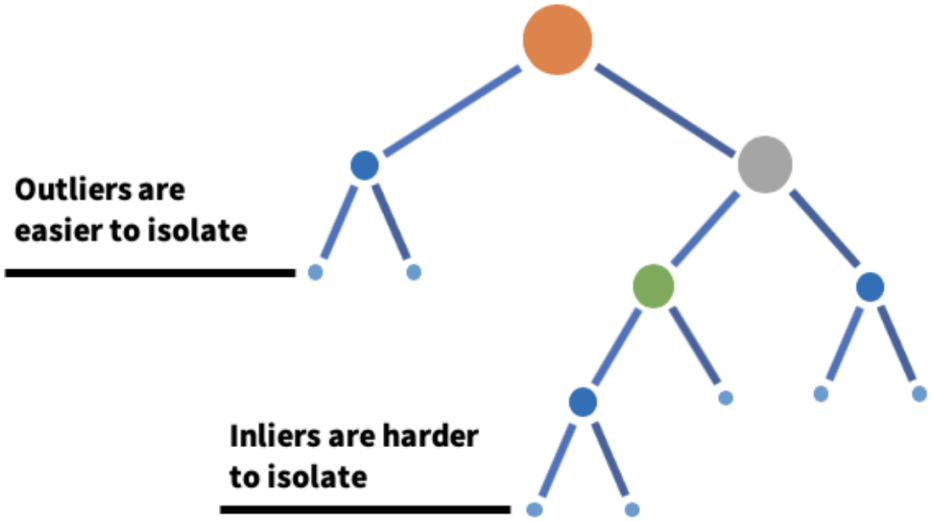
K-means є одним із найпростіших і популярних алгоритмів неконтрольованого машинного навчання. Щоб обробити початковий масив даних, алгоритм спочатку виконує випадковий вибір центроїдів, які використовуються як початкові точки для кожного кластера, а потім виконує ітераційні (повторювані) обчислення для оптимізації позицій центроїдів для кожної з ітерацій. На практиці даний алгоритм дуже швидкий (один із найшвидших доступних алгоритмів кластеризації), але він потрапляє в локальні мінімуми. Через це при повторних запусках цього алгоритму буде отримано різні аномальні дані, що іноді може бути як корисно, так і погано. Приклад кластеризації даних з використанням цього алгоритму наведено на *рис. 1.2*.



Source: https://i.stack.imgur.com/clDB3.png

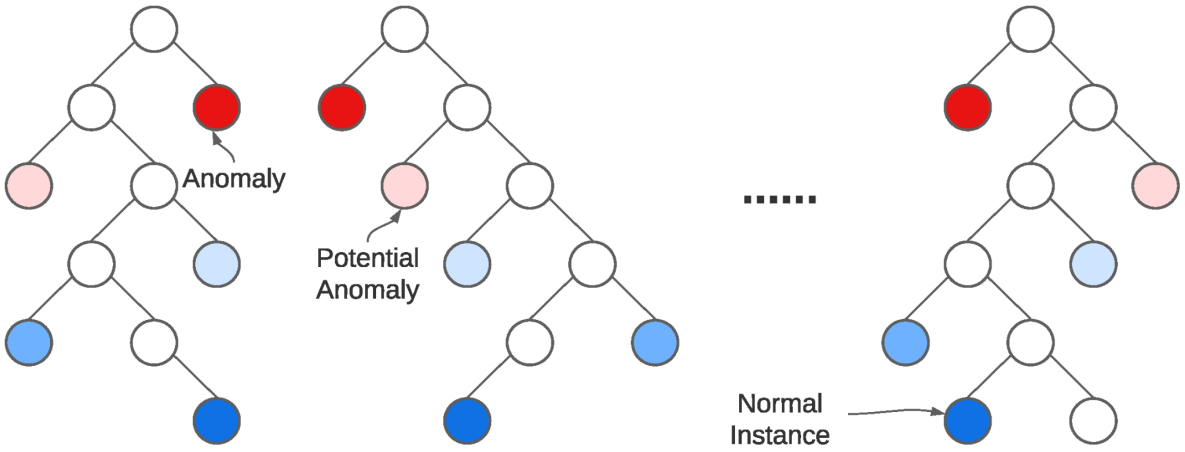
Рис. 1.2. Приклад кластеризації даних з використанням алгоритму виявлення аномалій K-means

IForest – алгоритм неконтрольованого машинного навчання для виявлення аномалій (викидів) у наборах даних. Випадково відібрані дані обробляються в деревовидній структурі. Зразки, які потрапляють глибше в дерево, мають меншу ймовірність бути аномаліями, а інші навпаки. Іншими словами – Isolation Forest є бінарним деревом рішень. Алгоритм починає роботу з навчання даних шляхом генерації ізольованих дерев. Коли алгоритм отримує набір даних, випадковий масив даних перетворюється в бінарне дерево, після чого виконується розгалуження дерева. Наступним кроком є відсіювання даних спочатку для вибраного дерева рішень, а потім усього наступного масиву. Всі процеси розгалуження будуть продовжуватися рекурсивно до тих пір, поки одна з точок не буде вважатися ізольованою. Даний алгоритм класифікує аномалії наступним чином: значення -1 – присвоюється аномальним даним, 1 – присвоюється для даних, в яких не виявлено підозрілих активностей. Для прикладу, на *рисунках 1.3-1.4* наведено принцип роботи цього алгоритму.



Source:
https://content.linkedin.com/content/dam/engineering/site-assets/images/blog/posts/2019/08/IsolationForest1.png

Рис. 1.3. Візуалізація деревовидної структури алгоритму Isolation Forest (IForest)



Source:
https://www.mdpi.com/computers/computers-11-00054/article_deploy/html/images/computers-11-00054-g003.png

Рис. 1.4. Візуалізація деревовидної структури алгоритму Isolation Forest (IForest)

До менш популярних і водночас простіших в реалізації алгоритмів можна віднести алгоритми SeasonalAD та QuantileAD. В певних сценаріях аналізу масивів даних вони можуть відігравати досить важливу роль. Розглянемо їх більш детально.

Seasonal Decomposition – алгоритм для виявлення даних, які порушили сезонність. Даний детектор виявляє аномальні дані в часових рядах. Іншими словами у масиві, до якого буде застосовано цей алгоритм, дані повинні надходити з однаковим інтервалом часу (щосекунди/щогодини/щодня і тд). Алгоритм аналізує кожен з проміжків часу, порівнює його з іншими часовими рядами після чого переходить до виявлення аномальних даних.

QuantileAD (Quantile Anomaly Detection) – алгоритм для виявлення аномалій, який порівнює кожне значення часового ряду з логованими квантилями. Він потребує два обов’язкових параметри, які будуть безпосередньо впливати на виявлення аномалій. В якості параметрів задається відсоток верхньої та нижньої межі. Після чого в цих межах дані відмічатимуться як аномальні. Межі генеруються відповідно до максимального та мінімального значень з масиву даних.

В результаті отримуємо, що кожен з цих алгоритмів по-своєму виявляє аномальні дані, тому для більш точного виявлення аномальних даних потрібно проводити тестування цих алгоритмів. Водночас в процесі тестування потрібно спробувати досягти максимально точних результатів виявлення. Для максимізації виявлення аномалій потрібно передбачити комбіноване тестування алгоритмів.

1.4. Аналіз вже існуючих систем виявлення аномалій

Станом на сьогодення виявлення аномалій (Anomaly Detection) знаходить своє застосування в різних сферах, різних проектах та компаніях. Як було зазначено раніше – алгоритми виявлення аномалій можуть використовуватися з різною метою, наприклад:

- Виявлення вразливостей в безпеці;
- Виявлення підозрілої активності джерела генерування інформації;
- Виявлення підозрілих фінансових транзакцій у бізнес-сферах та ін.

Виявлення аномалій може бути використано для створення деякої системи, яка виявлятиме вторгнення до певного ресурсу.

Система виявлення вторгнень – це звичайні механізми кібербезпеки, призначені для збору, обробки та аналізу інформації, отриманої від комп'ютерних хостів або мереж, для виявлення зловмисних дій, таких як порушення безпеки, включаючи атаки, що виникають всередині, або поза інфраструктурою [17, 18].

В контексті виявлення вразливостей в безпеці можна навести такий приклад: Pajouh et al. представили модель виявлення вторгнень на основі дворівневого модуля зменшення розмірності та дворівневої класифікації. Ця модель також розроблена для виявлення зловмисних дій, таких як атаки U2R (*User to root*) і R2L (*Remote to local*). Для зменшення розмірності використовувався компонентний аналіз і лінійний дискримінаційний аналіз. Набір даних NSL-KDD використовувався для проведення всього експерименту. Для виявлення підозрілої поведінки за допомогою модуля дворівневої класифікації було застосовано версію Naive Bayes і Certainty Factor K-Nearest Neighbor [15].

Розглядаючи дослідження, котрі пов'язані з моніторингом систем водних резервуарів – можемо взяти до уваги наукову роботу "IoT for Water Management: Towards Intelligent Anomaly Detection" дослідниками якої є Aurora Gonzalez-Vidal, Jesus Cuenca-Jara and Antonio F. Skarmeta. Динаміка пропозиції та попиту на воду робить критично важливим для урядів можливість оцінювати та краще управляти водопостачанням, що в свою чергу вимагає розумнішого підходу для досягнення кращих результатів протягом життєвого циклу управління водними ресурсами. Саме в цій сфері діяльності Інтернет речей (IoT) займає основне місце. Завдяки IoT, можливості взаємозв'язку та комунікації, які сьогодні вбудовані майже у всі речі, дозволяють обмінюватись інформацією ненав'язливо та ефективно [16] .

У вищевказаній науковій роботі було виконано виявлення слабких та сильних сторін методів різної природи, щоб знайти аномалії саме в часових

рядах споживання води, а також щоб перевірити їх ефективність. Алгоритми, які базуються на моделях ARIMA та HOT-SAX було протестовано дослідниками на 30-ти часових рядах, які були анотовані. Спочатку споживання води було накопиченим значенням, а вимірювання відбувалось не регулярно. В процесі роботи вони агрегували споживання кожні 2 години та враховували вимірювання за 1 місяць часу, щоб провести аналіз [16].

Обидва підходи, котрі були протестовані науковцями добре виявляють аномалії: 90% знайдено за допомогою так званої моделі ARIMA (Auto Regressive Integrated Moving Average), а 80% за допомогою HOT-SAX. Однак обидва цих підходи демонструють високий рівень помилкових спрацьовувань, тобто вони виявляють аномалії, які не були зазначені експертом [16].

Таким чином, можемо зробити висновок, що виявлення аномалій знаходить своє застосування у різних сферах діяльності, починаючи від виявлення аномалій та обробки інформації про рівень води в резервуарах і завершуючи виявленням вразливостей в безпеці тих чи інших інтернет-систем. Використання та розробка таких алгоритмів і механізмів дозволяє автоматизувати безліч процесів, котрі виконує людина. Варто зазначити, що при належній реалізації механізмів виявлення аномалій можна виявляти підозрілу активність деякої системи навіть в тих випадках, коли звичайний користувач не здатний помітити події, котрі можуть виконуватись по інший бік програмного коду (вторгнення, хакінг і тому подібне).

1.5. Порівняльна характеристика ефективності алгоритмів

Для того, щоб виконати відбір найефективнішого алгоритму виявлення аномалій необхідно виконати узагальнене тестування кожного з вказаних раніше алгоритмів.

На теперішній час існує велика кількість ресурсів, на яких було проведено тестування алгоритмів, котрі були вказані раніше. З даних, які зображено на *рис. 1.5.* можемо спостерігати, що в більшості сценаріїв (при використанні різних масивів даних) алгоритм IForest є лідером.

*conditional formatting is a row-wise comparison

ROC Performances (average of 10 independent trials)

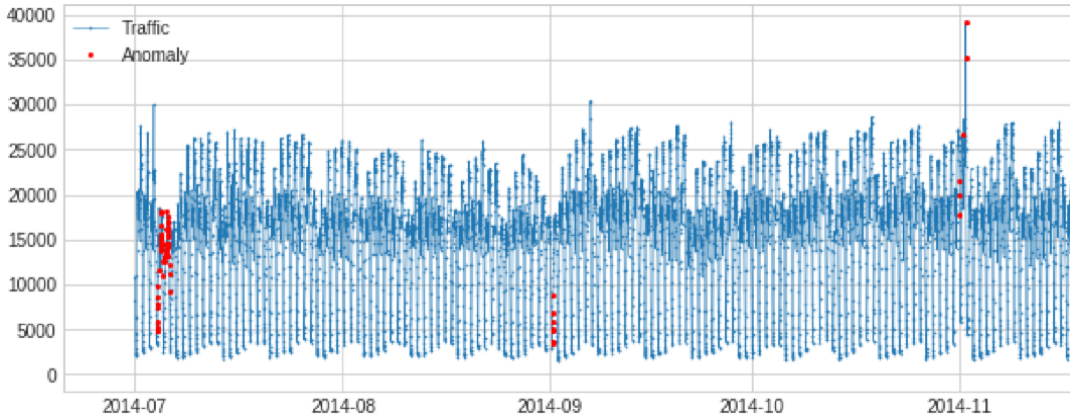
Data	#Samples	# Dimensions	Outlier Perc	ABOD	CBLOF	FB	HBOS	IForest	KNN
arrhythmia	452	274	14.60	0.7688	0.7835	0.7781	0.8219	0.8005	0.7861
cardio	1,831	21	9.61	0.5692	0.9276	0.5867	0.8351	0.9213	0.7236
glass	214	9	4.21	0.7951	0.8504	0.8726	0.7389	0.7569	0.8508
ionosphere	351	33	35.90	0.9248	0.8134	0.873	0.5614	0.8499	0.9267
letter	1,600	32	6.25	0.8783	0.507	0.866	0.5927	0.642	0.8766
lympho	148	18	4.05	0.911	0.9728	0.9753	0.9957	0.9941	0.9745
mnist	7,603	100	9.21	0.7815	0.8009	0.7205	0.5742	0.8159	0.8481
musk	3,062	166	3.17	0.1844	0.9879	0.5263	1	0.9999	0.7986
optdigits	5,216	64	2.88	0.4667	0.5089	0.4434	0.8732	0.7253	0.3708
pendigits	6,870	16	2.27	0.6878	0.9486	0.4595	0.9238	0.9435	0.7486
pima	768	8	34.90	0.6794	0.7348	0.6235	0.7	0.6806	0.7078
satellite	6,435	36	31.64	0.5714	0.6693	0.5572	0.7581	0.7022	0.6836
satimage-2	5,803	36	1.22	0.819	0.9917	0.457	0.9804	0.9947	0.9536
shuttle	49,097	9	7.15	0.6234	0.6272	0.4724	0.9855	0.9971	0.6537
vertebral	240	6	12.50	0.4262	0.3486	0.4166	0.3263	0.3905	0.3817
vowels	1,456	12	3.43	0.9606	0.5856	0.9425	0.6727	0.7585	0.968
wbc	378	30	5.56	0.9047	0.9227	0.9325	0.9516	0.931	0.9366
			mean	0.7031	0.7636	0.6767	0.7819	0.8179	0.7758
			median	0.7688	0.8009	0.6235	0.8219	0.8159	0.7986
			sd	0.2038	0.1890	0.1966	0.1866	0.1590	0.1764

*Source: https://miro.medium.com/v2/resize:fit:1161/1*SbyjIQ5_WNwM1X9TzxhQGA.png*

Рис. 1.5. Порівняння алгоритмів виявлення аномальних даних

Алгоритм DBSCAN, як було зазначено раніше – є алгоритмом просторової кластеризації, через це виявлення аномальних даних не є стабільним, а саме: при повторному запуску алгоритму з використанням одного і того ж масиву даних буде отримано різні набори даних з аномальними викидами. Для більш точного визначення викидів з використанням цього алгоритму потрібно знайти оптимальний розмір кластера та оптимальну відстань між зразками (кластерами), але під час повторного перезапуску алгоритму масиви з викидами не будуть ідентичними.

Алгоритм SeasonalAD працює виключно з часовими рядами, тому для коректної роботи цього алгоритму, як правило, необхідно проводити ренормалізацію даних. В більшості випадків його використання підходить для тих масивів даних, які вже з самого початку містять впорядковані часові ряди. На *рис 1.6.* зображено приклад роботи даного алгоритму.



Source: https://arundo-adtk.readthedocs-hosted.com/en/stable/_images/notebooks_demo_40_0.png

Рис. 1.6. Виявлення аномальних даних (тестовий масив даних) з використанням алгоритму SeasonalAD

Алгоритм QuantileAD виявляє аномалії лише у верхніх та нижніх межах масиву даних. Це обумовлено його особливістю роботи. Для прикладу, на *рис. 1.7* відображено принцип роботи алгоритму. Згідно наведеного прикладу бачимо, що алгоритм визначає моменти часу, впродовж яких температура була більшою за 99% та меншою за 1% від верхньої та нижньої меж відповідно до аналізуючого масиву значень.

Source: https://arundo-adtk.readthedocs-hosted.com/en/stable/_images/notebooks_demo_10_0.png

Рис. 1.7. Виявлення аномальних даних (тестовий масив даних) з використанням алгоритму QuantileAD

Після завершення тестувань алгоритмів, які базуються на масивах даних з інтернет-мережі, можемо зробити висновок, що найкращим рішенням є використання алгоритмів DBSCAN та IForest. Обрані алгоритми в подальшому дозволять виявити аномальні дані серед пристроїв інтернету речей.

РОЗДІЛ ІІ. ІНФОРМАЦІЙНА СИСТЕМА МОНІТОРИНГУ ВОДНИХ РЕЗЕРВУАРІВ

2.1. Огляд інформаційної системи моніторингу

Віддалені пристрої інтернету речей значно розширюють управлінські повноваження фермерів. Пристрої інтернету речей в нашому випадку автоматизують завдання перевірки рівнів води та роботи водяних насосів. З цієї причини компанія Meadowlark Solutions розробила складну інформаційну систему моніторингу віддалених датчиків (IoT пристроїв) під назвою Tank Toads спеціально для тваринницької галузі, які спроможні вимірювати рівні води в резервуарах (*див. рис. 2.1.*).

Рис. 2.1. Демонстрація роботи пристрою інтернету речей

Датчики вимірюють рівень води щогодини. Щоранку клієнт отримує зведене текстове повідомлення з високим, низьким і середнім рівнем води за останні 24 години. Клієнт також отримує попередження, якщо рівень води стає занадто високим, занадто низьким або якщо пристрій несправний. Автоматизація цих процесів відбувається у системі обробки інформації на сервері. Сервер відповідає за керування всією мережею пристроїв у масштабований спосіб. Сервер відстежує пристрої, записує дані, обробляє ці дані та надсилає їх відповідним одержувачам. Ця ключова частина технології необхідна для належного керування IoT пристроями.

Фільтрування великих обсягів діагностичних даних утомливе та неточне, якщо виконувати його вручну. Процес фільтрації цих даних можна знайти в веб-інтерфейсі Meadowlark. Інтерфейс сортує те, що важливо, і відправляє це адміністратору та клієнтам. Інформаційно-аналітична система обробляє дані, що надходять від віддалено підключених пристроїв, фільтрує інформацію та збирає її в зручну для читання форму. Потім система доставляє цю інформацію через SMS-шлюз на мобільні телефони клієнтів. Клієнти можуть запитувати додаткові дії, такі як отримання додаткових показань, зміна налаштувань або дистанційне керування підключеним обладнанням.

Інформаційна система моніторингу пристроїв IoT пристроїв розроблена для того, щоб було зручно та швидко спостерігати за змінами тих чи інших діагностичних даних пристроїв інтернету речей. З використанням цієї системи також можна коригувати роботу пристроїв IoT, рисувати графіки, які містять дані від пристроїв за різний період часу, або ж виконати експорт даних до окремих файлів для виконання подальшого аналізу [5, 9].

Система моніторингу складається з комплексу сторінок. Кожна сторінка такої системи прив’язана до окремих функцій, таких як: “Логи”, “Налаштування аккаунту”, “Моніторинг функціональності пристроїв”, “Увімкнення/Вимкнення пристроїв” та ін. На відповідних сторінках містяться великі обсяги діагностичних даних від пристроїв IoT, які дозволяють вибудовувати все більший і більший функціонал проекту.

Збір даних та забезпечення коректної роботи платформи моніторингу пристроїв IoT відбувається з використанням багатьох алгоритмів і механізмів, ось деякі з них:

- *Механізм реплікацій баз даних між віртуальним сервером та хмарним сервісом Microsoft Azure Sql Database*. Даний механізм відповідає за створення дублікату даних в іншу базу даних, яка знаходиться в хмарному сервісі Azure. Таким чином всі внесені зміни на сервері-видавці практично моментально будуть відображені на сервері-підписнику. Тому у разі недосяжності сервера-видавця користувачі і/або програмне

забезпечення не зможе отримати доступ до даних, але завжди буде оперативна можливість використати резервний хмарний сервер/сервіс [6];

- *Алгоритм проекції запитів*. Даний алгоритм націлений в основному на оптимізацію всіх доступів до бази даних. Він базується на стратегії оптимізації коду з доступу до даних (Data Access Layer, DAL), з якого генеруються SQL запити до СКБД, шляхом використання проекцій [7];
- *Алгоритм асинхронного обміну в клієнт серверній архітектурі за допомогою технології SignalR*. Алгоритм асинхронного обміну вирішує проблему оперативного інформування підключених користувачів про зміни стану (рівень води, заряд батареї, користувацькі повідомлення, тощо) на веб-сторінці реалізованого додатку. При його належній реалізації Користувач web-додатку не зобов'язаний оновлювати сторінку для отримання нових даних, а навпаки – сторінка повинна миттєво сповіщати користувача про наявність нової інформації [8].

В заключенні можемо з упевненістю сказати, що даний проект має обширний функціонал, за допомогою якого адміністратор сайту може керувати роботою пристроїв інтернету речей (IoT), керувати користувачами, переглядати логи, які були сформовані пристроями впродовж деякого проміжку часу та виконувати інші дії.

2.2. Теоретичне обґрунтування появи аномальних даних

У контексті описаної IoT системи моніторингу водних резервуарів можна спостерігати різні типи аномалій у відповідних масивах даних. Ці аномалії можна розділити на такі категорії:

- Раптові зміни рівня води: ці аномалії можуть виникати, коли відбувається раптове підвищення або зниження рівня води, що може свідчити про несправність водяного насоса або проблеми з водопостачанням.

- Постійно низький рівень води: ці аномалії можуть виникнути, коли рівень води залишається постійно низьким, що може свідчити про витік у резервуарі для води або проблему з системою розподілу води.
- Постійно високий рівень води: ці аномалії можуть виникнути, коли рівень води залишається стабільно високим, що може свідчити про проблему з резервуаром для води або проблему з водопостачанням.
- Незвичайні явища при використанні води: ці аномалії можуть виникнути, коли у споживанні води спостерігаються незвичні дії, наприклад раптові стрибки або падіння споживання води. Ці моделі можуть вказувати на проблему з худобою або проблему з системою розподілу води.

Ми можемо використовувати ті самі алгоритми машинного навчання для виявлення аномалій як у даних рівня води так і у даних заряду батареї. Насправді було б корисно проаналізувати обидва набори даних разом, оскільки вони, ймовірно, пов’язані, і будь-які аномалії в одному наборі можуть свідчити про проблему з іншим.

Наприклад, якщо рівень заряду батареї постійно низький, це може означати, що пристрій не працює або датчик споживає більше енергії, ніж очікувалося. Подібним чином, якщо рівень води стабільно низький або високий, це може означати, що насос не працює належним чином, що може бути пов’язано з низьким рівнем заряду батареї.

Використовуючи алгоритми машинного навчання для виявлення аномалій у даних рівня води та рівня заряду акумулятора, ми можемо використовувати один алгоритм для обох або використовувати різні алгоритми для кожного. Найважливіше – переконатися, що використання відповідних алгоритмів дозволить вирішити проблему з виявленням рідкісних подій.

Можна додати фактор відстеження кількості отриманих та відправлених байтів інформації, як додатковий інформаційний фактор про продуктивність системи. Аналізуючи дані про тривалість або інтенсивність мережевої роботи IoT пристроїв в поєднанні з даними про рівень води та рівень заряду батареї, ми можемо виявити закономірності та кореляції, які можуть бути не очевидними

при окремому аналізі даних. Один із можливих способів представити тривалість з’єднання – це відсоток часу, протягом якого пристрій було підключено до мережі протягом певного періоду часу.

Крім того, включення фактора оцінки тривалості підключення також може підвищити загальну стійкість і надійність моделі, надаючи додатковий вимір для виявлення аномалій. Враховуючи кілька факторів, модель може бути більш стійкою до хибно-позитивних і хибно-негативних результатів і визначати ширший діапазон аномалій.

Отже, загальні етапи розробки алгоритму виявлення аномалії в нашому випадку:

- Збір даних: виконуємо збір історичних даних про рівень води, рівень заряду батареї та кількість отриманих та відправлених байтів IoT пристроїв за певний проміжок часу;
- Попередня обробка даних: виконання попередньої обробки даних, щоб усунути відсутні значення та будь-які інші проблеми, які можуть вплинути на продуктивність алгоритму. Крім того, необхідно нормалізувати або масштабувати дані, щоб переконатися, що вони мають формат, який може використовуватися алгоритмом;
- Розробка програмних функцій: вибір відповідних масивів з даних, які підходять для виявлення аномалій. Такими масивами можуть бути дані про рівень води, рівень заряду батареї, термін служби підключення;
- Вибір моделі: вибір відповідного алгоритму для виявлення аномалій на основі масивів даних;
- Навчання моделі: Навчання вибраного алгоритму на попередньо оброблених даних за допомогою вибраних масивів даних;
- Налаштування моделі: на основі результатів налаштовуємо параметри моделі, щоб покращити її продуктивність;
- Розгортання моделі: інтеграція остаточної моделі у систему моніторингу.

2.3. Аналіз роботи пристроїв IoT щодо генерації даних

Впродовж тривалої експлуатації пристроїв IoT, як правило, виникають разного роду явища, що можуть певним чином вплинути на збір інформації від пристрою. До таких явищ можна віднести, для прикладу, природні явища, які ніколи не були і не будуть стабільними на нашій планеті. Саме через зміну погодних факторів пристрій спроможний виконувати збір унікальних екземплярів даних, які фактично не можна класифікувати як аномальні дані.

Досліджуючи масиви даних, що містять інформацію про рівень води в резервуарі, рівень заряду пристрою, кількість отриманих байтів від пристрою, було виявлено підозрілу активність, яку складно віднести до аномальної.

Для прикладу, розглянемо масив даних, що містить інформацію про рівень води. Як зазначалося раніше – цей масив даних складається з інформації, яка є нестабільною переважно через погодні умови. Аналізуючи графіки, було виявлено нестандартну поведінку пристрою, яка, як виявилося, виникала через пориви вітру в момент зчитування даних з датчика. Як наслідок, пориви вітру сприяли створенню хвиль в резервуарі. Водночас з цим пристрій виконував збір статистичних даних, котрі, по суті, можна вважати за унікальні екземпляри даних (*див. рис. 2.2*).

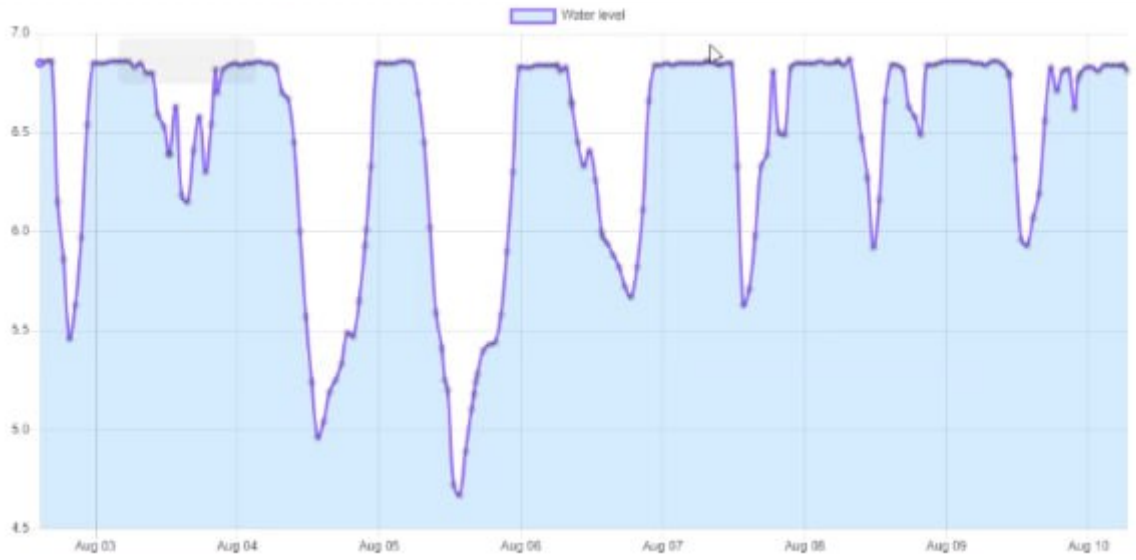


Рис. 2.2. Візуалізація даних від пристрою IoT

Якщо більш детально розглянути *рис. 2.2*, то можемо помітити різки всплески рівня води. Це можна пояснити тим, що резервуар має великий діаметр і пристрій виконує збір інформації в період часу, коли дме вітер. Вітер в свою чергу створює хвилі в резервуарі, а пристрій саме в цей момент проводить вимірювання рівня води. Таким чином ми отримуємо “спотворені” дані від пристрою, котрі можна по-різному класифікувати та ідентифікувати. Також варто зазначити, що ці дані впливають на точність виявлення аномалій у вибірці даних.

В якості наступного прикладу розглянемо ще один графік, який пов’язаний з рівнем води та демонструє “ідеальну” поведінку пристрою *(див рис. 2.3)*.

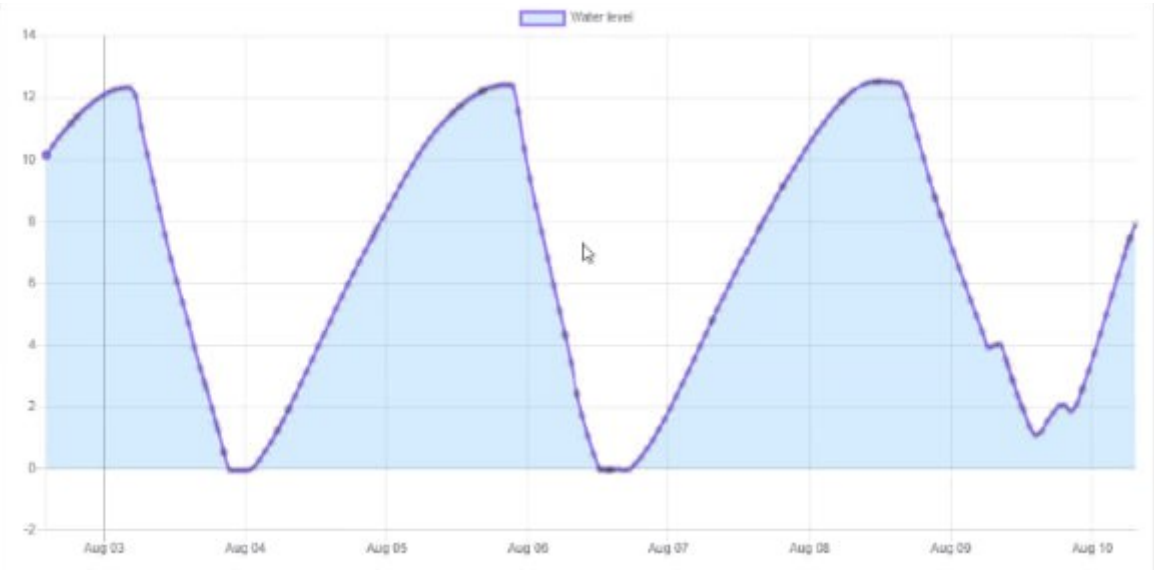


Рис. 2.3. Візуалізація “ідеального” збору інформації пристроєм IoT

Аналізуючи вищенаведений графік *(див. рис. 2.3)*, можемо зробити висновок, що до ідеального набору даних можна віднести такий набір даних, який містить в собі дані з прив’язкою до часу, котрі в свою чергу можуть формувати щось на кшталт правильної синусоїди. Це може означати те, що робота пристрою IoT та механізмів спустошення/наповнення резервуару працює належним чином. Графік, який містити таку синусоїду може сигналізувати про те, що при спустошенні резервуару вмикається водяний насос, який впродовж тривалого часу проводить наповнення резервуару до

певної мітки. Коли рівень води в резервуарі досягає цієї мітки – насос припиняє наповнення резервуару. Далі датчики очікують виявлення нижньої межі рівня води, після чого подають сигнал на ввімкнення насосу, який проводить повторне наповнення резервуару. Всі ці дії є циклічними, а це означає, що в певні проміжки часу мають бути сформовані дані, котрі спроможні утворити правильну синусоїду, або щось наближене до неї.

2.4. Приклади аномальних даних від пристроїв IoT

Через певний час, після запуску системи TankToad, на різноманітних графіках системи окремих пристроїв стало помітно нетипові сплески або прогалини в надсиланні даних. Як приклад, розглянемо деякі з них.

На *рис. 2.4 - 2.5* зображено статистичні дані від пристрою інтернету речей. Проаналізувавши наведені рисунки можемо побачити, що рівень води в резервуарі та рівень заряду пристрою не завжди є стабільним, іноді може бути різко високим, іноді навпаки.

Рис. 2.4. Візуалізація статистичних даних про рівень води, отриманих від пристрою інтернету речей АС-45

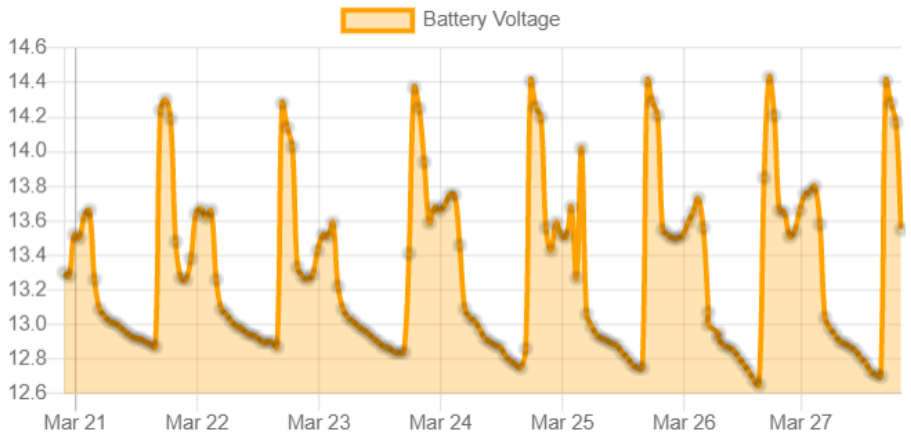


Рис. 2.5. Візуалізація статистичних даних про рівень заряд батареї, отриманих від пристрою інтернету речей AC-45

На *рис 2.6* відображено кількість відправлених та отриманих байтів впродовж певного проміжку часу від цього ж пристрою.

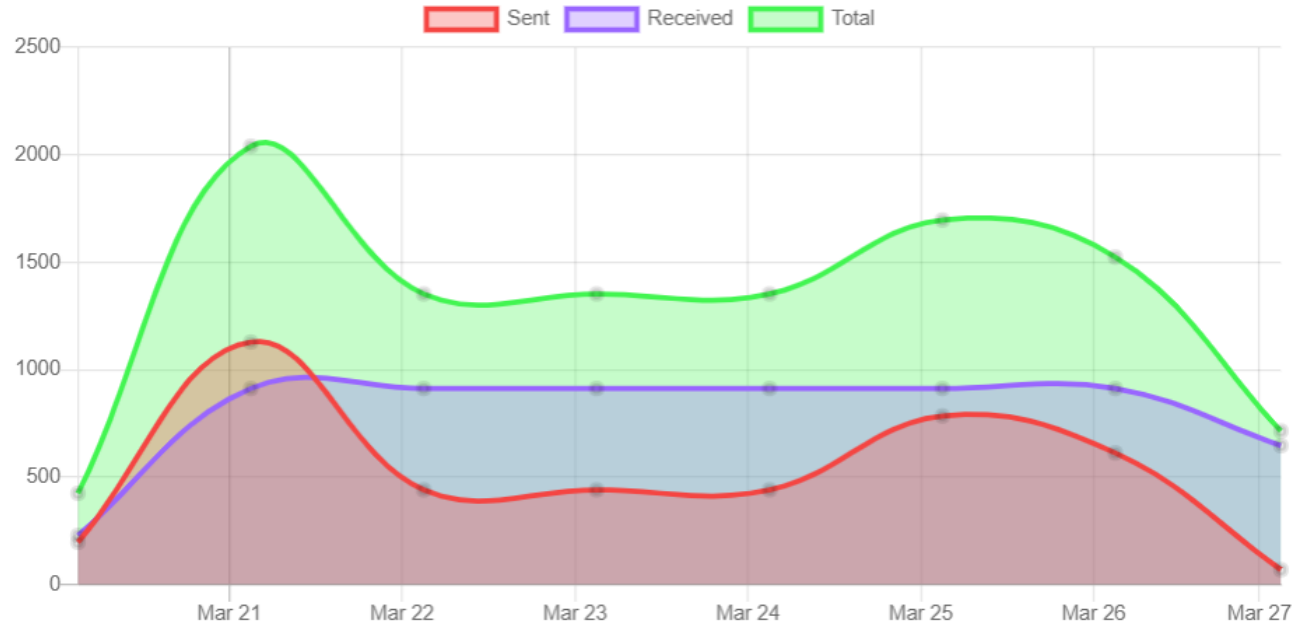


Рис. 2.6. Кількість відправлених та отриманих байтів від пристрою інтернету речей AC-45 за одиницю часу

Аналогічним чином можемо проаналізувати інший пристрій інтернету речей, який, ймовірно, містить аномальні дані *(див. рис. 2.7 - 2.8).*

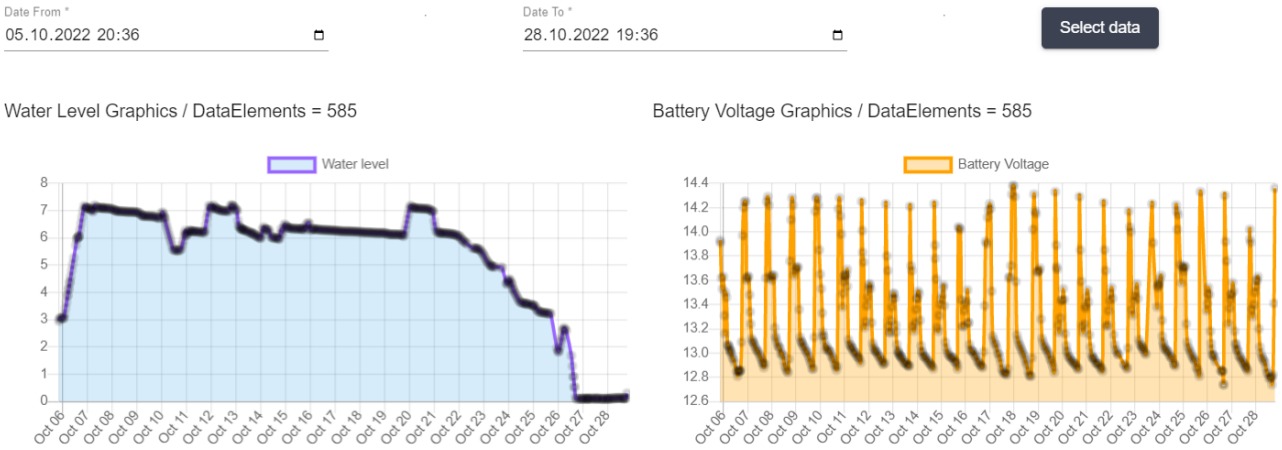


Рис. 2.7. Візуалізація статистичних даних, отриманих від пристрою інтернету речей АС-55

Проаналізувавши *рис. 2.7,* можемо зробити висновок про те, що даний пристрій інтернету речей мав негаразди в своїй роботі, або ж з резервуаром, з якого виконувався збір даних сталася якась проблема.

Тепер оглянемо кількість відправлених та отриманих байтів від цього пристрою (*див. рис. 2.8*).

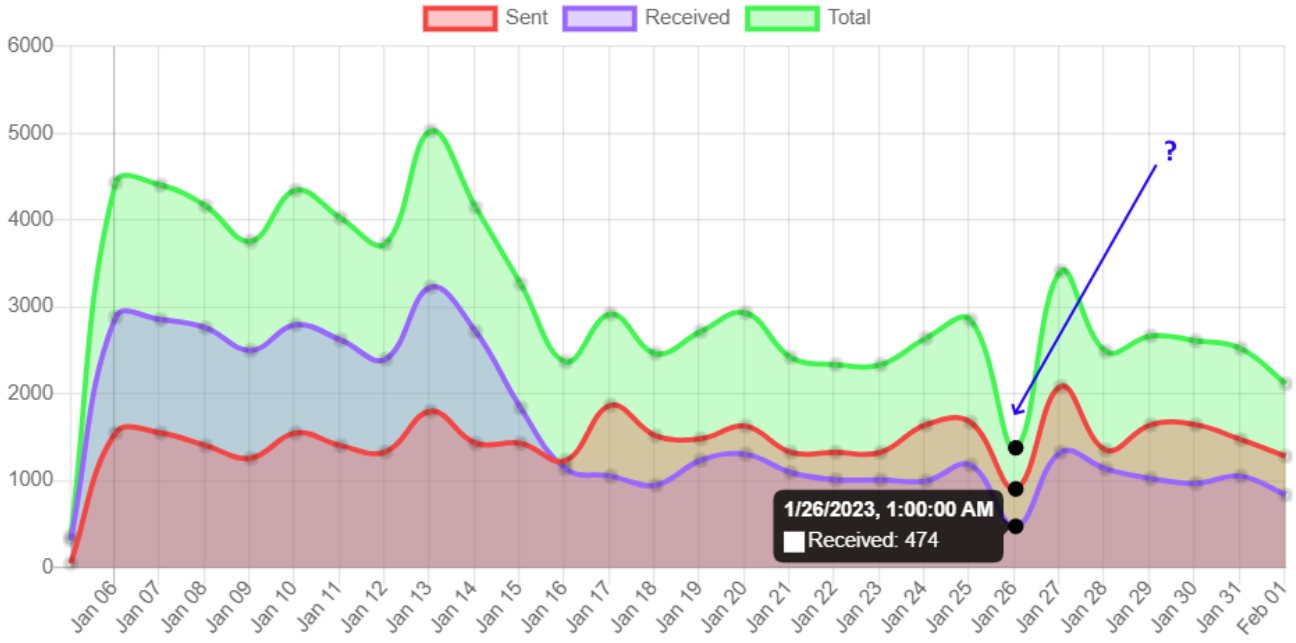


Рис. 2.8. Кількість відправлених та отриманих байтів від пристрою інтернету речей АС-55 за одиницю часу

Проаналізувавши вищенаведений графік можемо помітити, що орієнтовно 26 січня сталася неочікувана проблема з пристроєм. Судячи з графіку видно, що кількість отриманих і відправлених байтів різко знизилася в порівнянні з

іншими часовими рядами. Це свідчить про проблеми із зв'язком, які потрібно виправити. Причиною таких проблем може бути будь-який із факторів: негаразди в роботі оператора стільникового/супутникового зв'язку, невдале розташування антени, тимчасовий вихід з ладу внаслідок втручання сторонніх осіб, тощо.

Ознайомившись з принципом роботи пристроїв інтернету речей, можемо зробити висновок, що алгоритми виявлення аномальних даних можна застосувати для:

- виявлення аномалій у масиві, який містить інформацію про рівень води;
- виявлення аномалій у масиві, який містить інформацію про рівень заряду батареї;
- виявлення аномалій у масиві, який містить інформацію про кількість відправлених та отриманих байтів від пристрою.

2.5. Застосування алгоритмів виявлення аномалій в даних IoT пристроїв

Для більш точного виявлення аномальних даних серед великого набору інформації необхідно було провести тестування вже існуючих алгоритмів виявлення аномалій. В результаті тестування виокремили алгоритми, які є найбільш оптимальні в контексті нашого проекту. Ними стали алгоритми DBSCAN (Density-based spatial clustering of applications with noise) та IForest (Isolation forest).

Процес тестування алгоритмів виявлення аномальних даних дозволив ознайомитися з недоліками та перевагами алгоритму, який тестується, а також дослідити його тонкощі роботи.

Алгоритм DBSCAN може видавати неочікувані результати при роботі з великими обсягами даних наших IoT пристроїв. Даний алгоритм базується на методах, які використовують функції кластеризації даних, що в свою чергу

потребує велику кількість обчислювальних ресурсів. Для коректної роботи цього алгоритму з використанням великого масиву даних – потрібно виконати розбиття всього масиву даних на частини і тільки потім виявити аномальні дані. Далі необхідно знову провести ренормалізацію даних і так повторювати дії рекурсивно до тих пір, поки масив даних не буде повністю проаналізовано. Недоліком такого методу є те, що даний спосіб може давати немало помилкових викидів, що напряму відобразиться на точності виявлення. Навіть за невеликого масиву даних алгоритм DBSCAN може помилково позначити дані як аномальні, але в будь-якому випадку кількість правильно виявлених аномалій домінує над кількістю помилкових викидів.

Алгоритм IForest досить швидко виявляє аномальні дані наших IoT пристроїв навіть у великих наборах даних. Звичайно, що даний алгоритм може помилково позначити дані як аномальні, але в подальшому, як приклад, можна буде виконати класифікацію аномальних даних, провести повторне навчання і відфільтрувати виявлені аномалії. Це дасть змогу донавчити нашу систему виявлення аномалій.

Після завершення всіх необхідних тестувань, вибору відповідних наборів даних, на яких виконуватимуться тестування на реальних даних, було виконано реалізацію алгоритмів виявлення аномалій в інформаційній системі моніторингу водних резервуарів.

На даному етапі було застосовано алгоритми виявлення аномальних даних до таких масивів даних:

- масивів даних, які містять інформацію про рівень води;
- масив даних, який містить інформацію про рівень заряду батареї;
- масив даних, який містить інформацію про кількість відправлених та отриманих байтів від пристрою.

Пристрої, до яких було застосовано алгоритми виявлення аномальних даних:

- Мобільний пристрій інтернету речей. Дані передаються GSM каналами зв'язку. Серійний номер пристрою – AC-55;

- Супутниковий пристрій інтернету речей. Дані передаються супутниковим каналом Iridium. Серійний номер пристрою – AS-8.

Після того як всі вхідні масиви та алгоритми було обрано та реалізовано – перейшли до проведення фінальних тестувань.

Для реалізації поставлених цілей було використано мову програмування Python та ряд додаткових бібліотек. За основу реалізації алгоритмів виявлення аномальних даних було взято алгоритми з бібліотек adtk та sklearn. Встановлення зв'язків програми з базою даних було виконано за допомогою бібліотеки pyodbc.

2.6. Тестування роботоздатності модуля виявлення аномалій та аналіз отриманих результатів

Для початку проаналізуємо пристрій АС-55. На *рис 2.9 – 2.10* наведено виявлення аномальних даних з використанням алгоритмів IForest та DBSCAN. Червоним кольором на всіх рисунках позначено виявлені аномальні дані.

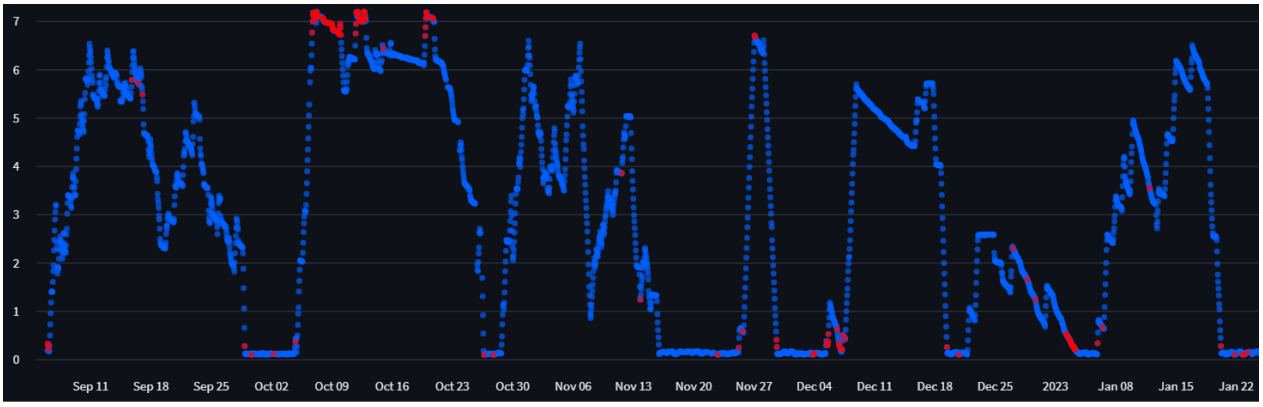


Рис. 2.9. Виявлення аномальних даних в масиві даних про рівень води для пристрою АС-55 (червоний колір позначає аномальні дані)

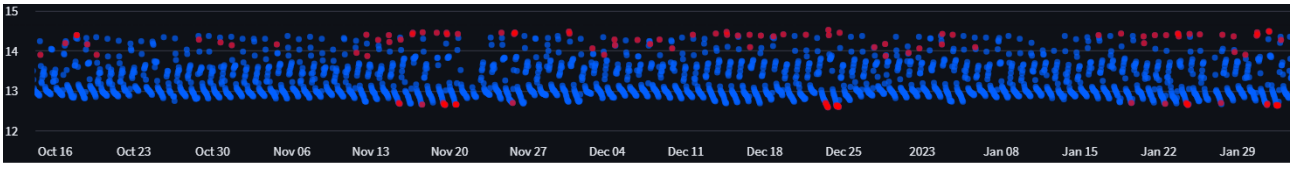


Рис. 2.10. Виявлення аномальних даних з використанням масиву даних про рівень заряду батареї для АС-55 (червоний колір позначає аномальні дані)

Рис. 2.11. демонструє результати виявлення аномалій в масиві даних про кількість відправлених та отриманих байтів пристрою АС-55.

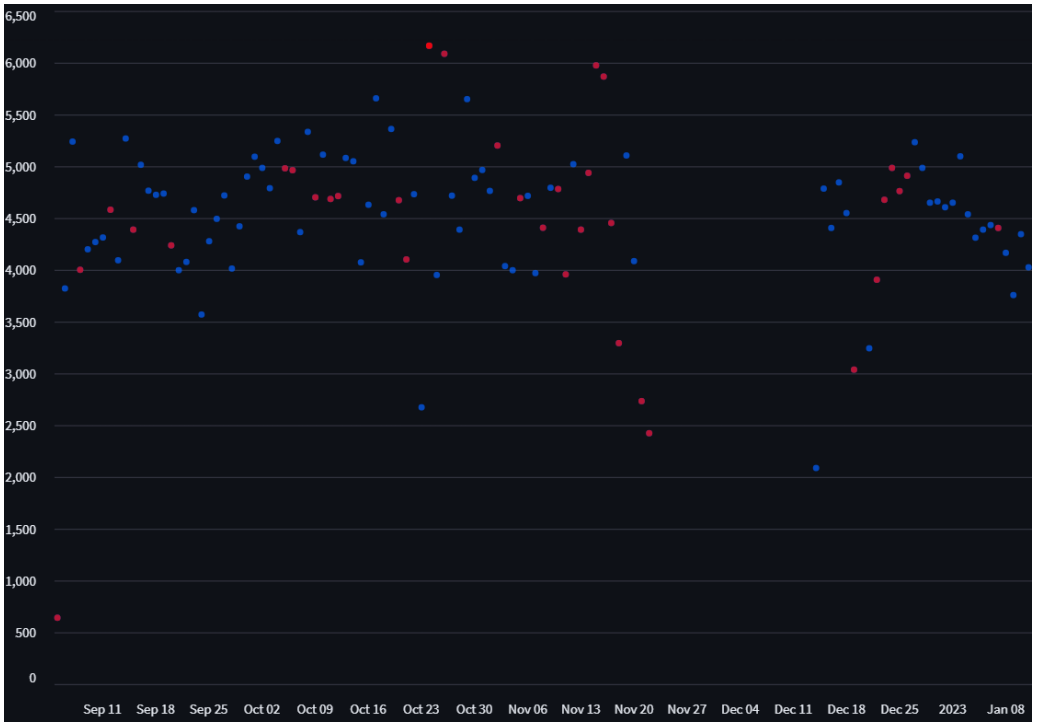


Рис. 2.11. Виявлення аномальних даних з використанням масиву даних про кількість відправлених та отриманих байтів пристрою АС-55 (червоний колір позначає аномальні дані)

Тепер спробуємо дослідити коректність виявлення аномальних даних. Для цього використаємо графіки з системи моніторингу TankToad (*див. рис. 2.12 -2.14*).

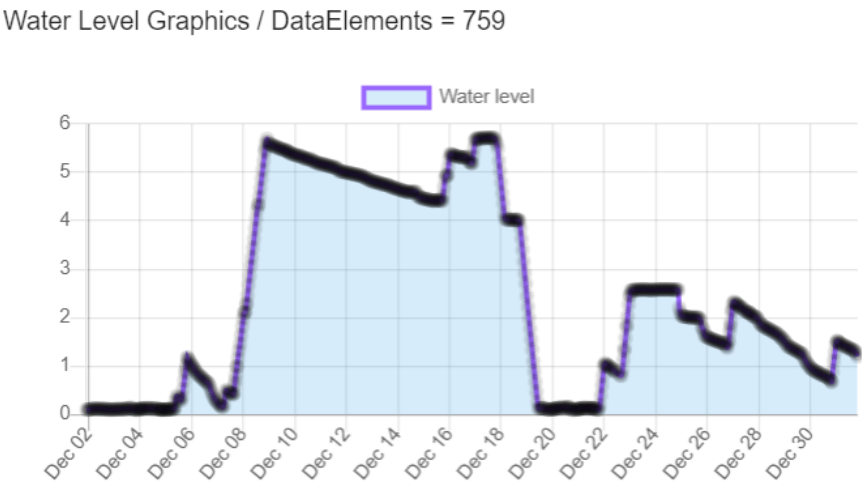


Рис. 2.12. Інформація про рівень води пристрою АС-55

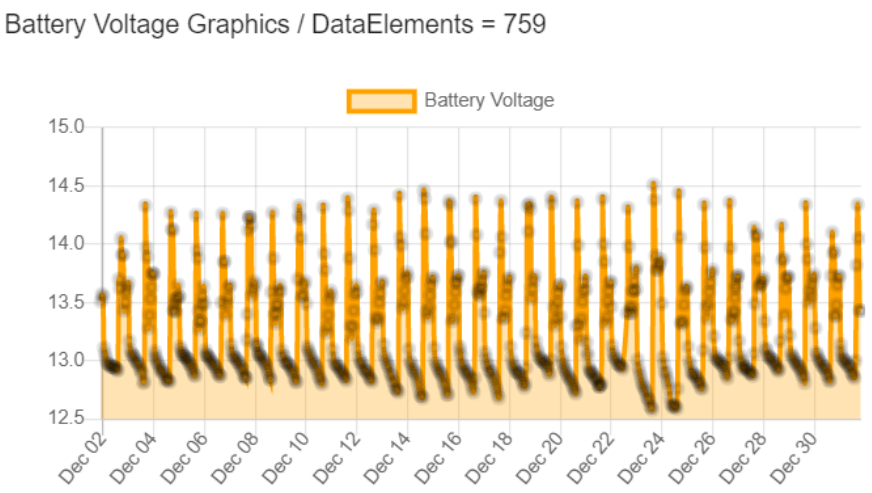


Рис. 2.13. Інформація про рівень заряду батареї пристрою AC-55

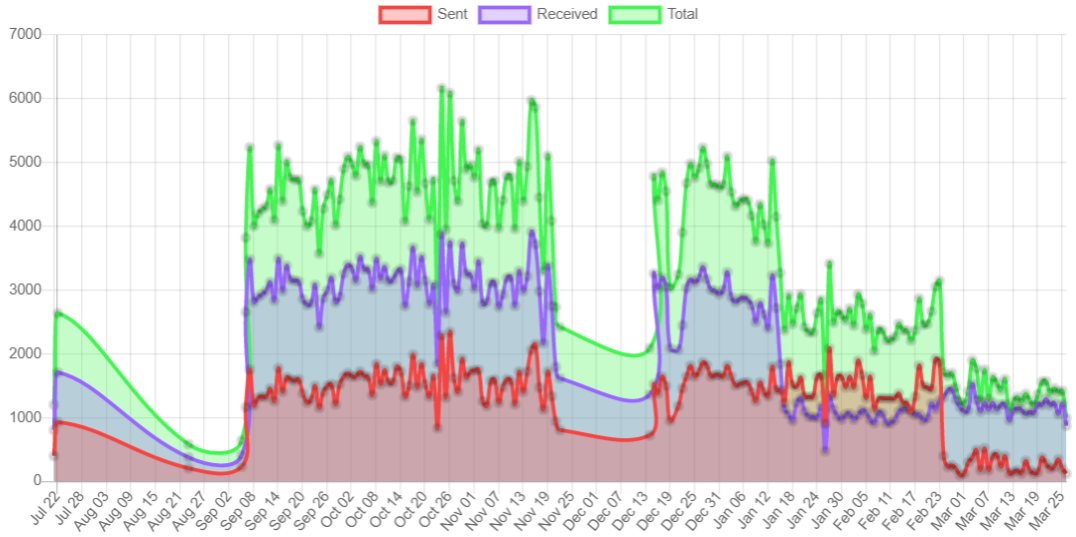


Рис. 2.14. Інформація про кількість відправлених та отриманих байтів пристрою AC-55

Аналіз рівня води. Розглянемо *рис. 2.9* та *рис 2.12.* На цих графіках відображено інформацію про рівень води (відображено як аномальні, так і нормальні дані). Проаналізувавши ці два рисунки можемо зробити висновок, що рівень води може варіюватися в різних межах, на нього може впливати велика кількість факторів, тому виявлення аномальних даних для цього масиву даних може бути досить не точним. Незважаючи на ці фактори алгоритмом виявлення аномальних даних вийшло зафіксувати суттєві зміни в поведінці цих даних, але не всі виявлені викиди є правдивими.

Аналіз рівня заряду батареї. Проаналізуємо *рисунки 2.10* та *2.13* бачимо, що рівень батареї відносно стабільний на протязі всього життєвого циклу

пристрою IoT, але все ж алгоритми IForest і DBSCAN виявили аномальні дані. Ці аномалії можуть бути пов’язані з перезарядом (більше 14 Вольт) або розрядом пристрою (нижче 12.7 Вольт). Тому ми бачимо, що знайдені аномалії в основному розміщені на верхніх та нижніх межах графіка.

Аналіз кількості отриманих та відправлених байтів. Проаналізувавши вищенаведені графіки (*рис 2.11* та *рис 2.14*) можемо помітити, що виявлення аномальних даних за період від ~25 листопада 2022 року до 13 грудня 2022 року було знайдено правильно. Адже в цей період часу даних від пристрою не було отримано, або вони були помилкові. Інші аномальні дані серед цього масиву також не є помилковими, тому що згідно *рис 2.14* можемо помітити суттєві зміни в рівнях масиву отриманих та відправлених байтів. Це свідчить про наявність певних проблем як апаратного так і програмного характеру.

Далі виконаємо аналіз даних від пристрою AS-8. На *рис 2.15 – 2.17* наведено виявлення аномальних даних з використанням алгоритмів IForest та DBSCAN.

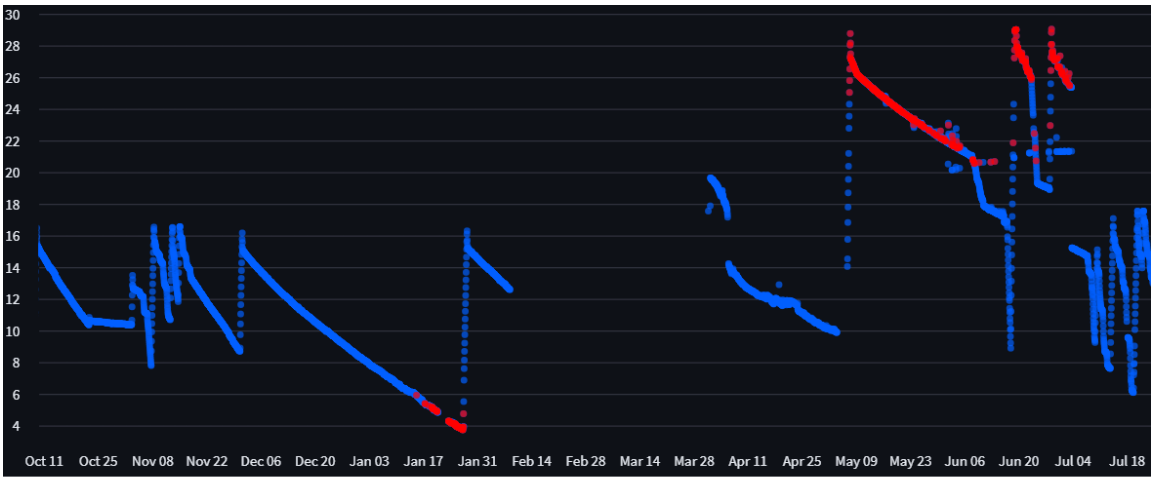


Рис. 2.15. Виявлення аномальних даних з використанням масиву даних про рівень води для пристрою AS-8 (червоний колір позначає аномальні дані)

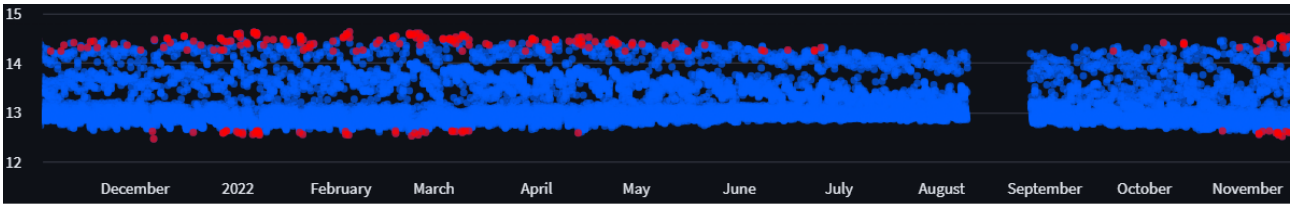


Рис. 2.16. Виявлення аномальних даних з використанням масиву даних про рівень заряду батареї пристрій AS-8 (червоний колір позначає аномальні дані)

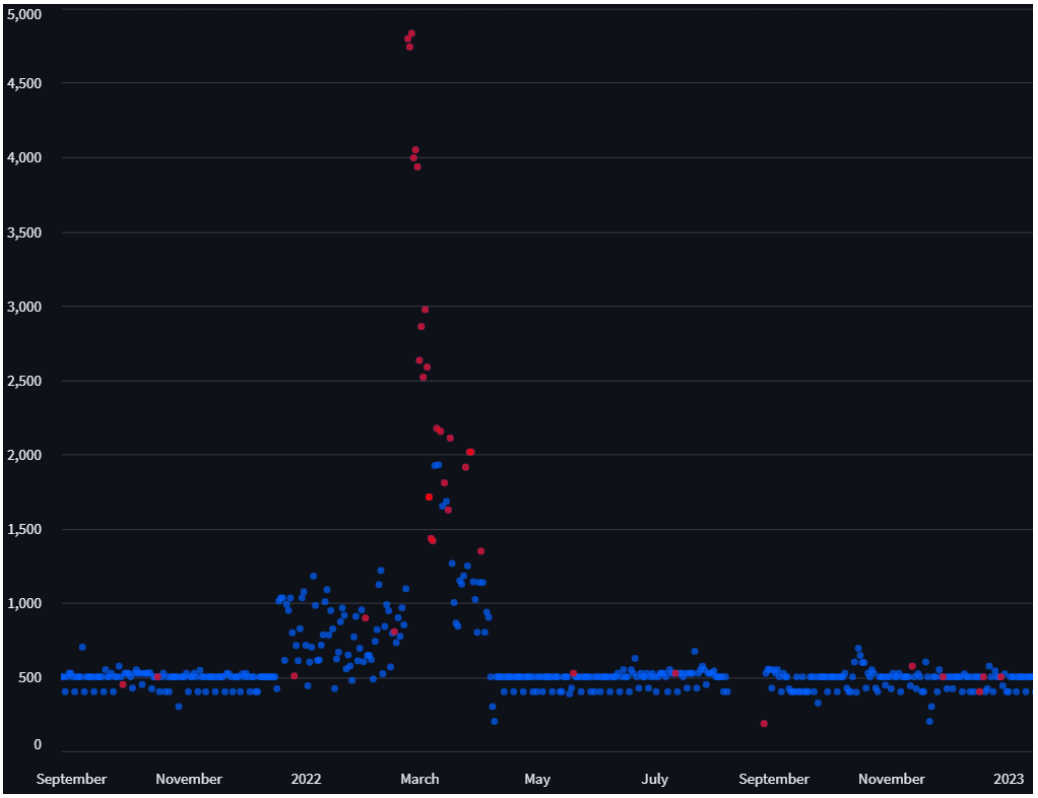


Рис. 2.17. Виявлення аномальних даних з використанням масиву даних про кількість відправлених та отриманих байтів для пристрою AS-8 (червоний колір позначає аномальні дані)

Наступним кроком є виконання збору статистичних даних з системи моніторингу *(рис. 2.18 – 2.20)*.

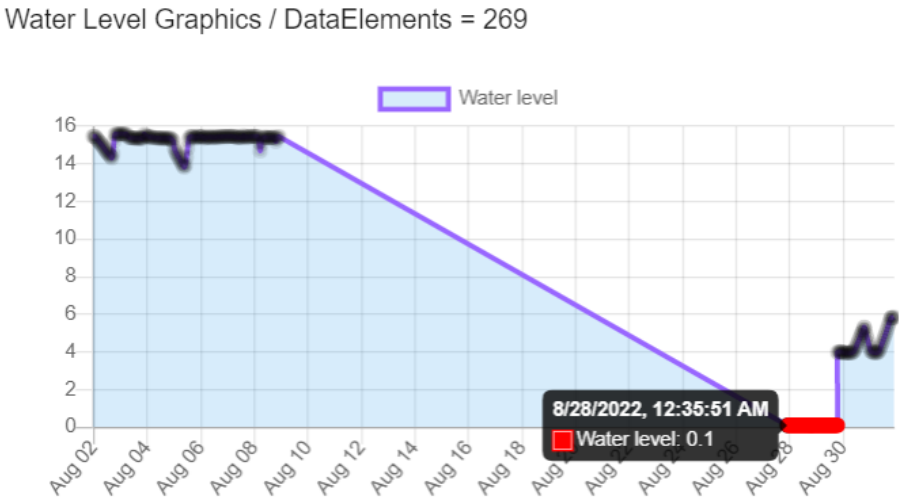


Рис. 2.18. Інформація про рівень води пристрою AS-8

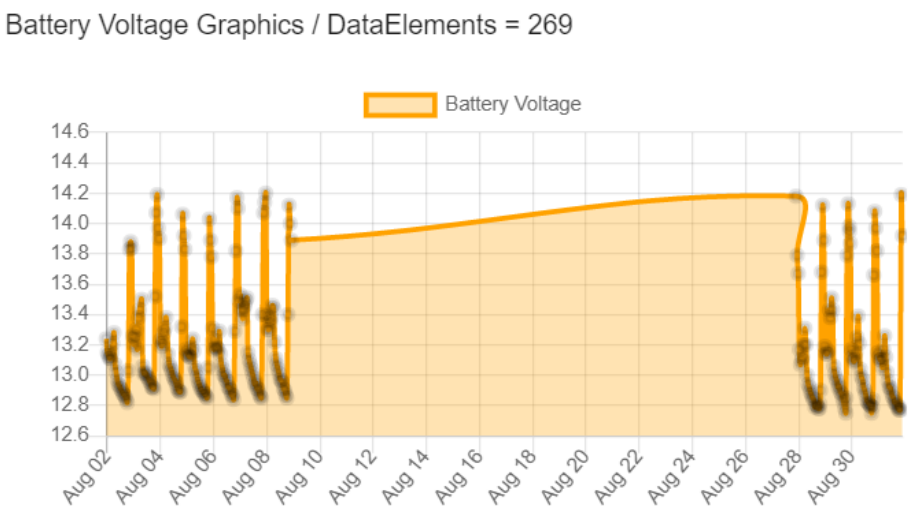


Рис. 2.19. Інформація про рівень заряду батареї пристрою AS-8

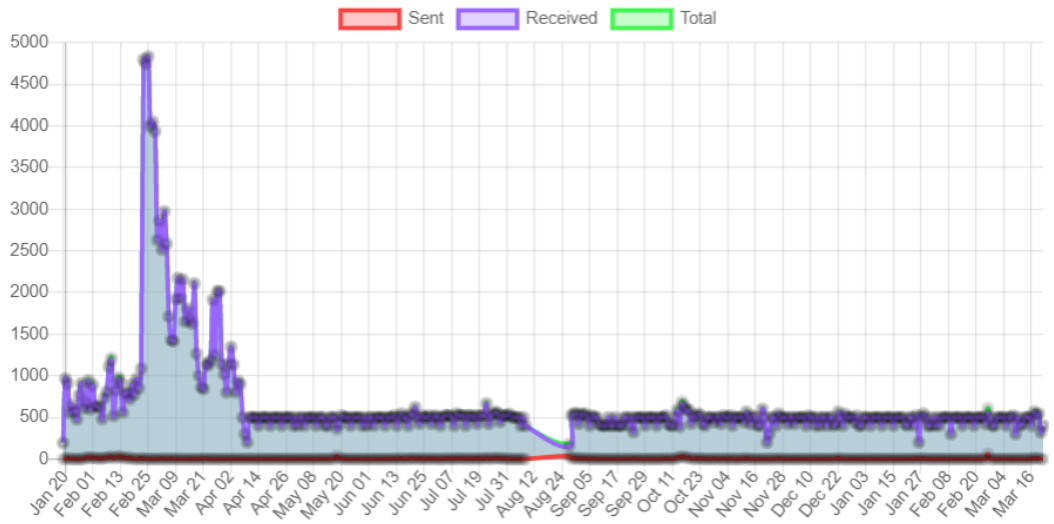


Рис. 2.20. Інформація про кількість відправлених та отриманих байтів від пристрою AS-8

Аналіз рівня води. Аналізуємо графіки, які зображені на *рисунках 2.15 та 2.18.* Можемо помітити, що знову ж таки, виявлення аномалій для рівня води є неточним, тому що рівень води може досить стрімко змінюватися, але не дивлячись на це дуже різкі перепади значень алгоритми виявили успішно. Один з таких перепадів був 28/08/2022.

Аналіз рівня заряду батареї. Розглядаємо *рисунки 2.16 та 2.19*. Бачимо, що алгоритми виявили аномальні дані лише на верхніх та нижніх межах масиву даних. Коли пристрій дані не надсилав – алгоритми не класифікували ці події як аномальні дані, що є не досить добре.

Аналіз кількості отриманих та відправлених байтів. Проводимо аналіз *рисунків 2.17 та 2.20*. Можемо помітити, що алгоритми виявлення аномалій правильно виявили різке підвищення трафіку від пристрою, що може сигналізувати про неправильну роботу пристрою. Інші аномальні дані, які позначені червоним кольором, переважно не є помилковими, тому що в основному вони сигналізують саме про різкі падіння значень, отриманих від пристрою.

В результаті завершення інтеграції було виявлено, що в даному випадку алгоритми для виявлення аномалій краще застосовувати для масивів даних, які містять інформацію про рівень заряду батареї (BatteryVoltage) та кількість отриманих та відправлених даних (TotalBytes). Саме при аналізуванні цих масивів даних алгоритми здатні більш коректно виявляти рідкісні екземпляри даних, що свідчать про проблеми в роботі та вимагають уваги адміністратора.

РОЗДІЛ ІІІ. ІНТЕГРАЦІЯ АЛГОРИТМІВ ВИЯВЛЕННЯ АНОМАЛІЙ ДО СИСТЕМИ МОНІТОРИНГУ ВОДНИХ РЕЗЕРВУАРІВ

3.1. Постановка завдання

Базуючись на функціоналі вже наявної системи моніторингу, її базі даних клієнтів, а також цілодобовим збором даних – постає потреба в інтеграції реалізованого модуля виявлення аномалій до вже існуючої системи моніторингу водних резервуарів. Подібне розширення функціоналу системи сприятиме користувачам в адмініструванні та аналізі поведінки пристрою без потреби запускати окремі файли, які відповідають за отримання необхідних масивів інформації.

Раніше проведені тестування модуля виявлення аномалій було виконано виключно з використанням мови Python та бібліотеки streamlit, за допомогою якої, власне, створювався окремий веб-сервер, що полегшував розробку алгоритмів в межах лише одного середовища.

Після проведення всіх належних тестувань реалізованого модуля виявлення аномалій з використанням окремого веб-серверу – необхідно виконати об’єднання двох окремих проектів, а саме: модуля виявлення аномалій та системи моніторингу водних резервуарів.

Систему водних резервуарів TankToad було реалізовано з використанням великої кількості засобів, бібліотек та модулів. Дана система складається з Frontend та Backend частин, котрі взаємопов’язані між собою.

Клієнтська частина проекту, іншими словами Frontend, розроблена з використанням платформи Node.js, мов програмування TypeScript та JavaScript, мови веб-розмітки HTML5 і мов стилів SCSS та CSS. Всі раніше перелічені засоби працюють в поєднанні з веб-фреймворком Angular (Angular TypeScript).

Angular — написаний на TypeScript front-end фреймворк з відкритим кодом, який розробляється під керівництвом Angular Team у компанії Google, а також спільнотою приватних розробників та корпорацій [13]. Даний фреймворк

постійно оновлюється, актуалізується та водночас є потужним засобом для розробки веб-додатків. Цей фреймворк базується на платформі Node.js, котра в свою чергу тісно пов’язана з менеджером пакунків npm. Менеджер пакунків npm містить незліченну кількість публічних та приватних пакетів, які при належному використання здатні полегшити розробку веб-додатку. Для прикладу, Angular Material є одним з пакунків цієї платформи.

Angular Material використовується в проекті системи моніторингу водних резервуарів для полегшення розробки. До ключових його особливостей можна віднести ряд модулів, які включають в себе вже готові компоненти, директиви, бінди та стилі для роботи з різними веб-елементами. Окрім даного пакунку в проекті TankToad використовуються й ряд інших, які необхідні для належного функціонування системи в цілому.

Серверна частина проекту (Backend) використовує технологію ASP.NET. ASP.NET є технологією створення веб-застосунків та веб-додатків, розробником якої є компанія Microsoft. Вона є складовою частиною платформи Microsoft.Net. Розробники веб-застосунків можуть побудувати серверну логіку з використанням будь-яких мов програмування, що належать до комплекту .NET Framework. До цього комплекту належать такі мови програмування: C#, JScript.NET та Visual Basic.NET.

Бекенд системи моніторингу водних резервуарів було розроблено з використанням мови програмування C# і ряду nuget пакунків. Бекенд проекту виступає проміжним каналом зв’язку між клієнтом та базою даних. Він контролює та обробляє всі запити до внутрішнього API. Однією з ключових особливостей бекенду є те, що він містить в собі механізм асинхронного конвейеру команд, який запускається регулярно і виконує ряд певних задач. Даний механізм дозволяє запускати виконуваний код в заздалегідь заданий розробником час. Одними з функцій, котрі виконує описаний раніше механізм асинхронного конвейеру команд є:

- Щотижневе надсилання діагностичних даних від пристроїв інтернету речей на пошти адміністраторів проекту;

- Списання кредитів з балансу користувачів;
- Інформування користувачів у разі виникнення проблем з пристроями;
- Щоденний підрахунок та класифікація великих наборів даних, отриманих від пристроїв та ін.

Система управління базами даних (СУБД) MS SQL Server керує всіма транзакціями, які надходять до бази даних проекту від бекенду. Адміністрування бази даних відбувається з використанням SQL Server Management Studio (SSMS), основним призначенням якого є полегшення взаємодії адміністратора бази даних з ядром СУБД. Вищевказана система управління базами даних працює з базою даних типу Transact-SQL (T-SQL).

Transact-SQL — процедурне розширення мови SQL, створене компанією Microsoft і Sybase. SQL був розширений наступними додатковими можливостями, такими як:

- Керуючі оператори;
- Локальні і глобальні змінні;
- Різні додаткові функції для обробки рядків, дат, математики, тощо.
- Підтримка аутентифікації Microsoft Windows.

Мова Transact-SQL є ключем до використання MS SQL Server. Всі застосунки, які взаємодіють з екземпляром MS SQL Server, незалежно від їхньої реалізації і інтерфейсу користувача, відправляють з сервера інструкції Transact-SQL [14].

Отже, підсумовуючи все вищесказане, можемо зробити висновок про те, що постає потреба у проведенні інтеграції модуля аномалій, котрий був реалізований на мові Python, до вже функціонуючого проекту, котрий розроблений з використанням інших засобів, відмінних від тих, котрі були використані у модулі виявлення аномальних екземплярів даних. На етапі інтеграції потрібно спрогнозувати та усунути можливі помилки, котрі можуть виникнути під час запуску окремого контрольованого бекендом процесу модуля виявлення аномалій.

3.2. Аналіз вже існуючих способів інтеграції алгоритмів до проектів побудованих на базі технології ASP.NET Core

Під час дослідження вже існуючих способів інтеграції алгоритмів розроблених з використанням мови Python до проектів, побудованих на базі технології ASP.NET, було знайдено декілька способів інтеграції, а саме:

1) Використання динамічних сценаріїв з використанням технології IronPython;
2) Використання динамічних сценаріїв з використанням технології Python.NET;
3) Використання вже вбудованих функцій ASP.NET для запуску окремого процесу.

Кожен з перелічених способів інтеграції має свої нюанси, недоліки та відмінності в реалізації. Розглянемо їх більш детально.

У разі реалізації динамічних сценаріїв з використанням технологій IronPython або Python.NET буде створено ізольоване середовище для запуску Python-скриптів. Подібного роду середовище має свої переваги та недоліки. Однією з переваг подібного рішення є те, що: ізольовані середовища створюють внутрішні Python-процеси в межах пулу одного веб-застосунку. В межах цього ж пулу вони обробляються та імплементують результат виконуваного Python-скрипта в додаток ASP.NET. З використанням подібного середовища можна виконувати повний контроль над створеним процесом через додаток ASP.NET. В ньому ж можна керувати проміжним етапом виконання скрипта та ін. Таким чином у разі виникнення проблем у роботі самого Python-скрипта – розробник матиме можливість обробити ці винятки в межах додатку ASP.NET. Що ж стосується недоліків подібних сценаріїв, то їх є декілька, а саме:

1) При ініціалізації запуску того чи іншого скрипта інсталяція всіх залежностей (модулів/бібліотек) буде проводитися з нуля, що в результаті створюватиме додаткове навантаження на серверну частину;

2) Сценарій інсталяції всіх залежностей скрипта потрібно прописувати окремими командами та запускати їх через відповідні засоби технологій IronPython або Python.NET.

Обробити перелічені вище недоліки можна по-різному, для прикладу можна виконати обмеження використання ресурсів процесора (CPU) конкретним процесом, а сценарії інсталяції залежностей забіндити.

Незважаючи на перелічені раніше недоліки ізольованих середовищ виконання скриптів, було взято до уваги ще один найважливіший “мінус” використання динамічних сценаріїв на базі засобів IronPython або Python.NET. Дані засоби рідко оновлюються, що в свою чергу викликає ряд додаткових проблем в роботі скрипта, для прикладу: IronPython на даний момент підтримує лише скрипти, котрі були розроблені на Python версії 3.4.x, в той час як модуль виявлення аномалій, котрий потрібно інтегрувати до системи моніторингу водних резервуарів використовує бібліотеки, котрі підтримуються лише в більш нових версіях середовищ виконання Python. Аналогічна ситуація з засобом Python.NET. Через недолік, котрий мають засоби виконання скриптів було прийнято рішення про проведення інтеграції розроблених алгоритмів виявлення аномалій за допомогою вже вбудованих функцій ASP.NET для запуску окремого Python процесу.

У випадку використання третього способу інтеграції Python-скриптів до додатку ASP.NET також було виявлено певні нюанси. До них можна віднести те, що процес, котрий створюється додатком ASP.NET не буде котролюватися пулом веб-додатку, тобто запускатиметься окремо від основного проекту. Це вимагає побудови додаткової логіки та механізмів для запуску Python-скрипта. Це можна пояснити тим, що помилки або коректні результати обробки великих наборів даних, які генеруватиме скрипт, потрібно вручну транслювати в додаток ASP.NET Core. Але незважаючи на це, при належній обробці даних та передбачуванні можливих помилок в процесі виконання, цілком можливо запустити модуль виявлення аномалій та інтегрувати його до системи моніторингу таким способом. Слід зазначити, що в порівнянні з ізольованими

середовищами запуску даний спосіб інтеграції не виконуватиме інсталяцію всіх бібліотек і залежностей при кожному запуску проекту, тобто ці модулі будуть інстальовані лише один раз. В свою ж чергу це зменшить навантаження на сервер, тому що не потрібно буде виконувати повторну інсталяцію пакунків.

Після проведення досліджень та аналізу вже існуючих способів інтеграції Python-скриптів до проектів, побудованих на базі технології ASP.NET, було прийнято рішення виконати інтеграцію розробленого модуля виявлення аномалій до системи моніторингу водних резервуарів з використанням вже вбудованих функцій ASP.NET для запуску окремого процесу.

3.3. Розробка Backend-частини проекту

Проаналізувавши більшість способів запуску Python-скриптів, а також взявши до уваги доступну кількість серверних ресурсів і кількість ресурсів, яких потребує модуль виявлення аномалій, було прийнято рішення про запуск розробленого модуля один раз на день. Запуск відбуватиметься тоді, коли навантаження на сервер є найменшим.

Для того, щоб виконати збір аномальних даних, необхідно дотримуватись такого порядку дій:

1. Виконати запуск модуля виявлення аномальних даних з використанням основного бекенду проекту;
2. Після того як модуль виявлення аномалій було запущено – виконується збір даних від пристроїв IoT. Модуль виявлення аномалій встановлює паралельне підключення до бази даних та отримує з неї необхідну для своєї роботи інформацію.
3. Відповідно до реалізованого способу запуску розроблений модуль надсилає виявлені аномалії в форматі JSON в командний рядок головного процесу та повідомляє про успішне/не успішне завершення скрипта. В свою ж чергу Backend обробляє всі ці винятки. У випадку успішного завершення скрипта і при знаходженні аномалій – бекенд конвертує

отримані результати в JSON формат. У випадку не успішного завершення скрипта, або ж якщо аномалій для пристрою не було виявлено – бекенд проекту автоматично відсіює помилки та зупиняє збір аномальних даних для конкретного девайсу.

4. Коли третій етап було успішно пройдено, тоді бекенд встановлює підключення з ядром СУБД і надсилає до нього відповідні команди для збереження виявлених аномалій в окрему таблицю.

Всі вищевказані дії виконуються асинхронно, тобто кожна наступна інструкція команд очікує завершення попередньої. На *рис. 3.1* проілюстровано послідовність дій, котрі виконує в основному бекенд проекту.

Використовуючи паралельні потоки обробки даних користувач може й надалі взаємодіяти з фронтендом проекту, навіть якщо в цей час виконується збір (виявлення) аномалій.

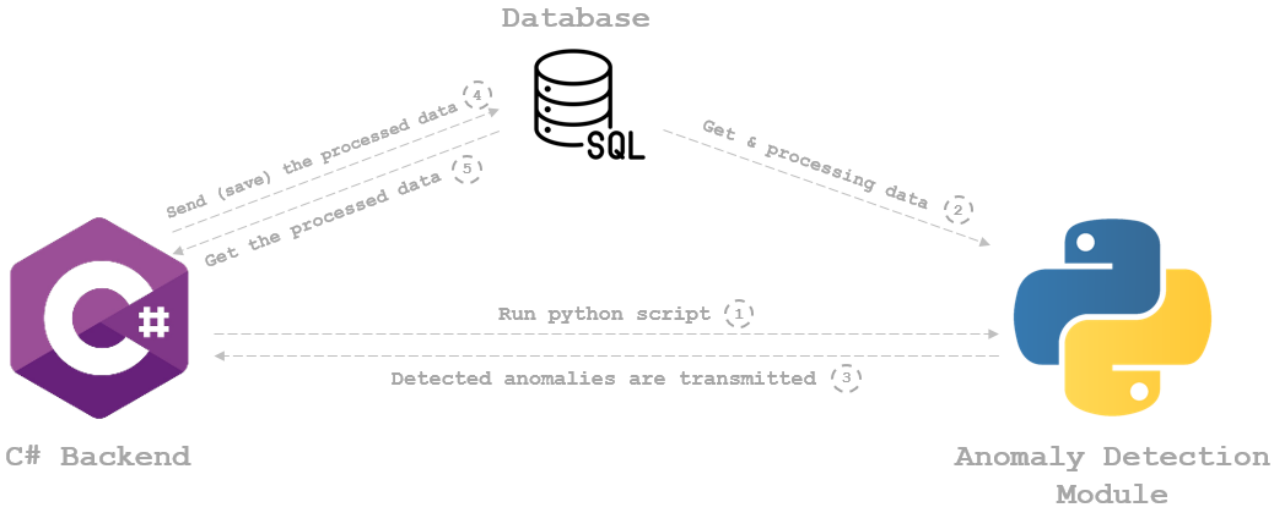


Рис. 3.1. Ілюстрація роботи модуля виявлення аномалій

Як зазначалося раніше – розроблений спосіб збору аномальних даних виконується лише один раз на день з використанням асинхронного конвейеру команд. Збір даних виконується для всіх пристроїв. Щодня в таблиці, котра містить масиви з аномальними даними, створюються нові поля, а не замінюються вже раніше знайдені за минулі дні аномалії.

Витягнення збережених раніше даних з таблиці виконується відповідно до потреб користувача, тобто якщо користувач взаємодіє з

Frontend-компонентою, яка відповідає за візуалізацію аномальних даних, тоді й формується POST-запит до API на отримання даних в JSON форматі. В тілі POST-запиту першочергово вказуються ID пристрою, для якого потрібно зібрати діагностичні дані, а також вказуються назви масивів даних з виявленими аномаліями, які фронтенд планує отримати в результаті успішної відповіді від бекенду.

Обробку сформованих користувачем запитів виконує контроллер бекенду. Він же проводить перевірку правильності формування запиту, а також обробку всіх винятків у разі відсутності потрібних даних в тій чи іншій таблиці бази даних. В тілі контроллера формується підключення до бази даних та до конкретної її таблиці. В даному випадку контроллер встановлює зв’язок з таблицею, яка містить в собі масиви з аномальними даними. Витягнення даних з таблиці виконується відповідно до вказаних на фронтенді назв масивів даних (ті, які фронтенд потенційно планує отримати). З конкретної таблиці отримуються останні записи (записи, котрі були створені раніше) для конкретного девайсу, за умови, що очікувані фронтендом назви масивів даних відповідають тим, котрі формуються модулем виявлення аномалій. Якщо потрібні масиви даних знайдено – контроллер їх отримує з бази даних, десеріалізує та відправляє на Frontend. На *рис 3.2* наведено фрагмент коду даного контроллера, котрий виконує раніше зазначений функціонал.

```
[HttpPost]
0 references
public async Task<IActionResult> PostAnomalyDataByDeviceId(PostDataAnomaliesModel AnomalyData)
{
    var AnomalyContent = new List<object>();
    foreach (var i in AnomalyData.FieldDetectAnomalies)
    {
        var a = _context.AnomalyDetection.Where(x => x.DeviceId == AnomalyData.DeviceId && i == x.AnomalyColName)
            .AsEnumerable().LastOrDefault();

        if (a != null)
        {
            var Dates = JsonConvert.DeserializeObject(a.Dates).ToString();
            var Values = JsonConvert.DeserializeObject(a.Values).ToString();
            string delimiter = ", ";

            var b = new
            {
                Anomalies = JsonConvert.DeserializeObject(a.Anomalies),
                Dates = Dates.Split(delimiter).ToList(),
                Values = Values.Split(delimiter).ToList(),
                a.AnomalyColName,
                a.DeviceId,
                a.CreatedDate,
            };

            AnomalyContent.Add(b);
        }
    }

    if (AnomalyContent == null)
    {
        return Ok(new
        {
            data = "No Content"
        });
    }
```

Рис. 3.2. Фрагмент коду метода контроллера бекенду

Подібна реалізація бекенд-частини дозволить користувачу швидко отримувати діагностичні результати для конкретного пристрою, тому що в момент формування запиту потрібно буде всього лиш витягнути та десеріалізувати декілька записів з бази даних, а не запускати модуль виявлення аномалій з нуля, котрий здатний формувати велике навантаження на серверну частину при великій кількості запитів до нього.

3.4. Розробка Frontend-частини проекту

Після завершення інтеграції розробленого модуля виявлення аномалій до серверної частини проекту, постає потреба в розробці веб-компонентів, з якими вже взаємодіятиме користувач. Загалом, для реалізації зручного та інтуїтивно зрозумілого функціоналу, було прийнято рішення про створення окремого розділу на сторінці певного девайсу. Даний розділ є окремою веб-компонентою, яка підвантажуватиме дані (надсилатиме запити до API) в залежності від потреб користувача *(див. рис. 3.3)*.

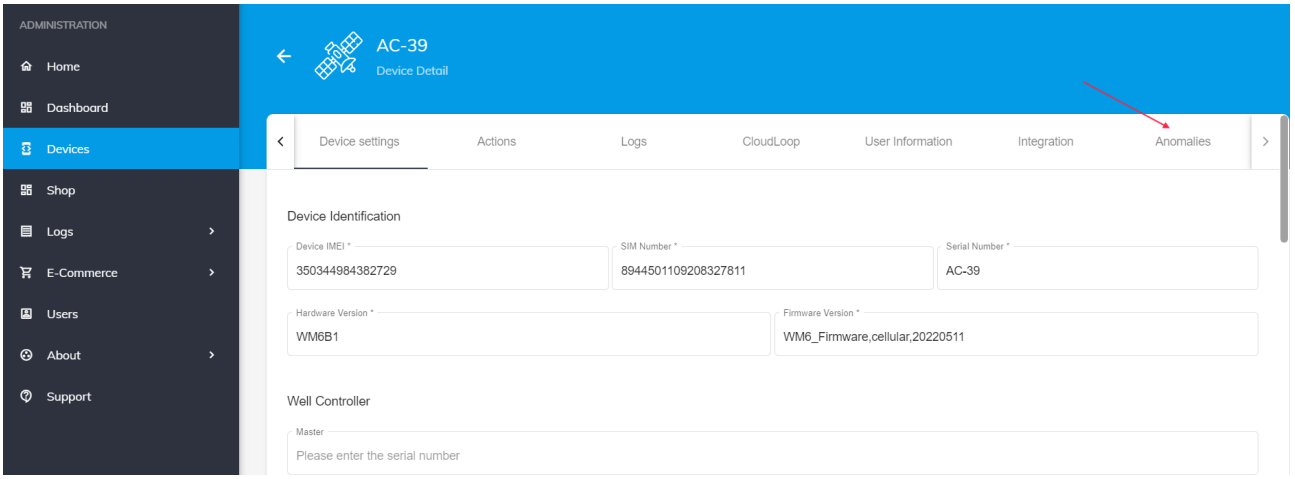


Рис. 3.3. Розміщення компонента по детекції аномалій

Отримання даних від бекенду відбувається через окремий сервіс-файл на Frontend-частині проекту. Даний сервіс-файл підвантажується при виклику з головного компонента виявлення аномалій. Для прикладу, на *рис. 3.4* продемонстровано рядки головного компонента, котрі виконують функції отримання та попередньої обробки даних отриманих від бекенду.

```
this.$subscribeRequest =
  this._anomalyDetectionService.getAnomalyDataByDeviceId(DeviceId, this.CheckAnomalies)
    .subscribe((res: any) => {
      this.AnomalyDetection = res == "No Content" ? [] : res

      if (this.AnomalyDetection.length != 0) {
        this.DataCollected = new Date(this.AnomalyDetection[0].CreatedDate)

        this.AnomalyDetection.forEach(x => {
          this.ActiveHolsts.push("graph")

          let anomalies = x.Anomalies
          anomalies = anomalies.sort((a, b) => new Date(b.date).getTime() - new Date(a.date).getTime())

          this.dataSources.push(new MatTableDataSource<any>(x.Anomalies))
        })
      }

      this.isLoadedContent = true
    }, err => {
      this.isLoadedContent = true
    })
```

Рис. 3.4. Фрагмент коду головного компонента

На фронтенд-частині розроблено два типи подання отриманих даних від бекенду. Якщо розглядати більш детально, то користувач матиме можливість взаємодіяти з вибірками даних за допомогою графіків та таблиць.

Веб-елементи (графіки та таблиці) динамічно формуються в залежності від кількості масивів з виявленими аномаліями, тобто, якщо модуль виявлення

аномалій виявив підозрілу активність в масиві даних, що містить інформацію про рівень води в певному резервуарі, тоді фронтенд сформує веб-елементи конкретно під цю інформацію, веб-елементи під пусті масиви не будуть сформовані та відображені користувачеві.

Графіки на Frontend-частині було розроблено з використанням додаткового модуля Chart.js. Даний модуль є досить зручним та зрозумілим у використанні. На сайті розробників даного модуля міститься велика кількість документації до нього, що дозволяє швидко дізнатися як саме потрібно працювати з ним та його функціями. Модуль відмінно виконує візуалізацію даних безпосередньо для користувача, а також дозволяє зберігати сформований графік у форматі окремого image-файлу у разі потреби. В якості прикладу наведено *рис. 3.5*, на якому відображено побудову графіка, що візуалізує дані про рівень води.

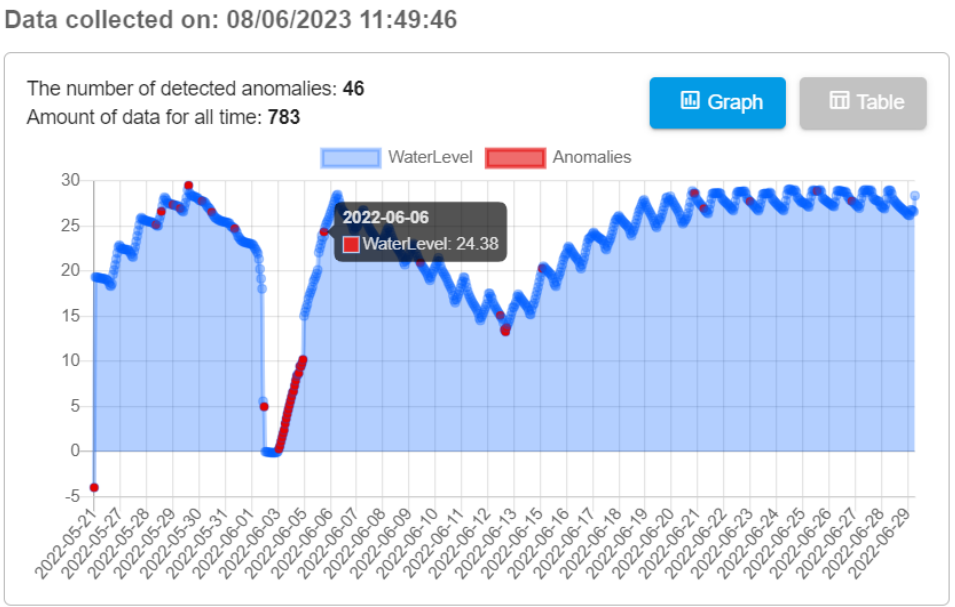


Рис. 3.5. Візуалізація виявлених аномальних даних про рівень води з використанням модуля Chart.js

У випадку, коли пристрій тривалий час функціонує він потенційно може генерувати дуже великі набори даних, тому було також розроблено табличне подання інформації. За допомогою табличного подання користувач може переглядати всі виявлені аномалії. Згенерована таблиця містить в собі лише аномальні дані, та відображає ключові поля (дата та значення), котрі потрібні

для аналізу поведінки пристрою. Приклад візуалізації виявлених аномалій з використанням табличного подання наведено на *рис. 3.6*.

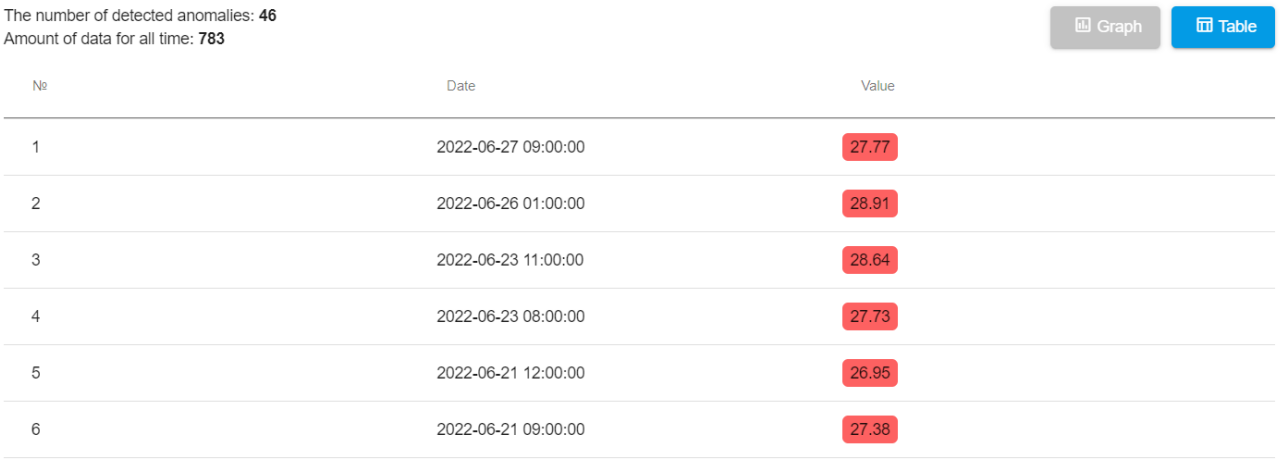


Рис. 3.6. Демонстрація виявлених аномальних даних з використанням табличного подання інформації

Аналогічним чином формуються й інші веб-елементи під масиви даних, котрі містять інформацію про рівень заряду батареї пристрою та кількість отриманих байтів від нього, за умови, що аномальні дані було виявлено. Таке формування веб-елементів пов’язане з тим, що на етапі нормалізації вхідної інформації всі дані було “підігнано” до одного вигляду, та реалізовано схожі правила та винятки для формування вихідної інформації від модуля виявлення аномалій. На *рис. 3.7* проілюстровано весь ключовий функціонал компонента.

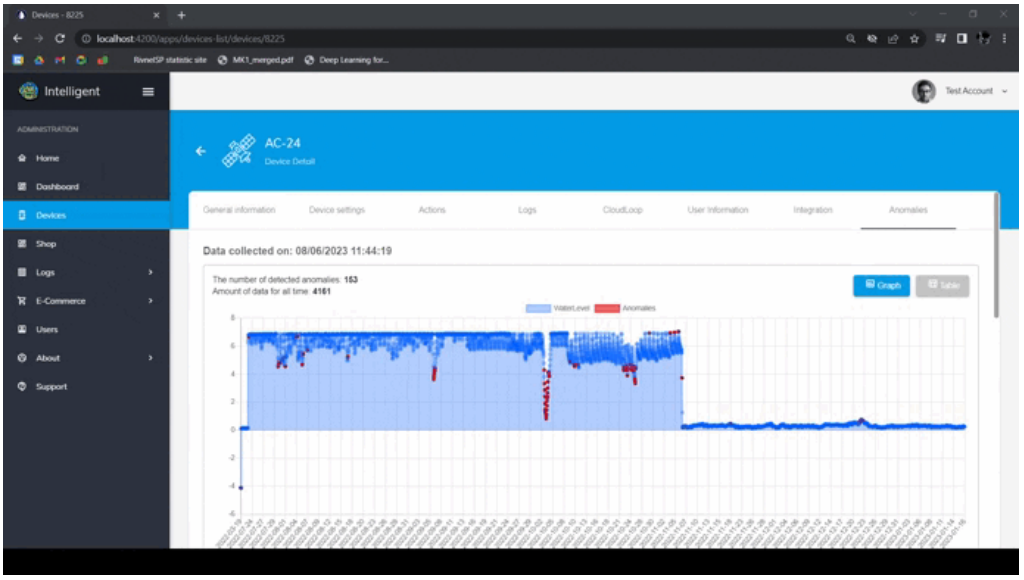


Рис. 3.7. Вигляд розробленого Frontend-компонента для модуля виявлення аномальних даних

Розроблений функціонал Frontend-частини проекту дозволить користувачам зручно переглядати діагностичні дані обраного пристрою впродовж усього його життєвого циклу.

За потреби розроблені в цій роботі програмні засоби можна динамічно розширювати, що дозволяє додавати та створювати новий функціонал як на клієнтській частині проекту, так і на серверній, який може бути пов’язаний зі вже існуючими модулями.

Також варто зазначити, що щоденне логування записів виявлення аномалій дозволить в майбутньому розробити нові засоби, які відображатимуть, для прикладу, порівняльну характеристику вже існуючих діагностичних даних, котрі в свою чергу є і будуть прив’язаними до пристроїв інтернету речей. Розробка подібних засобів зможе більш детально розкрити унікальні аспекти роботи того чи іншого пристрою.

ВИСНОВКИ

В результаті виконання роботи було приділено велику кількість уваги даним від пристроїв IoT. На основі цих даних було виконано їх нормалізацію, щоб підготувати дані до подальшої роботи. Проаналізовано раніше нормалізовані дані та обрано алгоритми DBSCAN та IForest, які відігравали ключову роль у виявленні аномалій. Розроблено механізм виявлення аномалій з використанням мови програмування Python, допоміжних бібліотек numpy, sklearn, adtk та pyodbc. Відтестовано розроблені алгоритми на реальних даних від пристроїв інтернету речей. В результаті тестувань алгоритмів виявлено, що обрані раніше алгоритми найефективніше виявили аномалії у масиві даних, який містив у собі дані про кількість відправлених та отриманих байтів від пристрою(-їв) IoT. Було досліджено, що обрані алгоритми виявлення аномалій не є ефективними у знаходженні рідкісних даних серед масивів, котрі містять інформацію про рівень води та рівень заряду батареї пристрою. Тобто, масиви, що містять в собі інформацію про рівень води та рівень заряду батареї пристрою можуть складатися з різного роду даних, які генеруються у багатьох випадках і залежать від великої кількості факторів, що в свою чергу ускладнює процес виявлення аномалій. Розроблену програму було пов’язано з базою даних проекту моніторингу IoT пристроїв, що дозволить маніпулювати алгоритмами довільним чином, зручно коригувати унікальні налаштування для кожного з пристроїв і виконати поглиблену інтеграцію алгоритмів виявлення аномалій до уже наявної системи моніторингу.

В подальшому проводитиметься робота над класифікацією аномальних даних, а також фільтруванням помилкових рідкісних екземплярів. Всі ці дії дозволять підвищити точність виявлення рідкісних подій (аномалій).

Передбачається, що функціонал розроблених алгоритмів буде розширюватися, що дозволить виконати автоматичне інформування користувачів проекту про ті чи інші негаразди з пристроєм. Як приклад – власники проекту зможуть відправляти лист на електронну адресу, або ж розробити бот для сповіщень в одну із соціальних мереж.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. sklearn.cluster.DBSCAN [Електронний ресурс] : [Веб-сайт] – Просторова кластеризація додатків із шумом на основі щільності – Електронні дані. Режим доступу до ресурсу: scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html (дата звернення 08.07.2023) – Назва з екрана.
2. Isolation Forest Algorithm [Електронний ресурс] : [Веб-сайт] – Isolation Forest is the best Anomaly Detection Algorithm for Big Data Right Now – Електронні дані. Режим доступу до ресурсу: towardsdatascience.com/isolation-forest-is-the-best-anomaly-detection-algorithm-for-big-data-right-now-e1a18ec0f94f (дата звернення 07.07.2023) – Назва з екрана.
3. ADTK Detectors [Електронний ресурс] : [Веб-сайт] – ARUNDO Detectors – Електронні дані. Режим доступу до ресурсу: arundo-adtk.readthedocs-hosted.com/en/stable/notebooks/demo.html#QuantileAD (дата звернення 09.07.2023) – Назва з екрана.
4. Журнали та книги (ScienceDirect) [Електронний ресурс] : [Веб-сайт] – Research on anomaly detection and real-time reliability evaluation with the log of cloud platform – Електронні дані. Режим доступу до ресурсу: www.sciencedirect.com/science/article/pii/S1110016821008711 (дата звернення 12.07.2023) – Назва з екрана.
5. Zhukovskyy Viktor, Printz Damon, Demchuk Yuriy, Holiuk Kostiantyn, Human-Computer Interaction in IoT System for Water Tank Monitoring and Controlling. // 2022 IEEE 16th International Scientific Conference on Informatics. – Poprad, Slovakia: IEEE, 23.11.2022 - 25.11.2022.
6. Цифровий репозиторій Національного університету водного господарства та природокористування [Електронний ресурс] : [Веб-сайт] – Демчук Юрій. Реалізація механізму реплікацій баз даних між віртуальним сервером та хмарним сервісом Microsoft Azure Sql Database. Науково-практична конференція навчально-наукового інституту автоматики, кібернетики та

обчислювальної техніки, 12–19 травня 2022 року, Рівне, с. 16 – Електронні дані. Режим доступу до ресурсу: ep3.nuwm.edu.ua/24903/ (дата звернення 15.07.2023) – Назва з екрана.

7. Цифровий репозиторій Національного університету водного господарства та природокористування [Електронний ресурс] : [Веб-сайт] – Губач Максим. Про реалізацію проекції запитів на прикладі платформи моніторингу Tank Toad. Науково-практична конференція навчально-наукового інституту автоматики, кібернетики та обчислювальної техніки, 12–19 травня 2022 року, Рівне, с. 18 – Електронні дані. Режим доступу до ресурсу: ep3.nuwm.edu.ua/24903/ (дата звернення 15.07.2023) – Назва з екрана.
8. Цифровий репозиторій Національного університету водного господарства та природокористування [Електронний ресурс] : [Веб-сайт] – Голюк Костянтин. Реалізація асинхронного обміну в клієнт серверній архітектурі за допомогою технології SignalR. Науково-практична конференція навчально-наукового інституту автоматики, кібернетики та обчислювальної техніки, 12–19 травня 2022 року, Рівне, с. 17 – Електронні дані. Режим доступу до ресурсу: ep3.nuwm.edu.ua/24903/ (дата звернення 15.07.2023) – Назва з екрана.
9. Zhukovskyy Viktor, Printz Damon, Zhukovska Nataliia, Hubach Maksym, Rajab Hesham, IoT based Intelligent Information-Analytical System Architecture for Water Tank Monitoring. // 2021 International Conference on Information Technology (ICIT). – Amman, Jordan: IEEE, 14.07.2021 - 15.07.2021. С. 924–928.
10. AVI Networks [Електронний ресурс] : [Веб-сайт] – What is Anomaly Detection? – Електронні дані. Режим доступу до ресурсу: avinetworks.com/glossary/anomaly-detection/ (дата звернення 13.07.2023) – Назва з екрана.
11. Wikipedia [Електронний ресурс] : [Веб-сайт] – Anomaly Detection – Електронні дані. Режим доступу до ресурсу: en.wikipedia.org/wiki/Anomaly_detection (дата звернення 16.07.2023) – Назва з екрана.

12. R. Abrantes, P. Mestre, and A. Cunha, “Exploring dataset manipulation via machine learning for botnet traffic,” Procedia Computer Science, vol. 196, pp. 133–141, 2022.
13. Wikipedia [Електронний ресурс] : [Веб-сайт] – Angular (фреймворк) – Електронні дані. Режим доступу до ресурсу: uk.wikipedia.org/wiki/Angular_(%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) (дата звернення 17.07.2023) – Назва з екрана.
14. Wikipedia [Електронний ресурс] : [Веб-сайт] – Transact-SQL (T-SQL) – Електронні дані. Режим доступу до ресурсу: uk.wikipedia.org/wiki/Transact-SQL (дата звернення 19.07.2023) – Назва з екрана.
15. Журнали та книги (ScienceDirect) [Електронний ресурс] : [Веб-сайт] – Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches – Електронні дані. Режим доступу до ресурсу: www.sciencedirect.com/science/article/pii/S2542660519300241 (дата звернення 22.07.2023) – Назва з екрана.
16. Aurora Gonzalez-Vidal, Jesus Cuenca-Jara and Antonio F. Skarmeta, IoT for Water Management: Towards Intelligent Anomaly Detection. // 2019 IEEE 5th World Forum on Internet of Things (WF-IoT).
17. A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, B. D. Payne, Evaluating computer intrusion detection systems: A survey of common practices, ACM Computing Surveys (CSUR) 48 (1) (2015) 1–41.
18. Inês Martins, João S. Resende, Patricia R. Sousa, Simão Silva, Luis Antunes, João Gama, Host-based IDS: a review and open issues of an anomaly detection system in IoT.

ДОДАТКИ

Додаток 1
Лістинг коду модуля виявлення аномалій

```
from sklearn.ensemble import IsolationForest
from adtk.detector import SeasonalAD
from adtk.data import validate_series
from sklearn.cluster import DBSCAN
from sklearn.datasets import make_blobs
from numpy import where
from adtk.detector import QuantileAD
from functools import reduce
import sys

alt.data_transformers.disable_max_rows()

#
# Get information from csv file & generate pd.Series
#
requestData = []



def read_csv_file(fileName, colnames, skiprowsCount, indexColumn):
    if (fileName == ""):
        return None

    fileReader = pd.read_csv(r'' + fileName, parse_dates=True,
                             dayfirst=True, names=colnames, skiprows=skiprowsCount)
…
#
# Get Other data & generate pd.Series (create input arrays)
#
def DB_Context_getData(DeviceId):
```

```
    if (DeviceId == "" or DeviceId == None):
        return None

    connection_string = f"DRIVER={{ODBC Driver 18 for SQL Server}};Server={server};Initial
Catalog={catalog};Persist Security
Info=False;UID={username};PWD={password};MultipleActiveResultSets=True;Encrypt=yes;Trus
tServerCertificate=yes;Connection Timeout=30;"
    cnxn = pyodbc.connect(connection_string)

    DeviceSettings = pd.read_sql(f"""
    SELECT *
    FROM {catalog}.dbo.DeviceSettings
    WHERE Id ="""" + str(DeviceId) + """";
    """, cnxn)

    if (DeviceSettings["Id"].values[0] == "" or DeviceSettings["Id"].values[0] == None):
        return None

    SerialNumber = DeviceSettings["SerialNumber"].values[0]
    IMEI = DeviceSettings["IMEI"].values[0]
    DataConversionConstantA = DeviceSettings["DataConversionConstantA"].values[0]
    DataConversionConstantB = DeviceSettings["DataConversionConstantB"].values[0]
    ConnectionType = DeviceSettings["ConnectionType"].values[0]
…
def Data_Normalization(Data):
    if (Data == None):
        return None

    DB_Data = Data[0]
    GCOM_Packages = Data[1]
    DeviceSettings = Data[2]

    DataConversionConstantA = DeviceSettings["DataConversionConstantA"]
```

```
    DataConversionConstantB = DeviceSettings["DataConversionConstantB"]

    BatteryVoltageConstant = 0.0041503906
…
    #
    # Getting values from a series
    #
    WaterLevel = DB_Data["WaterLevel"].values
    BatteryVoltage = DB_Data["BatteryVoltage"].values

    #
    # Round number to format x.xx
    #
    def r(value):
        return round(value, 2)

    #
    # Converting values
    #
    WaterLevel = [r(DataConversionConstantA * float(iterable -
                DataConversionConstantB)) for iterable in WaterLevel]
    BatteryVoltage = [r(iterable * BatteryVoltageConstant)
                  for iterable in BatteryVoltage]
…
def IForest_Detector(values):
    try:
        Iforest = IsolationForest(max_samples=200000,
                                  random_state=0,
                                  contamination=0.035,
                                  max_features=1,
                                  n_estimators=2000,
```

```
                    bootstrap=False,
                    verbose=1,
                    n_jobs=-1)

        Iforest.fit(values)
        y_pred = Iforest.predict(values)
        y_pred_adjusted = ['yes' if x == -1 else 'no' for x in y_pred]
        return y_pred_adjusted
    except:
        return []
…
def anomalyDetection(values, colname, Data, SerialNumber):
    # Fill NaN values in pd.Series
    values = values.fillna(dict.fromkeys(
        values.columns.tolist(), values[colname].ffill()))

    # Initial graph & added draw all selection values
    X = values[colname].index
    Y = values[colname].values

    val = pd.DataFrame({
        'x': X,
        'y': Y
    })

    _chart = alt.Chart(val).mark_circle(interpolate="basis", color="#0160ff").encode(
        x=alt.X('x', title='Date'),
        y=alt.Y('y', title="Values"),
    ).interactive(bind_y=False)
…
def _anomalyDetectionHandler(Data, fieldsAnomalies, SerialNumber):
```

```
    if (len(Data) == 0 | len(fieldsAnomalies) == 0):
        return None

    defaultFileData = Data[0]
    fieldsDetection = fieldsAnomalies

    # Detect anomalies in WaterLevel & BatteryVoltage
    for i in fieldsDetection:
        data = defaultFileData[[i]]
        anomalyDetection(values=data, colname=i,
                         Data=Data, SerialNumber=SerialNumber)

    # Detect anomalies in TotalBytes
    data = Data[1]
    anomalyDetection(values=data, colname="TotalBytes",
                     Data=Data, SerialNumber=SerialNumber)
def ___main___(DeviceId):
    DB_Data = DB_Context_getData(DeviceId)
    DB_Data = Data_Normalization(DB_Data)

    if (DB_Data == None or len(DB_Data) == 0 or DB_Data == ""):
        return None
    # Specify the fields for which you want to detect anomalies
    fieldsAnomalies = ["WaterLevel", "BatteryVoltage"]
    DeviceData = DB_Data[0]
    GCOM_Packages = DB_Data[1]
    DeviceSettings = DB_Data[2]
    SerialNumber = DeviceSettings["SerialNumber"]
    _anomalyDetectionHandler(
        Data=[DeviceData, GCOM_Packages], fieldsAnomalies=fieldsAnomalies,
SerialNumber=SerialNumber)
…
```