Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Модель виявлення аномалій у даних про споживання електроенергії

(тема)

Виконав:

студент II курсу, групи СПм-22-5

Братищенко М.Р.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Філімончук Т.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Братищенку Микиті Руслановичу
(прізвище, ім’я, по батькові)

1. Тема роботи Модель виявлення аномалій у даних про споживання електроенергії

затверджена наказом по університету від “ 01 ” квітня 2024 р. № 257 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 червня 2024 р.

3. Вхідні дані до роботи 1) статистичні дані про споживання електроенергії;
2) існуючі методи та алгоритми виявлення аномалій; 3) навчальні набори даних.

4. Перелік питань, що потрібно опрацювати у роботі
1) аналіз предметної області;
2) аналіз існуючих методів виявлення аномалій в даних;
3) вибір інструментів для моделі виявлення аномалій в даних про енергоспоживання;
4) тестування запропонованої моделі CADM-ECD;
5) об’єктивна та суб’єктивна оцінка моделі CADM-ECD.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

Слайд-презентацій – 12 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	02.04.24-08.04.24	
2	Аналіз існуючих методів виявлення аномалій	09.04.24-16.04.24	
3	Вибір інструментів для моделі виявлення	17.04.24-22.04.24	
	аномалій в даних про енергоспоживання		
4	Тестування моделі CADM-ECD	23.04.24-06.05.24	
5	Об'єктивна та суб'єктивна оцінка моделі	07.05.24-23.05.24	
	CADM-ECD		
6	Оформлення матеріалів кваліфікаційної роботи	24.05.24-03.06.24	
7	Подання кваліфікаційної роботи керівникові	04.06.24-07.06.24	
	та її попередній захист		
8	Подання кваліфікаційної роботи на	08.06.24-12.06.24	
	рецензування		

Дата видачі завдання 01 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Філімончук Т.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 74 с., 14 рис., 14 джерел.

ЕНЕРГОСПОЖИВАННЯ, АНОМАЛІЯ, ВИЯВЛЕННЯ АНОМАЛІЙ, МОДЕЛЬ, СТАТИСТИЧНІ ДАНІ, МАШИНЕ НАВЧАННЯ, МЕТОДИ.

Метою кваліфікаційної роботи є досідження моделі для виявлення аномалій в даних про енергоспоживання.

У ході виконання кваліфікаційної роботи було досліджено методи та алгоритми для виявлення аномалій в даних. Вони допомагають ідентифікувати нетипові значення в різноманітних наборах даних та дозволяють працювати з ними далі (побудова графіків, аналіз з метою мінімізувати виникнення цих аномалій, проведення обчислень на їх основі тощо). Було проведено порівняльний аналіз доступних методів, в результаті чого були обрані найбільш ефективні з них. Модель відповідає останнім вимогам завдання, при розробці були використані сучасні технології.

Використовуючи комбінований підхід, який об'єднав методи виявлення аномалій в даних, модель показала очікувані результати на тестових наборах даних. Це підтвердило її здатність ідентифіковувати відхилення від нормальних значень у показниках споживання електроенергії.

У підсумку, проведена кваліфікаційної робота не тільки підтверджує ефективність обраної моделі для виявлення аномалій в даних про енергоспоживання, але і вказує на потенціал її можливого застосування в реальних умовах. Це може стати важливим кроком в електроенергетиці.

ABSTRACT

Master’s thesis: 74 pages, 14 figures, 14 sources.

ENERGY CONSUMPTION, ANOMALY, ANOMALY DETECTION, MODEL, STATISTICS, MACHINE LEARNING, METHODS.

The purpose of the qualification work is to test the model for detecting anomalies in energy consumption data.

In the course of the qualification work, methods and algorithms for detecting anomalies in data were investigated. They help to identify atypical values in various data sets and allow you to work with them further (building graphs, analysing to minimise the occurrence of these anomalies, performing calculations based on them, etc.) A comparative analysis of the available methods was carried out, and the most effective ones were selected. The model meets the latest requirements of the task, and modern technologies were used in its development.

Using a combined approach that combines methods for detecting anomalies in data, the model showed the expected results on test data sets. This confirmed its ability to identify deviations from normal values in electricity consumption.

In summary, the qualification work carried out not only confirms the effectiveness of the selected model for detecting anomalies in energy consumption data, but also indicates the potential for its possible application in real-world conditions. This could be an important step in the electricity sector.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БРІКС – міжнародна організація найбільших за площею та населенням країн, що розвиваються (Бразилія, Росія, Індія, КНР, ПАР)

ВДЕ – вхідні дані щодо енергоспоживання

МВА – модуль виявлення аномалій

МГВ – модуль графічного відображення

ООП – об'єктно-орієнтоване програмування

ПРАІЛ – підмодуль з реалізацією алгоритму «Ізоляційний ліс»

ППМАКС – підмодуль побудови моделі авторегресії ковзного середнього

ARIMA – модель авторегресійної інтегрованої ковзньої середньої (англ., AutoRegressive Integrated Moving Average)

Isolation Forest – метод для виявлення викидів (аномальних значень) в наборах даних

K-means – алгоритм кластеризації машинного навчання

LOF – алгоритм машинного навчання, оцінює ступінь аномальності кожної точки даних на основі її сусідніх точок (англ., Local Outlier Factor)

OC-SVM – однокласовий метод опорних векторів (англ., One-class Support vector machine)

SVM – метод опорних векторів, що використовується для класифікації об’єктів (англ., Support vector machine)

UNC – підхід до кластеризації даних без попереднього маркування (без навчання з учителем) (англ., Unsupervised Niche Clustering)

VAR – векторна авторегресія (англ., Vector autoregrassion)

CADM-ECD – коммпозитна модель виявлення аномалій у даних про споживання електроенергії (англ. Composite Anomaly Detection Model for Electrical Consumption Data)

ВСТУП

На сьогоднішній день людство використовує електроенергію багато в яких сферах свого життя, починаючи від побутових електроприладів, і закінчуючи електрообладнанням на підприємствах та електростанціях. Проте треба пам’ятати, що надмірне споживання електроенергії це погано, а особливо якщо є різкі скачки напруги – можуть погоріти електроприлади або електрообладнання вийде з ладу.

Сучасні люди постійно шукають нові способи використання енергії для покращення свого життя, тому попит на неї зростає. У більшості випадків компаніям та галузям важко контролювати всі свої пристрої одночасно, що може призвести до втрати електроенергії в будь-який час. В результаті операційні витрати будуть більшими, ніж необхідно. Крім того, втрата електроенергії сприяє глобальному потеплінню через вивільнення вуглецю, коли енергія генерується шляхом спалювання вугілля, газу та нафти.

Збір та аналіз таких даних може допомогти ефективніше використовувати ресурси, розробляти стратегії для зменшення споживання електроенергії та підвищення енергоефективності. Самі ж дані можна збирати як і в режимі реального часу (знімати показники з різних приладів або електрообладнання), так і за певний проміжок часу. Після чого отримані дані можна агрегувати не тільки по тому, скільки електроенергії було спожито (кВт/год), а ще за таким покажчиками, як: напруга, сила струму, потужність, частота струму тощо.

Ця робота має на меті запропонувати модель виявлення аномалій в даних енергоспоживання, яка буде базуватись на основі комбінації існуючих методів аналізу та виявлення аномалій в даних. Модель буде оцінена на декількох наборах даних енергоспоживання з різними проміжками часу зчитування цих показників. Це дасть змогу проаналізувати запропоновану модель та оцінити її ефективність, точність, час, який необхідний для формування результатів та їх коректність.

1 ПОСТАНОВКА ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Інформаційні системи енергоменеджменту або системи енергоменеджменту будівель – це інструменти для безперервного моніторингу енергоменеджменту, які регулярно збирають та аналізують дані про енергію. Однак, оскільки неможливо отримати фіксовані характеристики та не можна посилатися на поточну статичну модель енергії, вони дають низьку точність прогнозування споживання електроенергії.

Отже модель, яка здатна завчасно виявити відхилення від історичного використання енергії, є важливою та корисною на сьогоднішній день. Тому що електроприлади стали невід’ємною частиною нашого життя.

Виявлення аномалій може запобігти великому споживанню електроенергії для досягнення енергозбереження, нагадувати користувачам про виявлення несправних електроприладів або змінювати неправильні схеми споживання електроенергії, знижувати витрати користувачів на енергоспоживання та сприяти обізнаності щодо безпеки споживання електроенергії [1].

Дані про споживання електроенергії зазвичай зберігаються за допомогою різних систем та технологій, в залежності від потреб та можливостей конкретного об'єкта або системи. Ось деякі загальні способи збереження даних про споживання електроенергії:

- лічильники електроенергії можуть збирати дані на місці самого споживача. Сучасні "розумні" лічильники можуть автоматично передавати дані через мережу зв'язку, що спрощує збір і моніторинг інформації;
- системи автоматизації будівель використовуються у великих промислових та комерційних об'єктах, які включають в себе збір даних про енергоспоживання.
- системи управління енергоефективністю доцільно використовувати для промислових підприємств та великих установ. Вони включають в себе збір та аналіз даних про енергоспоживання для оптимізації роботи систем;

- хмарні технології: збір даних можливо здійснювати в хмарних сервісах, де їх легко аналізувати та використовувати з різних місць і пристроїв;

- централізовані системи моніторингу підходять до великих енергетичних мереж чи країн задля збору та аналізу даних енергоспоживання.

Щодо формату зберігання різних показників енергоспоживання за певний проміжок часу, то вони можуть зберігатися у різних форматах в залежності від конкретних потреб системи, програмного забезпечення (ПЗ) та вимог користувача. Ось деякі з популярних форматів для зберігання таких даних: CSV, JSON, XML, Parquet, Apache Avro. У тому випадку, коли необхідно зберігати великі обсяги даних, доцільно використовувати реляційні бази даних, такі як MySQL, PostgreSQL чи Microsoft SQL Server. Статистичні дані можуть бути структуровані та збережені в таблицях, а за допомогою мови SQL є можливість робити різноманітні запити для їх отримання, аналізу та виконання різних операцій.

Важливо також зосередитись на тому, щоб забезпечити високий відсоток точності при визначенні, чи були нетипові значення аномалією в той чи інший період часу в показниках електроспоживання. Наприклад, можна розглядати використання методів машинного навчання, які призначені як раз таки для визначення аномалій.

Однак, створення такої універсальної моделі є великим викликом. Вона має враховувати численні фактори, що впливають на точність розрахунків, і бути спроможною ефективно обробляти величезні набори даних. Не дивлячись на це, розробка такої моделі може значно покращити енергоменеджмент будь-то побутових пристроїв, чи навіть електрообладнання на підприємствах або електростанціях. На основі результатів роботи моделі можна робити висновки стосовно того, чи є несправності в електроустановках або навіть запобігти перенавантаження електромереж та електроприладів.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ВИЯВЛЕННЯ АНОМАЛІЙ

2.1 Аномалії в даних та їх види

На початку аналізу існуючих методів та рішень щодо врегулювання енергоспоживання та ідентифікації нетипових значень варто ознайомитись із статистичними даними про енергоспоживання у різних країнах світу за останні роки.

У документі [2] представлені та проаналізовані дані, взяті з кількох досліджень про енергоспоживання будівель у США, ЄС та БРІКС країн (Бразилія, Росія, Індія, Китай, ПАР). Більшість поточних досліджень споживання енергії стосується статистики конкретної країни. Однак міжнародні порівняння корисні для виявлення історичних, фактичних тенденцій та тенденцій споживання енергії. Дані представлені у звітах Світового банку, Програми ООН з навколишнього середовища, Міжурядової групи експертів зі змін клімату та міжнародного енергетичного агентства. Вони порівнюються з національними звітами, а також із дослідженнями. Цей аналіз показує, що країни БРІК вже подолали загальне енергоспоживання розвинутих країн. Але розширення їх будівельного фонду викликає нагальну потребу в енергоефективності в будівлях. Водночас, можна зробити висновок, що заходів, прийнятих у розвинених країнах, недостатньо для того, щоб гарантувати значне скорочення енергії споживання в будівлях та на підприємствах.

Виявлення аномалій – це метод розпізнавання даних, які відрізняються від звичайних. Аномалії в даних – це ситуації, які не відповідають визначеній звичайній моделі поведінки. Для того щоб проаналізувати та ідентифікувати аномалії треба сформувати набір даних за певний період часу чи в реальному часі (або використовувати вже готові дані). Аномалії, загалом, також відомі як викиди, девіанти, неузгодженості або винятки. Як правило, аномалії бувають наступних типів:

- точкові аномалії, які є найпростішим і дуже поширеним випадком. Точкові аномалії часто представляють екстремум, нерегулярність або відхилення, що трапляються випадковим чином і не мають особливого значення;

- контекстуальні аномалії, які часто виявляються в часових рядах та просторових даних. Це випадки, які можна розглядати як аномальні у якомусь конкретному контексті. Але варто зазначити, що спостереження однієї і тієї ж точки в різних контекстах не завжди дасть ознаки аномальної поведінки;

- колективні аномалії – це група корельованих, взаємопов’язаних або послідовних випадків, коли кожен конкретний екземпляр сам по собі не повинен бути аномальним, але їх колективне виникнення є аномальним.

2.2 Аналіз існуючих рішень для виявлення аномалій

Існують три основні категорії методів виявлення аномалій: контрольована, напівконтрольована та без нагляду.

Методи виявлення аномалій без нагляду (неконтрольовані алгоритми) визначають аномалії на непозначеному наборі даних, виходячи з припущення, що більшість зразків у цьому набору є нормальними, і шукаючи зразки, що виглядають якнайменше відповідними решті набору даних. Зазвичай ці методи застосовуються в таких галузях, як:

- системи виявлення вторгнень: ці типи систем бувають у вигляді програмного або апаратного забезпечення, яке відстежує мережний трафік на наявність ознак порушень безпеки або зловмисної активності. Алгоритми машинного навчання можна навчити виявляти потенційні атаки на мережу в режимі реального часу, захищаючи інформацію користувачів та функції системи. Ці алгоритми можуть створювати візуалізацію нормальної роботи на основі даних часових рядів, які аналізують точки даних через задані інтервали протягом тривалого часу. Стрибки мережного трафіку або несподівані патерни можуть бути позначені та розглянуті як потенційні порушення безпеки;

- виробництво: переконатися, що обладнання функціонує належним

чином, дуже важливо для виробництва продукції, оптимізації забезпечення якості та підтримки ланцюгів поставок. Алгоритми навчання без нагляду можна використовувати для прогнозованого технічного обслуговування, отримуючи немарковані дані з датчиків, прикріплених до обладнання, і роблячи прогнози щодо потенційних збоїв або несправностей. Це дозволяє компаніям проводити ремонт до того, як станеться критична поломка, скорочуючи час простою обладнання.

Методи контрольованого виявлення аномалій вимагають набори даних, що позначено як "нормальні" або "аномальні", та включають навчання класифікатора (ключовою відмінністю від інших задач класифікації є незбалансований характер виявлення викидів). Приклади використання цих методів наступні:

- роздрібна торгівля: використання маркованих даних про продажі за попередній рік може допомогти спрогнозувати майбутні цілі продажів. Це також може допомогти встановити орієнтири для конкретних працівників відділу продажів на основі їхніх минулих показників та загальних потреб компанії. Оскільки всі дані про продажі відомі, їх можна проаналізувати, щоб отримати уявлення про продукти, маркетинг і сезонність;

- прогноз погоди: використовуючи історичні дані, алгоритми керованого навчання можуть допомогти у прогнозуванні погодних умов. Аналіз останніх даних, пов'язаних з барометричним тиском, температурою та швидкістю вітру, дозволяє метеорологам створювати більш точні прогнози, які враховують мінливі умови.

Методи напівконтрольованого виявлення аномалій створюють модель, що представляє нормальну поведінку, виходячи із заданого нормального навчального набору даних, і потім перевіряють правдоподібність того, що тестовий екземпляр було породжено вивченою моделлю. Дані методи найбільше підходять до таких сфер, як:

- медицина: використовуючи алгоритми машинного навчання, медичні працівники можуть позначати зображення, які містять відомі захворювання або

розлади. Однак, оскільки зображення відрізняються від людини до людини, неможливо позначити всі потенційні причини появи недугів. Після навчання ці алгоритми можуть обробляти інформацію про пацієнта та робити висновки на немаркованих зображеннях та позначати потенційні причини виникнення хвороб;

- виявлення шахрайств: прогностичні алгоритми можуть використовувати напівконтрольоване навчання, яке вимагає як маркованих, так і немаркованих даних для виявлення шахрайства. Оскільки активність користувача за кредитною карткою маркується, її можна використовувати для виявлення незвичних витрат. Однак рішення для виявлення шахрайства не покладаються виключно на транзакції, які раніше були позначені як шахрайські, вони також можуть робити припущення на основі поведінки користувача.

2.2.1 Алгоритм «Ізоляційний ліс»

Ізоляційний ліс (Isolation Forest) – це один із алгоритмів неконтрольованого машинного навчання, який використовується для виявлення аномалій у наборі даних [3]. На відміну від керованих алгоритмів машинного навчання, Isolation Forest не потребує жодних міток чи класифікації для даних, які потрібно проаналізувати. Алгоритм або відокремлює аномалії, розглядаючи аномалії як випадки, які менш імовірні, або приписує значення, які дуже відрізняються від зазвичай приписуваних. Варто зазначити, що цей метод невимогливий щодо пам’яті. Також він добре працює в задачах великої розмірності, які мають велику кількість нерелевантних атрибутів.

Зазвичай, щоб виявити аномалії, дані, які є нормальними, профілюються в набори даних, а аномаліями вважаються зразки, які не відповідають цьому нормальному профілю. Isolation Forest використовує інший підхід: замість того, щоб будувати модель нормальних екземплярів, він явно ізолює аномальні точки в наборі даних. Алгоритм ізолює точки в даних, обираючи випадкову ознаку. Як правило, аномальні точки

спостерігаються в наборі даних рідше, ніж нормальні. Вони розташовані далі від нормальних точок даних. Аномалії повинні бути ближче до кореня дерева, це пов'язано з тим, що вони мають коротший середній шлях.

Алгоритм роботи методу Isolation Forest складається з двох кроків.

Крок 1: для створення ізольованих дерев (iTrees) використовується навчальний набір даних.

Крок 2: кожна вибірка в наборі даних є iTrees з попереднього кроку, компілюється, і зразку присвоюється відповідна оцінка аномалії.

Оцінка аномалії розраховується за формулою (вираз 2.1):

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}, \tag{2.1}$$

де: h(x) – довжина шляху спостереження x;

c(n) – середня довжина шляху невдалого пошуку бінарного дерева та кількість зовнішніх вузлів.

Кожному вузлу дерева (рисунок 2.1) присвоюється "оцінка аномалії" з урахуванням наступного правила: якщо оцінка близька до 1, то воно позначається як аномалія. Вузли з оцінкою менше 0.5 зазвичай не позначаються.

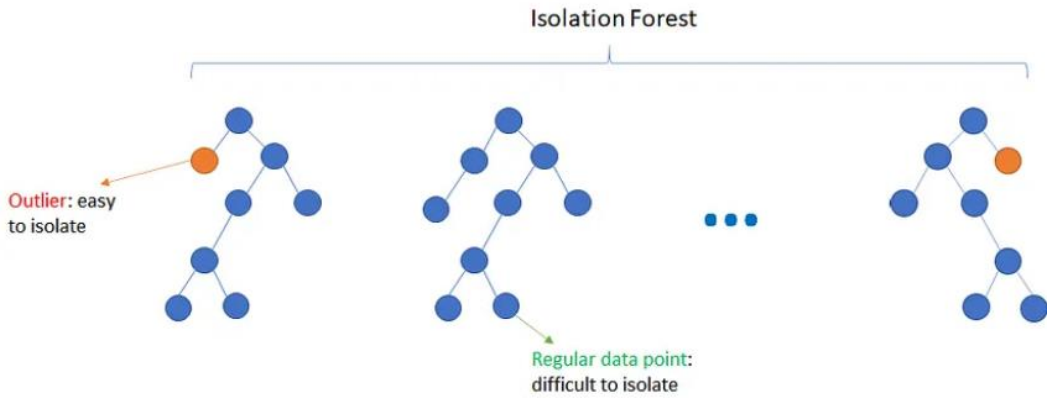


Рисунок 2.1 – Приклад визначення аномального вузла в дереві

На рисунку 2.2 можна побачити алгоритм Isolation Forest при визначення аномалій в даних щодо споживання електроенергії.

Класифіковані значення "0" використовувалися для позначення нормального споживання енергії, тоді як "1" вказувало на аномальне споживання енергії.

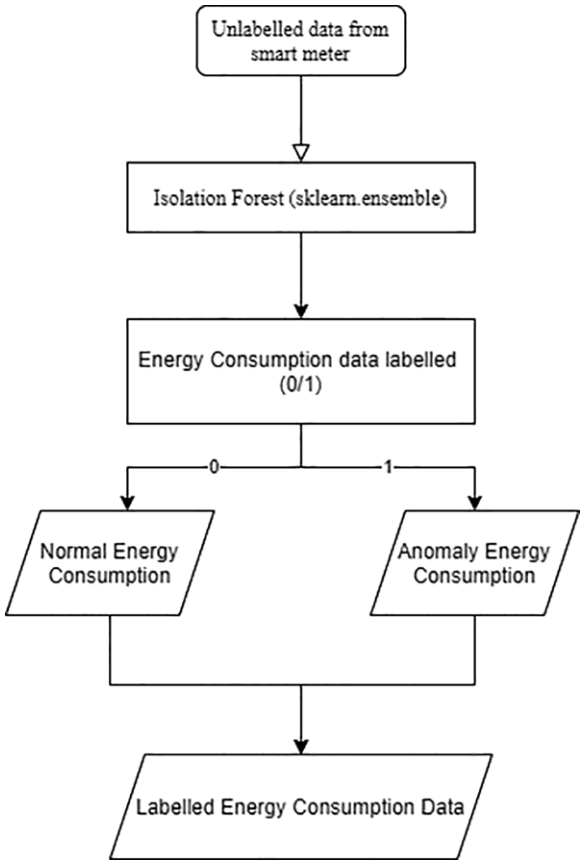


Рисунок 2.2 – Блок-схема маркування даних за алгоритмом Isolation Forest

2.2.2 Алгоритм кластеризації K-means

K-Means – це один із алгоритмів, що використовується для кластеризації та розбиває дані на кілька груп. Алгоритм K-Means є одним із методів неієрархічної кластеризації даних, який може групувати дані в кілька кластерів на основі подібності даних [4]. Цей механізм дозволяє групувати дані з однаковими характеристиками в один кластер, а дані, які мають різні характеристики, групуються в інші кластери. Щоб визначити мітку кластера будь-яких даних, обчислюється відстань між центрами кожного кластеру. Є кілька способів, якими можна скористатися для обчислення відстані, наприклад Евклідова відстань, відстань Манхеттен та відстань Чебічі. Метод K-Means спрямований на мінімізацію суми квадратів відстаней між усіма

точками та центром кластера. Ця процедура складається з наступних кроків.

Крок 1: обираємо k із заданих n шаблонів як початкові центри кластерів. Призначаємо кожному шаблону, що залишився, до одного з k кластерів (шаблон призначається найближчому центру/кластеру).

Крок 2: обчислюємо центри кластерів на основі поточного призначення шаблонів.

Крок 3: відносимо кожен із n шаблонів до їх найближчого центру/кластеру.

Крок 4: якщо немає змін у призначенні шаблонів кластерам протягом двох послідовних ітерації, завершуємо процедуру, інакше повертаємось до кроку 2.

В дослідженні [5] автори взяли тестові дані з показниками споживання електроенергії одним підприємством на протязі 1 року і розділили на 4 частини (на кожну пору року). На цих даних і було проведено тренування та апробація результатів кластеризації.

Для кожного сценарію було виміряно сумарну квадратичну помилку (SSE) та кількість ітерацій. SSE описує значення стандартного відхилення кожного кластера до центру обробки даних. Більше значення SSE означає, що ступінь подібності даних в одному кластері нижче. Кількість ітерацій описує довжину кластерів на протязі процесу формування. Результати дослідження наведено в таблиці 2.1.

Таблиця 2.1 – Результати кластеризації

№	Сценарій	SSE	Кількість ітерацій
1	4 кластери без усунення аномалій	0.174	20
2	4 кластери з усуненням аномалій	0.752	14
3	5 кластерів без усунення аномалій	0.134	21
4	5 кластерів з усуненням аномалій	0.509	22

2.2.3 Керований алгоритм Support Vector Machine

Support vector machine (SVM) є керованим алгоритмом машинного навчання, який часто використовують для лінійної та нелінійної класифікації. SVM використовує гіперплощини в багатовимірному просторі, щоб розділити точки даних на класи. Інакше кажучи, він знаходить оптимальну гіперплощину, яка максимізує відстань між класами. Межа – це відстань між гіперплощиною та найближчими точками даних з кожного класу, які називаються опорними векторами. SVM також може обробляти декілька класів, використовуючи стратегії "один проти одного" або "один проти всіх", де вона створює декілька бінарних класифікаторів і об'єднує їхні результати. SVM також може виконувати регресію, використовуючи іншу функцію втрат та виводячи неперервне значення замість дискретної мітки.

SVM зазвичай застосовується, коли в проблему залучено більше ніж один клас. Однак у виявленні аномалій він також використовується для проблем одного класу. Модель натренована визначати "норму" і може зрозуміти, чи належать незнайомі дані до цього класу, чи являють собою аномалію. Багато факторів сприяли високій популярності SVM сьогодні. Наприклад, рішення розріджене, що робить його дійсно ефективним у порівнянні з іншими підходами на основі ядра. Також він може використовувати нелінійне перетворення в формі ядра, яке навіть дозволяє розглядати SVM як техніку зменшення розмірності [6]. Однокласовий SVM розроблений для випадків, коли відомий лише один клас і треба виявляти будь-що поза цим класом. Це відомо як визначення новизни та відноситься до автоматичної ідентифікації непередбачених або аномальних явищ, тобто викиди, вбудовані у велику кількість нормальних даних. На відміну від традиційного SVM, One-Class SVM (OC-SVM) дозволяє визначити межу, яка досягає максимальне розділення між зразками "відомий клас" та "походження". Лише невелика частина точок даних може лежати з іншого боку межі прийняття рішення: ці точки вважаються викидами (рисунок 2.3).

Існує ще одна концепція роботи OC-SVM, яка використовує не опорний вектор, а так звану "гіперсферу" класу. Ідея полягає в тому, щоб мінімізувати гіперсферу одного класу прикладів у навчальних даних та вважати всі інші приклади за межами гіперсфери викидами або такими, що не відповідають розподілу навчальних даних.

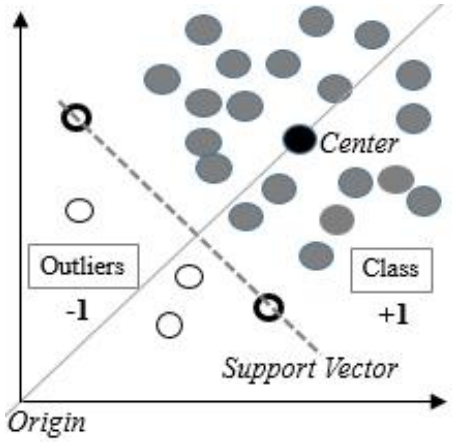


Рисунок 2.3 – Класифікація точок відносно опорного вектора класу

Однокласова класифікація (OCC) на основі SVM ґрунтується на визначенні найменшої гіперсфери (з радіусом r та центром c), що складається з усіх точок даних. Цей метод називається опорно-векторним описом даних (SVDD).

Математичний вираз для обчислення гіперсфери з центром c та радіусом r має вигляд (вираз 2.2):

$$\min_{r,c} r^2 \text{ subjects to, } \|\Phi(x_i) - c\|^2 \le r^2 + \xi_i \forall_i = 1,2,...,n. \tag{2.2}$$

Вираз 2.2 намагається мінімізувати радіус гіперсфери. Однак, наведене формулювання є дуже обмежувальним для викидів, тому більш гнучке формулювання, яке допускає викиди до певної міри, має вигляд (вираз 2.3):

$$\begin{gathered} \min_{r,c} r^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i, \\ \text{subjects to, } \|\Phi(x_i) - c\|^2 \le r^2 + \xi_i \forall_i = 1,2,...,n. \end{gathered} \tag{2.3}$$

Тут функція $\Phi(x_i)$ є перетворенням гіперсфери x відліків. На рисунку 2.4 показано, як формула утворює гіперсферу з радіусом r та центром c. Об'єкти на межах гіперсфери є опорними векторами, а два об'єкти, які лежать за межами мають відхилення більше 0. Отже вони вважаються викидами.

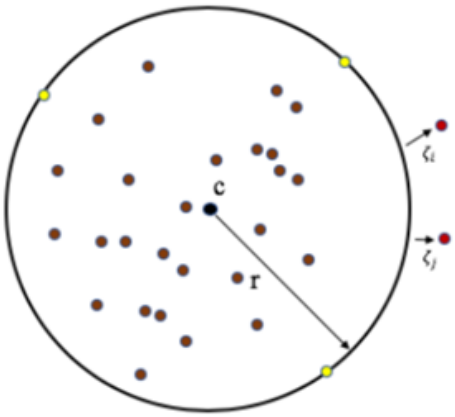


Рисунок 2.4 – Гіперсфера, сформована OC-SVM для опанування здатності класифікувати дані поза розподілом навчання на основі гіперсфери

2.2.4 Неконтрольований метод Local Outlier Factor

Алгоритм LOF – це неконтрольований метод виявлення викидів, який оцінює унікальність кожної події на основі відстані від k-найближчих сусідів [7]. Алгоритм LOF здатний виявляти викиди незалежно від розподілу даних, оскільки він робить певні припущення щодо їх розподілу. Основна ідея полягає в тому, що щільність навколо стороннього об'єкта суттєво відрізняється від щільності навколо своїх сусідів. Це перевага, коли дані, які аналізуються, не позначені або неможливо позначити через великий обсяг даних. Це досить поширено в комп'ютерних мережах, де ряд генерованих мережних пакетів дуже високий.

На рисунку 2.5 наведено схему обробки даних та виявлення аномалій. Перший крок складається з навчання NSL-KDD та розділення наборів даних на звичайні та атакуючі. Тільки для навчання використовуються записи, що відповідають звичайним даним.

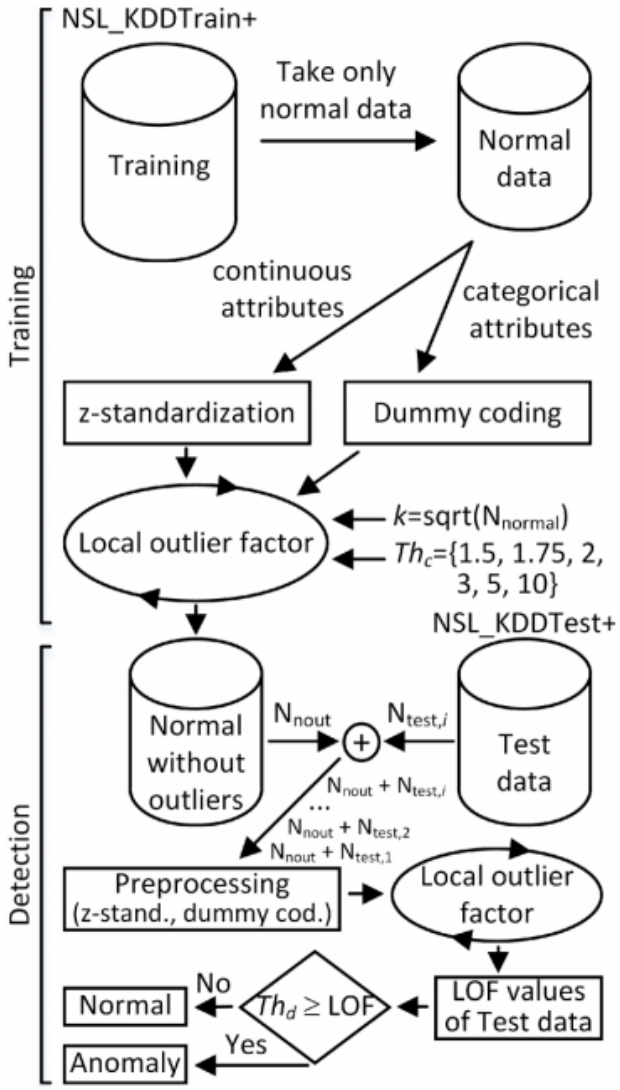


Рисунок 2.5 – Обробка даних та виявлення аномалій в LOF

Далі, необхідно видалити атрибути 8 та 20 зі звичайного набору даних, оскільки всі значення цих атрибутів дорівнюють 0, тобто вони не мають передбачуваної сили. Наступний крок – це переведення даних на стандартизовану Z шкалу, де середнє значення буде дорівнювати 0, а стандартне відхилення 1. Це необхідно задля визначення допустимих значень та відхилень (аномалій). Даний процес застосовується до всіх числових значень і номінальних атрибутів (2 – тип протоколу, 3 – сервіс, 4 – прапор). Після обробки буде отримано 75 атрибутів. Перед застосуванням алгоритму LOF необхідно вказати два параметри: кількість найближчих сусідів k та граничне значення, щоб виявляти чи запис виходить за межі чи ні. При виборі кількість найближчих сусідів, рекомендується щоб k-значення дорівнюватиме

квадратному кореню з усіх даних, які використано для модельного навчання. За алгоритмом LOF записи, які мають граничне значення відхилення більше 1, вважаються викидами. Для нормальної обробки даних існує шість порогових значень: Thc (cleaning) = {1,5; 1,75; 2; 3; 5 та 10}.

Алгоритм LOF застосовується з обраною кількістю найближчих сусідів k та граничним значенням. На основі розрахованих значень, записи зі значеннями граничного відхилення, які вище або дорівнюють Thc видаляються. Тоді k-значення для нормального типу даних, з якого були викиди видаляються, перераховуються, і знову виконується алгоритм LOF. Цикл повторюється до тих пір, поки не залишиться жодних записів, які будуть перевищювати встановлене граничне значення. Після навчання, отриманий набір даних далі використовується для виявлення аномалій.

Після фази навчання було підготовлено шість наборів даних, які складаються з даних нормального типу з видаленими аномаліями. Кількість викидів, знайдених та видалених за допомогою обраних порогових значень наведено на рисунку 2.6 (числа в дужках – це відсотки записів, які було видалено з набору навчальних даних).

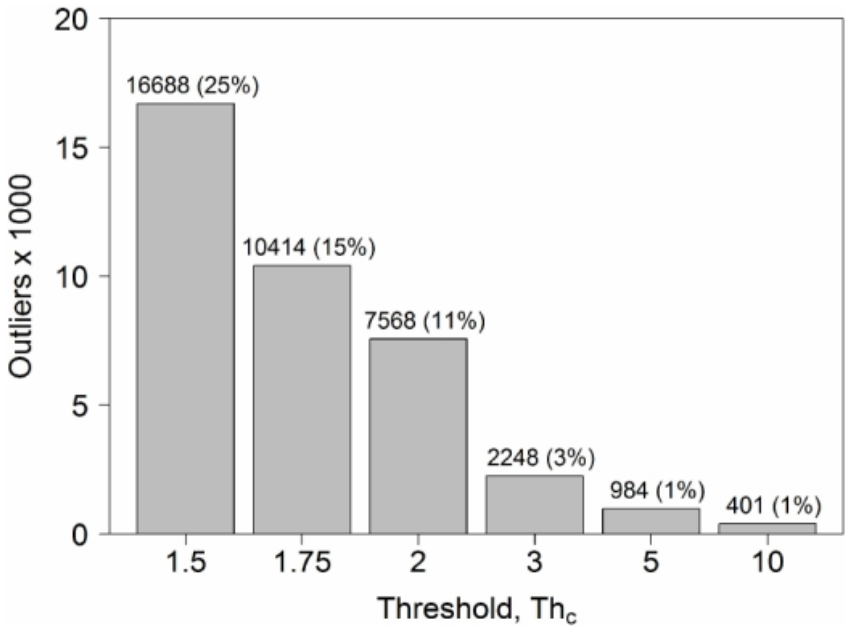


Рисунок 2.6 – Кількість аномалій, які було видалено з нормальних даних використовуючи значення Thc

З рисунку 2.6 видно, що перші три набори даних із пороговими значеннями 1.5, 1.75 та 2.0 повинні мати найбільший вплив на результати виявлення, оскільки ці набори даних зменшуються на 25, 15 та 11 відсотків відповідно.

2.2.5 Метод кластирезації Unsupervised Niche Clustering

Unsupervised Niche Clustering (UNC) – це надійний метод кластеризації, який використовує еволюційний алгоритм зі стратегією заняття ніш [8]. Даний алгоритм допомагає знаходити кластери за допомогою стійкої функції пристосованості, в той час як техніка заняття ніш дозволяє створювати та підтримувати ніші (кластери-кандидати). Оскільки UNC базується на генетичній оптимізації, він набагато менш сприйнятливий до неоптимальних рішень, ніж традиційні методи. Основною перевагою алгоритму є здатність обробляти шум та автоматично визначати кількість кластерів. Метод UNC особливо корисний при великому обсязі даних, де треба визначити приховані шаблони або структури без попередньої розмітки даних.

Автори статті [9] поєднали UNC з теорією нечітких множин для виявлення аномалій та застосували його для ідентифікації мережних вторгнень. Вони пов'язані з кожним кластером, згенерованим за допомогою UNC. Це функції-члени, які відповідають гаусовій формі, використовуючи еволюційний центр та радіус кластера. Такі функції приналежності кластерів визначатимуть рівень нормалізації вибірки даних.

В UNC проблема кластеризації вирішена шляхом зміни мети від пошуку простору розв'язків для C кластерів до пошуку цього простору для будь-якого одного кластера. Для цього UNC знаходить та підтримує щільні області (кластери) у просторі розв'язків використовуючи еволюційний алгоритм (EA) та техніку нішування. Як і в природі, ніші в даному контексті відповідають різним підпросторам середовища (кластерам), які можуть підтримувати різні типи (зразки даних).

Після створення кластерів використовується чітка характеристика для того, щоб визначити кластер, до якого буде віднесено зразок. Нова вибірка x віднесена до кластеру c_i, якщо x потрапляє у межі кластера c_i та відстань від x до центру c_i є мінімальною відстанню серед між усіма кластерами, до яких належить вибірка. Точка даних потрапляє у кластер c_i, якщо відстань від точки даних до центру кластера менша або дорівнює радіусу кластера. Оскільки основним припущенням UNC є те, що кожен кластер слідує гаусівському розподілу відстаней до центру кластера, можна охарактеризувати кожен прогнозований кластер за допомогою нечіткої функції належності (вираз 2.4), яка має Гауссову форму.

$$\mu_i(x) = \exp\left(-\frac{d(x_i c_i)^2}{2\sigma_i^2}\right). \qquad (2.4)$$

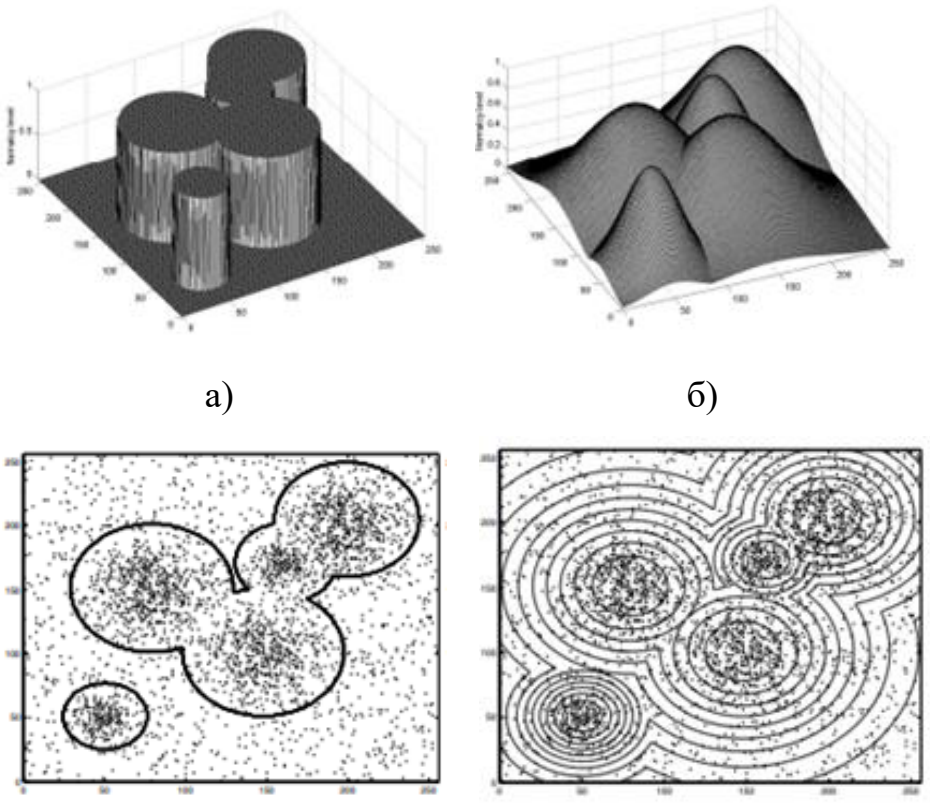
Як згадувалося раніше (вираз 2.4), радіус кластера c_i задається величиною Kσ2i, де K можна визначити як X2n,0.995 за даними розмірності множин (n).

Отже, вибірка x належить до кожного кластера з певним ступенем приналежності. Для визначення остаточного кластера, до якого належить вибірка, можна використовувати кілька методів дефаззифікації, наприклад, MAX-дефаззифікацію. При MAX-дефаззифікації зразок класифікується в кластері з найвищим значенням приналежності. Для задачі виявлення аномалій важливо визначити, чи належить вибірка даних до якогось кластера чи ні (нормальна вона чи аномальна), а не важливо зрозуміти, до якого саме кластера була віднесена вибірка даних. Тому нормальний клас визначається об'єднаною множиною всіх кластерів, згенерованих алгоритмом кластеризації. Нормальний клас розраховується як нечітка множина за допомогою нечіткого оператора max-OR (вираз 2.5). Вибірка x вважається нормальною зі ступенем μnormal(x).

$$\mu_{normal}(x) = \max\left\{\mu_i(x) \middle| \forall_i = 1,2,...,c\right\}. \qquad (2.5)$$

Оскільки генерується нечітка характеристика нормального класу, будемо використовувати механізм порогового значення для отримання остаточної чіткої характеристики. Якщо нечітке значення перевищує певний поріг θ, x вважається нормальним, інакше – ненормальним. Таким чином, θ – це відповідний поріг, який приймає значення в інтервалі [0, 1], що може бути обраний відповідно до необхідної точності.

На рисунку 2.7 порівнюються чіткі та невиразні характеристики кластерів, згенерованих за допомогою UNC для синтетичного набору даних.



а) б)

в) г)

Рисунок 2.7 – Функції належності для типового прогону UNC+MDE для набору даних з 35% шумового забруднення: а) чіткі, б) невиразні, в) чіткі контури, г) неясні контури

2.2.6 Методи на основі авторегресії

Виявлення аномалій на основі регресійних моделей є підкатегорією параметричних методів [10], що включає низку методів, які широко

застосовуються до даних часових рядів. Ці методи базуються на двоетапному підході. Спочатку на навчальних даних будується регресійна модель. Наявність аномалій у навчальних даних може вплинути на параметри регресії, а отже, регресійна модель може дати неточні результати. Популярний метод для обробки таких аномалій при підборі регресійних моделей називається стійкою регресією. Методи стійкої регресії можуть виявляти аномалії, оскільки аномалії, як правило, мають більші залишки від стійкої апроксимації. Подібний надійний підхід до виявлення аномалій був застосований в авторегресійних моделях інтегрованого ковзного середнього (ARIMA).

Потім модель, що була отримана, використовується на тестових послідовностях для обчислення залишків, наприклад, різниці між прогнозованим та реальним значенням. На основі залишків остаточно визначаються оцінки аномалій. До цієї категорії можна віднести методи виявлення аномалій, які засновані на традиційних моделях прогнозування часових рядів, таких як векторна авторегресія (VAR) [11] та авторегресійне інтегроване ковзне середнє (ARIMA).

Векторна авторегресія (VAR) – це модель динаміки кількох часових рядів, у якій поточні значення цих рядів залежать від минулих значень цих самих часових рядів. Модель запропонована Крістофером Сімсом як альтернатива системам одночасних рівнянь, які припускають суттєві теоретичні обмеження. VAR-моделі вільні від обмежень структурних моделей, проте проблема VAR-моделей полягає в різкому зростанні кількості параметрів зі збільшенням кількості аналізованих часових рядів і кількості лагів (запізнілих значеннь).

Як і в авторегресійній моделі, кожна змінна має рівняння, що моделює її еволюцію в часі. Це рівняння включає минулі значення змінної, запізнілі значення інших змінних у моделі та член помилки. VAR-моделі не вимагають стільки знань про фактори, що впливають на змінну, як структурні моделі з одночасними рівняннями. Єдине необхідне попереднє

знання – це перелік змінних, які, за гіпотезою, можуть впливати одна на одну в часі [12]. VAR-модель описує еволюцію набору k змінних, які називаються ендогенними змінними, у часі. Кожен період часу пронумеровано, t = 1, ..., T. Змінні зібрано у вектор y_t довжиною k. Вектор моделюється як лінійна функція від попереднього значення. Компоненти вектора позначаються як $y_{i,t}$, що означає спостереження в момент часу t i-ї змінної.

VAR моделі характеризуються порядком, який вказує на кількість попередніх часових періодів, що використовуються в моделі. Лаг – це значення змінної в попередньому періоді часу. Отже, в загальному випадку VAR p-го порядку відноситься до VAR моделі, яка включає лаги для останніх p періодів часу. VAR p-го порядку позначається як "VAR(p)", а іноді називається "VAR з p лагами". VAR модель p-го порядку записується як (вираз 2.6):

$$y_i = c + A_1 y_{t-1} + A_2 y_{t-2} + ... + A_p y_{t-p} + e_t, \quad (2.6)$$

де y_{t-1} – змінні, які вказують на те, що значення змінної на і періодів раніше та називаються "і-м лагом" y_t;

c – k-вектор констант, що слугує перехопленням моделі;

A_i – інваріантна в часі (k×k)-матриця;

e_t – k-вектор членів помилки.

Члени помилки повинні задовольняти трьом умовам:

- $E(e_t) = 0$: кожен член помилки має середнє значення, рівне нулю;

- $E(e_t e_t^{/}) = \Omega$: сучасна коваріаційна матриця членів помилок – це k×k додатньо-напіввизначена матриця, що позначається Ω;

- $E(e_t e_{t-k}^{/}) = 0$ для кожного ненульового k: тут немає кореляції в часі. Зокрема, немає послідовної кореляції в термінах індивідуальних помилок.

Процес вибору максимального лагу p у VAR моделі потребує особливої уваги, оскільки від правильності обраного порядку лагу залежить результат.

Модель авторегресії ковзного середнього (ARIMA) та систематичний підхід до її побудови був запропонований у 1970-х роках Джорджем Боксом і Гвілімом Дженкінсом. Вона призначена для аналізу стаціонарних часових рядів на основі оцінки лінійної залежності прогнозованих значень від історичних. Часовим рядом можуть бути будь-які дані в розрізі часу, наприклад, продажі товарів, кількість замовлень, потік клієнтів тощо.

Для використання моделі часовий ряд має бути стаціонарним, тобто його середнє значення та дисперсія мають бути постійними.

Модель Бокса-Дженкінса широко застосовують під час прогнозування часових рядів. Основне завдання при цьому полягає в оцінці параметрів моделі. Методологія побудови ARIMA-моделі часового ряду, що досліджується, включає такі основні етапи:

- побудова пробної моделі;
- оцінювання параметрів моделі та перевірка адекватності моделі;
- використання моделі для прогнозування.

Також вона передбачає, що часовий ряд містить три складові: авторегресійну, інтегровану та ковзаючу середню, які в моделі позначені як p, d, та q відповідно:

- величина p називається порядком авторегресії, вона дає змогу відповісти на запитання, чи буде черговий елемент ряду близьким до значення X, якщо до нього були близькі p попередніх значень;
- величину d називають порядком інтегрування, вона показує, наскільки елемент ряду близький за значенням до d попередніх значень, якщо різниця між ними мінімальна;
- параметр q – це порядок ковзного середнього, який дозволяє встановити похибку моделі як лінійну комбінацію значень помилок, що спостерігалися раніше.

Авторегресія – це складова моделі часового ряду, в якій його прогнозоване значення може бути виражене у вигляді лінійної комбінації історичних значень цього ж ряду і випадкової помилки.

Зазвичай модель згадується як ARIMA(p, d, q), де p, d та q – цілі невід'ємні числа, що характеризують порядок для частин моделі (відповідно авторегресійної, інтегрованої та ковзної середньої). Для часового ряду X(t) модель може бути записана так (вираз 2.7):

$$(\Delta^d x_t) = \sum_{t=1}^{p} a_t (\Delta^d x_{t-1}) + \varepsilon_t + \sum_{j=1}^{q} b_j (\Delta^d x_{t-j}), \quad (2.7)$$

де: Δ^d – оператор різниці порядку d (послідовне взяття d разів різниць першого порядку – спочатку від самого ряду, потім від отриманих різниць першого порядку, потім від другого порядку та ін.);

a_t – коефіцієнти авторегресійної частини моделі;

ε_t – значення помилки (вважаються незалежними однаково розподіленими випадковими величинами з нормального розподілу з нульовим середнім);

b_j – коефіцієнти ковзної середньої.

Існує також розширена модель ARIMAX, яка враховує зовнішні чинники під час побудови прогнозу. Це математична модель для аналізу часових рядів, що об'єднує в собі інтегровану авторегресію, ковзну середню і можливість врахування додаткових зовнішніх факторів.

3 ВИБІР ІНСТРУМЕНТІВ ДЛЯ МОДЕЛІ ВИЯВЛЕННЯ АНОМАЛІЙ В ДАНИХ ЕНЕРГОСПОЖИВАННЯ

3.1 Вибір технологій та методів

Неможливо уявити існування людства без електроенергії та її використання. Вона пронизує буквально всі сфери нашого існування, від побуту до виробництва та транспорту. Без електроенергії багато сфер життя перестали б функціонувати ефективно. Загалом, електроенергія стала тим запальним двигуном, який тримає сучасне суспільство в русі, допомагаючи нам забезпечити комфорт, продуктивність та інновації.

Але постає питання щодо регулювання споживання електроенергії людьми. Це може бути представлено наступним чином:

- аналіз регулярності та шаблонів у споживанні енергії;
- виявлення непередбачуваних змін у тенденціях споживання енергії;
- виявлення незвичайних змін у поведінці споживачів;
- ідентифікації надмірного споживання енергії, що може бути наслідком витоку енергії або інших проблем;
- реагування на незвичайні події, такі як аварії або крадіжки енергії.

Споживання електроенергії може бути джерелом важливої інформації для виявлення аномалій у системі електропостачання. Але для аналізу та обробки цих даних недостатня лише присутність людини – цей процес можна автоматизувати. Тут з’являється необхідність у виборі відповідних інструментів для цього. Вибір інструментів для виявлення аномалій в показниках про енергоспоживання є складною задачею, яка вимагає не тільки знань в електроенергетиці через специфіку цих даних, а ще і можливості технологій. Невірний вибір може призвести до викривлення фактичних даних або некоректної інтерпретації результатів. Це в свою чергу буде причиною неефективного споживання чи непомітних крадіжок електроенергії, або навіть до виводу електрообладнання з ладу через перепади у напрузі.

3.1.1 Мова програмування C#

Мова C# – це мова програмування, що поєднує об'єктно-орієнтовані та контекстно-орієнтовані концепції. Мова була розроблена корпорацією Microsoft в кінці 90-х років як частина загальної стратегії .NET. Вперше вона була випущена у вигляді альфа-версії в середині 2000 року. Головним розробником C# був Андерс Хейльсберга – один з провідних у світі фахівців з мов програмування, який може похвалитися рядом помітних досягнень в цій галузі. Поточною версією мови є версія C# 12.0, яка вийшла у листопаді 2023 року у складі .NET 8.0.

Мова C# безпосередньо пов'язана з C, C++ та Java. І це не випадково. Адже це три найбільш широко поширених і визнаних у всьому світі мови програмування. Крім того, на момент створення C# практично всі професійні програмісти вже володіли C, C++ або Java. Завдяки тому що C# побудований на настільки міцній і зрозумілій підставі, перейти на цю мову з C, C++ або Java не становило особливих труднощів.

C# можна вважати нащадком мови C, від якої було успадковано синтаксис, багато ключових слів та операторів. Крім того, C# побудований на вдосконаленій об'єктній моделі, яка визначена в C++.

У C# є чимало нових засобів, але найважливіше з них пов'язане з вбудованою підтримкою програмних компонентів. Насправді C# може вважатися компонентно-орієнтованою мовою програмування, оскільки в неї впроваджена інтегрована підтримка написання програмних компонентів. Наприклад, до складу C# входять засоби прямої підтримки таких складових частин програмних компонентів, як властивості, методи та події. Але найважливішою компонентно-орієнтованої особливістю цієї мови, ймовірно, є можливість роботи в безпечному середовищі багатомовного програмування. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, покажчики на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML.

C# розроблявся як мова програмування прикладного рівня для CLR і залежить від можливостей самої CLR. Це стосується, перш за все, системи типів C#, яка відображає FCL. Присутність або відсутність тих чи інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована в відповідні конструкції CLR.

CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені "класичні" мови програмування. Наприклад, прибирання сміття не реалізоване в самому C#, а проводиться CLR для програм, написаних на C# так само як це робиться для програм на VB.NET, F# та ін.

Мова програмування C# переважно використовується для створення корпоративного програмного забезпечення, фінансових проєктів, наприклад для банків і бірж, зокрема мобільних додатків, хмарних сервісів. Крім того, є ще багато галузей де можна застосовувати C#, наприклад такі, як: blockchain, розширення для інших мов програмування (використовуваних як прошарок між бібліотекою C # і мовою, можливості якої під конкретні цілі планується розширювати), розробка комп'ютерних ігор (за допомогою Unity), embedded system, IoT, machine learning (використовуючи ML.NET) та ін. Також цю мову програмування можна застосовувати і в науці (проведення складних експериментальних розрахунків, криптографія, розпізнавання образів тощо).

3.1.2 Бібліотека ML.NET

ML.NET – це безкоштовна бібліотека машинного навчання для мов програмування C# та F#. Вона також підтримує моделі Python при використанні разом з NimbusML. Попередній випуск ML.NET містив рішення для конструювання ознак (наприклад, створення N-грам), двійкової та мультикласової класифікацій, регресійного аналізу. Згодом були додані додаткові завдання ML, такі як виявлення аномалій та системи рекомендацій, а інші підходи, такі як глибоке навчання, будуть включені до майбутніх версій даної бібліотеки.

ML.NET надає .NET розробникам можливості аналізу та прогнозування за допомогою машинного навчання на основі моделей. Фреймворк побудований на базі .NET Core та .NET Standard, успадкувавши можливість кросплатформної роботи на Linux, Windows та macOS. Хоча фреймворк ML.NET є новим, його історія почалася у 2002 році як дослідницький проєкт Microsoft під назвою TMSN (Text mining search and navigation) для використання всередині продуктів Microsoft. Пізніше, приблизно у 2011 році, він був перейменований на TLC (The learning code). ML.NET був створений на основі бібліотеки TLC і в значній мірі перевершив своїх попередників.

Розробники можуть самі навчати модель машинного навчання або повторно використовувати існуючу модель, створену третьою стороною, і запускати її в будь-якому середовищі в автономному режимі (без підключення до мережі Інтернет). Це означає, що розробникам не потрібно мати досвід роботи з наукою про дані, щоб використовувати фреймворк. Починаючи з версії 0.3 в ML.NET з'явилася підтримка формату моделей глибокого навчання Open Neural Network Exchange (ONNX) з відкритим вихідним кодом. Цей реліз включає інші помітні покращення, такі як машини факторизації, LightGBM, ансамбливе навчання, перетворення LightLDA та класифікація OVA. Інтеграція ML.NET з TensorFlow можлива починаючи з версії 0.5. У збірці 0.7 додано підтримку x86 та x64 додатків, включаючи розширені можливості рекомендацій за допомогою матричної факторизації.

Перший стабільний реліз 1.0 фреймворку був анонсований на конференції Build у 2019 році. Він включав додавання інструменту для побудови моделей (Model builder) та можливостей автоматизованого машинного навчання (AutoML). У Build 1.3.1 було представлено попередній перегляд навчання глибоких нейронних мереж з використанням прив'язок даних C# для Tensorflow та Database Loader, який дозволяє навчати моделі на базах даних. Наступна версія 1.4.0 ML.NET дозволила проводити оцінку моделей на процесорах ARM та навчання глибоких нейронних мереж на графічних процесорах для Windows та Linux.

ML.NET дозволяє користувачам експортувати навчені моделі у формат Open Neural Network Exchange (ONNX). Це дає можливість використовувати моделі в різних середовищах, які не підтримують ML.NET. З'явилась змога запускати такі моделі в клієнтській частині браузера за допомогою ONNX.js, клієнтського фреймворку на JavaScript для моделей глибокого навчання, створених у форматі Onnx.

Фреймворк ML.NET складається з:

- компонентів для завантаження даних;
- задач машинного навчання;
- використання та оцінки моделей;
- інструментів та розширень.

Детальну структуру фреймворка ML.NET наведено на рисунку 3.1.

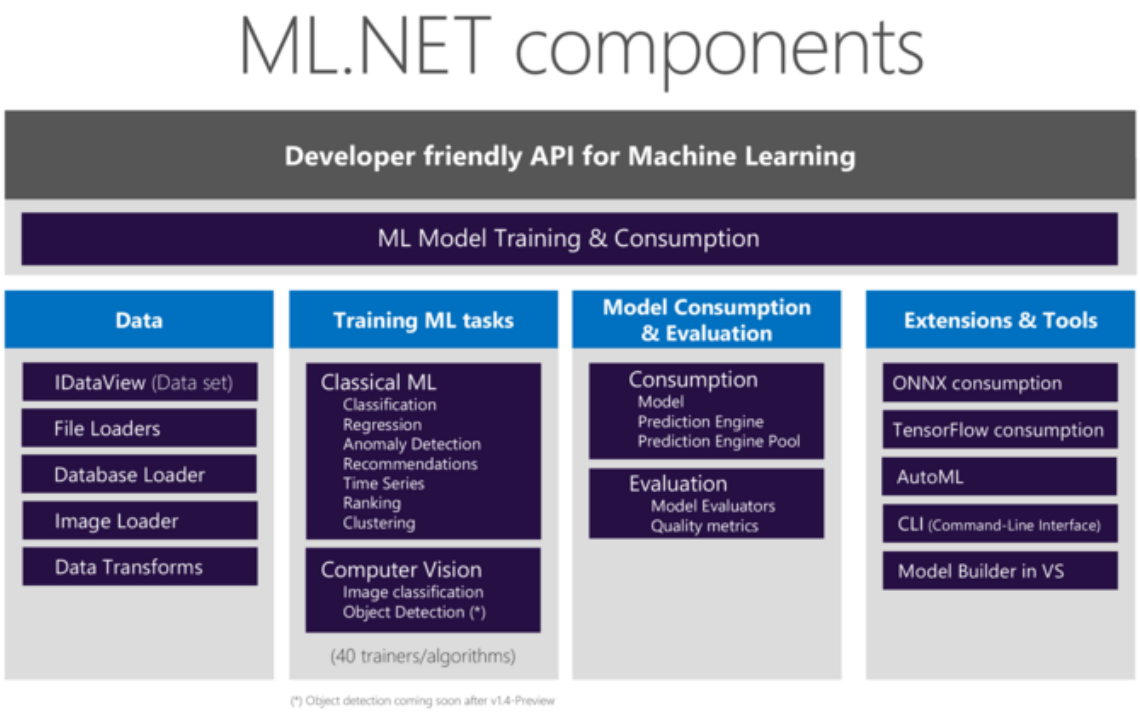


Рисунок 3.1 – Компоненти фреймворку ML.NET

3.1.3 Інтерфейс командної строки ML.NET CLI

ML.NET CLI – це інтерфейс командної строки, який використовує ML.NET AutoML для навчання моделі та вибору найкращого алгоритму для даних. ML.NET Model Builder Preview – це розширення для Visual Studio, яке

використовує ML.NET CLI та AutoML для виведення найкращої моделі ML.NET за допомогою графічного інтерфейсу.

Після встановлення фреймворку користувач обирає задачу машинного навчання та завантажує навчальний набір даних, і він генерує модель ML.NET, а також код C# для запуску, щоб використовувати модель у додатку. Поточна версія ML.NET CLI підтримує наступні задачі: виявлення аномалій, класифікація, регресія, рекомендація, класифікація зображень, класифікація тексту, прогнозування тощо.

Як показано на рисунку 3.2, дуже просто згенерувати високоякісну модель ML.NET (серіалізовану модель у вигляді .zip-файлу), а також зразок коду на C# для запуску/оцінки цієї моделі. Крім того, генерується також код C# для створення/навчання цієї моделі, щоб користувач міг досліджувати і повторювати алгоритм та налаштування, які використовуються для згенерованої "найкращої моделі".

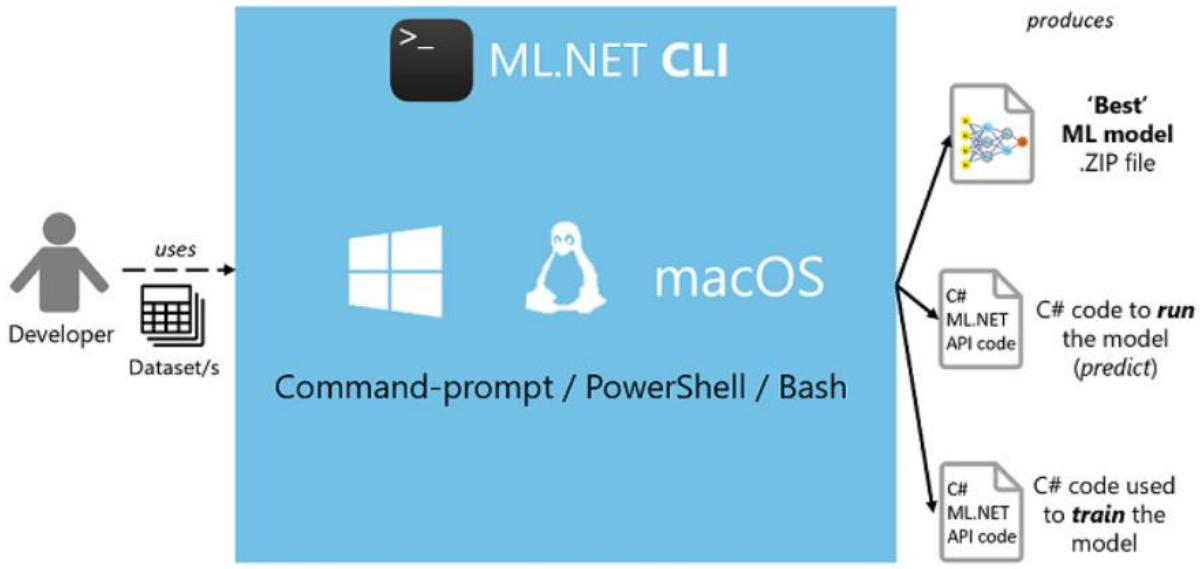


Рисунок 3.2 – Принцип роботи ML.NET CLI

Варто зазначити, що взаємодія користувача з інтерфейсом командної строки ML.NET проходить у декілька етапів. Спочатку користувач обирає задачу машинного навчання, для рішення якої треба згенерувати та натренувати модель. Для цього треба ввести відповідну команду у консолі.

Після цього консоль ML.NET CLI виведе список обов'язкових та опціональних параметрів задля подальшої роботи, такі як: шлях до навчального набору даних; максимальний час тренування моделі; шлях до директорії в яку потрібно зберегти результати; перелік стовпців в наборі даних які треба ігнорувати тощо.

На рисунку 3.3 можна побачити процес тренування моделей різними алгоритмами машинного навчання для вирішення задачі мультикласової класифікації. В даному прикладі отримані результаті можна оцінити за такими метриками як MacroAccuracy та Duration.

```
Start Training
start multiclass classification
Evaluate Metric: MacroAccuracy
Available Trainers: LGBM,FASTFOREST,FASTTREE,LBFGS,SDCA
Training time in seconds: 10
|      Trainer                              MacroAccuracy Duration    |
|--------------------------------------------------------------------|
|0     FastTreeOva                          0.5751     0.6000        |
|1     FastTreeOva                          0.5854     0.2860        |
|2     FastTreeOva                          0.6822     0.4950        |
|3     FastForestOva                        0.6059     0.3550        |
|4     FastTreeOva                          0.5826     0.5150        |
|5     LightGbmMulti                        0.6465     0.1350        |
|6     LightGbmMulti                        0.6483     0.1390        |
|7     FastTreeOva                          0.6737     1.1230        |
|8     FastTreeOva                          0.6380     0.4040        |
|9     LightGbmMulti                        0.7010     0.1160        |
[Source=AutoMLExperiment, Kind=Info] cancel training because cancellation token is invoked...
|--------------------------------------------------------------------|
|                        Experiment Results                          |
|--------------------------------------------------------------------|
|                             Summary                                |
|--------------------------------------------------------------------|
|ML Task: multiclass classification                                  |
|Dataset: DataSets\credit_customers.csv|
|Label : class                                                       |
|Total experiment time :     9.0000 Secs                             |
|Total number of models explored: 11                                 |
|--------------------------------------------------------------------|
|                      Top 5 models explored                         |
|--------------------------------------------------------------------|
|      Trainer                              MacroAccuracy Duration    |
|--------------------------------------------------------------------|
|9     LightGbmMulti                        0.7010     0.1160        |
|2     FastTreeOva                          0.6822     0.4950        |
|7     FastTreeOva                          0.6737     1.1230        |
|6     LightGbmMulti                        0.6483     0.1390        |
|5     LightGbmMulti                        0.6465     0.1350        |
|--------------------------------------------------------------------|
[Source=AutoMLExperiment, Kind=Info] cancel training because cancellation token is invoked...
save SampleClassification.mbconfig to ML.NET_CLI\SampleClassification
Generating a console project for the best pipeline at location : ML.NET_CLI\SampleClassification
```

Рисунок 3.3 – Тренування моделі для мультикласової класифікації

3.1.4 Бібліотека побудови графіків OxyPlot

OxyPlot – це бібліотека для побудови графіків та діаграм у .NET з відкритим кодом на Github. Вона призначена для створення наукових, статистичних та технічних графіків, які можуть бути інтегровані в настільні та мобільні додатки. Завдяки своїй кросплатформності OxyPlot підтримує .NET Framework, .NET Core, Xamarin.iOS, Xamarin.Android та ін.

OxyPlot була створена у 2010 році як простий компонент для побудови графіків у WPF, зосереджений на простоті, продуктивності та візуальному вигляді. Стиль графіків натхненний книгами Едварда Тафта та Стівена Фью (їх принципи були важливими при розробці цієї бібліотеки).

OxyPlot в першу чергу орієнтована на двовимірні системи координат, саме тому в назві є «xy». Також є підтримка декартової та полярної систем координат. Щодо кругових діаграм, то їх те ж можна будувати у OxyPlot. Графіки можна експортувати в наступні растрові та векторні формати файлів: .png, .svg, .pdf. Щодо переваг OxyPlot, то вони наступні:

- легкість у використанні: простий API дозволяє легко створювати та налаштовувати графіки;
- гнучкість: можливість створення різноманітних типів графіків, таких як лінійні графіки, гістограми, кругові діаграми, графіки з помилками, 3D графіки тощо;
- висока продуктивність: оптимізована для швидкої обробки та відображення великих обсягів даних;
- інтерактивність: підтримує інтерактивні функції, такі як масштабування, панорамування та інші взаємодії з графіками;
- відкритий вихідний код, доступний на GitHub.

Але OxyPlot має певні обмеження. Елементи керування діаграмами не спостерігають за змінами у властивостях та колекціях. Користувач повинен вручну оновити діаграми при зміні даних. Також не підтримуються анімація, градієнтні та штрихові пензли.

3.2 Метрики порівняння та оцінки результатів виявлення аномалій

При оцінці моделі виявлення аномалій у даних про енергоспоживання можна використовувати різні метрики залежно від конкретного контексту та вимог задачі. Зазвичай метрики поділяються на об’єктивні та суб’єктивні. Об'єктивні метрики вимірюють продуктивність моделі на основі об'єктивних критеріїв та кількісних даних, тоді як суб'єктивні метрики оцінюють продуктивність моделі на основі суб'єктивної оцінки експерта або користувача.

Об'єктивні метрики:

- точність (Accuracy): метрика вимірює загальну точність моделі, тобто відношення кількості правильно класифікованих аномалій до загальної кількості аномалій;

- точність виявлення аномалій (Precision): метрика вимірює точність моделі у виявленні аномалій, тобто відношення кількості правильно виявлених аномалій до загальної кількості точок, які модель визначила як аномалії;

- повнота виявлення аномалій (Recall): метрика вимірює частку аномалій, які були виявлені моделлю, відносно загальної кількості реальних аномалій у даних;

- F1-показник (F1-score): гармонічне середнє точності та повноти, яке дозволяє оцінити збалансованість моделі між точністю та повнотою виявлення аномалій;

- Roc-крива (Receiver Operating Characteristic Curve) та площа під roc-кривою (Area under the roc curve, Auc-roc): метрики використовуються для оцінки роботи моделі при різних порогах відсічення та вимірюють здатність моделі відрізняти між аномаліями та нормальними даними;

- матриця помилок (Confusion matrix), яка показує кількість вірних та хибних класифікованих аномалій та нормальних даних, дозволяє отримати детальну інформацію про продуктивність моделі;

- середній час виявлення аномалії, який потрібен моделі для виявлення аномалії після її виникнення.

Суб'єктивні метрики:

- легкість в розумінні результатів: оцінка наскільки легко користувачам розуміти та інтерпретувати результати виявлення аномалій;

- придатність до використання: оцінка того, наскільки ефективно модель вирішує конкретні потреби та задачі користувача;

- надійність: оцінка стабільності та надійності моделі під час різних умов експлуатації;

- відповідність до контексту: оцінка того, наскільки добре модель враховує контекст та особливості енергетичних систем користувача.

Ці метрики можуть бути використані для оцінки ефективності та придатності моделі виявлення аномалій у даних про енергоспоживання з різних точок зору: об'єктивного аналізу продуктивності та задоволення потреб користувача.

4 МОДЕЛЬ ВИЯВЛЕННЯ АНОМАЛІЙ У ДАНИХ ПРО СПОЖИВАННЯ ЕЛЕКТРОЕНЕРГІЇ CADM-ECD

4.1 Огляд існуючої моделі виявлення аномалій у даних про енергоспоживання

Модель – це абстрактне представлення реального явища або процесу [13]. Моделі використовуються для прогнозування, аналізу даних, розуміння поведінки системи тощо. Наприклад, лінійна регресія може бути моделлю для прогнозування ціни на нерухомість на основі площі будинку, його розташування тощо.

Модель машинного навчання – це алгоритм або математична конструкція, яка використовується для виявлення патернів у вхідних даних та прийняття виведених рішень на основі цих патернів. Вона є центральною складовою багатьох застосувань штучного інтелекту, включаючи розпізнавання образів, розпізнавання мови, рекомендації на основі даних, прогнозування та багато іншого. Модель машинного навчання навчається на вхідних даних шляхом корекції своїх параметрів з метою мінімізації помилки або максимізації заданого критерію продуктивності. Після навчання модель може бути використана для прогнозування або класифікації нових даних.

В розрізі виявлення аномалій у даних про енергоспоживання, термін модель визначається як математичне або алгоритмічне представлення нормальної поведінки системи енергоспоживання. Ця модель використовується для порівняння з реальними даними, щоб ідентифікувати аномальні або незвичайні зміни у споживанні енергії.

В розумінні виявлення аномалій модель може бути представлена як алгоритм, який автоматично виявляє відхилення від очікуваних нормальних патернів в споживанні енергії. Це може включати в себе використання різноманітних методів машинного навчання, статистичних аналізів часових рядів або комбінації різних підходів.

Для прикладу, варто розглянути узагальнений кортеж для існуючої моделі Anomaly Detection Model for Electrical Consumption Data (ADM-ECD) виявлення аномалій (вираз 4.1):

$$\text{ADM-ECD} = \{\text{ВДЕ, АМНВА, ВР}\}, \quad (4.1)$$

де ВДЕ – вхідні дані щодо енергоспоживання;

АМНВА – алгоритм машинного навчання для виявлення аномалій;

ВР – відображення результатів.

Наведений кортеж ілюструє узагальнений вигляд існуючої моделі. У прикладі наведено використання алгоритму виявлення аномалій, і це може бути будь-який алгоритм машинного навчання (ізоляційний ліс, кластеризація та кластерний аналіз, метод опорних векторів тощо).

Але варто зауважити, що використання лише одного методу або алгоритму для моделі може не дати очікуваного результату, на відміну від використання комбінації методів. Якщо ж об’єднати декілька методів виявлення аномалій в одну модель, то це може дати більш точні результати або ж принаймні збільшиться відсоток коректності роботи моделі.

4.2 Модель Composite Anomaly Detection Model for Electrical Consumption Data

Composite Anomaly Detection Model for Electrical Consumption Data (CADM-ECD) – це комбінована модель для виявлення аномалій в даних про споживання електроенергії. Вона розробляється задля більш ефективної обробки показників енергоспоживання та ідентифікації різких перепадів в напрузі.

На основі аналізу предметної області, а також після огляду існуючих методів виявлення аномалій в даних, можна запропонувати власну модель. Вона об’єднає у собі 2 раніше оглянуті способи виявлення аномалій: ARIMA та Isolation Forest. Головна ідея така: спочатку треба побудувати рекурентну

модель ARIMA для прогнозування часового ряду споживання електроенергії. Потім необхідно використати метод Isolation Forest для виявлення аномалій у залишках між фактичними даними та прогнозованими значеннями ARIMA. Саме такою і є модель CADM-ECD (вираз 4.2):

$$\text{CADM-ECD} = \{\text{ВДЕ, МВА, МГВ}\}, \quad (4.2)$$

де ВДЕ – вхідні дані щодо енергоспоживання;

МВА – модуль виявлення аномалій**;**

МГВ – модуль графічного відображення.

В наведеному кортежі варто детальніше оглянути модуль виявлення аномалій та його підмодулі (вираз 4.3). Тому що в цьому модулі відбувається основний процес роботи моделі, а саме аналіз вхідного набору даних енергоспоживання та виявлення аномалій в ньому:

$$\text{МВА} = \{\text{ППМАКС, ПРАІЛ }\}, \quad (4.3)$$

де ППМАКС – підмодуль побудови моделі авторегресії ковзного середнього (ARIMA);

ПРАІЛ – підмодуль з реалізацією алгоритму «Ізоляційний ліс».

Варто зазначити, що задля виявлення аномалій у вхідному наборі даних спочатку повинен спрацьовувати підмодуль ППМАКС, а вже потім ПРАІЛ. Тому що принцип виявлення аномалій в запропонованій моделі це порівняння спрогнозованого часового ряду ARIMA, отриманого в ППМАКС з фактичними даними. А потім підмодуль ПРАІЛ вирішує, які розбіжності в цьому порівняні можна вважати викидами (аномаліями), а які ні.

Щодо модуля графічного відображення (МГВ), то він необхідний для візуалізації даних та побудови графіків на їх основі. При чому це будуть не просто графік фактичних даних про споживання електроенергії, а ще і відображення прогнозованих значень ARIMA сформованих в ППМАКС. Ну і звичайно це візуальне або кольорове виділення точок на графіку, які вдалось

ідентифікувати як аномалії в результаті роботи ПРАІЛ.

Базово алгоритм роботи моделі CADM-ECD складається з таких кроків:

Крок 1: завантаження даних про споживання електроенергії.

Крок 2: розділення даних на навчальний та тестувальний набори.

Крок 3: побудова моделі ARIMA для прогнозування споживання електроенергії.

Крок 4: прогнозування значень споживання електроенергії за допомогою побудованої ARIMA моделі.

Крок 5: визначення залишків між фактичними та прогнозованими значеннями.

Крок 6: використання Isolation Forest для виявлення аномальних залишків.

Крок 7: виявлення аномальних точок на основі оцінки Isolation Forest.

Крок 8: відображення графіка з фактичними даними, прогнозом, залишками та виявленими аномаліями.

Таким чином, модель CADM-ECD є унікальним рішенням для ідентифікації аномалій в даних про енергоспоживання. Це зумовлено тим, що застосовується комплексний та інтегрований підхід, який в свою чергу може дати кращі результаті ніж використання одного методу виявлення аномалій при аналізі даних. Також можливість побудови графіків з виділенням аномальних точок є більш наочним, ніж дані у вигляді набору чисел. Тому через комбінований підхід кожен модуль моделі потребує окремого огляду.

4.2.1 Модуль виявлення аномалій

Як зазначалось вище, в модулі виявлення аномалій моделі CADM-ECD проходить основний процес, а саме аналіз вхідних даних про енергоспоживання та ідентифікація нетипових значень. Іншими словами, тут зосереджено вся логіка обробки даних та агрегації результатів. Можна сказати, що МВА є ядром моделі CADM-ECD. Результати, які були отримані після роботи МВА переходять далі в модуль МГВ для того, щоб побудувати

відповідні графіки фактичних і спрогнозованих значень та виявлених аномалій (різких перепадів напруги).

Задля більшого розуміння функціонування МВА варто детальніше розглянути його підмодулі.

4.2.1.1 Підмодуль побудови моделі авторегресії ковзного середнього ARIMA

Підмодуль побудови моделі авторегресії ковзного середнього (ППМАКС) це одна з двох складових МВА. Його призначення – це побудова регресійної моделі ARIMA на основі вхідних даних – показників напруги, які були отримані з електрообладнання або в результаті використання електроенергії споживачами за певний проміжок часу. Кінцева мета цього підмодуля це обчислення «залишків»: різниці між фактичними даними та спрогнозованими значеннями ARIMA.

Модель ARIMA використовується для прогнозування часових рядів і вона є потужним інструментом для аналізу та передбачення даних, що залежать від часу (в нашому випадку це дані про споживання електроенергії). В моделі ARIMA виконує роль прогнозування майбутніх значень споживання електроенергії на основі історичних даних. Це дозволяє мати базові очікування щодо майбутніх значень споживання електроенергії.

Відхилення фактичних даних від прогнозованих значень (залишки) є ключовими для виявлення аномалій. Значні відхилення можуть вказувати на аномальні події або поведінку. Визначення межи відхилення та чи можна вважати конкретний залишок аномалією передбачено у підмодулі ПРАІЛ.

Також ARIMA дозволяє враховувати складні часові залежності в даних, що робить його ефективним для аналізу та прогнозування часових рядів.

Процес побудови моделі ARIMA та її використання можна охарактеризувати наступним чином. Спочатку йде збір та підготовка даних про споживання електроенергії. Зазвичай це числовий ряд показників напруги. Далі йде обробка та форматування даних для подальшого аналізу

(наприклад, заповнення пропусків, нормалізація). Перед побудовою моделі ARIMA треба зробити розділення даних на дві частини: навчальний набір (наприклад, перші 80% даних) та тестовий набір (останні 20% даних). Але відсоткове співвідношення може відрізнятись.

Коли дані були успішно сформовані та розділені на навчальний і тестовий набори необхідно визначити параметри ARIMA(p, d, q), де:

- p (порядок автокореляції): кількість лагів, що використовуються в моделі;
- d (порядок інтеграції): кількість разів, що ряд потрібно диференціювати, щоб зробити його стаціонарним;
- q (порядок середнього ковзання): кількість лагів помилки прогнозу, що включаються в модель.

Після обчислень параметрів (важливо щоб це були цілі невід'ємні числа) йде сам процес навчання моделі ARIMA на відповідному наборі, який раніше був сформований. Наступний крок – це використання моделі, яка була навчена ARIMA, для прогнозування значень на тестовому наборі та отримання прогнозованих значень для заданого горизонту прогнозування (наприклад, на 12 місяців вперед). Це дозволяє мати базові очікування щодо майбутніх значень споживання електроенергії.

Останній етап готової моделі ARIMA – це обчислення залишків між фактичними значеннями споживання електроенергії та прогнозованими значеннями (залишки = фактичні значення – прогнозовані значення). Значні відхилення можуть вказувати на аномалії. Ці залишки є фінальним результатом роботи ППМАКС, які передаються далі до підмодуля ПРАІЛ.

На блок-схемі рисунку 4.1 проілюстровано вищеописаний алгоритм роботи підмодуля ППМАКС. Тут наведено усі етапи побудови моделі ARIMA та формування результатів роботи цього підмодуля (обчислення залишків). Блок-схема містить умовний блок перевірки на стаціонарність даних: якщо дані стаціонарні – слідуємо далі. Коли ж дані не стаціонарні то проводимо необхідні дії для цього і знову потрапляємо у цей умовний блок.

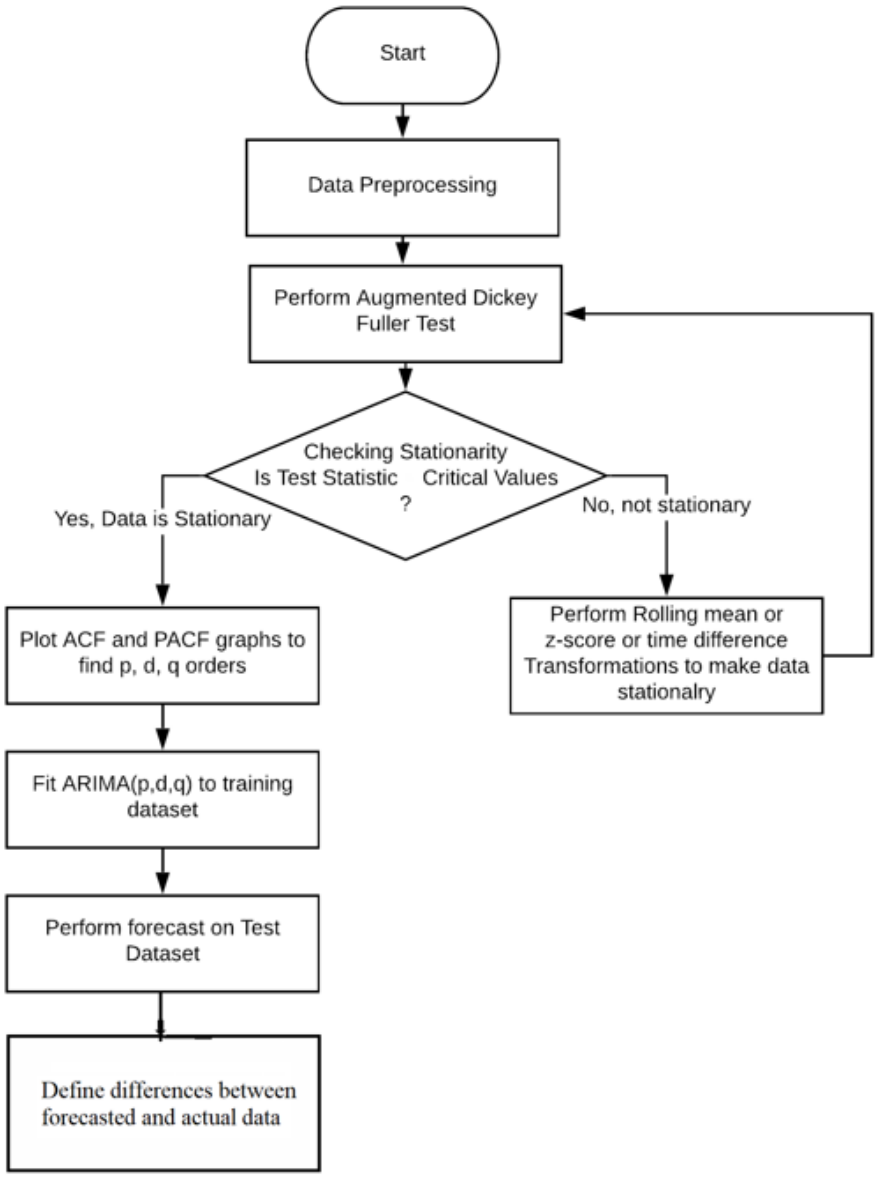


Рисунок 4.1 – Алгоритм роботи підмодуля ППМАКС

Щодо програмної імплементації, то фреймворк ML.NET базово не має підтримки ARIMA. Але можна запропонувати власний варіант реалізації на мові програмування C# з використанням алгоритму SSA, який є у фреймворку ML.NET. Варто зазначити, що під час розробки застосовується об’єктно-орієнтований підхід з характерним розбиттям даних на класи та необхідні методи.

Спочатку доцільно буде розробити класи, що характеризують певні дані або зберігатимуть у собі результати обчислень. На лістингу 4.1 наведено код з описом основних класів та їх властивостей.

Лістинг 4.1 – Реалізація класів та опис їх властивостей

```
public class ElectricityData
{
    public float Consumption { get; set; }
    public DateTime Date { get; set; }
}
public class ElectricityForecast
    {
        public float[] ForecastedConsumption { get; set; }
    }
    public class ResidualData
    {
        public float Residual { get; set; }
    }
```

Перший клас ElectricityData буде містити у собі вхідні дані про електроспоживання. Кожен об’єкт цього класу матиме 2 властивості: Consumption для значення напруги (В) та Date для дати та часу, коли було зафіксовано цю напругу.

Для збереження спрогнозованих значень моделі ARIMA розроблено клас ElectricityForecast з лише однією властивістю – ForecastedConsumption у вигляді масиву чисел з плаваючою точкою.

Останній клас необхідний для зберігання обчислених значень: клас ResidualData буде міститиме у собі підраховані залишки (різницю між прогнозованих та реальних значень напруги).

Перейдемо до реалізації методів програми. Виходячи з вище описаних класів зрозуміло, що потрібно виконувати певні дії та обчислення оперуючи цими даними.

Метод LoadData() слугує для завантаження тестового набору даних а також ініціалізації списку об’єктів класу ElectricityData (лістинг 4.2). Якщо необхідно працювати з реальними даними про споживання електроенергії, в цьому методі можна зробити завантаження даних з файлу або бази даних, попередньо змінивши код цього методу.

Лістинг 4.2 – Метод LoadData()

```
public static List<ElectricityData> LoadData()
{
    var data = new List<ElectricityData>();
    for (int i = 0; i < 100; i++)
```

```
    {
        data.Add(new ElectricityData
        {
            Date = DateTime.Now.AddMonths(-i),
            Consumption = (float)(100+20*Math.Sin(i/12.0*2*Math.PI))
        });
    }
    data.Reverse();
    return data;
}
```

Побудова моделі та її тренування, а також прогнозування значень буле реалізовано в основному блоці програми. Для обчислення різниці між фактичними та прогнозованими значеннями було розроблено метод CalculateResiduals(), код якого наведено у лістингу 4.3.

Лістинг 4.3 – Метод CalculateResiduals()

```
public static List<float> CalculateResiduals(List<ElectricityData> data,
ElectricityForecast forecast, int trainSize)
    {
        var residuals = new List<float>();
        for (int i = 0; i < data.Count - trainSize; i++)
        {
            residuals.Add(data[trainSize + i].Consumption -
forecast.ForecastedConsumption[i]);
        }
        return residuals;
    }
```

Останній метод DisplayResults() (лістинг 4.4) виводить у консоль отримані результати, а саме: дату, фактичні та прогнозовані значення, залишки.

Лістинг 4.4 – Метод DisplayResults()

```
public static void DisplayResults(List<ElectricityData> data,
ElectricityForecast forecast, List<float> residuals)
    {
        Console.WriteLine("Дата\tФактичні значення\tПрогнозовані
значення\tЗалишки\t");
        for (int i = 0; i < residuals.Count; i++)
        {
            Console.WriteLine($"{data[data.Count - residuals.Count +
i].Date}\t{data[data.Count - residuals.Count +
i].Consumption}\t{forecast.ForecastedConsumption[i]}\t{residuals[i]}\t}");
        }
    }
```

Основний код програми знаходиться у класі Program (лістинг 4.5), в якому міститься метод Main(), де проходить вся робота програми, а саме: ініціалізація даних, побудова та навчання моделі, прогнозування значень,

виклик методів для їх обробки та відображення кінцевих результатів у консолі.

Лістинг 4.5 – Основний код програми для ППМАКС

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.ML;
using Microsoft.ML.Data;
using Microsoft.ML.Transforms.TimeSeries;
public class Program
{
    public static void Main()
    {
        var mlContext = new MLContext();
        // Завантаження даних
        var data = LoadData();
        // Побудова та навчання моделі
        var forecastingPipeline = mlContext.Forecasting.ForecastBySsa(
            outputColumnName:
nameof(ElectricityForecast.ForecastedConsumption),
            inputColumnName: nameof(ElectricityData.Consumption),
            windowSize: 12,
            seriesLength: data.Count,
            trainSize: (int)(data.Count * 0.8),
            horizon: 12,
            confidenceLevel: 0.95f,
            confidenceLowerBoundColumn: "LowerBoundConsumption",
            confidenceUpperBoundColumn: "UpperBoundConsumption");
        var forecastTransformer =
forecastingPipeline.Fit(mlContext.Data.LoadFromEnumerable(data));
        var forecastEngine =
forecastTransformer.CreateTimeSeriesEngine<ElectricityData,
ElectricityForecast>(mlContext);
        // Прогнозування
        var forecast = forecastEngine.Predict();
        // Визначення залишків
        var residuals=CalculateResiduals(data,forecast,(int)data.Count*0.8));
        // Відображення результатів
        DisplayResults(data, forecast, residuals);
    }
```

При необхідності можна вносити зміни в дані фрагменти коду в залежності від потреб та результату, що очікується.

4.2.1.2 Підмодуль з реалізацією алгоритму «Ізоляційний ліс»

Підмодуль з реалізацією алгоритму «Ізоляційний ліс» (ПРАІЛ) – друга складова МВА. Головна задача ПРАІЛ – це аналіз залишків (отриманих з ППМАКС) та класифікація їх як аномалії.

Isolation Forest – це алгоритм машинного навчання, призначений для

виявлення аномалій у даних. На відміну від інших методів, Isolation Forest використовує випадкове підмноження даних та випадковий вибірковий поділ для побудови дерев рішень, що робить його дуже ефективним для виявлення аномалій. Поділ на підмножини проходить до тих пір, поки всі точки не будуть ізольовані. У моделі, що запропонована, алгоритм Isolation Forest використовується для виявлення аномалій у залишках, обчислених між фактичними та прогнозованими значеннями споживання електроенергії.

Описати процес роботи Isolation Forest можна наступним чином. Для кожного дерева iTree обирається випадкова підмножина вихідних даних. Це дозволяє уникнути ефекту переобучення та підвищити різноманітність дерев у лісі. Для кожного вузла дерева обирається випадкова ознака (атрибут) з підмножини даних. Для вибраної ознаки визначається випадковий поріг, який розділяє дані на дві підмножини.

Даний поділ повторюється рекурсивно, поки всі точки в підмножині не будуть ізольовані (кожна точка в окремому листі дерева) або не буде досягнута максимальна глибина дерева.

Після побудови дерев оцінка аномалії для кожної точки визначається на основі середньої довжини шляху від кореня до листа, де знаходиться точка. Короткі шляхи відповідають аномаліям, оскільки для їх ізоляції потрібно менше поділів. Інакше кажучи, для кожного залишку обчислюється оцінка аномалії. Чим вище оцінка, тим більш імовірно, що точка є аномальною. Точки, оцінка яких перевищує певний поріг, позначаються як аномалії. Отже, спочатку необхідно побудувати ізоляційні дерева (iTrees). Алгоритм для цього наведений на блок-схемі нижче (рисунок 4.2). Окрім послідовних дій тут присутні 2 умовні блоки. Перший – перевірка, чи досяг поточний підвузол максимальної висоти дерева або він ізольований (true – переходимо до наступного кроку; false – повертаємося до кроку з розбиттям підвибірки). Друга ж перевірка призначена для з’ясуванні чи всі підвузли досягли максимальної висоти дерева або вони ізольовані (true – переходимо до наступного кроку; false – повертаємося до кроку з розбиттям підвибірки).

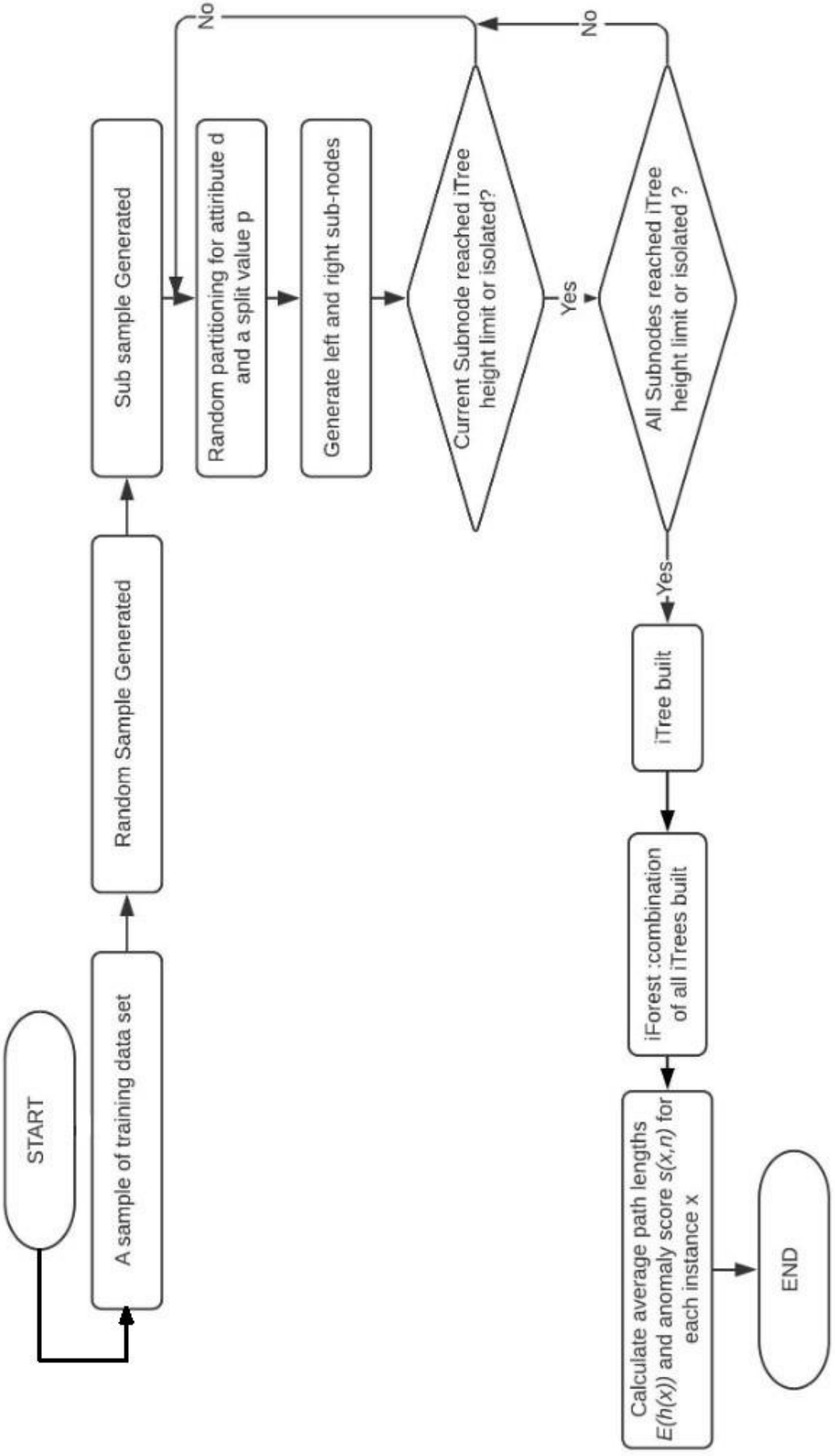


Рисунок 4.2 – Алгоритм побудови ізоляційних дерев iTrees

В результаті побудови дерев варто дати аномальну оцінку їх вузлам. На рисунку 4.3 зображено блок-схему цього процесу.

Рисунок 4.3 – Виведення аномальних оцінок для вузлів дерев ITrees

Для підмодуля ПРАІЛ також може бути розроблена програмна реалізація за допомогою мови C# та фреймворку ML.NET. Вона буде заснована на вищеописаному алгоритмі.

Так як підхід Isolation Forest передбачає побудову ізоляційного лісу, який складається з дерев, то потрібно створити відповідні класи: IsolationTree та IsolationForest (лістинг 4.6).

Лістинг 4.6 – Класи IsolationTree та IsolationForest

```
public class IsolationTree
{
    public int FeatureIndex { get; set; }
    public double SplitValue { get; set; }
    public IsolationTree Left { get; set; }
    public IsolationTree Right { get; set; }
    public List<double[]> Instances { get; set; }
    public bool IsLeaf => Left == null && Right == null;
}
public class IsolationForest
{
    private readonly int _maxDepth;
    private readonly int _numTrees;
    private readonly Random _random;
    public IsolationForest(int numTrees, int maxDepth)
    {
        _numTrees = numTrees;
        _maxDepth = maxDepth;
        _random = new Random();
    }
    public List<IsolationTree> Trees { get; private set; }
}
```

Клас IsolationTree являє собою вузол дерева. Вузол може бути листом (листовий вузол) або мати дочірні вузли (внутрішній вузол). Внутрішній вузол зберігає індекс ознаки (FeatureIndex) та значення поділу (SplitValue), а також посилання на лівий (Left) та правий (Right) дочірні вузли. Листовий вузол зберігає список інстанцій, які до нього потрапили.

Щодо класу IsolationForest, то він зберігає параметри numTrees (кількість дерев) та maxDepth (максимальну глибину дерев). Також в ньому буде знаходитись список дерев – об’єктів з типом даних IsolationTree. Так як програма слідує принципам ООП, то доцільно буде розмістити необхідні методи в середині класу IsolationForest, при тому, що деякі властивості та методи можуть мати різні модифікатори доступу (public, private, protected або internal). Тепер перейдемо до розробки та опису цих методів.

Перший метод Fit() створює дерева, обираючи випадкові підмножини даних та передаючи їх для побудови дерев (лістинг 4.7).

Лістинг 4.7 – Метод Fit() класу IsolationForest

```
    public void Fit(List<double[]> data)
    {   Trees = new List<IsolationTree>();

        for (int i = 0; i < _numTrees; i++)
        {   var sample = data.OrderBy(x
=>_random.Next()).Take(data.Count/2).ToList();
```

```
            Trees.Add(BuildTree(sample, 0));
        }
    }
```

Метод BuildTree() рекурсивно будує дерево, випадково обираючи ознаку та значення поділу для кожного вузла. Код цього методу наведено у лістингу 4.8.

Лістинг 4.8 – Метод BuildTree() класу IsolationForest

```
    private IsolationTree BuildTree(List<double[]> data, int depth)
    {
        if (depth >= _maxDepth || data.Count <= 1)
        {
            return new IsolationTree { Instances = data };
        }
        int featureIndex = _random.Next(data[0].Length);
        double minValue = data.Min(x => x[featureIndex]);
        double maxValue = data.Max(x => x[featureIndex]);
        double splitValue = _random.NextDouble()*(maxValue -
minValue)+minValue;
        var left = data.Where(x => x[featureIndex] < splitValue).ToList();
        var right = data.Where(x => x[featureIndex] >= splitValue).ToList();
        return new IsolationTree
        {
            FeatureIndex = featureIndex,
            SplitValue = splitValue,
            Left = BuildTree(left, depth + 1),
            Right = BuildTree(right, depth + 1)
        };
    }
```

Наступні 2 методи – AveragePathLength() та AnomalyScores() можна охарактеризувати так: в методі AveragePathLength() обчислюється середня довжина шляху для кожної точки через всі дерева лісу. В той же час виклик цього методу для кожного значення винесено у метод AnomalyScores() задля більшої зручності. Лістинг 4.9 демонструє код цих методів.

Лістинг 4.9 – Методи AveragePathLength() та AnomalyScores()

```
    private double AveragePathLength(double[] instance)
    {
        double pathLength = 0;
        foreach (var tree in Trees)
        {
            pathLength += PathLength(instance, tree, 0);
        }
        return pathLength / Trees.Count;
    }
    public double[] AnomalyScores(List<double[]> data)
    {
        return data.Select(x => AveragePathLength(x)).ToArray();
    }
```

На наведеному лістингу 4.10 можна побачити останні 2 методи класу IsolationForest: методи PathLength() та ExpectedPathLength(). Перший метод рекурсивно обчислює довжину шляху для кожної точки в дереві. Тоді як другий метод обчислює очікувану довжину шляху для листового вузла на основі його розміру.

Лістинг 4.10 – Методи PathLength() та ExpectedPathLength()

```
    private double PathLength(double[] instance, IsolationTree tree, int
currentDepth)
    {
        if (tree.IsLeaf)
        {
            return currentDepth + ExpectedPathLength(tree.Instances.Count);
        }
        if (instance[tree.FeatureIndex] < tree.SplitValue)
        {
            return PathLength(instance, tree.Left, currentDepth + 1);
        }
        else
        {
            return PathLength(instance, tree.Right, currentDepth + 1);
        }
    }
    private double ExpectedPathLength(int size)
    {
        if (size > 2)
        {
            return 2 * (Math.Log(size-1) + 0.5772156649) - (2*(size-1)/size);
        }
        else if (size == 2)
        {
            return 1;
        }
        else
        {
            return 0;
        }
}
```

Тепер, коли було розроблено всі необхідні програмні компоненти, можна переходити до основного блоку програми ПРАІЛ (лістинг 4.11). В цьому блоці коду йде виклик методу LoadData() – ще один метод для попереднього завантаження даних з файлу або бази даних. Після чого йде процедура ініціалізації об’єкту класу IsolationForest та виклик методів Fit() та AnomalyScores(). Це призведе до певного ланцюга виклику раніше описаних методів. Результати виявлення аномалій будуть виведені у консоль. На цьому виконання програми підмодуля ПРАІЛ дійшло до свого кінця.

Лістинг 4.11 – Основний блок програми для ПРАІЛ

```
using System;
using System.Collections.Generic;
using System.Linq;

public class Program
{
    public static void Main()
    {
        var data = LoadData();
        var iForest = new IsolationForest(numTrees: 100, maxDepth: 10);
        iForest.Fit(data);

        var scores = iForest.AnomalyScores(data);
        foreach (var score in scores)
        {
            Console.WriteLine(score);
        }
 }
```

В результаті роботи підмодулів ППМАКС та ПРАІЛ маємо аномальні значення – різкі перепади напруги. Отриманні аномалії разом з прогнозованими та фактичними значеннями напруги далі переходять до модуля МГВ для того щоб побудувати графіки та відобразити на них різке падіння або зростання напруги в певні моменти часу.

4.2.2 Модуль графічного відображення

Модуль графічного відображення (МГВ) – це невід’ємна складова моделі CADM-ECD, яка необхідна задля візуального представлення отриманих результатів з МВА та побудови відповідних графіків. Це дасть змогу наглядно продемонструвати як і часовий ряд вхідних даних про споживання електроенергії, так і виявлені аномальні перепади напруги.

Для реалізації цього модуля було вирішено обрати бібліотеку OxyPlot, тому що її функціонал цілком підходить для поставлених задач по побудові графіків та відображення аномалій на них.

Перед програмною реалізацією на мові C# варто додати бібліотеку OxyPlot до проєкту. Для цього можна скористатися NuGet Package Manager або додати залежності до файлу .csproj проєкту. Тільки після цих дій можна в повній мірі використовувати можливості OxyPlot. На лістингу 4.12 наведено приклад реалізації побудови графіків та позначення аномалій.

Лістинг 4.12 – Фрагмент коду для побудови графіків

```
public static void DisplayResults(List<double[]> data, double[] scores,
double threshold)
    {
        var model = new PlotModel { Title = "Anomaly Detection Results" };
        var normalSeries = new ScatterSeries { MarkerType =
MarkerType.Circle, MarkerFill = OxyColors.Blue };
        var anomalySeries = new ScatterSeries { MarkerType =
MarkerType.Circle, MarkerFill = OxyColors.Red };
        for (int i = 0; i < data.Count; i++)
        {
            var point = new ScatterPoint(data[i][0], data[i][1]);
            if (scores[i] > threshold)
            {
                anomalySeries.Points.Add(point);
            }
            else
            {
                normalSeries.Points.Add(point);
            }
        }
        model.Series.Add(normalSeries);
        model.Series.Add(anomalySeries);
        var plotView = new OxyPlot.Wpf.PlotView
        {
            Model = model,
            Width = 600,
            Height = 400
        };
        var window = new Window
        {
            Title = "Anomaly Detection",
            Content = plotView,
            Width = 600,
            Height = 400
        };
        var app = new Application();
        app.Run(window); }
```

В даному методі DisplayResults() реалізовано побудова та відображення графіка на основі отриманих нормальних та аномальних точок в якості аргументів метода.

Спочатку ініціалізується об'єкт PlotModel для графіку, потім створюються дві серії точок: normalSeries для нормальних точок та anomalySeries для аномальних точок. Точки додаються до відповідних серій на основі їх аномальної оцінки. Далі створюється PlotView для відображення графіку. Фінальний крок це ініціалізація та запуск вікна Window, що містить PlotView.

При виконанні цього коду на екрані з’явиться вікно з побудованими графіками(на основі нормальних та аномальних точок).

4.3 Тестування моделі CADM-ECD

Тестування моделі машинного навчання – це процес оцінки продуктивності моделі з метою забезпечення її якості, точності та відповідності очікуванням. Воно включає різні етапи та методи для перевірки ефективності та узагальнення моделі на нових, раніше невідомих даних.

Перевірка роботи моделі машинного навчання дещо відрізняється від тестування програмного забезпечення. По-перше, поведінка MLS значною мірою залежить від таких факторів, як наявні набори навчальних даних, вибір гіперпараметрів, архітектура моделі, алгоритм та оптимізатор [14]. З іншого боку, вихідний код зазвичай дуже лаконічний та менш схильний до помилок, оскільки складається з простої послідовності викликів функцій API, а політика прийняття рішень (тобто власне алгоритм) виводиться з навчальних даних. Крім того, поведінка, що демонструється, кодується за допомогою певних параметрів в моделі, що дуже важко інтерпретувати та налагоджувати для людини. Класичні методи тестування нелегко застосувати до будь-якого з вищезгаданих артефактів, за винятком вихідного коду.

В контексті тестування моделі CADM-ECD, буде досить складно провести юніт-тести або перевірити роботу кожного модуля окремо. Адже ці модулі та підмодулі працюють як єдиний механізм в даній моделі. Тому вирішено скористатись таким підходом тестування, як «чорна скринька» (Black-box tests). Тому що тести чорної скриньки не залежать від внутрішньої архітектури чи стану системи, яка знаходиться під перевіркою. Задля перевірки моделі машинного навчання під час таких тестів є доступ лише до вхідних та вихідних даних. Всі необхідні дії будуть проходити всередині моделі. Результати будуть представлені у вигляді побудованого графіку.

Щодо вхідного набору даних, то було обрано дані про споживання електроенергії одного з підприємств за період 1 січня 2022 року по 11 лютого 2022 року. Деякі з цих значень наведено в лістингу 4.13.

Після завантаження набору даних у модель CADM-ECD відбуваються

наступні дії всередині цієї моделі, а саме: розбиття вхідних даних на навчальний та тестовий набори, прогнозування очікуваних значень (ARIMA) та виявлення аномалій шляхом порівняння фактичних та прогнозованих величин з застосуванням методу Isolation Forest. Результати роботи моделі CADM-ECD зображено на рисунку 4.4.

Лістинг 4.13 – Фрагмент вхідного набору даних про енергоспоживання

```
Date and time         Energy Consumption (kW-h)
2022-01-01 01:00:00   208.003144
2022-01-01 02:00:00   219.574760
2022-01-01 03:00:00   244.817864
...                          ...
2022-01-05 02:00:00   202.538242
2022-01-05 03:00:00   208.039787
2022-01-05 04:00:00   400.000000
2022-01-05 05:00:00   173.044819
...                          ...
2022-01-09 07:00:00   226.730559
2022-01-09 08:00:00    50.000000
2022-01-09 09:00:00   195.212416
2022-01-09 10:00:00   221.993192
...                          ...
2022-01-21 18:00:00   179.639162
2022-01-21 19:00:00   198.442905
2022-01-21 20:00:00   450.000000
2022-01-21 21:00:00   199.315154
2022-01-21 22:00:00   221.926937
...                          ...
2022-01-23 00:00:00   152.608262
2022-01-23 01:00:00   217.281046
2022-01-23 02:00:00   155.207919
2022-01-23 03:00:00   208.029981
...                          ...
2022-02-11 11:00:00   208.257416
2022-02-11 12:00:00   196.032022
```

Таким чином, було побудовано графік з фактичними (синій колір) та прогнозованими (червоний колір) значеннями споживання електроенергії. Виявлені аномальні точки було виділено зеленим кольором. Це ті значення, які кардинально відрізняються від прогнозованих. Крім цього, різкі перепади можна побачити наглядно на графіку.

Отже, модель CADM-ECD працює справно та може впоратись з виявленням аномальних значень споживання електроенергії в наборі даних за певний відрізок часу. Але важливо, щоб відповідний набір даних містив у собі такі значення, як величина спожитої електроенергії (або інша величина, наприклад напруга) та її зафіксована дата в певний момент часу.

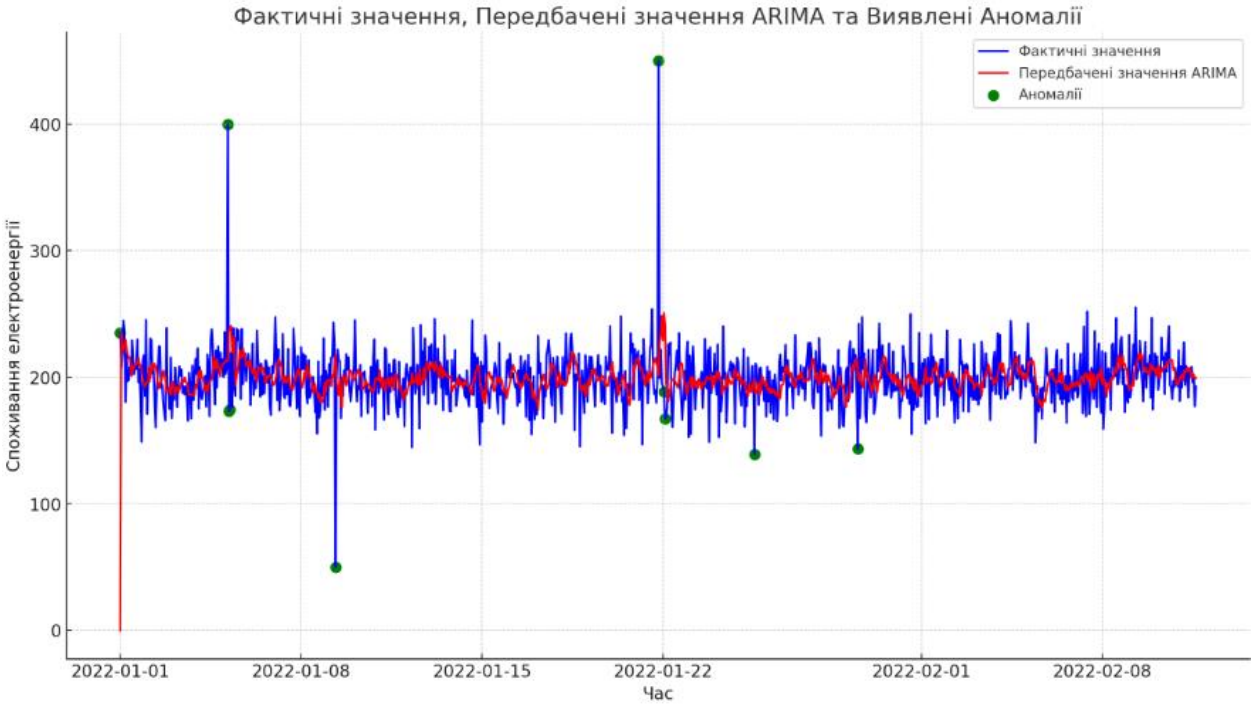


Рисунок 4.4 – Результати роботи моделі CADM-ECD

4.4 Об’єктивна перевірка результатів роботи моделі

Для об’єктивної оцінки роботи моделі CADM-ECD необхідно обчислити певні метрики, які часто використовуються для перевірки моделей машинного навчання з метою виявлення аномалій в даних. Перелік цих метрик наступний: Accuracy, Precision, F1-Score, Recall, ROC AUC.

Щоб розрахувати наведені метрики було розроблено наступний скрипт на мові програмування Python. Його фрагмент наведено у лістингу 4.14. За основу беруться 2 масиви даних: true_anomalies – це фактичні аномалії та data[‘Anomaly’] – це агреговані значення, які було отримано в результаті роботи моделі CADM-ECD (тобто ідентифіковані аномалії). Кожна метрика обчислюється на основі цих 2 наборів даних (використовуючи необхідні для кожної метрики методи, які були попередньо імпортовані з бібліотеки sklearn.metrics). Після чого отримані значення виводяться у консоль.

Лістинг 4.14 – Реалізація розрахунків метрик для моделі CADM-ECD

```
import pandas as pd
import numpy as np
from sklearn.metrics import precision_score, recall_score, accuracy_score,
```

```
f1_score, roc_curve, auc

# Обчислення метрик
precision = precision_score(true_anomalies, data['Anomaly'])
recall = recall_score(true_anomalies, data['Anomaly'])
accuracy = accuracy_score(true_anomalies, data['Anomaly'])
f1 = f1_score(true_anomalies, data['Anomaly'])

# Обчислення ROC-кривої
fpr, tpr, thresholds = roc_curve(true_anomalies, data['Anomaly'])
roc_auc = auc(fpr, tpr)
print("Precision:", precision)
print("Recall:", recall)
print("Accuracy:", accuracy)
print("F1 Score:", f1)
print("ROC AUC:", roc_auc)
```

Результати обчислень метрик наступні:

- Accuracy: загальна точність моделі складає 0.993 – це досить непоганий результат, тому що ця метрика показує скільки аномалій було класифіковано серед загальної кількості аномалій в наборі даних;

- ROC AUC: при різних порогах відсічення здатність моделі відрізняти аномалії та нормальні дані складає 0.9965;

- Precision: частка правильно виявлених аномалій серед усіх точок складає 0.3 – це дещо нижче ніж очікувалось, але в цілком результат можна вважати валідним;

- Recall: частка правильно виявлених аномалій серед усіх справжніх аномалій дорівнює 1.0, тобто те, що модель маркує реальні аномалії як аномалії можна вважати з точністю у 100%;

- F1-Score: гармонійне середнє між точністю та повнотою (Precision та Recall) складає 0.4615, тобто збалансованість моделі на достатньому рівні, але це число може бути більше при певних змінах та покращеннях.

Наведені результати підкреслюють потенціал моделі CADM-ECD та більш точні результати ніж при використанні інших моделей. Розрахованих значень цілком достатньо для об’єктивної оцінки розробленої моделі. Але хоч ці метрики і можуть відобразити певні кількосні оцінки, все ж таки варто давати ще і суб’єктивну оцінку роботи моделі. Тому що не завжди можна коректно оцінити модель тільки по об’єктивним оцінкам та розрахованим метрикам.

4.5 Суб’єктивна перевірка результатів роботи моделі

Для того, щоб надати суб’єктивну оцінку моделі CADM-ECD, потрібно проаналізувати отримані результати під час тестування моделі. Перш за все легкість розуміння результатів – побудований графік в кінці роботи моделі чітко відображає фактичні, прогнозовані та аномальні значення споживаної електроенергії. При цьому аномалії можна чітко відобразити на графіку, що дозволяє легко ідентифікувати відхилення в даних. Модель можна адаптувати для різних частот даних (годинних, щоденних, щотижневих тощо) та для різних контекстів виявлення аномалій (фінансові дані, дані про споживання ресурсів, медичні дані тощо). Щодо надійності та точності, то ARIMA є надійним інструментом для моделювання та прогнозування часових рядів з добре вираженими трендами та сезонними компонентами. В той час як алгоритм Isolation Forest ефективно виявляє аномалії навіть у великих наборах даних, завдяки своїй здатності ізолювати аномальні точки.

Так як модель CADM-ECD це поєднання методів ARIMA та Isolation Forest, тому потрібно навести суб’єктивну оцінку для кожного з них.

ARIMA легко інтерпретується як модель часових рядів, що надає прогнозовані значення. Вона здатна точно прогнозувати тренди та сезонні коливання в даних. Використання ARIMA для генерації прогнозів дозволяє ефективно враховувати внутрішні закономірності в даних про споживання електроенергії. Також добре працює з лінійними часовими рядами, але може бути менш ефективною з нелінійними або сильно змінними даними.

З цього випливає, що ARIMA вимагає стаціонарності даних або попереднього перетворення, що може бути складним для деяких типів даних. Ще один недолік це те, що ARIMA є обчислювально складною для великих наборів даних, особливо при високих порядках моделі. Тому що вона вимагає ретельного вибору порядків моделі (p, d, q), що може бути не тривіальним завданням.

Isolation Forest це ефективний метод для виявлення аномалій, особливо при роботі з великими наборами даних. Використання залишків (резідуалів) від ARIMA дозволяє Isolation Forest зосередитися на аномаліях, які не пояснюються трендом або сезонністю. Тобто допомагає визначити аномальні точки, які можна інтерпретувати як потенційно критичні події або несправності в системі.

Isolation Forest забезпечує чітке визначення аномалій, але інтерпретація деяких рішень може бути складнішою. Ще цей метод є більш гнучким щодо типів даних та виявлення аномалій, але вимагає правильного налаштування параметрів для оптимальної роботи. Проте Isolation Forest все ж може вимагати значних обчислювальних ресурсів для великих обсягів даних.

У підсумку, поєднання ARIMA та Isolation Forest в запропонованій моделі CADM-ECD для виявлення аномалій в даних про споживання електроенергії є потужним підходом, який дозволяє врахувати як сезонні та трендові компоненти, так і нестандартні аномалії. Така гібридна модель може бути дуже корисною для виявлення несправностей та підвищення надійності енергосистем. Однак, цей підхід вимагає ретельного налаштування та значних обчислювальних ресурсів, що слід враховувати при його застосуванні.

ВИСНОВКИ

В ході кваліфікаційної роботи було досліджено таке питання як виявлення аномалій в даних про споживання електроенергії. А також запропоновано рішення цієї задачі у вигляді гібридної моделі для визначення аномалій в цих даних.

Спочатку було проведено аналіз популярних існуючих рішень для визначення аномалій в показниках щодо споживання електроенергії. При чому це було як з метою теоретичного ознайомлення принципу роботи того чи іншого методу, так і експериментальна перевірка тестових даних задля визначення коректності та точності виявлення аномалій. Звернута увага на обмеження таких методів обробки даних, а також на факти, висновки, рекомендації, закономірності з раніше відомих досліджень.

В результаті аналізу було запропоновано та розроблено власну гібридну модель, яка поєднує в собі декілька методів виявлення аномалій. Під час розробки цієї моделі було використано сучасні технології та алгоритми машинного навчання. Після тренування та тестування можна сказати, що запропонована модель відповідає вимогам кваліфікаційної роботи та виконує необхідні задачі щодо визначення аномальних значень у наборах даних про енергоспоживання. Це дає змогу також підкреслити надійність, точність та ефективність роботи моделі.

Завершуючи, можна вважати, що розроблена модель для виявлення аномалій у даних про споживання електроенергії має неабияке наукове та практичне значення. Адже тема енергетики була, є, і буде актуальною. Тим паче попередження про різкі показники напруги дозволять попередити вихід зі строю електроприладів та завчасно їх вимкнути. Або у випадку збору статистичних даних це дасть змогу зрозуміти, коли і як часто показники мали пікові значення, що дасть розуміння щодо подальшого обслуговування та експлуатації електрообладнання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Братищенко М.Р., Філімончук Т.В., Майстренко Г.В., Сітніков В.І. Аналіз методів виявлення аномалій у даних про споживання електроенергії // Системи управління, навігації та зв'язку. Збірник наукових праць. Полтава: ПНТУ, 2024. Випуск 1 (75). С. 45-49. doi: https://doi.org/10.26906/ SUNZ. 2024.1.045.

2. Berardi U. Building energy consumption in US, EU, and BRIC countries // Department of Architectural Science, Faculty of Engineering and Architectural Science. 2015. №118. P. 128-136.

3. Lim J.Y., Tan W.-N., Tan Y.-F. Anomalous energy consumption detection using a Naïve Bayes approach // Faculty of Engineering, Multimedia University. Cyberjaya, Selangor, 63100, Malaysia. 2022. №1. P. 4.

4. Zhang J., Zhang H., Ding S., Znang X. Power Consumption Predicting and Anomaly Detection Based on Transformer and K-Means // College of Mathematics and Information Technology, Hebei University. 2021. Volume 9, Article № 779587. P. 3-7.

5. Amril Y., Fadhilah A.L., Fatmawati, Setianil N., Ranil S. Analysis Clustering of Electricity Usage Profile Using K-Means Algorithm // IOP Conference Series: Materials Science and Engineering. 2016. №105. P.2-7. doi: 10.1088/1757-899X/105/1/012020.

6. Lamrini B., Gjini A., Daudin S., Armando F., Pratmarty P., Travé-Massuyès L. Anomaly Detection Using Similarity-based One-Class SVM for Network Traffic Characterization // Université de Toulouse, CNRS, Toulouse, France. 2018. №1. P. 2-4.

7. Auskalnis J., Paulauskas N., Baskys A. Application of Local Outlier Factor Algorithm to Detect Anomalies in Computer Network // Elektronika I Elektrotechnika. 2018. 24(3). P. 96-99. doi: https://doi.org/10.5755/j01.eie.24.3. 20972.

8. Nasraoui O., Leon E., Krishnapuram R. Unsupervised Niche Clustering:

Discovering an Unknown Number of Clusters in Noisy Data Sets // Evolutionary Computation in Data Mining, Springer Berlin Heidelberg. 2005. Volume 1. P. 2-30.

9. Leon L., Nasraoui O., Gomez J. Anomaly detection based on unsupervised niche cluster-ing with application to network intrusion detection // Proceedings of the IEEE Conference on Evolutionary Computation. 2007. Volume 1. P. 502.

10. Chandola V., Banerjee A., Kumar V. Anomaly detection: A survey // Association for Computing Machinery Computing Surveys. 2009. №41. P. 1-58.

11. Melnyk I., Matthews B., Valizadegan H., Banerjee A., Oza N. Vector Autoregressive Model-Based Anomaly Detection in Aviation Systems // JAIS Aerospace Information Systems. 2016. №13. P. 161-173.

12. Feng L., Xu S., Zhang L., Wu J., Zhang J., Chu C., Wang Z., Shi H. Anomaly detection for electricity consumption in cloud computing: framework, methods, applications, and chal-lenges // Wireless Com Network. 2020. №194. P. 2-10. doi: https://doi.org/10.1186/s13638-020-01807-0.

13. West M. Developing high quality data models. // The European Process Industries STEP Technical Liaison Executive (EPISTLE). 2011. Book Version 2.0. P. 5.

14. Riccio V., Jahangirova G., Stocco A., Humbatova N., Weiss M., Tonella P. Testing machine learning based systems: a systematic mapping. // Empirical Software Engineering. 2020. P. 5918. doi: https://doi.org/10.1007/s10664-020-09881-0.