КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра комп'ютерної інженерії

До захисту допущено: «На правах рукопису»
Завідувач кафедри _____ Юрій Бойко
« _ » _____ 2023 р.

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«ЗАСТОСУВАННЯ АЛГОРИТМІВ ВИЗНАЧЕННЯ АНОМАЛІЙ ДЛЯ ОБРОБКИ ДАНИХ СЕНСОРІВ ІНТЕРНЕТУ РЕЧЕЙ»

Виконав:
студент 2-го курсу магістратури
денної форми навчання
спеціальності 123 Комп'ютерна інженерія
ОНП «_____»
Олександр Марченко _____

Науковий керівник:
асистент
Юрчик Юрій Костянтинович _____

Рецензент:
кандидат технічних наук, доцент
Лебедев Денис Юрійович _____

Засвідчую, що у цій магістерській роботі
немає запозичень з праць інших авторів без
відповідних посилань
Студент _____

Робота допущена до захисту в ЕК рішенням кафедри _____
від «__»_____ 2023 р., протокол № __.

Завідувач кафедри _____,
кандидат фізико-математичних наук, доцент
Бойко Юрій Володимирович

(підпис)

Київ – 2023

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра містить 59 сторінок, 23 рисунки, 2 формули, 4 таблиці, 5 додатків, використано 8 інформаційних джерел.

Ключові слова: алгоритм, аномалія, дані, Інтернет речей, метод, сенсор руху.

Актуальність роботи полягає у постійному ускладненні систем Інтернету Речей та загальної кількості пристроїв, що є частиною цих систем.

Об’єктом роботи є система Інтернету Рчей, що виконує моніторинг переміщення людей в приміщенні офісного типу.

Предметом роботи є аномальні результати вимірювання параметрів навколишнього середовища (офісного приміщення) та способи їх виявлення

Метою роботи є дослідження алгоритмів виявлення аномалій та придатність їх використання у мережах Інтернету Речей.

За результатами роботи було визначено ефективність виявлення аномалій в даних сенсору системи Інтернету речей для алгоритмів: К – середніх, Ковзного середнього, Авторегресійного ковзного середнього.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

IoT - Інтернет речей, Internet of Things

Алгоритм ARIMA – алгоритм авторегресивного ковзного середнього, autoregressive integrated moving average algorithm

Алгоритм MA – алгоритм середнього ковзного, moving average algorithm

ВСТУП

Стрімкий розвиток IoT систем призводить до збільшення загального обсягу IoT пристроїв, їх різноманіття і географічної розподіленості. Це призводить до збільшення кількості та різноманітності позаштатних ситуацій в IoT системі.

У цьому випадку практично неможливо передбачити усі можливі позаштатні ситуації при проєктуванню системи. Проте ці ситуації можна визначити як аномальні відносно штатного режиму роботи системи. При цьому визначення аномальності даних доцільно робити щонайближче до місця їх генерації, таким чином збільшуючи швидкість реакції системи на аномальні події.

Для реалізації вищезгаданого підходу, необхідно обрати алгоритм визначення аномалій, що має достатню точність і мінімальні вимоги стосовно обчислювальних ресурсів.

У якості джерела даних для подальшої обробки було створено дослідницьку модель з використанням датчику руху, а саме HC-SR501. Дані отримані з цього датчика мають певні закономірності, а саме, щотижневу періодичність, що варто враховувати у роботі.

1 ЗАГАЛЬНА СТРУКТУРА IoT СИСТЕМИ

1.1 Загальна структура IoT

IoT є мережею серверів, цифрових пристроїв, комп'ютерів та датчиків, що можуть обмінюватися даними через мережу Інтернет. Це дає змогу керувати процесами у різних сферах життя та автоматизовувати їх, забезпечуючи надійність, ефективність, і безпеку [1].

Це мережа різноманітних пристроїв, підключених до мережі Інтернет що реалізують різні моделі взаємодії: «Річ - Річ», «Річ - Користувач», «Річ - Пристрій» і «Річ – веб-сервер».

В даному випадку річчю можна вважати будь який електронний пристрій, що може передавати дані та є частиною загального інтернету речей.

В загальному випадку структура найпростішої IoT системи виглядає наступним чином (рис. 1): річ (датчик) збирає певну інформацію, залежно від свого призначення та передає її на пристрій (контролер), який керує кількома датчиками, та акумулює їх дані, зазвичай є комп’ютером із обмеженими характеристиками; після чого ця річ передає наявну в неї інформацію, через мережу Інтернет до віддаленого, або стаціонарного сервера - де відбувається обчислення.

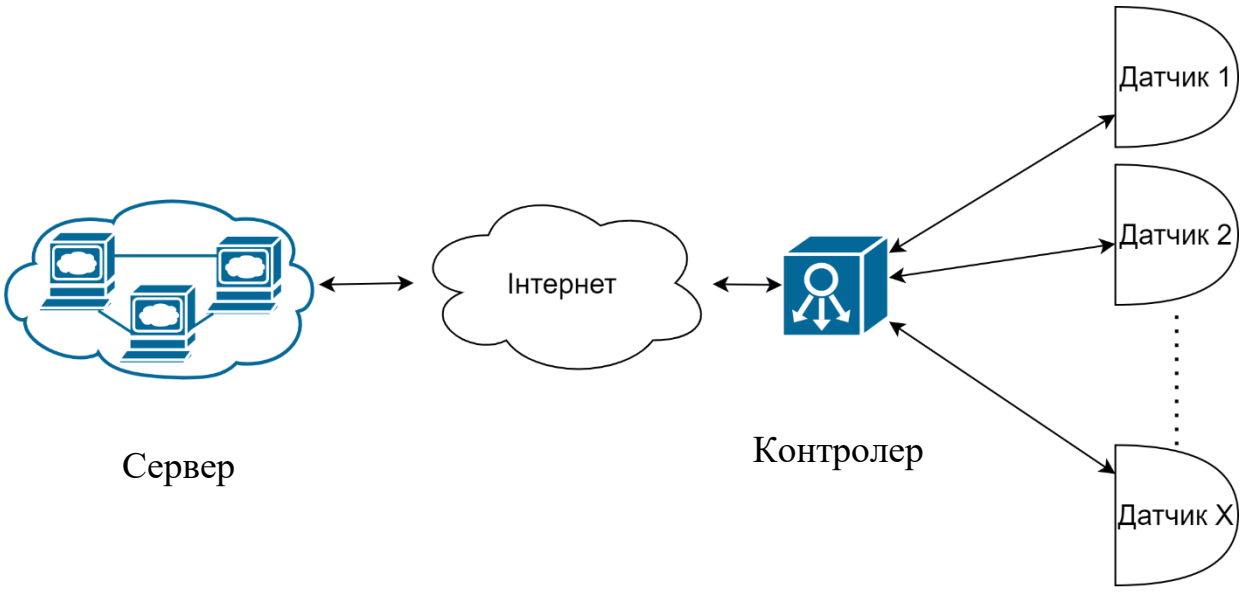


Рис. 1 – Архітектура системи IoT.

Зі збільшенням кількості речей в мережі IoT з’явилася проблема передачі великих масивів даних, та збільшення часу реакції на інциденти. Внаслідок чого об’єднанням компаній ARM, Cisco, Dell, Intel, Microsoft і Прінстонського університету у 2015 році було введено концепцію туманних обчислень, що передбачала перенесення частини обчислень на рівень речей [5].

Як канал зв'язку використовується мережа на базі протоколу IP, а місця, де встановлюються датчики настільки різноманітні, що не завжди можна використовувати дротову інфраструктуру.

Фундаментальними характеристиками IoT є:

1. Взаємозв’язаність: всі пристрої взаємодіють через глобальні або локальні канали інформаційного обміну.

2. Сервіси, що орієнтовані на пристрої: IoT здатні забезпечити узгодженість та об’єднати датчики чи навіть цілі пристрої для взаємодії.

3. Гетерогенність: пристрої в IoT надзвичайно різноманітними і можуть мати різну обчислювальну потужність, належати різним мережам і апаратним платформам, мати різний тип та призначення, це не має бути перешкодою для взаємодії.

4. Динамічність: постійно може змінюватися місцезнаходження пристроїв, їх стан, кількість та тип підключених пристроїв також може динамічно змінюватися.

5. Масштабність: кількість з'єднань типу «Річ – Річ» та «Річ – Пристрій» в сотні разів перевищує кількість з’єднань типу «Річ – Користувач». Тому на перший план виходять питання інтерпретації даних, з метою їх подальшого застосування.

1.2 Типи сенсорів руху

Однією із основних і практично невід’ємною частиною IoT системи є різноманітні сенсори. Одним із найпростіших типів сенсорів, що використовуються в мережі IoT є датчики руху.

В сучасному світі датчики руху є невід'ємною частиною багатьох пристроїв. Вони використовуються для розпізнавання руху людини, транспорту, тварин, тощо.

Одним з найпоширеніших типів датчиків руху є інфрачервоні датчики руху. Вони використовують інфрачервоне випромінювання для виявлення руху об'єкта. Ці датчики часто використовуються в системах безпеки, щоб автоматично вмикати світло або сповіщати про можливу небезпеку. Вони також

використовуються у вимикачах світла і системах автоматичного контролю клімату [3].

Інший тип датчиків руху - ультразвукові датчики. Вони використовують ультразвукові хвилі для виявлення руху. Ці датчики використовуються у робототехніці, системах контролю трафіку і відеоспостереження.

Також існують магнітні датчики руху. Вони використовуються в системах безпеки для виявлення відкриття дверей або вікон. Вони використовуються в робототехніці та інших системах, де необхідно вимірювати рух об'єкта.

Датчики руху на базі камер з обробкою зображень, використовують алгоритми машинного навчання для виявлення руху. Ці датчики використовуються в системах безпеки, системах контролю доступу і відеоспостереження.

Для збирання статистичної інформації з датчиків руху і подальшої її обробки, використовується формат даних, що має прив’язку до часу.

1.3 Формат даних Time Series

Time Series (дані часового ряду) - це формат даних, які вимірюються та записуються послідовно впродовж певного проміжку часу. Кожне числове значення вимірюваної величини - пов'язане з певним моментом часу і може бути використане для аналізу попередніх трендів чи прогнозування майбутніх значень.

Для аналізу часових рядів використовуються різні методи, включаючи статистичний аналіз, машинне навчання та прогнозування [2].

При роботі з часовими рядами необхідно враховувати сезонність, тренди та інші фактори, які можуть впливати на дані. Для більш точного прогнозування та аналізу часових рядів може бути необхідно використовувати спеціальні методи та алгоритми.

Одним з найбільш поширених методів аналізу часових рядів є алгоритм ARIMA. Цей алгоритм використовується для моделювання та прогнозування часових рядів, ґрунтуючись на їх минулих значеннях.

Іншим методом аналізу часових рядів є метод MA, який використовується для виявлення трендів та згладжування шуму даних.

Крім того, в деяких випадках можна використовувати нейронні мережі для прогнозування часових рядів, особливо якщо дані мають складні патерни або нелінійні залежності.

Важливим аспектом аналізу часових рядів є попередня обробка даних. Це може включати видалення викидів, заповнення пропущених значень, перетворення даних чи будь які інші модифікації початкових даних.

У зв’язку із великою гетерогенністю і географічним розподілом елементів IoT системи практично неможливо наперед передбачити усі позаштатні ситуації, що можуть виникнути під час роботи системи. Можливим вирішенням цієї проблеми є пошук аномальних подій (значень вимірюваних величин) у IoT системі.

Зважаючи на великі розміри IoT систем, може бути доцільним виконання пошуку аномальних значень параметрів навколишнього середовища якнайближче до джерел генерації цих значень. Тобто обробляти значення отримані безпосередньо від сенсора. Це дозволяє заощадити пропускну здатність каналів зв’язку та потенційно зменшити час реакції системи на подію [8].

Для визначення аномальності конкретного результату виміру параметрів навколишнього середовища існують різні математичні методи. Обираючи той чи інший метод у якості інструменту необхідно зважати на особливості IoT систем. Особливо на обмеженість обчислювального ресурсу контролерів.

2 АНОМАЛІЇ ТА МЕТОДИ ТА ЇХ ПОШУКУ

2.1 Загальна інформація про аномалії

2.1.1 Види аномалій у даних

Аномалії — це закономірності в даних, які не відповідають чітко визначеному поняттю нормальної поведінки [4].

Рисунок 2 ілюструє аномалії в простому двовимірному наборі даних. Дані мають дві нормальні області, N1 і N2, оскільки більшість спостережень знаходяться в цих двох областях. Точки, які досить віддалені від цих областей, наприклад, точки O1 і O2, а також точки в регіоні O3, є аномаліями.

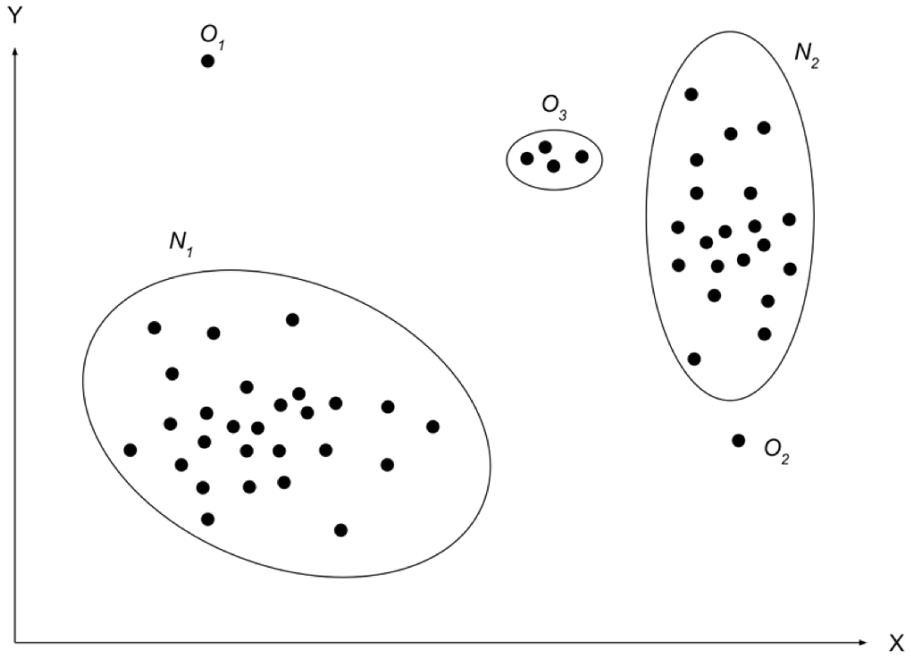


Рис. 2 - Приклад аномалій у двовимірному наборі даних.

Аномалії можуть бути спричинені різними причинами, такими як зловмисна діяльність або збій системи. У будь якому випадку виявлення аномалій

є важливим процесом і розглядається як перевага в різних системах прийняття рішень.

Аномалії можна класифікувати за такими трьома категоріями [4]:

- Точкові аномалії.
- Колективні аномалії.
- Контекстуальні аномалії.

Якщо один об’єкт можна спостерігати на тлі інших об’єктів як аномалію, то це точкова аномалія. Це найпростіша категорія аномалій, велика кількість досліджень концентруються саме довкола їх пошуку. Беручи до уваги приклад, представлений на рисунку (рис. 2), точки O_1 та O_2 є точковими аномаліями.

Якщо деякі пов’язані об’єкти можна об’єднати з іншими об’єктами як аномалію. Аномальним у цьому випадку не може бути окремий об’єкт, а лише сукупність об’єктів. Прикладом є регіон O3 (рис. 2).

Якщо об’єкт є аномальним у певному контексті - це контекстуальна аномалія. На рисунку 3 можна побачити періодичний контекст. У цьому випадку точка O1 є аномалією, оскільки вона відрізняється від періодичного контексту.

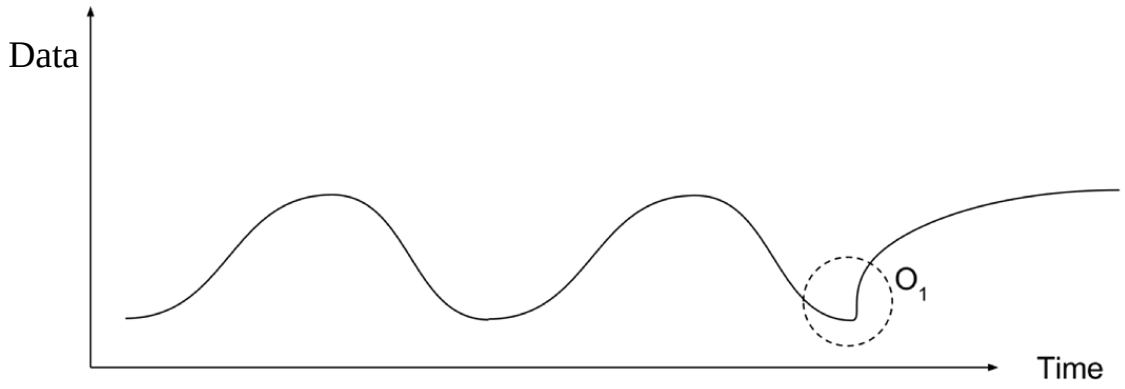


Рис. 3 – Приклад контекстуальної аномалії

2.1.2 Види алгоритмів виявлення аномалій

2.1.2.1. Метод класифікація

Реалізація цього методу полягає в припущенні, що нормальну поведінку системи можна описати одним чи кількома класами. Таким чином, точка, що не належить до жодного з класів є відхиленням від норми, тобто аномалією. Пошук аномалій відбувається у два етапи: навчання на певній вибірці та розпізнавання даних.

Класифікатор навчається на масиві перевірених даних, далі визначається приналежність до одного з відомих класів. В іншому випадку точка позначається, як аномалія. Механізмами, що використовуються найбільш широко для розпізнавання аномалій з допомогою класифікації є: нейронні мережі, Байєсові мережі, метод опорних векторів та метод на основі визначених правил.

Метод пошуку аномалій на основі нейронних мереж проходить у два етапи.

Перший етап: нейронна мережа навчається розпізнавати класи нормальної поведінки на тренувальній вибірці.

Другий етап: кожна точка потрапляє в систему як вхідний сигнал нейронної мережі. Система, на основі виявлених раніше закономірностей відносить точку до певного класу.

Для знаходження аномалій за допомогою розпізнавання тільки одного класу використовуються реплікативні нейронні мережі, або мережі глибинного навчання.

Байєсовою мережею є графічна модель, що відображає ймовірнісні залежності множини змінних і дозволяє робити висновок за допомогою цих змінних. Вона складається з двох частин: графічної структури, яка визначає набір залежностей, та набір імовірнісних розподілів, що визначають силу відносин. Залежність, закодована в графічній структурі. Таким чином, застосування Байєсової мережі при ідентифікації аномалій полягає в оцінці ймовірності віднесення точки до нормального або аномального класу.

Метод опорних векторів використовується для знаходження аномалій там, де нормальна поведінка задається тільки одним класом. Цей спосіб визначає межу регіону, в якому знаходяться дані, що вважаються нормальними. Для кожного об’єкту, що досліджується слід визначити його належність до певного регіону. Якщо точка виявляється за межами цієї області, вона вважається аномальною.

Останній метод заснований на генерації правил, які відповідають нормальній поведінці системи. Об’єкт, який не відповідає цим правилам, рахується аномальним. Алгоритм складається з двох кроків. Спершу система

навчається по згенерованим правиламза допомогою обраного алгоритму. Другим кроком йде пошук правила для кожної точки, яке найкраще підходить цій точці.

2.1.2.2. Метод кластеризації

Цей метод передбачає групування схожих точок у кластери і не вимагає знань про властивості можливих відхилень. Виявлення аномалій засновується на припущенні, що відстань між нормальними точками є меншою ніж відстань від нормальної точки до аномальної.

Однак цей метод має проблему визначення точних меж кластерів. Звідси слідує інше припущення: нормальні точки знаходяться ближче до центру кластера, а аномальні – значно далі. Якщо аномальні точки не є одиничними, вони можуть утворювати кластери, в яких щільність точок на квадратну одиницю площі буде відчутно меншою ніж у нормальному кластері. Одним із найпростіших реалізацій підходу на основі кластеризації є алгоритм К-середніх, що буде розглянутий в розділі 2.2.2.

2.1.2.3. Метод статистичного аналізу

Під час використання цього підходу досліджується сам процес, створюється його модель, яка потім порівнюється з реальною поведінкою. Якщо різниця між реальною і прогнозованою поведінкою системи вище встановленого порогу, можна зробити висновок про наявність відхилень.

В основі підходу йде припущення про те, що нормальна поведінка системи перебуватиме в зоні високої ймовірності, у той час як її аномалії – у зоні низької.

Методи статистичного аналізу поділяються на дві основні групи: параметричні та непараметричні.

В першій групі нормальні дані генеруються параметричним розподілом з параметрами та функцією щільності ймовірності. Аномалія є оберненою функцією розподілу. Ці методи часто ґрунтуються на Гаусовій або регресійній моделі, і навіть їх комбінації.

В другій групі передбачається, що структура моделі не визначена заздалегідь, але в той же час вона створюється з наданих даних і включає методи з урахуванням гістограм чи функцій ядра.

2.1.2.4. Метод найближчого сусіда

Для використання даної методики необхідно визначити міру схожості між точками, тобто відстань на якій об’єкти належатимуть одній групі. Два основних підходи ґрунтуються на наступних припущеннях:

- Відстань до найближчого сусіда.
- Використання відносної щільності

Для реалізації першого підходу для кожної точки визначається відстань до її найближчого сусіда. Найбільш віддалена точка вважається аномальною.

Другий підхід ґрунтується на оцінці щільності об’єктів, довкола кожної точки. Точки, що розташовані з низькою щільністю, оцінюються як аномальні, в той час як точка із вибірки з високою щільністю вважається нормальною.

2.1.2.5. Методи пошуку контекстуальних аномалій

Усі описані вище методи застосовуються для пошуку точкових аномалій, проте їх також можна застосувати для розпізнавання колективних та контекстуальних аномалій, при зведенні їх до точкових.

Під час пошуку контекстуальних аномалій, цей підхід реалізується за допомогою визначення контекстуальних атрибутів та перетворення даних на їх основі.

Після цього до перетворених даних можна застосувати один із методів ідентифікації точкових аномалій. Альтернативою даному методу є моделювання часових ряді, наприклад, побудова моделі ARIMA, або перетворення їх до символьних послідовностей.

При пошуку колективних аномалій можливо визначення підпослідовностей фіксованої довжини, як одиничних точок, проте при цьому робиться припущення, що всі ділянки, що є колективними аномаліями, мають однакову довжину.

2.2. Алгоритми для виявлення аномалій, що були реалізовані під час написання роботи

2.2.2. Алгоритм К-середніх

Алгоритм К-середніх — це алгоритм кластеризації, який намагається розділити набір даних на К попередньо визначених окремих підгруп, де кожна точка даних належить лише одній групі. Він визначає точки, котрі належать кластеру таким чином, щоб сума квадратів відстаней між точками і центроїдом кластера (середнє арифметичне всіх точок даних, які належать цьому кластеру) була мінімальною. Чим менша відмінність між точками, тим більш однорідними є кластери [6].

Алгоритм К-середніх працює наступним чином (рис. 4):

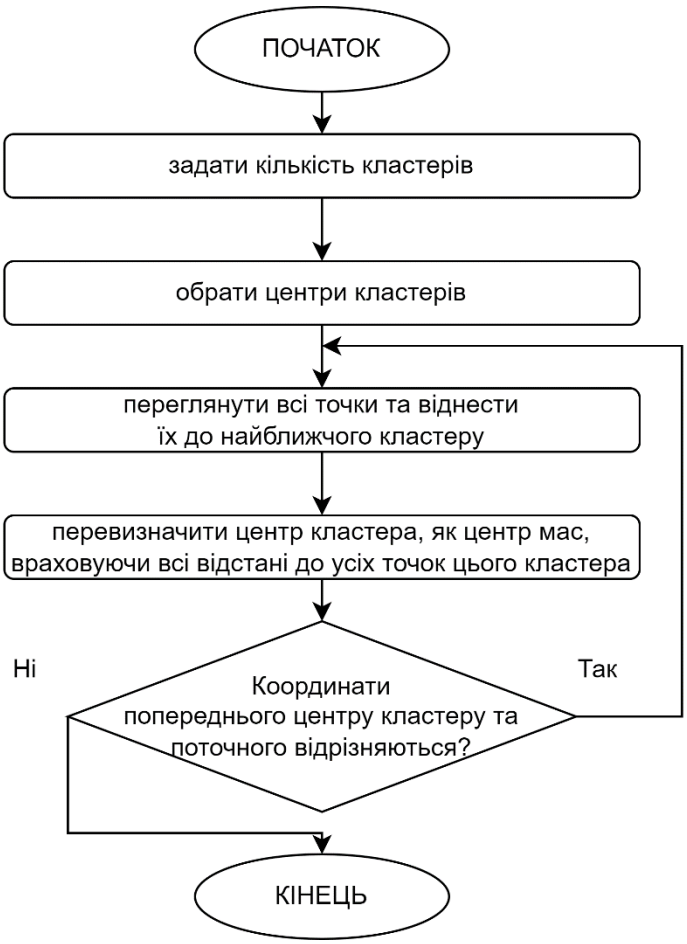


Рисунок 4 - Блок схема роботи алгоритму К-середніх

1. Спершу необхідно задати кількість кластерів
2. Після чого слід обрати точки, які будуть центрами кластерів, на цьому етапі центрами обираються випадкові точки з набору.
3. На наступному етапі алгоритм переглядає кожну точку та відносить її до кластеру, центроїд якого є найближчим.
4. Далі алгоритм перевизначає центроїд, з урахуванням відстаней до кожної точки, що належить цьому кластеру.
5. Після чого перевіряється зміна цендроїду кластера. Якщо змінився хоч один центроїд, хоч одного кластеру, необхідно повторити пункти 3 та 4.

2.2.3. Алгоритм MA

Алгоритм MA - це метод аналізу даних, який використовується для згладжування шумів та випадкових змін в часових рядах [7].

Принцип роботи алгоритму полягає в тому, що для кожної точки часового ряду задається, або обраховується період згладжування (середнє значення деякої попередньої кількості точок) (рис. 5). Цей період вибирається залежно від досліджуваних даних.

Рис. 5 – Блок схема роботи алгоритму MA

Для обчислення ковзного середнього для кожної точки, алгоритм зберігає певну кількість останніх значень часового ряду, рівну довжині періоду

згладжування N. Потім обраховується середнє значення цих N точок і використовується як нове значення для досліджуваної точки. Алгоритм слід повторити для кожної наступної точки.

2.2.4. Алгоритм ARIMA

Алгоритм ARIMA - це статистичний алгоритм прогнозування, який використовується для аналізу часових рядів. Він складається з трьох компонентів: авторегресії (AR), інтеграції (I) та ковзного середнього (MA).

Формула даної моделі:

$$(\Delta^d X_t) = c + \sum_{i=1}^{p} a_i \, \Delta^d X_{t-i} + \sum_{j=1}^{q} b_i \, \varepsilon_{t-j} + \varepsilon_t, \qquad (1)$$

де Δ^d означає диференціювання порядку d;

X_t є значенням часового ряду в момент часу t;

ε_t є білим шумом в момент часу t;

c, a_i, b_j - це коефіцієнти моделі ARIMA.

Компонента авторегресії (AR) використовується для опису залежності між попередніми значеннями часового ряду та поточним значенням. Це означає, що значення часового ряду в певний момент часу залежить від попередніх значень.

Компонента інтеграції (I) використовується для опису зміни рівня часового ряду з часом. Це означає, що видаляється тренд з даних, щоб зробити їх стаціонарними.

Компонента ковзного середнього (MA) використовується для опису залежності між поточним значенням часового ряду та попередніми значеннями шуму. Це означає, що значення часового ряду в певний момент часу залежить від шуму в попередні моменти часу.

Згідно з формулою (1) ARIMA складається з трьох параметрів: ARIMA (p, d,q), де:

p - це порядок компоненти авторегресії, який показує, скільки попередніх значень необхідно враховувати для прогнозу поточного значення.

d - це порядок компоненти інтеграції, який показує, скільки разів потрібно провести диференціювання для перетворення часового ряду стаціонарним.

q - це порядок компоненти ковзного середнього, який показує, скільки попередніх значень шуму необхідно враховувати для прогнозу поточного значення.

Загальні кроки роботи алгоритму можна продемонструвати у вигляді блок-схеми (рис. 6).

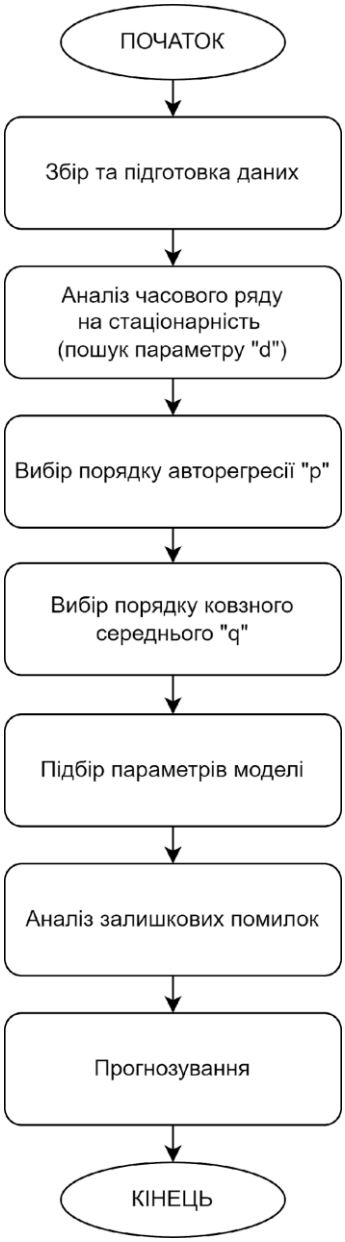


Рис. 6 - Блок схема роботи алгоритму ARIMA

Збір та підготовка даних: цей крок передбачає зібрання та підготовку часового ряду для подальшого аналізу та прогнозування. Це може включати в себе обробку відсутніх даних, приведення даних до стандартного формату, зменшення шуму та видалення аномальних даних.

Аналіз часового ряду: після підготовки даних проводиться аналіз стаціонарності ряду. Стаціонарний ряд - це ряд, у якого статистичні

характеристики, такі як середнє значення та дисперсія, залишаються стабільними в часі. Якщо ряд не є стаціонарним, то необхідно виконати інтегрування ряду деяку кількість разів до того, як ряд стане стаціонарним. Оптимальне значення порядку інтегрування "d" можна знайти з аналізу графіків ряду та його першої різниці.

Побудова моделі AR: вибір порядку авторегресії "p" для моделі AR відбувається шляхом аналізу автокореляції ряду. Автокореляція - це кореляція між значеннями в одному та різних часових моментах. Автокореляційна функція показує залежність між значеннями ряду від їх попередніх значень. Чим більше значення "p", тим більша залежність від попередніх значень.

Побудова моделі MA: вибір порядку ковзного середнього "q" для моделі MA відбувається шляхом аналізу часткової автокореляції ряду. Часткова автокореляція - це кореляція між значеннями в одному та середньому значенні сукупності часових моментів. Чим більше значення "q", тим більше залежність від попередніх значень проміжних складових.

Підбір параметрів моделі: після вибору порядків "p" та "q" для моделі ARMA, необхідно підібрати коефіцієнти моделі, що найкраще відображають часовий ряд. Для цього використовують метод максимальної правдоподібності, який знаходить набір параметрів, що максимізують ймовірність того, що модель з вибраними параметрами є правильною.

Аналіз залишкових помилок: залишкові помилки є різницею між прогнозованими значеннями та реальними значеннями часового ряду. Їх аналіз допомагає перевірити, наскільки добре побудована модель описує даний часовий ряд. Якщо залишкові помилки мають високу дисперсію, то можливо потрібно

побудувати більш складну модель з вищим порядком авторегресії або ковзного середнього.

Прогнозування: після побудови моделі та аналізу залишкових помилок можна приступити до прогнозування майбутніх значень часового ряду. Це робиться шляхом побудови прогнозів на певний часовий період з використанням побудованої моделі та вибраних параметрів.

Правильний вибір параметрів p, d та q залежить від досліджуваного часового ряду. Для визначення цих параметрів можна використовувати різні методи, такі як автоматичне визначення параметрів, експертну оцінку або емпіричну валідацію.

Після налаштування параметрів моделі ARIMA можна провести прогноз поточного значення часового ряду на майбутній час. Прогнозування зазвичай здійснюється шляхом запуску ARIMA на даних з попереднього періоду, які збираються до моменту прогнозування. Якщо прогноз поточного значення знайдений, то його можна використовувати для прийняття рішень або планування в майбутньому.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ ПОШУКУ АНОМАЛІЙ

3.1 Підготовка даних

Для генерації даних була зібрана дослідницька модель (рис. 7), що складалася із датчика руху HC-SR501та одноплатного комп’ютера Raspberry Pi.

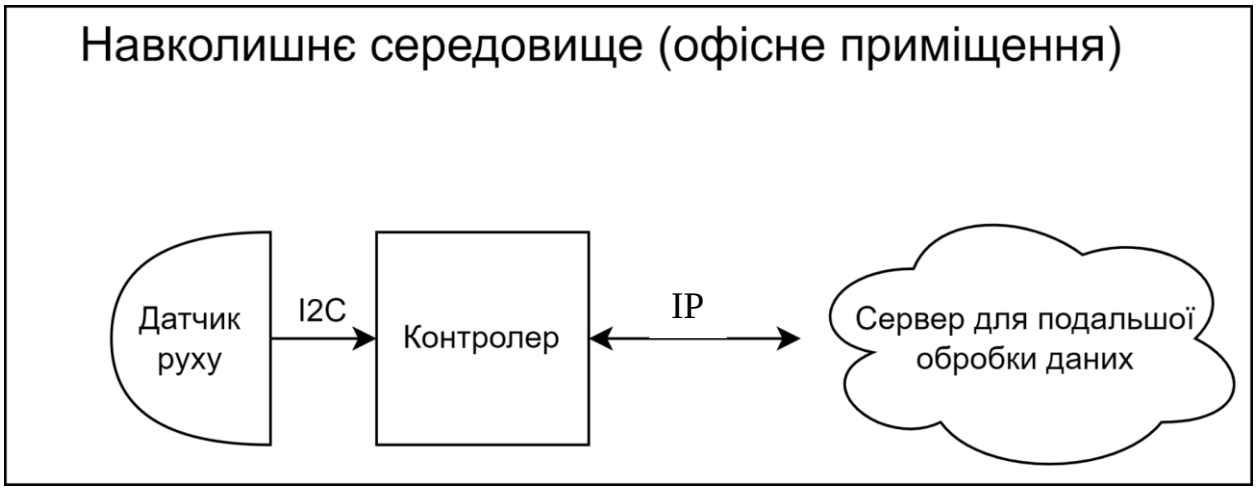


Рисунок 7 - Схема дослідницької моделі

Raspberry Pi є комп’ютером, всі необхідні елементи якого знаходяться в межах однієї плати. До неї було під’єднано інфрачервоний датчик руху, та інтернет кабель, через який дані передавались у сховище.

Дані збиралися щогодини, впродовж вісімдесяти чотирьох днів. Дані були у форматі Time Series (розділ 1.3). Дані було поділено на дві частини.

Було вирішено з частини даних зробити тренувальну вибірку, шляхом усунення всіх аномальних значень. Це дозволило проводити еталонні тести, які б

могли показати роботу алгоритмів в ідеальних умовах. Таким чином було отримано вибірки, що дозволяли провести тестування алгоримів із різними початковими умовами. Приклад нормальних та аномальних даних (Таблиця 1):

Таблиця 1 – Приклад даних з аномалією та без впродовж доби.

Доба без аномалій			Доба з присутньою аномалією	
Дата та час	Значення		Дата та час	Значення
3/29/2023 0:00:00	0		04/12/2023 0:00:00	0
3/29/2023 1:00:00	0		04/12/2023 1:00:00	0
3/29/2023 2:00:00	0		04/12/2023 2:00:00	0
3/29/2023 3:00:00	0		04/12/2023 3:00:00	0
3/29/2023 4:00:00	0		04/12/2023 4:00:00	0
3/29/2023 5:00:00	0		04/12/2023 5:00:00	0
3/29/2023 6:00:00	0		04/12/2023 6:00:00	0
3/29/2023 7:00:00	0		04/12/2023 7:00:00	0
3/29/2023 8:00:00	0		04/12/2023 8:00:00	0
3/29/2023 9:00:00	0		04/12/2023 9:00:00	0
3/29/2023 10:00:00	1		04/12/2023 10:00:00	1
3/29/2023 11:00:00	1		04/12/2023 11:00:00	1
3/29/2023 12:00:00	1		04/12/2023 12:00:00	1
3/29/2023 13:00:00	1		04/12/2023 13:00:00	1
3/29/2023 14:00:00	1		04/12/2023 14:00:00	1

3/29/2023 15:00:00	1		04/12/2023 15:00:00	1
3/29/2023 16:00:00	1		04/12/2023 16:00:00	1
3/29/2023 17:00:00	1		04/12/2023 17:00:00	1
3/29/2023 18:00:00	1		04/12/2023 18:00:00	1
3/29/2023 19:00:00	0		04/12/2023 19:00:00	0
3/29/2023 20:00:00	0		04/12/2023 20:00:00	1
3/29/2023 21:00:00	0		04/12/2023 21:00:00	1
3/29/2023 22:00:00	0		04/12/2023 22:00:00	0
3/29/2023 23:00:00	0		04/12/2023 23:00:00	0

Як можна помітити з таблиці щогодинні значення впродовж доби з різницею в 14 днів є практично ідентичними, за винятком аномалії о 20 та 21 годинах.

3.2 Програмна реалізація алгоритму К-середніх

Для реалізації алгоритму К-середніх було обрано бібліотеку «sklearn.cluster.kmeans» від ресурсу scikit-learn. Алгоритм через свою специфіку застосовується одразу на тестовій вибірці. А саме: алгоритм К-середніх є алгоритмом без учителя і розподіляє точки між кластерами у процесі своєї роботи. Кількість кластерів визначалось з допомогою двох методів: з допомогою методу силуету та методу ліктя.

Суть методу ліктя у прямому переборі кількості кластерів, та обрахунку суми квадратів відстані точок кластера до його центроїду. По мірі збільшення кількості кластерів це значення експоненційно знижується, проте деякі точки випадають із експоненційної кривої. Саме вони і є значеннями ймовірної кількості кластерів.

У коді, що демонструється на рисунку 8 проводиться зчитування даних з файлу за допомогою функції «pandas.read_csv» з бібліотеки «pandas. Далі, з допомогою функції «pandas.to_datetime» стовпець «Date_Time» конвертується до числового формату. Після чого значення стовпців «Data» та «Date_Time» зберігаються в змінних x та y відповідно.

У наступному рядку створюється список «data» з кортежів, де кожний кортеж містить значення зі стовпців «Data» та «Date_Time» відповідного рядка. Цей список потім можна використовувати для кластеризації.

```
D=pandas.read_csv('TestData with mistakes.csv')
D['Date_Time'] = pandas.to_numeric(pandas.to_datetime(D['Date_Time']))
x = D['Data']
y = D['Date_Time']
data = list(zip(x, y))
```

Рисунок 8 - Зчитування даних.

На рисунку 9 демонструється програмна реалізація методу ліктя. У коді створюється порожній список «inertias». Потім циклом відбувається перебір моделелей з різною кількістю кластерів «i», після чого кожен об'єкт тренується та додається до списку «inertias».

```
inertias = []
for i in range(1,45):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(data)
    inertias.append(kmeans.inertia_)
```

Рисунок 9 - Програмна реалізація методу ліктя

На рисунку 10 демонструється програмна реалізація методу силуета. Створюється порожній список «silhouette_scores», який буде містити значення коефіцієнта силуету для кожної кількості кластерів. Далі, проходить цикл для кожної кількості кластерів. Ініціалізується об'єкт «KMeans» з відповідною кількістю кластерів, за допомогою функції «fit_predict» прогнозуються кластери для вхідних даних «data», знаходяться центри кластерів, і обчислюється коефіцієнт силуету за допомогою функції «silhouette_score)». Значення коефіцієнта силуету для кожної кількості кластерів додається до списку «silhouette_scores».

```
silhouette_scores = []
num_clusters = range(2, 100)
for n_clusters in num_clusters:
    clusterer = KMeans(n_clusters=n_clusters)
    preds = clusterer.fit_predict(data)
    centers = clusterer.cluster_centers_
    score = silhouette_score(data, preds)
    silhouette_scores.append(score)
```

Рисунок 10 - Програмна реалізація методу силуета

У ході застосування цих двох методів було встановлено, що метод силуета не може встановити необхідну кількість кластерів (рис. 11). Проте метод ліктя дає цілком правдиву оцінку в 42 кластери. Переглянувши дані було встановлено, що оптимальною кількістю кластерів є 41 кластер.

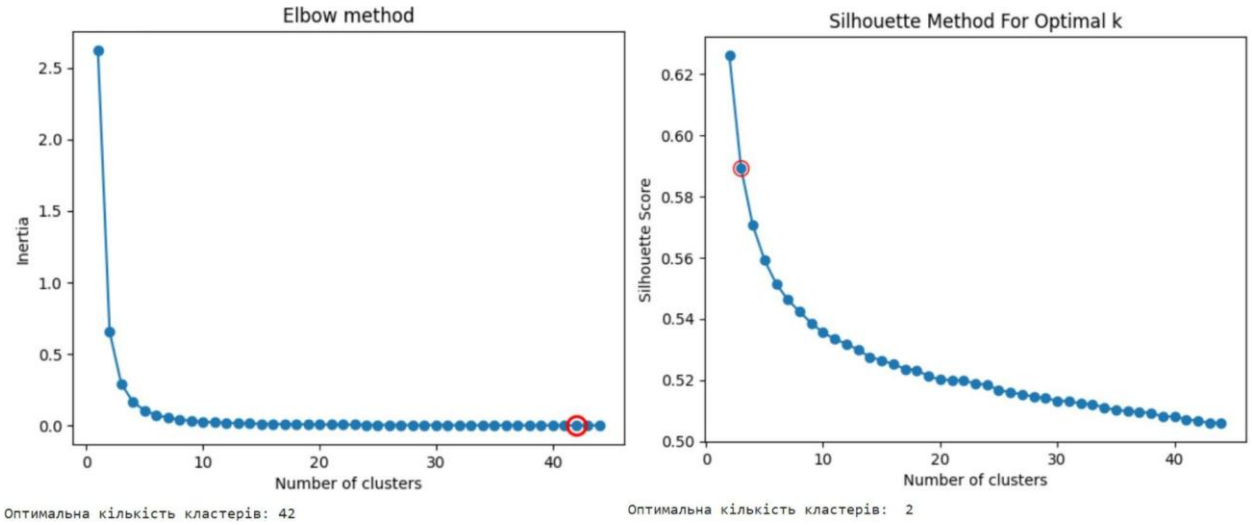


Рисунок 11 - Результат роботи методів знаходження кількості кластерів.

У даному коді (рис 12) здійснюється кластеризація даних методом К-середніх на 41 кластер. Спочатку створюється об'єкт «kmeans» з встановленим параметром «n_clusters» на значення 41, що вказує кількість кластерів, на які будуть розбиті дані. Потім метод «fit» виконує кластеризацію, і кожній точці з набору даних надається мітка класу, який вона належить від 0 до 40.

```
kmeans = KMeans(n_clusters=41)
kmeans.fit(data)
```

Рисунок 12 - Приклад коду який задає параметри алгоритму К-середнього.

Хоча алгоритм було запущено для розбиття даних як на 41 кластер так і на 42 кластери, відчутної різниці це не дало. На жаль алгоритм відпрацював некоректно, точки були розподілені між кластерами із великою кількістю хибних розподілень, що можна побачити на рисунку 13, де демонструється доба, яка була невдало розділена на кластери. Через наявність великих послідовностей нулів, що відповідають вихідним дням, крайні точки цих послідовностей знаходяться надто далеко від центроїда кластеру і алгоритм К-середніх намагається розбити цю послідовність на кілька кластерів, що призводить до некоректної роботи усього алгоритму. Також на рисунку помітно що алгоритм зміг знайти аномальну точку, проте відніс її до сусіднього кластера. Через некоректне розбиття точок на кластери алгоритм не може зробити висновок що ця точка є аномальною.

Рисунок 13 - Добовий приклад результату роботи алгоритму К-середніх.

3.3 Програмна реалізація алгоритму середнього ковзного

3.3.1 Опис внесених змін

Звичайний алгоритм середнього ковзного було модифіковано для вирішення поставленої задачі наступним чином: було зроблено припущення, що якщо розділити дані на певні періоди, то усереднивши значення, яке повертає датчик руху, і яке є ідентичним за один і той же час з різних періодів, при досить великій вибірці, можна отримати значення це цей час, яке є істинним. В такому випадку значення, що відрізнятимуться від отриманого значення на певну величину вважатимуться хибними. Тобто якщо в межах певного періоду значення зустрічається частіше ніж в 50% випадків, можна припустити, що саме воно і є істинним.

В таблиці 2 наведено приклад, порівняння значення за певний час. При встановленій точності «0.5», порівнюючи значення за конкретний час тестової вибірки із середнім значенням цього ж часу за кілька періодів тренувальної вибірки можна встановити, що значення за «Час 1» є гарантовано аномальним, значення за «Час 3» є гарантовано істинним, а значення за «Час 2» та «Час 4» можуть бути як істинними так і хибними.

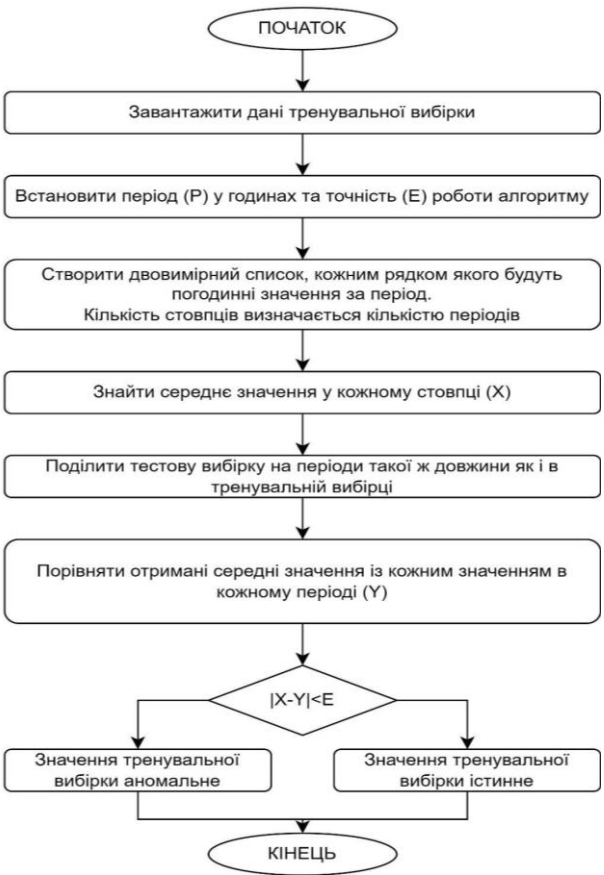
Таблиця 2 – Приклад роботи модифікованого алгоритму середнього ковзного

	Тренувальна вибірка	Тестова вибірка

	Період 1	Період 2	Період 3	Період 4	Середнє значення за час	Період 1
Час 1	1	1	1	1	1	0
Час 2	0	1	1	0	0.50	1
Час 3	1	1	0	1	0.75	1
Час 4	1	0	1	0	0.50	0

Проте цей недолік вирішується шляхом збільшення періодів вибірки, це дозволить отримати більше істинних значень за певний час, або шляхом зміни точності, проте це може спричинити хибнопозитивні спрацювання алгоритму, якщо кількість хибних значень наближається до кількості нормальних значень.

Код наведений у додатку А реалізує модифікований алгоритм середнього ковзного. Цей код завантажує дані з двох CSV файлів: один з нормальними даними «TestData» та другий з даними, які містять аномалії: «TestData_with_mistakes». Код обчислює середнє значення з періоду, заданого змінною «cycle_length» та порівнює кожне значення, отримані з файлу



«testdata_with_mistakes з відповідним середнім значенням (рис 14). Якщо різниця між даним значенням та середнім значенням перевищує 0.5, то вважається, що це аномалія, тому індекс цього значення додається до списку «anomaly_indices».

Рисунок 14 - Блок схема модифікованого алгоритму МА

3.3.2 Опис результатів роботи алгоритму

В ході тестування роботи алгоритму, його було протестовано у різних конфігураціях, а саме на тестовій вибірці, на тренувальній вибірці та з різними значеннями періодів та точності.

При еталонній ітерації (рис. 15) можна побачити результат роботи алгоритму при розбитті даних на щотижневі періоди та з точністю 0.5. Знайдено 27 аномалій, хибних спрацювань не помічено. Дана ітерація є такою, коли умови були штучно створені такими, щоб максимізувати кількість аномалій, що буде знайдено.

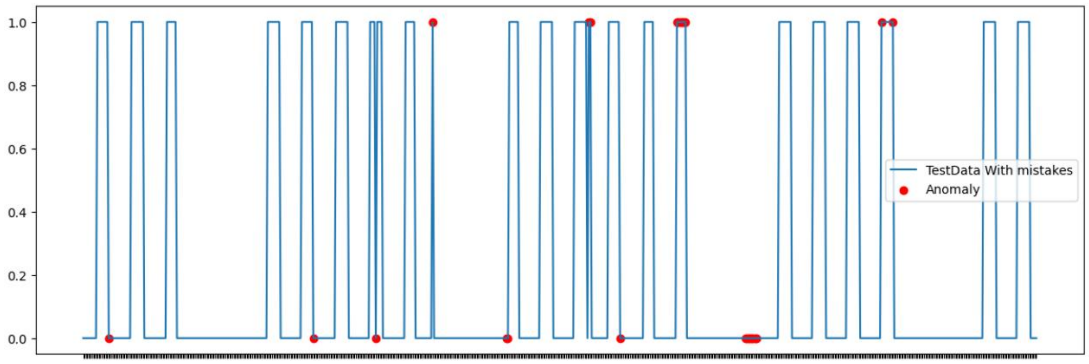


Рисунок 15 - Результат роботи модифікованого алгоритму середнього ковзного, еталонна ітерація.

При спробі зменшити період до однієї доби, та встановленій точності 0.5 (рис. 16), було встановлено, що середнє значення вираховується подобово і через це не враховується специфіка вихідних днів (суботи та неділі). Це призводить до хибно позитивних спрацювань. Знайдено 89 аномальних значень. Таким чином встановлено, що найбільш оптимальними є саме щотижневі інтервали.

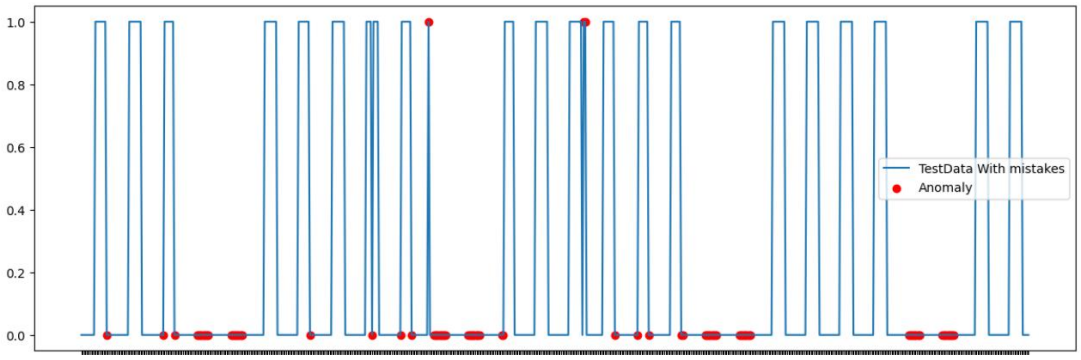


Рисунок 16 - Результат роботи модифікованого алгоритму середнього ковзного, добовий період, точність 0.5.

Наступним кроком була перевірка роботи алгоритму при зміні точності. Для цього було внесено зміни в тренувальну вибірку таким чином, щоб кількість аномальних значень в певний час була практично рівною кількості вірних значень. Це дозволить виявити вразливість алгоритму у випадках, коли кількість аномальних значень є близькою до кількості вірних значень, проте все ж є нижчою.

На рисунку 17 можна побачити результат роботи алгоритму при розбитті даних на щотижневі інтервали та з точністю 0.6. Знайдено 35 аномалій, що означає наявність хибно позитивних спрацювань алгоритму.

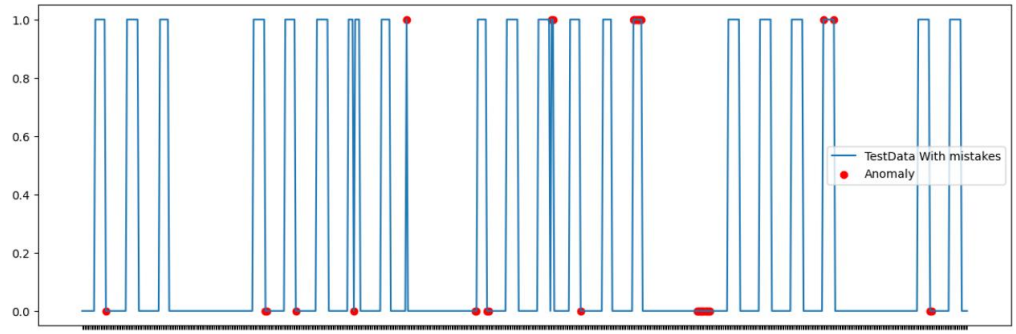
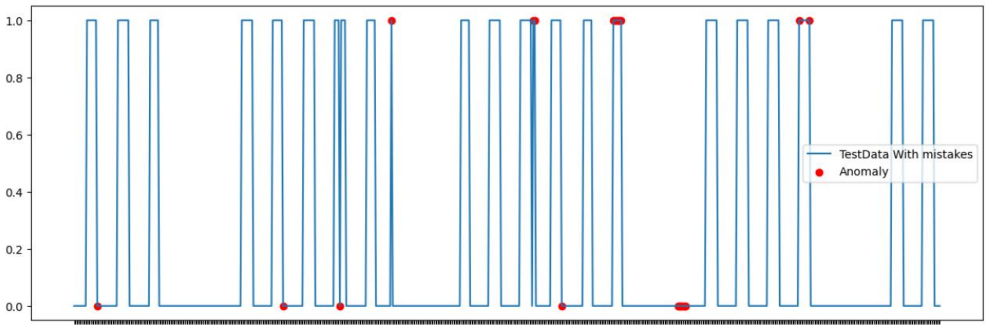


Рисунок 17 - Результат роботи модифікованого алгоритму середнього ковзного, тижневий період, точність 0.6.

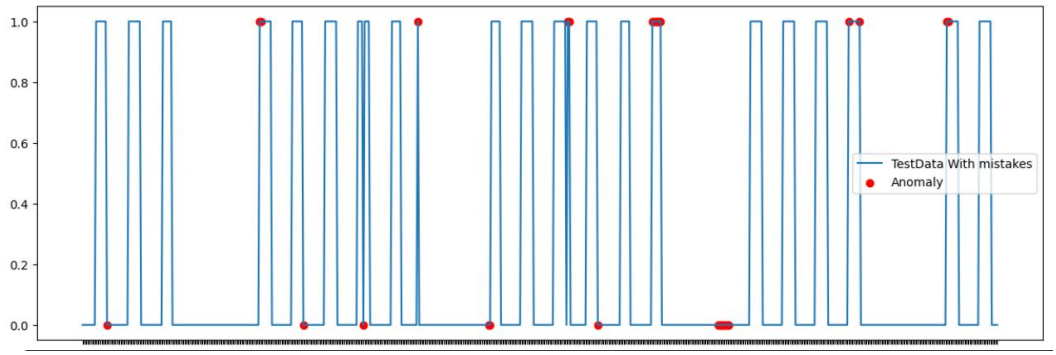
На рисунку 18 можна побачити результат роботи алгоритму при використанні оригінальної тестової вибірки як тренувальної. Тобто використовувались реальні значення із присутніми аномаліями. Для цієї ітерації алгоритм запускався лише на тестовій вибірці, щоб проімітувати ситуацію обмеженої кількості даних та ситуації, коли у тренувальній вибірці присутні аномалії, які не було усунуто. При розбитті даних на щотижневі інтервали та з точністю 0.5 - знайдено 23 аномалії, що означає наявність хибно негативних



спрацювань алгоритму.

Рисунок 18 - Результат роботи модифікованого алгоритму середнього ковзного, лише на тестовій вибірці, тижневий період, точність 0.5.

У ході наступної ітерації було також використано тестову вибірку як тренувальну (рис. 19). При розбитті даних на щотижневі інтервали та з точністю 0.6 - знайдено 31 аномалію, що свідчить про хибно позитивні спрацювання



алгоритму.

Рисунок 19 - Результат роботи модифікованого алгоритму середнього ковзного, лише на тестовій вибірці, тижневий період, точність 0.6.

Після проведених тестів можна встановити, що оптимальним варіантом для алгоритму середнього ковзного є використання тижневого інтервалу та точності 0.5. При використанні більшої точності зростає ймовірність виникнення хибно позитивних спрацювань алгоритму.

Зважаючи на значення (A), знайдене під час еталонної ітерації було розраховано відносну похибку (E) для значень, розрахованих при інших ітераціях (B), за формулою (2):

$$E = \frac{|B-A|}{A} * 100\% \quad (2)$$

Таким чином було встановлено, що точність алгоритму при запуску його на реальних даних з точністю 0.5 та 0.6 дає ефективність приблизно 85% (таблиця 3). Також максимальний час роботи алгроритму в середовищі «Google collab» склав 5 секунд. Якщо ж алгоритм у якості тренувальної вибірки отримає набір даних у якому відсутні аномальні значення, ефективність алгоритму для даних, отриманих із датчика руху може зрости до 100%, проте в даному випадку необхідно провести підготовчу роботу по формуванню такого пакету даних. Також алгоритм не має можливості оцінити сезонні коливання чи свята і помічатиме такі події як аномальні. Проте зважаючи на свою простоту алгоритм може використовуватися максимально близько до місця генерації даних.

Таблиця 3 – Порівняння ефективності роботи модифікованого алгоритму середнього ковзного, при різних початкових умовах.

	Еталонна ітерація інтервал: тиждень, точність: 0.5	інтервал: доба, точність: 0.5	інтервал тиждень, точність: 0.6	інтервал тиждень, точність: 0.5 запуск на тестовій вибірці	інтервал тиждень, точність: 0.6 запуск на тестовій вибірці
кількість знайдених аномалій	27	89	35	23	31
відносна похибка	0%	229.63%	29.63%	14.81%	14.81%

3.4 Програмна реалізація алгоритму ARIMA

3.4.1 Розрахунок параметра «p»

В якості наступного алгоритму було вирішено скористатися алгоритмом передбачення даних типу Time Series. В якості такого алгоритму було використано алгоритм ARIMA.

Для побудови графіку автокореляційної функції (рис. 20), тобто для розрахунку параметру «p», було використано бібліотеку «statsmodels.graphics.tsaplots». Код для пошуку автокореляції розміщено у додатку Б. З графіка, зображеного на рисунку 20 можна побачити, що є два досить великих піки зі значеннями 0 та 24.

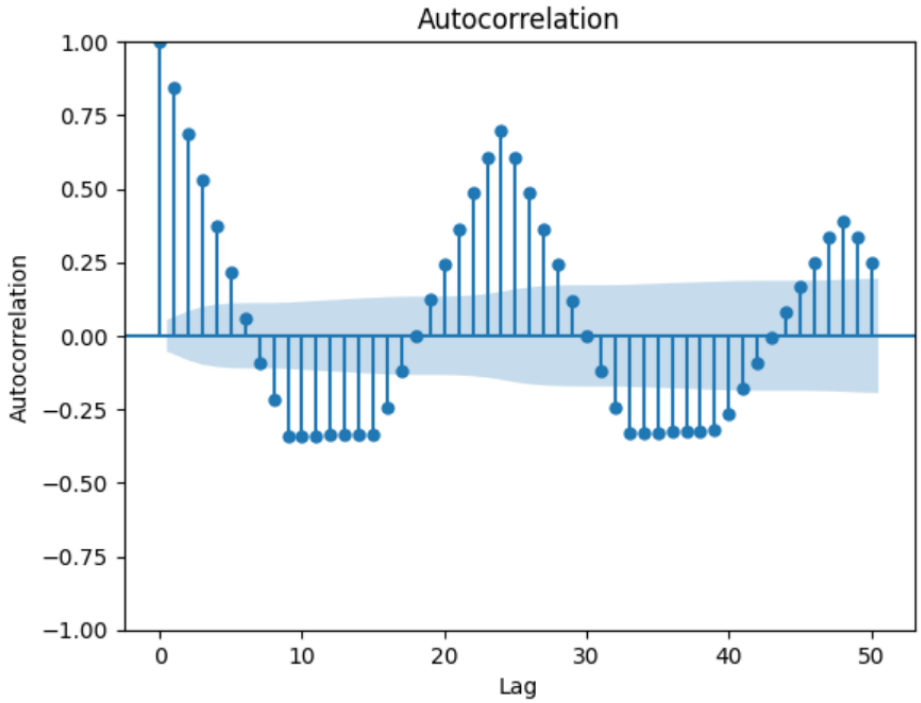


Рис. 20 – Графік автокореляційної функції

3.4.2 Розрахунок параметра «d»

Для перевірки стаціонарності часового ряду було використано тест Дікі-Фуллера.

Тест Дікі-Фуллера є одним з найпоширеніших тестів для перевірки наявності одиничного кореня в часовому ряді. Він використовується для оцінки стаціонарності ряду і визначення необхідності застосування диференціювання для отримання стаціонарного ряду.

Процедура тесту Дікі-Фуллера базується на побудові регресійної моделі, в якій залежна змінна представляється як різниця між спостереженнями в часовому ряді та їх попередніми значеннями. В моделі також враховуються інші фактори, такі як тренд або лаги ряду, що дозволяє врахувати потенційну автокореляцію в даних.

Результат тесту Дікі-Фуллера представляється двома основними величинами:

- ADF Statistic (статистика тесту Дікі-Фуллера): Це числове значення, яке порівнюється з критичними значеннями, що визначаються для рівнів значимості 1%, 5% і 10%. Якщо ADF Statistic менше за критичне значення, то ряд вважається стаціонарним.
- P-value (p-значення): Це ймовірність того, що гіпотеза про стаціонарність є правильною. Якщо p-значення менше заданого рівня значимості (звичайно 0.05), то ряд є стаціонарним.

У таблиці 4 наведено результати тесту Дікі-Фуллера, а код цього тесту розміщено у додатку В. У даному випадку, параметр «ADF Statistic» має негативне

значення, а параметр «p-value є меншим за 0.05, що є свідченнями на користь стаціонарності ряду.

Таблиця 4 – Результати роботи тесту Дікі-Фуллера

ADF Statistic	-3.094046
p-value	0.026997
Critical Value 1%	-3.435
Critical Value 5%	-2.864
Critical value 10%	-2.568

Варто зауважити, що значення параметра «ADF Statistic» є більшим за критичне значення на рівні 1% і меншим за критичне значення на рівнях 5і 10%. Це означає, що на рівнях значимості 5% і 10% можна відкинути нульову гіпотезу про наявність одиничного кореня в часовому ряді, але на рівні 1% цього зробити не можна. Таким чином, часовий ряд стаціонарний на рівнях значимості 5% і 10%. Тож параметр «d» дорінює 0.

3.4.3 Розрахунок параметра «q»

Параметр q в моделі ARIMA відповідає за кількість попередніх значень авторегресійної компоненти моделі. Для визначення оптимального значення q буде використано частковою автокореляційною функцією (PACF) часового ряду. Зазвичай, значущими вважаються лише ті коефіцієнти, які виходять за межі довірчого інтервалу. Якраз такими є перші два значення (рис. 21), що свідчить про

те, що параметр «q» дорівнює 2. Власне код реалізації часткової автокореляційної функції наведено у додатку Г.

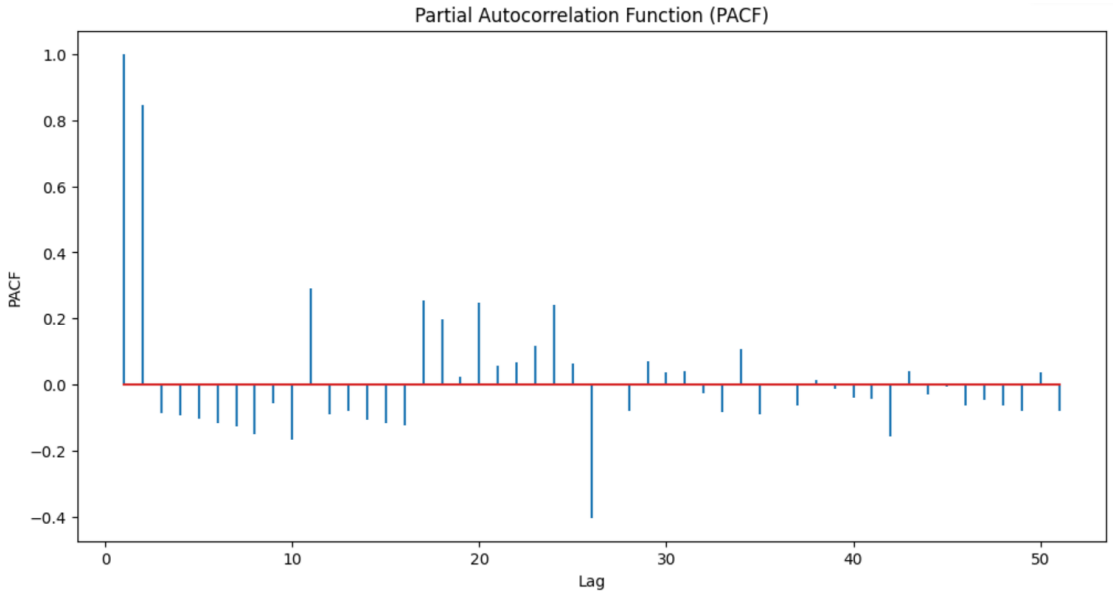


Рис. 21 – Графік часткової автокореляційної функції

3.4.4 Розрахунки та результати роботи моделі ARIMA

Позаяк при розрахунках параметра «p» було отримано два можливих результати, модель було запущено двічі. Відповідний код знаходиться у додатку Д.

Принцип роботи реалізованого алгоритму наступний (рис. 22): завантажити дані; згенерувати модель із параметрами, що були розраховані в пунктах 3.4.1,

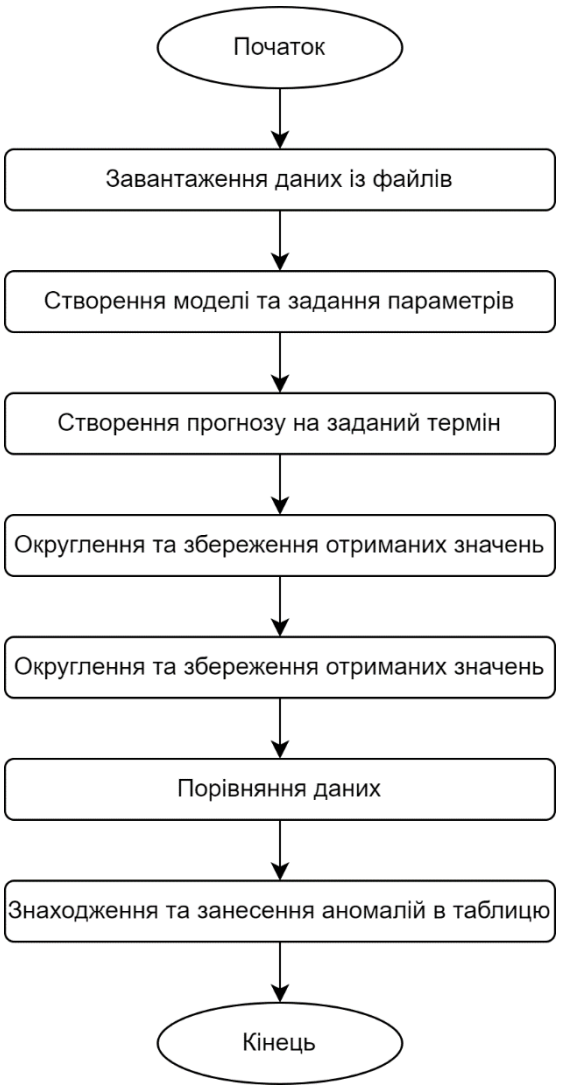


Рис. 22 – Блок схема реалізації роботи алгоритму ARIMA

3.4.2, 3.4.3; створення прогнозу на основі наявних даних; порівняння даних та занесення їх до тимчасової таблиці.

В результаті дослідження було виявлено, що при використанні значення параметра «p»=0 код видає результат через 9 секунд роботи в середовищі «Google

collab», при цьому при «p»=24 код виконується за 3 хвилини та 24 секунди. Зважаючи на те, що алгоритм у обох випадках видав однаковий результат, використання параметра «p»=0 є більш доцільним.

Результат роботи алгоритму продемонстровано на рисунку 23. Було знайдено 25 аномалій. Згідно формули (2), це становить 92.6%.

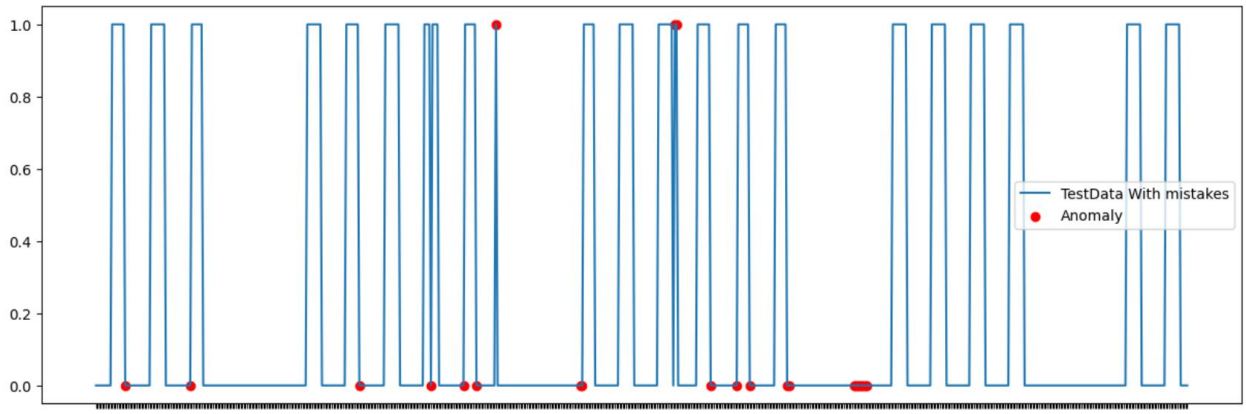


Рисунок 23 – Результат роботи алгоритму ARIMA

ВИСНОВКИ

У ході проведеного дослідження було встановлено, що:

Алгоритми кластеризації, а саме алгоритм К-середніх є непридатним для обробки даних, що надходять з датчика руху, в офісних приміщеннях

Модифікований алгоритм середнього ковзного та алгоритм ARIMA можуть виконати поставлену задачу з ефективністю 85-92%

Модифікований алгоритм середнього ковзного є простим у реалізації, та вимагає мінімальної кількості обчислювальних ресурсів на виконання, що дозволяє запускати його на контролері, якомога ближче до даних, що генеруються. Це потенційно дозволяє визначити позаштатні ситуації, які не були враховані на етапі проєктування та зменшити час реакції на подібні інциденти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Karen R. The Internet of Things an overview [Електронний ресурс] / R. Karen, S. Eldridge, L. Chapin // internetsociety.org. – 2015. – Режим доступу до ресурсу: https://d1wqtxts1xzle7.cloudfront.net/48790442/ISOC-IoT-Overview-20151014_0-libre.pdf?1473746977=&response-content-disposition=inline%3B+filename%3DThe_Internet_of_Things_An_Overview_Under.pdf&Expires=1684175584&Signature=O0WTIgqjlDR6j8vPNgSrLxKeN9KVgjXcLq398DaCt-cGhY8JtdZTVf7rYXjgjHhKLxTi5GI-4WOU5rq7Kp7l1m~x1r6L4grwQ3RZ8ms0~QYLMiZWfxKxCE8-Kj4ytNARFURaQLZ5lO1MvhJ52D2TpZN2UTrROujD8LPYeTIazgx4H39HNLGikL2z6CvwhQZv~N3a8fMK4JUxiqVjmfRDI~k8f7sgKjo~WS8FCVVQz8HyPsUVjn6IijwDehhjrCGZrwOaPYT390TKtqTVJl-bX92uWziUH-HgHUZIBivqMkD~Sl3nlHqcg4h6PRl7YLocVrdZwmhz5Y3v-jLvLZHBVw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA.

2. Kerem G. How is machine learning utilized for time series forecasting?. dataconomy.com. URL: https://dataconomy.com/blog/2022/11/25/time-series-forecasting-machine-learning/.

3. Different Types of Motion Sensors And How They Work. ElProCus - Electronic Projects for Engineering Students. URL: https://www.elprocus.com/working-of-different-types-of-motion-sensors/.

4. Chandola V., Banerjee A., Kumar V. Anomaly detection. ACM Computing Surveys. 2009. Vol. 41, no. 3. P. 1–58. URL: https://doi.org/10.1145/1541880.1541882

5. MSV J. Is Fog Computing The Next Big Thing In Internet of Things?. Forbes. URL: https://www.forbes.com/sites/janakirammsv/2016/04/18/is-fog-computing-the-next-big-thing-in-internet-of-things/?sh=6b98acb1608d.

6. Pelleg D. K-means and X-means implementations. CMU School of Computer Science. URL: http://www.cs.cmu.edu/~dpelleg/kmeans.html.

7. Yocelyn K. What are the Four Major Moving Averages? - Article. brokerxplorer. URL: https://www.brokerxplorer.com/article/what-are-the-four-major-moving-averages-3824.

8. Жураковський Б., Зенів І. Технології інтернту речей. ela.kpi.ua. URL: https://ela.kpi.ua/bitstream/123456789/42078/1/Zhurakovskyi_B_Zeniv_Tehnologii_internet_rechey.pdf.

ДОДАТОК А

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

df_testdata = pd.read_csv('TestData.csv')

df_testdata_with_mistakes = pd.read_csv('TestData with mistakes.csv')


# обрахування середнього ковзного

cycle_length = 168

mean_values = []

for i in range(cycle_length):

    data_values = df_testdata['Data'][i::cycle_length].values

    mean_value = np.mean(data_values)

    mean_values.append(mean_value)


# знаходження індексів аномальних файлів

anomaly_indices = []

for i in range(len(df_testdata_with_mistakes)):

    index_in_cycle = i % cycle_length

    mean_value = mean_values[index_in_cycle]
```

```
            data_value = df_testdata_with_mistakes['Data'][i]
            if abs(data_value - mean_value) > 0.5:
                anomaly_indices.append(i)


        # створення таблиці з аномаліями
        anomaly_table = df_testdata_with_mistakes.loc[anomaly_indices].copy()
        anomaly_table.rename(columns={'Date_Time': 'Anomaly_Date_Time', 'Data': 'Anomaly_Data'}, inplace=True)


        # вивід графіка
        plt.figure(figsize=(15,5))
        plt.plot(df_testdata_with_mistakes['Date_Time'], df_testdata_with_mistakes['Data'], label='TestData With mistakes')
        plt.scatter(anomaly_table['Anomaly_Date_Time'], anomaly_table['Anomaly_Data'], color='red', label='Anomaly')
        plt.legend()
        plt.show()
        # вивід таблиці з аномаліями
        print(anomaly_table, anomaly_table.count())
```

ДОДАТОК Б

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
from statsmodels.graphics.tsaplots import plot_acf
from pandas.plotting import register_matplotlib_converters

# Завантаження часового ряду з файлу або іншого джерела
time_series = pd.read_csv('TestData.csv')

# Перетворення значень в формат дати/часу
time_series['Date_Time'] = pd.to_datetime(time_series['Date_Time'])

# Налаштування конвертера Matplotlib для обробки дати/часу
register_matplotlib_converters()

# Побудова графіка автокореляційної функції (ACF)
plot_acf(time_series['Data'], lags=50)
plt.xlabel('Lag')
plt.ylabel('Autocorrelation')
plt.show()
```

ДОДАТОК В

```
import pandas as pd
from statsmodels.tsa.stattools import adfuller

# Завантаження часового ряду з CSV-файлу
time_series = pd.read_csv('TestData.csv')

# Перетворення значень в формат дати/часу
time_series['Date_Time'] = pd.to_datetime(time_series['Date_Time'])

# Виконання тесту Дікі-Фуллера
result = adfuller(time_series['Data'])

# Виведення результатів тесту Дікі-Фуллера
print('ADF Statistic:', result[0])
print('p-value:', result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

ДОДАТОК Г

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt

# Завантаження часового ряду
time_series = pd.read_csv('TestData.csv')

# Перетворення значень в формат дати/часу
time_series['Date_Time'] = pd.to_datetime(time_series['Date_Time'])

# Побудова PACF
pacf = sm.tsa.stattools.pacf(time_series['Data'], nlags=50)

# Графік PACF
plt.figure(figsize=(12, 6))
plt.stem(range(1, len(pacf) + 1), pacf, markerfmt=' ')
plt.xlabel('Lag')
plt.ylabel('PACF')
plt.title('Partial Autocorrelation Function (PACF)')
plt.show()

# Визначення параметра q
q = np.where(pacf < 0.2)[0][0]
print('Optimal value of q:', q)
```

ДОДАТОК Д

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from statsmodels.tsa.statespace.sarimax import ARIMA

# загрузка даних
df = pd.read_csv('TestData.csv', parse_dates=['Date_Time'], index_col='Date_Time')

# навчання моделі ARIMA
model = ARIMA(df, order=(0, 0, 2))
model_fit = model.fit(disp=False)

# генерація прогнозу
forecast = model_fit.forecast(steps=672)

# округлення
forecast = forecast.round().astype(int)

# перейменування значень
forecast[forecast == 1.0] = 1
forecast[forecast != 1.0] = 0

# створення нового датафрейму з прогнозом
new_df = pd.DataFrame({'Data': forecast}, index=pd.date_range(start=df.index[-
1]+pd.Timedelta(hours=1), freq='H', periods=len(forecast)))

# збереження нового датафрейму
```

```
new_df.to_csv('TestData2.csv', index_label='Date_Time')

# загрузка даних з файлів
df_testdata = pd.read_csv('TestData2.csv')
df_testdata_with_mistakes = pd.read_csv('TestData with mistakes.csv')

# порівняння даних
df_difference = df_testdata_with_mistakes['Data'] - df_testdata['Data']

# знаходження індексів аномальних значень
anomaly_indices = df_difference[df_difference != 0].index.tolist()

# створення таблиці з аномаліями
anomaly_table = df_testdata_with_mistakes.loc[anomaly_indices].copy()
anomaly_table.rename(columns={'Date_Time': 'Anomaly_Date_Time', 'Data': 'Anomaly
_Data'}, inplace=True)



# виведення таблиці з аномаліями та графіку
print(anomaly_table)
plt.figure(figsize=(15,5))
plt.plot(df_testdata_with_mistakes['Date_Time'], df_testdata_with_mistakes['Data'], lab
el='TestData With mistakes')
plt.scatter(anomaly_table['Anomaly_Date_Time'], anomaly_table['Anomaly_Data'], col
or='red', label='Anomaly')
plt.legend()
plt.show()
anomaly_table.count()
```