Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегравних технологій, автоматизації та мехатроніки
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Автоматизація процесів аналізу накопичених даних, отриманих за допомогою технології IoT
(тема)

Виконав: студент 2 курсу, гр. КІТПВм-19-1
Ахмад В. Х.
(прізвище, ініціали)

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології

освітньої програми Комп'ютерно-інтегровані технологічні процеси і виробництва
(код і повна назва напряму)

Тип програми освітньо-професійна
(повна назва освітньої програми)

Керівник проф. Невлюдов І.Ш.
(посада, прізвище, ініціали)

Зав. кафедри _____ Невлюдов І.Ш.

2020 р.

Харківський національний університет радіоелектроніки

Факультет	Автоматики і комп'ютеризованих технологій
Кафедра	Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки
Рівень вищої освіти	другий (магістерський)
Спеціальність	151 Автоматизація та комп'ютерно-інтегровані технології
Тип програми	освітньо-професійна
Освітня програма	Комп'ютерно-інтегровані технологічні процеси і виробництва

(код і повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри_____
(підпис)
«_____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Ахмаду Васіму Халєду
(прізвище, ім'я, по батькові)

1. Тема роботи Автоматизація процесів аналізу накопичених даних, отриманих за допомогою технології IoT

затверджена наказом по університету від 02.11. 2020 р. № 1511 Ст

2. Термін подання студентом роботи до екзаменаційної комісії . . 2020 р.

3. Вихідні дані до роботи об'єкт дослідження – промисловий інтернет речей; предмет дослідження – алгоритми аналізу великих даних; сервер баз даних – Postgresql; метод аналізу накопичених даних – з використанням технології машинного навчання

4. Перелік питань, що потрібно опрацювати в роботі:

4.1 Вступ.

4.2 Аналіз літератури за темою магістерської атестаційної роботи.

4.3 Обробка накопичених даних з використанням кластерного аналізу.

4.4 Методи прогнозування стану об'єкту автоматизації за допомогою засобів машинного навчання.

4.5 Експериментальні дослідження.

4.6 Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Демонстраційний матеріал представлений у форматі презентації PowerPoint (*.ppt) – 12 с. формату А4

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

Найменування розділу	Керівник (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	проф. Невлюдов І.Ш.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз літератури за темою роботи	3.09.2020	
2	Аналіз технології обробки даних в IOT	10.09.2020	
3	Аналіз аналогічних програмних рішень	20.09.2020	
4	Обробка накопичених даних на прикладі	10.10.2020	
	кластерного аналізу		
5	Вибір методів прогнозування стану об'єкту	22.10.2020	
	автоматизації		
6	Виконання експериментальних досліджень	16.11.2020	
7	Оформлення пояснювальної записки	30.11.2020	
8	Подання роботи до ЕК	08.12.2020	

Дата видачі завдання 03.11.2020 р.

Студент _____ Ахмад В. Х.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Невлюдов І.Ш.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 73 с., 7 табл., 24 рис., 2 дод., 13 джерел.

ІНТЕРНЕТ РЕЧЕЙ, ПРОМИСЛОВИЙ ІНТЕРНЕТ РЕЧЕЙ, АВТОМАТИЗАЦІЯ ПРОМИСЛОВОСТІ, БАЗА ДАНИХ, ПРОГРАМНИЙ ЗАСІБ.

Об'єкт дослідження – промисловий Інтернет речей.

Предмет дослідження – алгоритми аналізу великих даних.

Мета роботи – вибір алгоритму та розробка програми для реалізації задачі аналізу даних на хмарному сервері Інтернету речей.

У роботі проведено:

– дослід принципів обробки великих даних на серверах Інтернету речей;

– розроблений алгоритм для виконання аналізу накопичених даних;

– розроблена програма для аналізу даних на хмарному сервері Інтернету речей;

– проведене експериментальне дослідження використання обраного методу аналізу даних на тестовому сервері.

Методи дослідження – в якості методів дослідження були застосовані імітаційні методи програмного моделювання, методи машинного навчання та регресивний аналіз.

Результати магістерської атестаційної роботи апробовані в журналі ADED.

ABSTRACT

Explanatory note contains: 74 pp., 7 tables, 24 figures, 2 applications, 10 sources.

INTERNET OF THINGS, INDUSTRIAL INTERNET OF THINGS, INDUSTRIAL AUTOMATION, DATABASE, SOFTWARE.

The object of research – industrial internet of things.

Subject of research – big data analysis algorithms.

The purpose of the work – selection of an algorithm and development of a program for the implementation of the task of data analysis on the cloud server of the Internet of Things.

The work is carried out:

- explore the principles of big data processing on Internet of Things servers;
- developed an algorithm to analyze the accumulated data;
- developed a program for data analysis on the cloud server of the Internet of Things;
- an experimental study of the use of the selected method of analysis of VD on the test server.

Research methods – simulation methods of software modeling, machine learning methods and regression analysis were used as research methods.

The results of the master's certification work were tested in the journal ADED.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

IOT – Internet of things,

AWS – Amazon Web Services

SDK – Software development kit

MQTT – Message Queuing Telemetry Transport

HTTP – HyperText Transfer Protocol

API – Application Programming Interface

ВД – великі дані (Big Data)

ВСТУП

Взаємопов'язані фізичні пристрої представляють зовсім нові можливості промисловості і приватним особам. Завдяки Інтернету речей можна дистанційно керувати технологічним процесом, транспортними засобами і робочими пристроями за допомогою, наприклад, мобільним пристроям. З цим великим рівнем зв'язності стало можливим збирати величезну кількість даних з цих пристроїв, а точніше інформацію, яку дають вбудовані в них сенсори. Ці сенсори, такі як датчики температури, струму, положення, та інші, можуть видавати безперервний потік даних, але ці дані мають обмежене застосування без використання методів їх аналізу.

Отриману інформацію можна використовувати для оптимізації контрольованих в режимі реального часу процесів, для більш якісного планування обслуговування і роботи обладнання, а також для аналізу за допомогою спеціального програмного забезпечення. Надалі результати такого аналізу дозволять підвищити рівень безпеки, збільшити ефективність роботи обладнання і діагностувати проблеми в його функціонуванні.

«Великі дані» (ВД), що накопичуються на серверах і платформах Internet of things (IoT), і застосування технологій машинного навчання дозволяє автоматизувати процеси вдосконалення програмновиконуваних алгоритмів, тобто оптимізувати алгоритми управління накопичення технологічних даних, що надходять від широкої номенклатури пристроїв і автоматизованих систем управління.

Таким чином, актуальною є задача автоматизованого аналізу накопичених даних математичними методами обробки ВД.

Об’єкт дослідження – промисловий інтернет речей.

Предмет дослідження – алгоритми аналізу ВД.

Мета атестаційної роботи – вибір алгоритму та розробка програми для реалізації задачі аналізу даних на хмарному сервері Інтернету речей.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– дослідити принципи обробки ВД на серверах Інтернету речей;

– обрати алгоритм для виконання аналізу накопичених даних;

– розробити програму для реалізації задачі аналізу даних на хмарному сервері Інтернету речей;

– провести експериментальне дослідження використання обраного методу аналізу ВД на тестовому сервері;

– оформити атестаційну роботу магістра згідно ДСТУ 3008:2015 [1] та методичними вказівками з «Розробки й оформлення магістерської атестаційної роботи» для студентів другого (магістерського) рівня вищої освіти галузі знань 15 Автоматизація та приладобудування за спеціальністю 151 Автоматизація та комп’ютерно-інтегровані технології освітні програми: «Автоматизоване управління технологічними процесами», «Комп’ютерно-інтегровані технологічні процеси і виробництва», «Комп’ютеризовані та робототехнічні системи» [2].

Актуальність теми роботи – великі об’єми даних, що накопичуються на серверах і платформах Інтернету речей, і застосування технологій машинного навчання дозволяє автоматизувати процеси вдосконалення програмованих алгоритмів для вирішення задач управління технологічними процесами сучасних виробництв.

1 АНАЛІЗ ЛІТЕРАТУРИ ЗА ТЕМОЮ МАГІСТЕРСЬКОЇ АТЕСТАЦІЙНОЇ РОБОТИ

1.1 Аналіз існуючих архітектурних рішень серверів для IoT

Платформа Amazon Web Services (AWS) IoT забезпечує підключення пристроїв до сервісів AWS і інших пристроїв, захист даних і безпеку взаємодій, обробку даних пристроїв і дії з ними, а також взаємодія додатків з пристроями навіть при відсутності підключення до Інтернету [3]. На рисунку 1.1 приведена архітектура сервера для IoT фірми AWS.

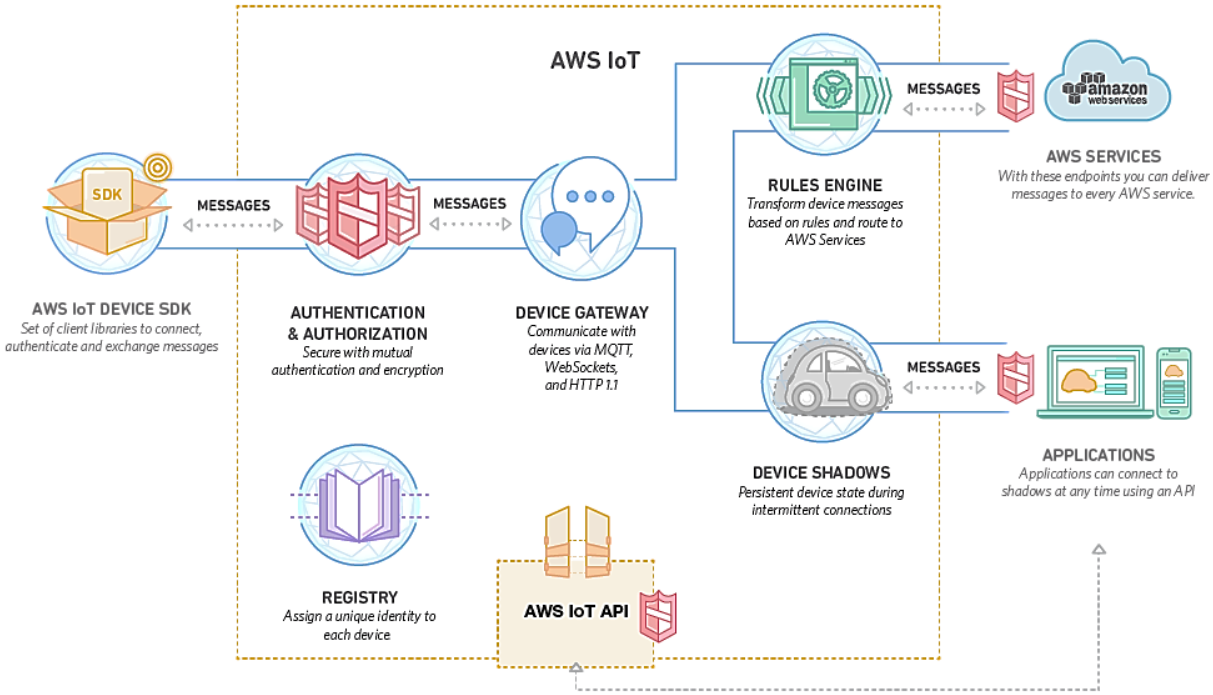


Рисунок 1.1 – Архітектура сервера для IoT фірми AWS

Платформа AWS IoT надає SDK (Software development kit) для простого і швидкого підключення апаратних пристроїв і мобільних додатків. AWS IoT SDK для пристроїв забезпечує підключення і аутентифікацію різних пристроїв, а також обмін повідомленнями з платформою AWS IoT по протоколам Message Queuing Telemetry Transport (MQTT), HyperText Transfer

Protocol (HTTP) або WebSockets. Пакет SDK для пристроїв підтримує мови C, JavaScript і Arduino і містить клієнтські бібліотеки, керівництво для розробників і керівництво по перенесенню для виробників. Також можна скористатися альтернативним SDK з відкритим вихідним кодом або написати власний SDK.

Шлюз пристроїв AWS IoT забезпечує безпечну і продуктивну взаємодію пристроїв з платформою AWS IoT. Шлюз пристроїв підтримує обмін повідомленнями за допомогою моделі публікації і підписки, що забезпечує взаємодію за схемами «один-до-одного» і «один-до-багатьох».

В рамках схеми взаємодії «один-до-багатьох» AWS IoT дозволяє підключеному пристрою транслювати дані багатьом підписникам даної теми. Шлюз пристроїв підтримує протоколи MQTT, WebSocket і HTTP 1.1, тому можна реалізувати підтримку пропрієтарних або застарілих протоколів. Шлюз пристроїв автоматично масштабується і може підтримувати більше мільярда пристроїв без необхідності виділення інфраструктури [3].

Платформа AWS IoT забезпечує взаємну аутентифікацію і шифрування між усіма точками підключення. Таким чином будь-який обмін даними між пристроями і AWS IoT відбувається тільки з перевіркою ідентифікації. AWS IoT підтримує метод аутентифікації AWS, який називається SigV4, а також аутентифікацію на основі сертифікатів стандарту X.509. Підключення по протоколу HTTP можуть використовувати будь-який з цих методів.

Підключення по протоколу MQTT використовують аутентифікацію на основі сертифікатів. Підключення по протоколу WebSockets можуть використовувати метод SigV4. AWS IoT дозволяє використовувати сертифікати, сформовані самим сервісом AWS IoT, а також сертифікати, підписані обраним центром сертифікації. Ви можете прив'язати обрану вами роль і/або політики до кожного сертифікату для надання авторизованого доступу пристроїв або додатків з можливістю відкликання дозволу доступу без необхідності працювати з пристроєм безпосередньо. Можна створювати

та використовувати сертифікати і політики пристроїв і керувати ними з Консолі або за допомогою Application Programming Interface (API).

Ці сертифікати пристроїв можна надавати, активувати і пов'язувати з відповідними політиками, налаштованими за допомогою AWS IAM. Це дозволяє при необхідності відразу ж відкликати дозвіл доступу конкретного пристрою.

AWS IoT також підтримує підключення з боку мобільних додатків користувачів за допомогою сервісу Amazon Cognito, який повністю забезпечує створення унікальних ідентифікаторів для користувачів ваших додатків і отримання тимчасових даних доступу AWS з обмеженими правами.

Реєстр встановлює ідентифікацію для пристроїв і дозволяє відстежувати метадані, такі як атрибути або можливості пристрою. Реєстр дозволяє унікальним чином ідентифікувати кожен пристрій відповідно до єдиного формату, не залежних від типу пристрою або його підключення. Він також підтримує використання метаданих, що описують можливості пристрою, наприклад, реєстрацію датчиком температури за шкалою Фаренгейта або Цельсія. Щоб уникнути видалення збережених в реєстрі метаданих, досить звертатися до відповідного запису реєстру або оновлювати її хоча б раз в 7 років.

За допомогою AWS IoT можна створити постійну віртуальну версію кожного пристрою, так звану «тінь», що містить його останній стан і дозволяє додаткам або іншим пристроям зчитувати повідомлення і взаємодіяти з цим пристроєм. «Тіні» пристроїв зберігають останній зареєстрований стан і бажаний майбутній стан кожного пристрою, навіть якщо він не приєднаний. Отримати останній зареєстрований стан пристрою або задати його бажаний майбутній стан можна за допомогою API або движка правил.

Використання «тіней» пристроїв спрощує створення додатків, взаємодіючих з вашими пристроями, завдяки наявності REST API, які завжди

доступні. Крім того, додатки можуть задавати бажаний майбутній стан пристрою без урахування його поточного стану. AWS IoT порівнює бажаний і останній зареєстрований стан і дає пристрою команду застосувати відмінність між ними.

За допомогою AWS IoT SDK для пристроїв можна легко синхронізувати стан пристроїв з їх «тінями» і реагувати на дані про бажані майбутні стани, що задані за допомогою «тіней».

При використанні «тіней» пристроїв можна безкоштовно зберігати стани ваших пристроїв протягом року. Щоб уникнути видалення «тіней» пристроїв, досить оновлювати їх хоча б раз на рік.

Движок правил дозволяє створювати додатки IoT для збору, обробки та аналізу даних, що генеруються підключеними пристроями, і виконання дій з ними в глобальних масштабах без необхідності керувати будь-якою інфраструктурою. Движок правил оцінює вхідні повідомлення, що публікуються в AWS IoT, перетворює і доставляє їх до іншого пристрою або хмарного сервісу з урахуванням заданих бізнес-правил. Правило можна застосовувати до даних від одного або від багатьох пристроїв і виконувати на його основі одну дію або ж безліч паралельних дій.

Движок правил також може перенаправляти повідомлення в кінцеві точки AWS, в тому числі AWS Lambda, Amazon Kinesis, Amazon S3, Amazon Machine Learning, Amazon DynamoDB, Amazon CloudWatch і Amazon Elasticsearch Service з вбудованими можливостями інтеграції з платформою Kibana. Доступ до зовнішніх кінцевих точок можливий за допомогою сервісів AWS Lambda, Amazon Kinesis і Amazon Simple Notification Service (SNS) [3].

Можна створювати правила за допомогою Консолі управління або писати їх, використовуючи SQL-подібний синтаксис. Правила можна розробити таким чином, щоб вони діяли по-різному в залежності від вмісту повідомлення. Наприклад, якщо показання температурного датчика перевищило деяке граничне значення, може запускатися правило для передачі даних в AWS Lambda. Правила можуть також передбачати облік

інших даних в хмарі, наприклад даних, отриманих від інших пристроїв. Приміром, можна задати виконання дії, якщо температура більше ніж на 15 % перевищує середній показник п'яти інших пристроїв.

У движку правил доступні десятки функцій для перетворення даних, а за допомогою сервісу AWS Lambda можна розширювати їх число до нескінченності. Наприклад, при роботі з широким діапазоном значень можна обчислювати середнє значення чисел, що входять. За допомогою правил можна також ініціювати виконання коду на Java, Node.js або Python в AWS Lambda, що відкриває надзвичайно гнучкі можливості потужної обробки даних від пристроїв.

1.2 Аналіз накопичених даних, отриманих за допомогою технології IoT

Машинне навчання (Machine Learning) являє собою клас методів штучного інтелекту, для якого характерно не пряме рішення задачі, а навчання в процесі застосування рішень безлічі подібних завдань [4, 5]. В процесі побудови таких методів можуть бути використані засоби математичної статистики, чисельних методів, методів оптимізації, інтелектуального аналізу даних, теорії ймовірностей, теорії графів. Теоретичні розділи машинного навчання були виділені в теорію обчислювального навчання. Машинне навчання використовується в тому випадку, коли явне програмування високопродуктивних алгоритмів є важкою або зовсім не здійсненною задачею. Однак якість настройки системи навчання безпосередньо залежить від якості і повноти навчальної вибірки.

За допомогою машинного навчання можливе вивчення закономірностей в даних, які згодом використовуються для виявлення аномального поведінки. Завдання машинного навчання зазвичай поділяються на наступні категорії в залежності від наявності навчального «сигналу» або «зворотного зв'язку», доступного для системи навчання.

Навчання з учителем (supervised learning) полягає в навчанні системи за допомогою прикладів «стимул-реакція». Як окремі випадки виділяють способи машинного навчання, при яких вхідний сигнал може бути доступний частково або ж обмежений спеціальної зворотним зв'язком:

– навчання з частковим залученням вчителя (semi-supervised learning), коли для частини прецендентів задається тільки тренувальний набір без вихідних значень (наприклад, методи на основі графів);

– активне навчання (active learning) є окремим випадком навчання з частковим залученням вчителя, особливістю якого є здатність інтерактивного запиту експерта. Використовується в тому випадку, коли у вибірці переважають немарковані дані, а ручне маркування представляється трудомісткти. Деякі алгоритми активного навчання засновані на методі опорних векторів (support vector machines);

– навчання з підкріпленням (reinforcement learning) – система навчається, взаємодіючи з деяким середовищем, де вона приймає в ході навчання сигнали підкріплення від системи. Навчання моделі представимо вигляді виконання наборів правил, на підставі яких з плином часу змінюється матриця взаємодії моделі і середовища (наприклад, алгоритм Q-leaming).

Навчання без вчителя (misupervised learning) передбачає навчання системи без втручання з боку експерту. Підходить для задач, об'єкти в яких детально описані, та потрібно встановити внутрішні взаємозв'язки між ними. Методи рішення – метод найближчих сусідів (k-means), графові алгоритми кластеризації (Ієрархічна кластеризація).

1.3 Математичні методі аналізу ВД

1.3.1 Методи виявлення аномалій

Інтелектуальний аналіз даних (Data Mining) є областю досліджень, що дозволяє виділити деяку нову значущу інформацію у великому обсязі даних.

Головним завданням даної області є отримання явно не видимих зв'язків і непередбачених тенденцій за допомогою різних методів обробки даних.

Виявлення аномалій відноситься до проблеми пошуку зразків даних, які не відповідають певному поняттю нормального, очікуваного поведінки і називаються викидами. Процес пошуку аномалій полягає у визначенні області, що представляє собою нормальна поведінка, н в подальшому віднесення до аномалій тих спостережень в даних, які не належать до цієї нормальної області. Таким чином, аномалії визначають не їх власні характеристики а порівняння з тим. що є норма.

Аномальні значення можут бути наслідком наступного:

– наявності шумових об'єктів (невірно класифікованих об'єктів);

– показання зламався датчика, тобто буде присутність об'єктів «інших» вибірок;

– помилок в даних (наприклад, неточності вимірювання, округлення або невірної запису).

На рисунку 1.2 представлений приклад завдання пошуку аномальних значень з двома ознаками. Видно, що шум (noise) є аномалією в слабкому сенсі, оскільки він може тільки лише розмивати кордони кластера, в той час як інтерес представляють аномалії в сильному сенсі, які ці кордони спотворюють.

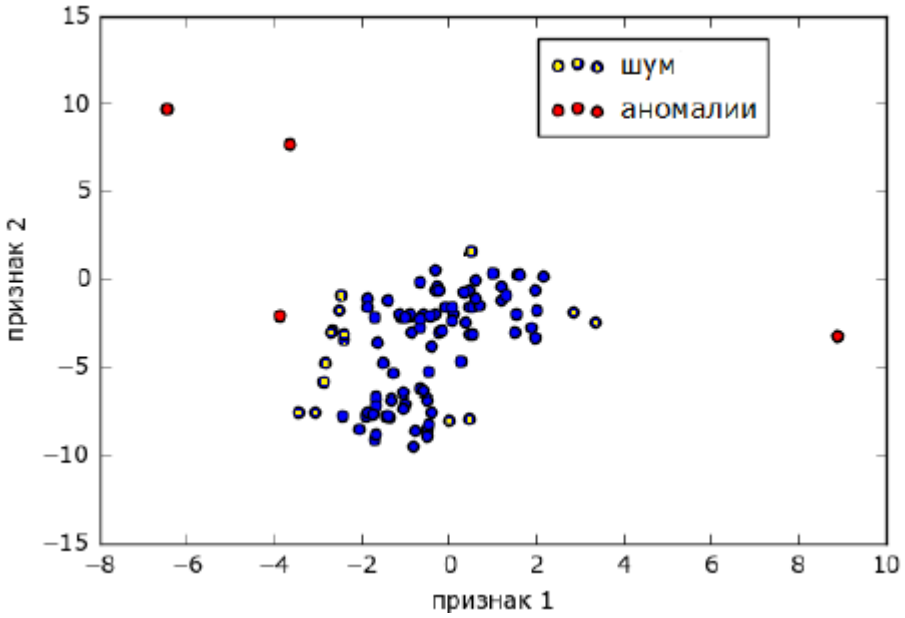


Рисунок 1.2 – Задача пошуку аномалій з двома признаками

Детектування аномалій використовується, наприклад, при пошуку відмов в системах безпеки; для контролю руху поїздів з метою підвищення безпеки руху на залізничному транспорті; при виявленні шахрайства при проведенні банківських транзакцій і вторгнень в комп'ютерній безпеці, для чого проводиться аналіз трафіку на аномальність; при виявленні ракових пухлин, які можуть бути виявлені за допомогою детектування аномалій на знімках магнітно-резонансної томографії.

При виявленні аномалій можуть виникнути такі складнощі:

– визначення нормальної області, яка охоплює всі можливі нормальні поведінки ускладнене, оскільки кордони між нормальним і аномальним поведінками часто неточні, а аномальне спостереження, що лежить поблизу кордону, може бути нормальним і навпаки;

– коли аномалії є результатом зловмисних дій, зловмисники часто пристосовуються таким чином, щоб аномальні спостереження здавалися нормальними, що ускладнює завдання визначення аномального поведінки;

– найчастіше дані містять шум. схожий на фактичні аномалії і з цієї причини важко від неї відмітний;

– наявність промаркованих, використовуваних для виявлення аномалій, даних для навчання і перевірки моделей;

– оскільки поняття нормальної поведінки розвивається, тим самим змінюючись, поточні поняття такої поведінки можуть бути недостатньо репрезентативним в майбутньому.

Через перераховані вище пункти, проблему виявлення аномалій в її найбільш загальній формі вирішити непросто. Фактично, більшість існуючих в даний час методів детектування аномалій вирішують конкретне формулювання проблеми. Вибір методу залежить від багатьох факторів, наприклад, характеру даних, доступністю промаркованих даних, типом і характером аномалій, що підлягають виявленню, та інше. Найчастіше ці чинники визначаються областю застосування, в якій необхідно виявляти аномалії.

На рисунку 1.3 представлені приклади аномалій a_1, a_2 і A до груп нормальних областей даних N_1 і N_2. Тут аномалії a_1, a_2 мають схожий характер, в той час як аномалія A від них відрізняється.

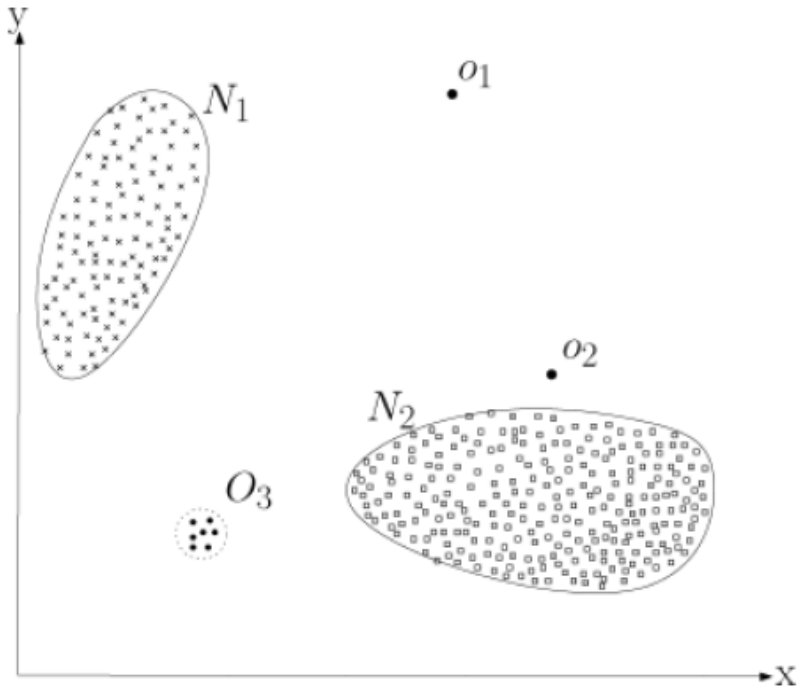


Рисунок 1.3 – Приклад аномальних значень в двомірному просторі

Найважливішим аспектом рішення задачі виявлення аномалій є визначення виду аномалій.

Якщо аномалія може бути класифікована по відношенню до решти набору даних, то таку аномалію називають одиночною, або точковою. На рисунку 1.2 a_1, a_2 і A являються одиночними аномаліями до нормальних множин об'єктів N_1 і N_2.

Прикладом застосування методів детектування одиночних аномалій може служити виявлення збоїв в роботі системи управляння рухом автономного мобільного роботу, наприклад, коли дані можуть бути представлені одним-двома ознаками – швидкістю руху мобільного робота і споживанням електроенергії. В такому випадку одиночної аномалією буде точкове підвищене споживання енергії в порівнянні з нормальними споживанням для даної швидкості руху роботу.

Якщо дані аномальні тільки в певному контексті, то така аномалія буде контекстуальною або умовною. Контекст визначається структурою набору даних і формулюванням проблеми. На рисунку 2.4 зображено приклад контекстуальної аномалії стосовно до задачі аналізу середньомісячної температури повітря на виробничій ділянці. В даному випадку тільки t_2 є контекстуальної аномалією, в той час як t_1 нею не є, незважаючи на рівність за значенням t_2.

Для даних можуть бути визначені наступні ознаки:

– контекстуальні ознаки необхідні при визначенні сусідніх прикладів, що представляють собою контекст, для поточного: наприклад, такою ознакою є момент часу в тимчасових рядах;

– не контекстуальні характеристики прикладу визначаються поведінковими ознаками. Наприклад, не контекстуальною характеристикою при описі середньої швидкості руху швидкість мобільного роботу в певному місці траєкторії.

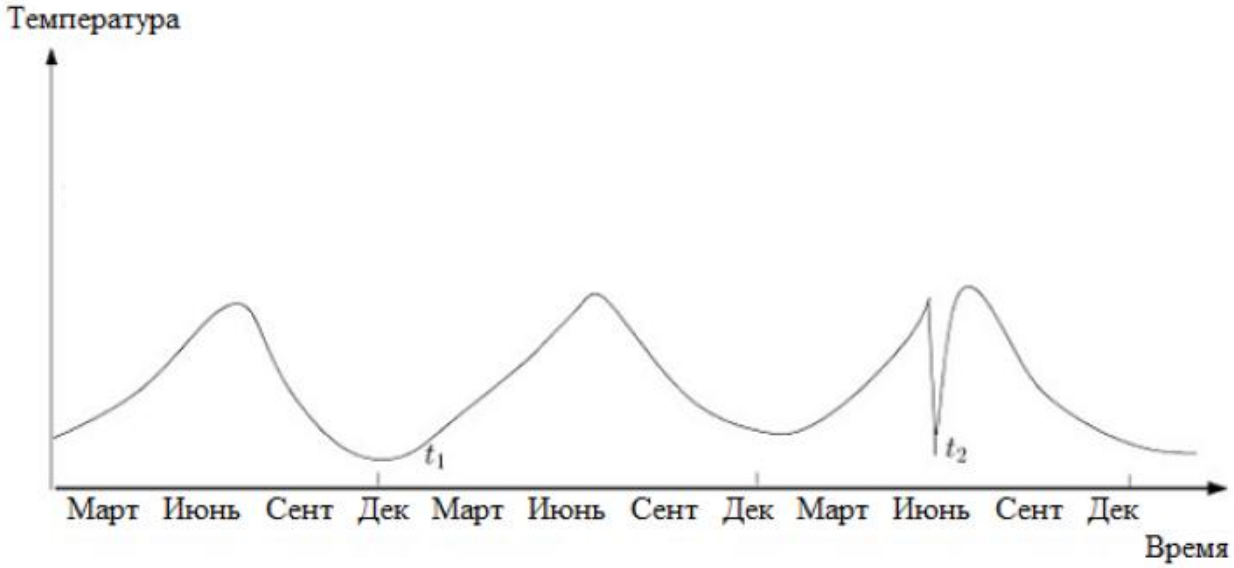


Рисунок 1.4 – Приклад контекстуальних аномалій

Серед підходів до виявлення контекстуальних аномалій можуть бути виділені два напрямки:

– використання методів детектування одиночних аномалій стосовно до контексту, коли виробляється перетворення в простір станів, що дозволяє виявляти аномалії без використання інформації про контекст;

– при обробці даних в контексті застосовуються методи моделювання часових рядів. Тоді для виявлення контекстуальних аномалій використовуються наступні регресійні методи: стійка регресія, авторегресійні моделі, авторегресійні моделі змінного середнього.

Нові дані перевіряються на аномальність шляхом порівняння з коваріаціоною матрицею авторегресійного процесу. При цьому якщо даний стан знаходиться поза межами допустимої помилки, він буде визнано класифікатором як аномальний. Незважаючи на велику обчислювальну складність на етапі навчання етап перевірки на аномальність менш витратний, ніж при перетворенні контекстуальних даних для пошуку одиночних аномалій.

Якщо пов'язані між собою дані аномальні по відношенню до іншого набору даних, то вони називаються груповий аномалією, при цьому її окремі

приклади можуть не бути поодинокими аномаліями. Групова аномалія представлена на фрагменті кардіограми (рисунок 1.5). Тут груповою аномалією є запис передчасного скорочення серця, яке в одиничному випадку є допустимим.

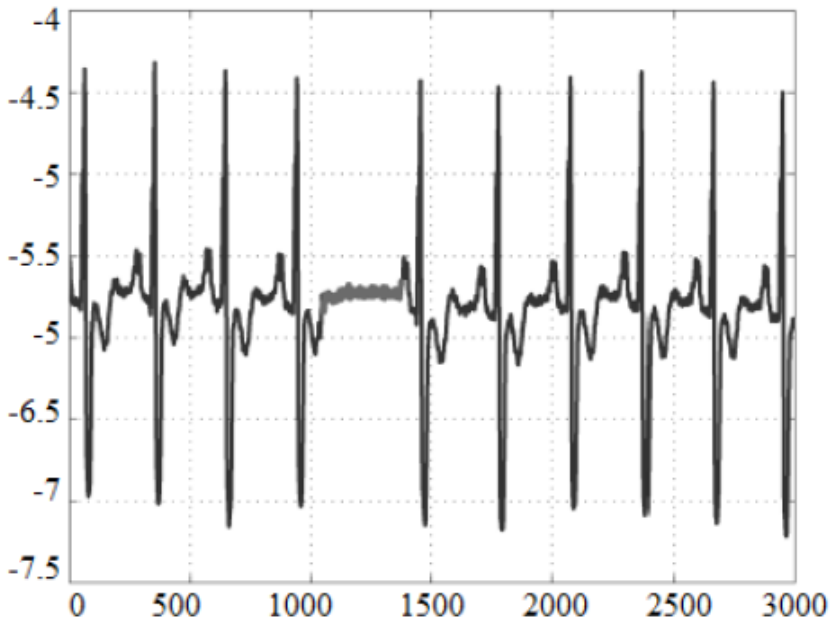


Рисунок 1.5 – Приклад групової аномалії

Поодинокі та групові аномалії могутній бути контекстуальними. В такому випадку завдання пошуку одиночних або групових аномалій може бути зведена до задачі виявлення контекстуальних аномалій.

Існують такі підходи до детектування аномалій.

Визначення аномалій за допомогою кластеризації без вчителя: дані обробляються по статистичному розподілу, а найбільш віддалені точки маркуються як потенційні викиди (рисунок 1.6).

Кластеризація без вчителя передбачає два можливих дії з виявленими аномаліями: в результаті діагностики аномальні значення можуть бути видалені з вибірки, в той час як при впровадженні вони можуть бути використані в моделі розподілу.

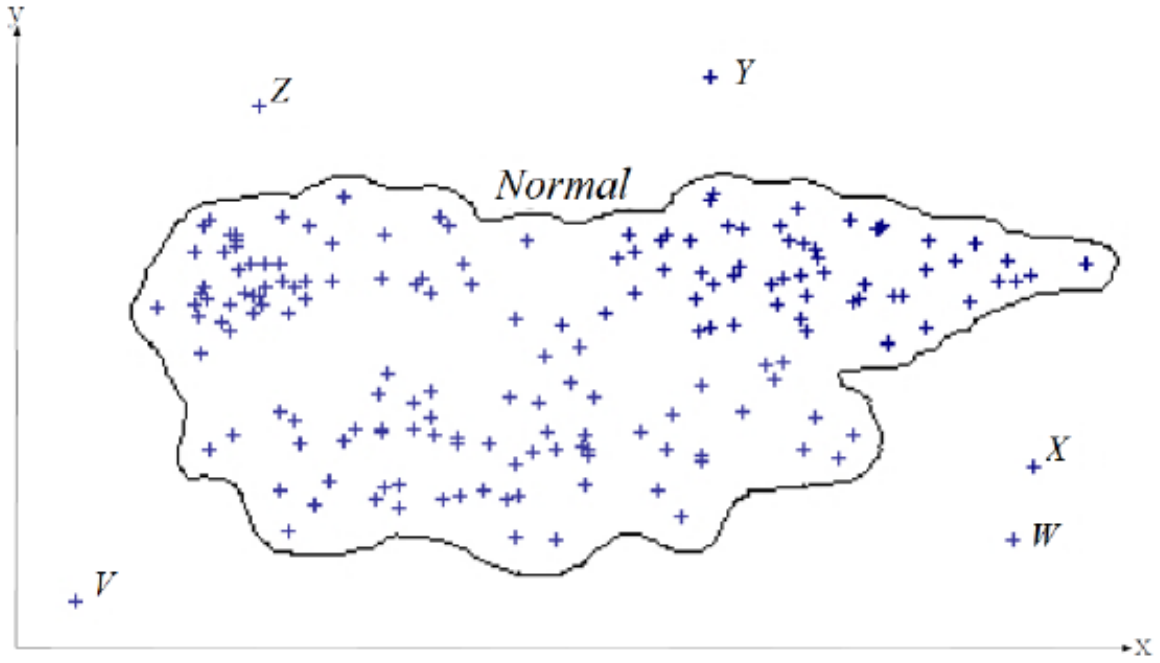


Рисунок 1.6 – Приклад застосування виявлення аномалій без вчителя

Детектування аномалій за допомогою класифікації з учителем вимагає маркованих відповідно до приналежності до аномальних значень даних. В результаті маркування можуть бути визначені кілька класів, що містять нормальні дані (рис. 1.7). Потенційною проблемою такого підходу може стати зміна статистичних характеристик даних, що несе за собою перебудову класифікатора.

Ця проблема вирішена за допомогою інкрементальних класифікаторів, що використовуються, наприклад, в еволюційних нейронних мережах в режимі реального часу, де побудована модель порівнюється зі вступниками на вхід гримерами за заданим алгоритмом.

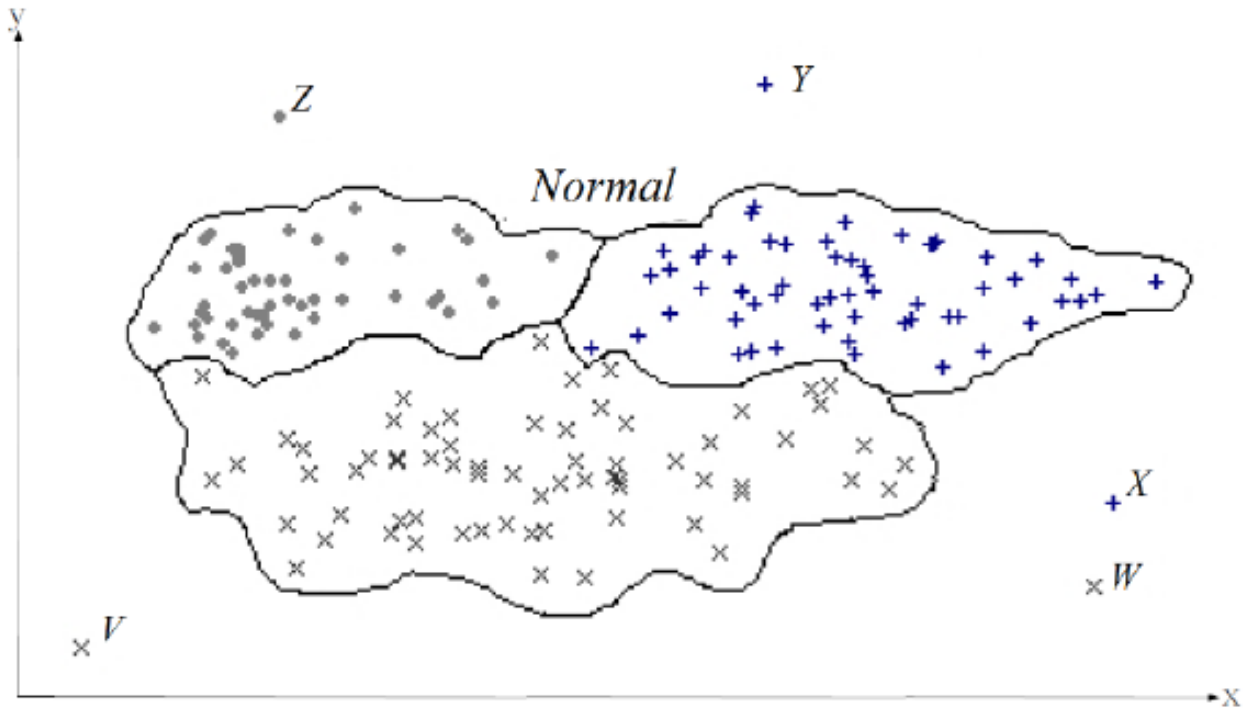


Рисунок 1.7 – Приклад застосування виявлення аномалій з учителем

Для пошуку новизни використовують навчання частково з учителем, навчаючись тільки на одному класі і припускаючи розпізнавання алгоритмом протилежного класу. Таким чином, новий приклад класифікується як нормальний якщо його можна порівняти з навчальною вибіркою, і розпізнається як новизна в зворотному випадку.

Рисунок 1.8 ілюструє нормальну область, необхідну для навчання частково з учителем і подібну інформації, що наведена на рисунку 1.6.

При порівнянні точок V, W, X, Y (рис. 1.6) з навченої моделлю на основі даних, представлених на рис. 1.8, вони будуть класифіковані як новизна в зв'язку з їх віддаленістю від навчальної вибірки.

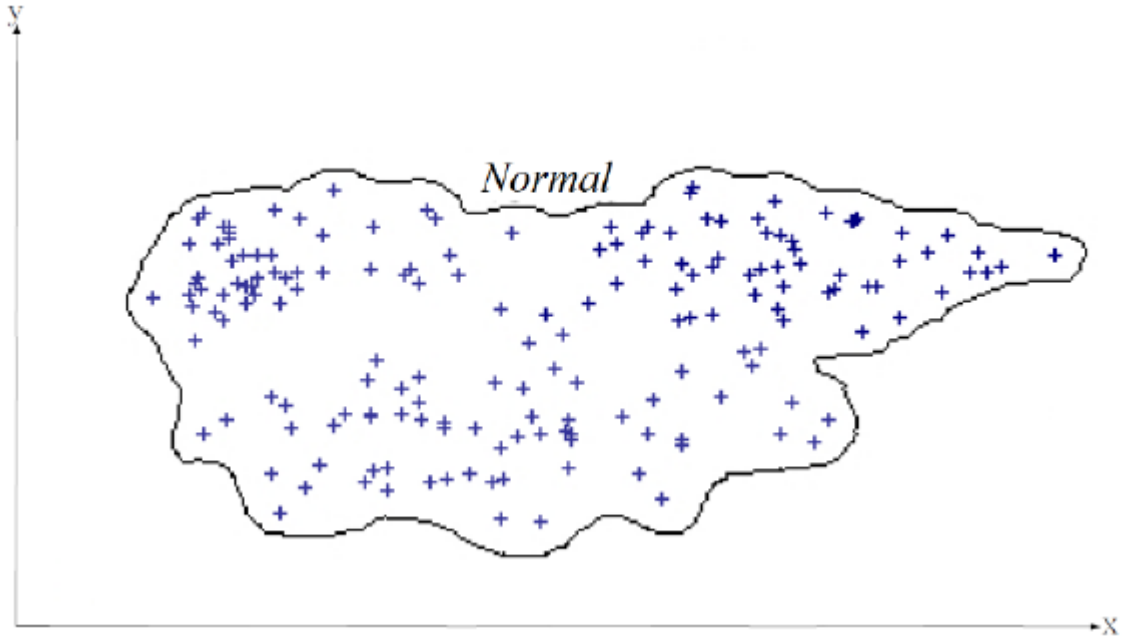


Рисунок 1.8 – Приклад застосування виявлення аномалій частково з учителем

При пошуку аномалій на основі статистичного аналізу виконується побудова профілів, що представляють реальну і ідеальну поведінки. Статистична система виявлення аномалій після обробки процесу оновлює реальний профіль і шляхом порівняння з ідеальним профілем за допомогою визначеної функції аномальності обчислює його ступінь аномальності. До переваг такого підходу відноситься відсутність вимоги апріорної інформації про ознаки аномалій, що дозволяє детектувати уразливості нульового дня, проти яких ще не розроблені захисні механізми.

До недоліків відноситься складність визначення порогу для оптимального детектування аномалій, неможливість ідентифікації аномалій в результаті зловмисних дій, схожих зі звичними діями або роботою і вимога наявності статистичних розподілів при недоступності всіх елементів досліджуваного процесу.

При пошуку аномалій на основі методів машинного навчання передбачається побудова системи, яка вдосконалюється на основі попередніх

знань. Високі обчислювальні витрати, а також складність адаптації до предметної області відносяться до недоліків методів машинного навчання.

Пошук аномалій на основі штучних нейронних мереж відноситься до класу статистичних алгоритмів, ідея яких була взята з принципу організації біологічних нейронних мереж. Перевагою підходу є стійкість до неточно введеної інформації, а також незалежність від наявності інформації про залежності прикладів вхідних даних. Недоліком є складне і довге навчання нейронних мереж, а також вимогливість до обсягу навчальної вибірки.

В основі генетичних алгоритмів пошуку аномалій лежить пошук приблизних рішень для оптимізації або рішення поставленої проблеми. Перевагою таких алгоритмів є стійкість до випадкових шумових змін вибірки даних: недоліком – складність підбору правил для вибору кращих рішень.

Оскільки безліч процесів представляється тільки нечіткими величинами, стає можливим пошук аномалій на основі нечіткої логіки, концепція «розмитості» даних якої дозволяє врахувати кордон між нормальними і аномальними даними. При цьому методі виникає складність з генерацією нечітких правил, що складно реалізуються на сучасних пристроях, які працюють з дискретної логікою.

1.4 Висновки по першому розділу магістерської атестаційної роботи

В результаті виконання першого розділу магістерської атестаційної роботи проведено аналіз принципів побудови додатків IoT та особливості застосування Інтернету речей на виробництві. Визначено загальні вимоги до сервера збору даних і надання послуг пристроям, які підтримують технологію Інтернету Речей (IoT).

Виконано аналіз існуючих архітектурних рішень серверів для IoT та аналіз методів обробки накопичених даних на сервері. Дано визначення терміну «Машинне навчання» та наведено приклади застосування різних математичних методів аналізу ВД.

2 ТЕОРИТИЧНЕ ПОДАННЯ ОБРОБКИ НАКОПИЧЕННЫХ ДАННЫХ З ВИКОРИСТАННЯМ КЛАСТЕРНОГО АНАЛІЗУ

2.1 Класифікація алгоритмів

Кластеризація (або кластерний аналіз) – це задача розбиття множини об'єктів на групи, які називаються кластерами. Усередині кожної групи повинні виявитися «схожі» об'єкти, а об'єкти різних групи повинні бути якомога більш відмінні. Головна відмінність кластеризації від класифікації полягає в тому, що перелік груп чітко не заданий і визначається в процесі роботи алгоритму.

Застосування кластерного аналізу для розроблюваної інтелектуальної системи в загальному вигляді зводиться до наступних етапів:

– відбір вибірки об'єктів для кластеризації;

– визначення безлічі змінних, за якими будуть оцінюватися об'єкти у вибірці. При необхідності – нормалізація значень змінних;

– обчислення значень міри схожості між об'єктами;

– застосування методу кластерного аналізу для створення груп схожих об'єктів (кластерів);

– представлення результатів аналізу.

Після отримання та аналізу результатів можливе корегування обраної метрики і методу кластеризації до отримання оптимального результату.

Основні класифікації алгоритмів кластеризації:

– ієрархічні і плоскі. Ієрархічні алгоритми (також звані алгоритмами таксономії) будують не одне розбиття вибірки на непересічні кластери, а систему вкладених розбиття. Т.ч. на виході ми отримуємо дерево кластерів, коренем якого є вся вибірка, а листям – найбільш дрібні кластера. Плоскі алгоритми будують одне розбиття об'єктів на кластери;

– чіткі і нечіткі. Чіткі (або непересічні) алгоритми кожному об'єкту вибірки ставлять у відповідність номер кластера, тобто кожен об'єкт належить тільки одному кластеру. Нечіткі (або пересічні) алгоритми кожному об'єкту ставлять у відповідність набір речових значень, що показують ступінь відносини об'єкта до кластерів. Тобто кожен об'єкт відноситься до кожного кластеру з певною ймовірністю.

2.2 Об'єднання кластерів

У разі використання ієрархічних алгоритмів постає питання, як об'єднувати між собою кластера, як обчислювати «відстані» між ними. Існує кілька метрик:

– одиночний зв'язок (відстань найближчого сусіда) у цьому методі відстань між двома кластерами визначається відстанню між двома найбільш близькими об'єктами (найближчими сусідами) в різних кластерах. Результуючі кластери мають тенденцію об'єднуватися в ланцюжка;

– повна зв'язок (відстань найбільш віддалених сусідів) у цьому методі відстані між кластерами визначаються найбільшою відстанню між будь-якими двома об'єктами в різних кластерах (тобто найбільш віддаленими сусідами). Цей метод зазвичай працює дуже добре, коли об'єкти походять з окремих груп. Якщо ж кластери мають подовжену форму або їх природний тип є «цепочечную», то цей метод непридатний;

– незважене попарне середнє. У цьому методі відстань між двумя різними кластерами обчислюється як середня відстань між усіма парами об'єктів в них. Метод ефективний, коли об'єкти формують різні групи, проте він працює однаково добре і в випадках протяжних («цепочечного» типу) кластерів;

– зважене попарне середнє. Метод ідентичний методу невиваженого попарного середнього, за винятком того, що при обчисленнях

розмір відповідних кластерів (тобто число об'єктів, що містяться в них) використовується в якості вагового коефіцієнта. Тому даний метод повинен бути використаний, коли передбачаються нерівні розміри кластерів;

– незважений центроїдного метод. У цьому методі відстань між двома кластерами визначається як відстань між їх центрами тяжіння;

– зважений центроїдного метод (медіана). Цей метод ідентичний попередньому, за винятком того, що при обчисленнях використовуються ваги для обліку різниці між розмірами кластерів. Тому, якщо є або підозрюються значні відмінності в розмірах кластерів, цей метод виявляється переважно попереднього.

2.3 Огляд алгоритмів кластеризації

2.3.1 Алгоритми ієрархічної кластеризації

Алгоритми впорядкування даних зазначеного типу виходять з того, що якесь безліч об'єктів характеризується певним ступенем зв'язності. Передбачається наявність вкладених груп (кластерів різного порядку).

Алгоритми, в свою чергу, поділяються на агломеративні (об'єднавчі) і дівізівние (розділяють). За кількістю ознак іноді виділяють монотетіческіе і політетіческіе методи класифікації. Як і більшість візуальних способів подання залежностей графи швидко втрачають наочність при збільшенні числа об'єктів.

Серед алгоритмів ієрархічної кластеризації виділяються два основних типи: висхідні і низхідні алгоритми. Спадні алгоритми працюють за принципом «зверху-вниз»: на початку всі об'єкти поміщаються в один кластер, який потім розбивається на всі більш дрібні кластери. Більш поширені висхідні алгоритми, які на початку роботи поміщають кожен об'єкт в окремий кластер, а потім об'єднують кластери в усі більші, поки всі об'єкти вибірки не будуть міститися в одному кластері.

Пояснення для випадку поділу на 2 кластера зображено на рисунку 2.1.

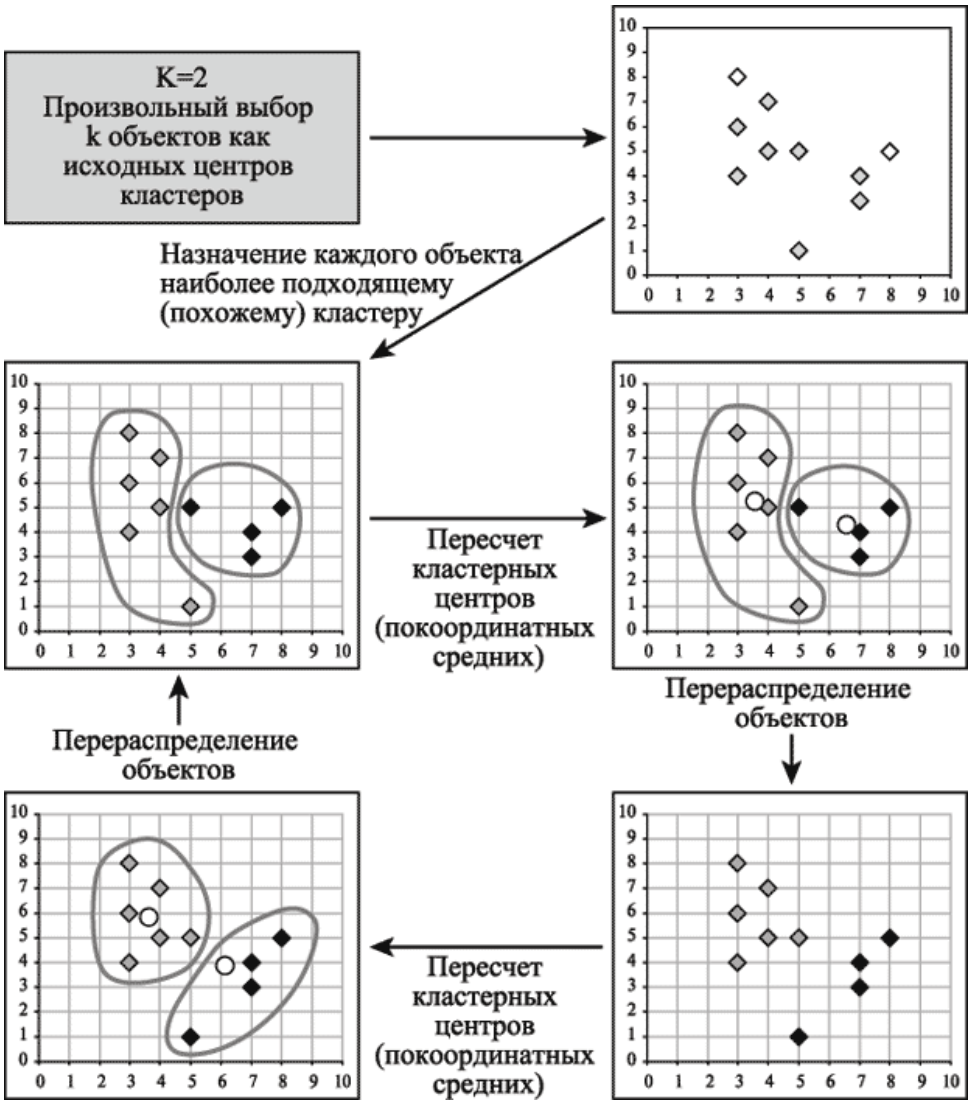


Рисунок 2.1 – Пояснення для випадку поділу на 2 кластера

Таким чином будується система вкладених розбиття. Результати таких алгоритмів зазвичай представляють у вигляді дерева – дендрограмми. Класичний приклад такого дерева – класифікація тварин і рослин.

Для обчислення відстаней між кластерами частіше все користуються двома відстанями: одиночній зв'язком або повним зв'язком (див. Огляд заходів відстаней між кластерами).

До недоліку ієрархічних алгоритмів можна віднести систему повних розбиття, яка може бути зайвої в контексті розв'язуваної задачі.

2.3.2 Алгоритми квадратичної помилки

Завдання кластеризації можна розглядати як побудова оптимального розбиття об'єктів на групи. При цьому оптимальність може бути визначена як вимога мінімізації середньоквадратичної помилки розбиття

$$e^2(X, L) = \sum_{j=1}^{k} \sum_{i=1}^{n_j} \left\| x_i^{(j)} - c_j \right\|^2,$$

де c_j — «центр масс» кластера j (точка з середніми значеннями характеристик для даного кластера).

Алгоритми квадратичної помилки відносяться до типу плоских алгоритмів. Найпоширенішим алгоритмом цієї категорії є метод k-середніх. Цей алгоритм будує задане число кластерів, розташованих якнайдалі один від одного. Робота алгоритму ділиться на кілька етапів:

– випадково вибрати k точок, які є початковими «центрами мас» кластерів;

– віднести кожен об'єкт до кластеру з найближчим «центром мас»;

– перерахувати «центримас» кластерів відповідно до їх поточним складом;

– якщо критерій зупинки алгоритму не задоволений, повернутися до п. 2.

Як критерій зупинки роботи алгоритму зазвичай вибирають мінімальне зміна среднеквадратической помилки. Так само можливо зупиняти роботу алгоритму, якщо на кроці 2 не було об'єктів, що перемістилися з кластера в кластер.

До недоліків даного алгоритму можна віднести необхідність задавати кількість кластерів для розбиття.

2.3.3 Нечіткі алгоритми

Найбільш популярним алгоритмом нечіткої кластеризації є алгоритм c-середніх (c-means). Він являє собою модифікацію методу k-середніх. Кроки роботи алгоритму:

– вибрати початкове нечітке розбиття n об'єктів на k кластерів шляхом вибору матриці приналежності U розміру *n x k*;

– використовуючи матрицю U, знайти значення критерію нечіткої помилки

$$E^2(X,U) = \sum_{i=1}^{n}\sum_{k=1}^{k} U_{ik} \left\| x_i^{(k)} - c_k \right\|^2,$$

де c_k – «центр мас» *нечіткого* кластера k

$$c_k = \sum_{i=1}^{n} U_k x_{i;}$$

– перегрупувати об'єкти з метою Зменшення цього значення крітерію нечіткої помилки;

– повертатися в п. 2 до тих пір, поки зміни матриці U не стануть незначними.

Цей алгоритм може не підійти, якщо заздалегідь невідомо число кластерів, або необхідно однозначно віднести кожен об'єкт до одного кластеру.

2.3.4 Алгоритми, засновані на теорії графів

Суть таких алгоритмів полягає в тому, що вибірка об'єктів представляється у вигляді графа G = (V, E), вершинам якого відповідають об'єкти, а ребра мають вагу, рівний «відстані» між об'єктами. Перевагою

графових алгоритмів кластеризації є наочність, відносна простота реалізації і можливість вносенія різних удосконалень, засновані на геометричних міркуваннях. Основними алгоритмами є алгоритм виділення зв'язкових компонент, алгоритм побудови мінімального покриває (остовного) дерева і алгоритм пошаровим кластеризації.

2.3.5 Алгоритм виділення зв'язкових компонент

В алгоритмі виділення зв'язкових компонент задається вхідний параметр R і в графі видаляються всі ребра, для яких «відстані» менше R. Сполученими залишаються тільки найбільш близькі пари об'єктів. Сенс алгоритму полягає в тому, щоб підібрати таке значення R, що лежить в діапазон всіх «відстаней», при якому граф «розвалиться» на кілька зв'язкових компонент. Отримані компоненти і є кластери.

Для підбору параметра R зазвичай будується гістограма розподілів попарних відстаней. У завданнях з добре вираженою кластерної структурою даних на гістограмі буде два піки – один відповідає внутрікластерним відстаням, другий – межкластерним відстані. Параметр R підбирається із зони мінімуму між цими піками. При цьому управляти кількістю кластерів за допомогою порога відстані досить важко.

2.3.6 Алгоритм мінімального покриває дерева

Алгоритм мінімального покриває дерева спочатку будує на графі мінімальне покриває дерево, а потім послідовно видаляє ребра з найбільшою вагою. На рисунку зображено мінімальне покривне дерево, отримане для дев'яти об'єктів.

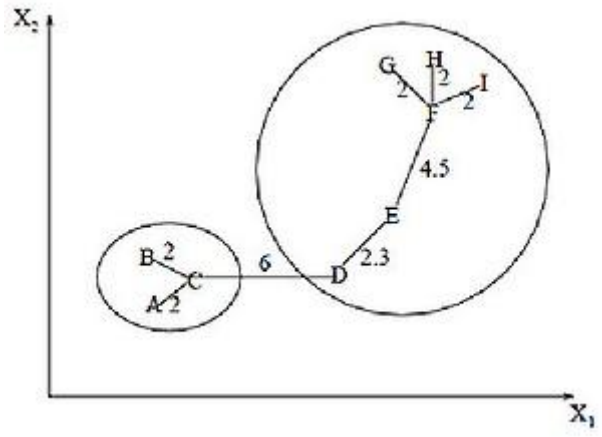


Рисунок 2.2 – Мінімальна покривне дерево

Шляхом видалення зв'язку з позначкою CD, з довжиною рівною 6 одиницям (ребро з максимальною відстанню), отримуємо два кластери: {A, B, C} і {D, E, F, G, H, I}. Другий кластер в подальшому може бути розділений ще на два кластери шляхом видалення ребра EF, яке має довжину, рівну 4,5 одиницям.

2.3.7 Пошарова кластеризация

Алгоритм пошаровим кластеризації заснований на виділенні зв'язкових компонент графа на деякому рівні відстаней між об'єктами (вершинами). Рівень відстані задається порогом відстані c. Наприклад, якщо відстань між об'єктами 0≤ρ (x, x ^ ') ≤1, то 0≤c≤1.

Алгоритм пошаровим кластеризації формує послідовність подграфов графа G, які відображають ієрархічні зв'язки між кластерами:

$$G^0 \subseteq G^1 \subseteq \cdots G^m,$$

де $G^t = (V, E^t)$ — граф на рівні c^t,

$$E^t = \{e_{ij} \in E : \rho_{ij} \leq c_t\},$$

c^t – t-ий поріг відстані;

m – кількість рівнів ієрархії;

$G^0 = (V, o)$, o – порожня множина ребер графа, що отримується при $t^0 = 1$;

$G^m = G$, тобто граф об'єктів без обмежень на відстань (довжину ребер графа), оскільки $t^m = 1$.

За допомогою зміни порогів відстані $\{c^0, ..., c^m\}$, где $0 = c^0 < c^1 < \ldots < c^m = 1$, можливо контролювати глибину ієрархії одержуваних кластерів. Таким чином, алгоритм пошаровим кластеризації здатний створювати як плоске розбиття даних, так і ієрархічне.

2.4 Висновки по другому розділу магістерської атестаційної роботи

В результаті виконання другого розділу магістерської атестаційної роботи виконана класифікація різних алгоритмов кластерного аналізу накопичених даних: алгоритми ієрархічної кластеризації, алгоритми квадратичної кластеризації, нечіткі алгоритми, алгоритми, що базуються на теорії графів, алгоритмізація виділення пов’язаних компонент, пошарова кластеризація. Наведені рекомендації щодо застосування кожного типу алгоритму.

3 МЕТОДИ ПРОГНОЗУВАННЯ СТАНУ ОБ'ЄКТУ АВТОМАТИЗАЦІЇ ЗА ДОПОМОГОЮ ЗАСОБІВ МАШИННОГО НАВЧАННЯ

3.1 Принципи використання методів машинного навчання

Процес обробки та аналізу даних отриманих за допомогою технології Інтернету речей відрізняється від традиційних математичних задач великим набором повторюваних даних. Накопичені об'єми інформації від інтелектуальних сенсорів виконують дуже важливу роль в прогнозуванні небезпечних ситуацій, як на виробництві, так і в повсякденному житті. В той же час, створення програмних інструментів для автоматизації аналізу даних та визначення граничних рішень має загальний шаблон [5]. Після створення чітких описів проблемних ситуацій, керованих отриманими даними, наступні кроки з впровадження автоматизованого засобу в інфраструктуру IoT зазвичай включають себе стадії, що показані на рисунку 3.1.

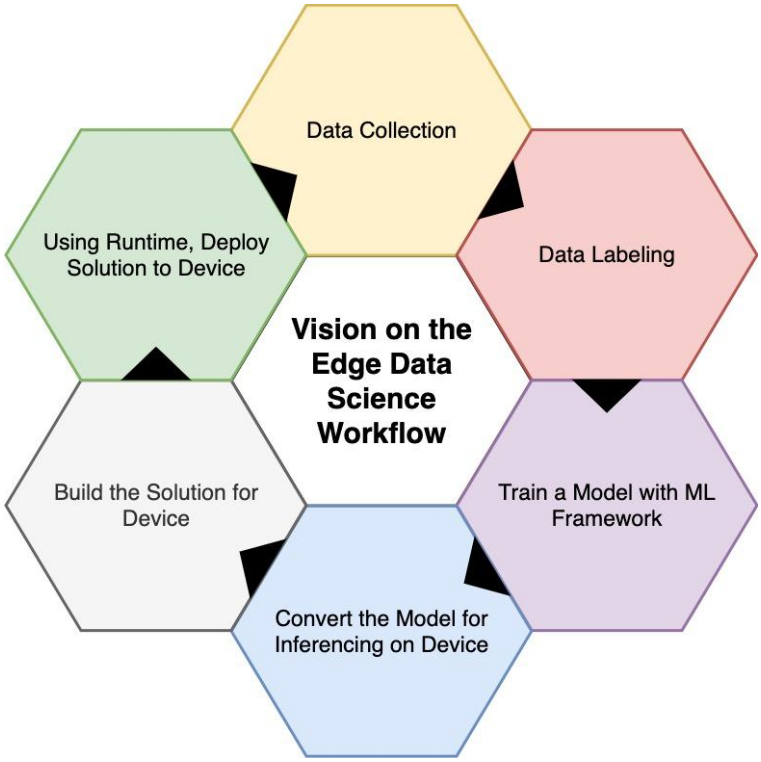


Рисунок 3.1 – Концепція процесу обробки та аналізу даних отриманих за допомогою технологією IoT

Стадія збору даних або придбання інформації може бути пошуком в оперативному режимі з поточних даних, що надходять зі встановленого на промисловому об’єкті пристрою або іншого репрезентативного джерела даних. Як правило, чим більше даних, тим краще. Крім того, чим більше варіативність, тим краще обробка.

Додавання міток до даних великого обсягу часто стає великою проблемою в проектах машинного навчання. Для проектів машинного навчання з компонентом комп'ютерного зору, таких як класифікація зображень або виявлення об'єктів, зазвичай потрібні мітки для тисяч зображень.

Маркування даних машинного навчання служить централізованим розташуванням для створення, адміністрування і моніторингу проектів додавання міток. Наприклад, за допомогою рішення Azur можна координувати дані, мітки і роботу команди, що дозволить ефективно управляти завданнями. Машинне навчання підтримує такі процеси: класифікація зображень (з декількома мітками або класами) і ідентифікація об'єктів з використанням обмежуючих прямокутників. Маркування даних відстежує хід виконання завдань додавання міток і підтримує черги незавершених завдань.

Для виконання навчання моделей, то пер за все необхідно вибрати платформу машинного навчання, наприклад, TensorFlow або PyTorch (з використанням API-інтерфейсів Python та C ++). Зазвичай це залежить від того, які ціни можуть бути доступними у відкритому або внутрішньому джерелах, а також від наявності практичних прикладів застосування машинного навчання. Наприклад, машинне навчання Azure може використовуватися для навчання моделі за допомогою будь-якої інфраструктури машинного навчання і підходу, так як вона не залежить від платформи і має прив'язки до мов Python і R, також існує багато бібліотек для популярних платформ.

Майже завжди необхідно переробляти модель для роботи з визначеною середою виконання. Перетворення моделей зазвичай включає в себе зручну оптимізацію, наприклад, швидке виведення та зменшення об'єму вже існуючих моделей. Цей шаг відрізняється для кожної платформи та середовища виконання машинного навчання. Існують загальнодоступні платформи взаємодії з відкритим кодом, такі як ONNX.

Створення рішень для пристроїв зазвичай будується на тому же типі пристроїв, що і в остаточному розгортанні, так як двійкові файли, створені для конкретної системи.

Після вибору середовища виконання, у поєднанні з вибором платформи машинного навчання, зазвичай можна розгорнути скомпільований рішення. Середовище виконання Azure IoT – це система на основі DOCKER, в якій середовища виконання машинного навчання можна розгортати як контейнери [7].

На схемі, що зображена на рисунку 3.2 показаний приклад процесу обробки та аналізу даних, де можна використовувати засоби з відкритим кодом для робочого процесу обробки та аналізу даних. Доступність і тип даних забезпечують більшу частину варіантів, включаючи обрані пристрої та обладнання.

Якщо робочий процес уже існує для спеціалістів при обробці та аналізі даних та розробниках додатків, вони можуть застосовуватись деякі інші процеси. Обрана мова реалізації проекту може допомогти визначити, який API або пакет SDK використовується для формування та навчання моделей машинного навчання. Це, у свою чергу, визначте тип моделей машинного навчання, тип пристроїв, тип IoT модуля, який потрібно використовувати. Наприклад, PyTorch має API C++ для навчання моделі, який добре працює у співпраці з API OpenCV C++.

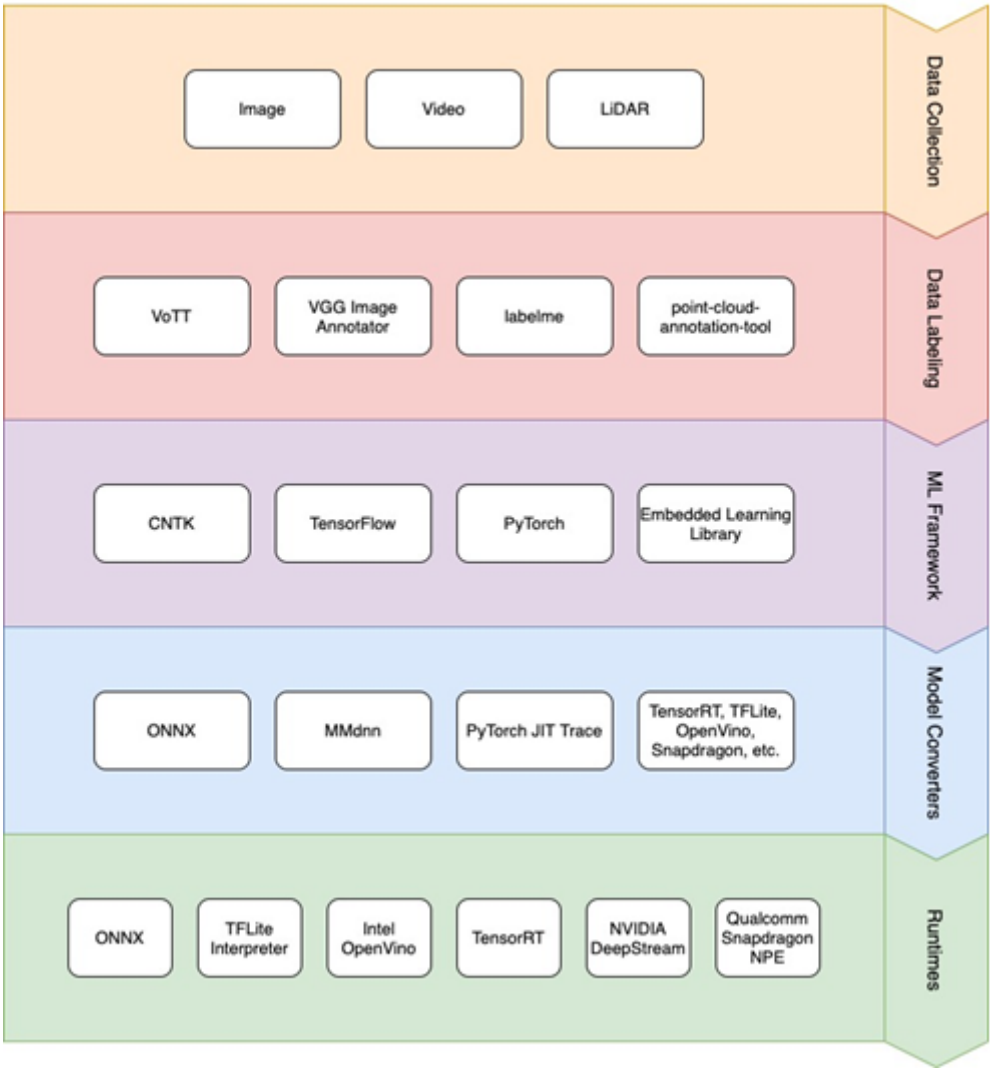


Рисунок 3.2 – Приклад процесу обробки та аналізу даних

3.2 Аналіз та вибір платформи для реалізації концепції машинного навчання.

Попередні дослідження показали, що зараз Amazon має найбільшу частку ринку IoT рішень. В той же час платформа Azure має багато інструментів для виконання задач пов’язаних з розробкою та впровадженням традиційних моделей обробки даних. Платформи від Google мають досить велику стабільність. Для додатків IoT Google має великий потенціал для ще більшого нарощування ресурсів в даній сфері, що дає нам привід вважати, що він і надалі буде конкурентним проти Amazon і Microsoft.

В таблиці 3.1 наведені порівняльні характеристики кожної з трьох платформ.

Таблиця 3.1 – Порівнювання IoT платформ

Назва платформи	Переваги	Недоліки
1	2	3
AWS	– домінування на хмарному ринку з найбільшою часткою більше 10 років; – масив функцій і хмарних сервісів не тільки вже є широкими, але й продовжує зростати; – є готові до використання функції "Готові підприємства"; – широка номенклатура документації про послуги та випадки використання; – можливість використання FreeRTOS робить все швидше і простіше.	– комплексне використання послуг та функцій; – переважна більшість послуг є платною.
Google	– міцна теоретична документація на модулі; – сумісний з усіма браузерами; – зручна панель інструментів. – стабільність підключення	– слабка практична документація для кінцевого користувача через відсутність прикладів; – відсутність послуг IoT на відміну від AWS або Azure.

Продовження таблиці 3.1

1	2	3
		– більш складні засоби аутентифікації вимагають від користувача генерації загальнодоступних і приватних ключів окремо.
Azure	– краща документація для кінцевого користувача та практичні програми; – перевага збирання проектів та користувачів, які вже використовують Office, SQL Server, SharePoint, .Net тощо; – потужний при розміщенні додатків у хмарі	– впровадження візуалізації даних у реальному часі є складним; – несумісність із веб-переглядачем Safari; – комплексний інтерфейс користувача; – запуск нової служби має більше часу створення, ніж інші постачальники; – деякі проблеми з підключенням служби PowerBI

Таким чином, виходячи з наведеного аналізу обираємо для реалізації нашого завдання платформу від AWS.

Враховуючи той факт, що нещодавно Microsoft реалізувала підтримку машинного навчання безпосередньо з програм, що написані мовою програмування C# прямо з середовища Visual Studio обираємо для практичного застосування інструмент, який має назву в Microsoft ML.NET.

Поточна версія ML.NET – це ML.NET preview 1.4, випущена у вересні 2019 року [6]. Machine Learning.Net – це рішення з відкритим вихідним кодом, що може працювати на різних платформах.

Для розробки машинного навчання з використанням ML.NET можна використовувати код на C# або F#. ML.NET має відкритий вихідний код і може розроблятися і працювати на Windows, Linux і macOS. Можно розробляти власні моделі машинного навчання, використовуючи ML.NET для консолі, персонального комп'ютера, Інтернету, мобільних пристроїв та для IOT. M L.NET також підтримує розширення та роботу з TensorFlow, Accord.NET і CNTK. Останній випускма ML.NET також підтримує завантаження і навчання даних з реляційних баз даних, таких як SQL Server, Oracle, MySQL і т.д. Також була створена остання версія ML.NET для розробки простого інтерфейсу користувача з використанням AutoML.

В даний час Microsoft випустила попередню версію ML.NET, і Microsoft продовжує додавати більше функцій в інфраструктуру ML.NET, поточної версії ML.NET є ML.NET 1.4.

Концепцією ML.NET включає наступні етапи для розробки програм машинного навчання:

– завантаження даних: для точного прогнозування результатів необхідно надати багато даних для навчання моделі. У ML.Net ми можемо надавати дані як для навчання, так і для перевірки по тексту (CSV / TSV, реляційна база даних (підтримуються SQL Server, Oracle, MySQL і т.д.)), двійкові файли, IEnumerable і т.д. [8];

– навчання: потрібно вибрати правильний алгоритм для навчання моделі в залежності від поточних потреб, для навчання і прогнозування результатів;

– оцінка: необхідно обрати конкретний тип машинного навчання обраної моделі навчання і прогнозування. Якщо потрібно працювати з певним сегментом даних, то можна обрати модель кластеризації, якщо потрібно знайти ціну прогнозування певних подій, можна обрати регресію і, якщо потрібно виконати аналіз отриманих даних, можна обрати модель класифікації;

– прогнозування результатів: на основі даних про навчання і тестування з навченої моделлю остаточний прогноз буде відображатися за допомогою програми ML.NET. Навчена модель буде збережена в двійковому форматі, який також може бути інтегрована з іншими додатками .NET .

На рисунку 3.3 показані зазначені вище етапи розробки програм машинного навчання.

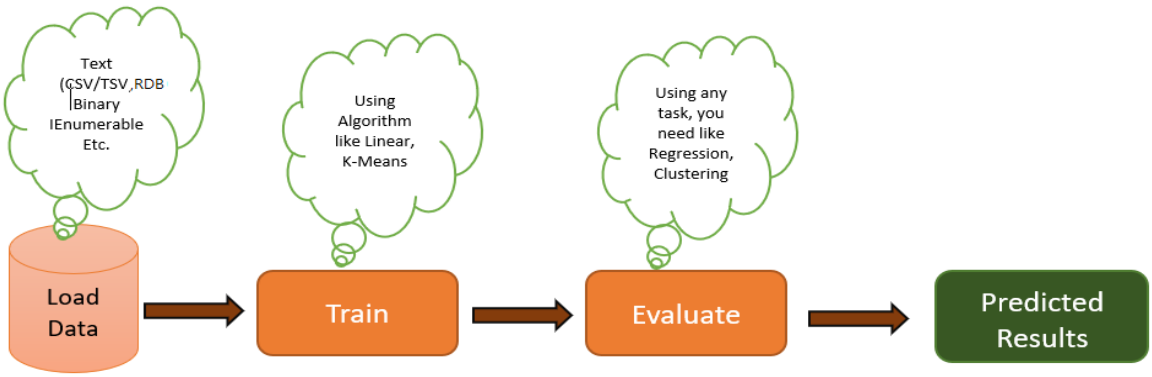


Рисунок 3.3 – Етапи розробки програм машинного навчання

Інтеграція підтримки машинного навчання в програмний проект виконується за допомогою засобу NuGet, що є в складі інтегрованого середовища Microsoft Visual Studio [7].

3.3 Висновки по третьому розділу магістерської атестаційної роботи

В результаті виконання третього розділу магістерської атестаційної роботи наведені принципи використання методів машинного навчання. Наведені основні стадії підготовки та виконання аналізу даних, тренування системи та виконання прогнозу.

Проведено аналіз та вибір платформи для реалізації концепції машинного навчання та обрали платформу від AWS. Для практичного застосування обираємо інструмент, який має назву в Microsoft ML.NET. Він дозволяє виконувати виклик функцій для машинного навчання

безпосередньо з програми що написані мовою програмування C# прямо з середовища Visual Studio.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Опис розробленої структури серверу IoT

Система обміну повідомленнями в мережі інтелектуальних пристроїв складається з серверу, датчиків та виконавчих пристроїв. Вони під’єднаються до загальної мережі, яку створює сервер та стають доступні системі керування інтелектуальними пристроями [9].

Сервер – це програмно-апаратний комплекс, який повинен мати два мережевих інтерфейсу, серед яких один обов’язково повинен мати підтримку внутрішньої мережі [10].

Сервер одночасно підключений до двох мереж: внутрішня мережа локальних пристроїв та зовнішня мережа системи контролю (схему показано на рисунку 4.1). Сервер керує мережу з шифруванням WPA2, до якої можуть долучитися тільки ті пристрої, які знають ім’я мережі та її пароль. Іншим інтерфейсом він підключається до локальної мережі користувача, з якої можна зайти на сторінку серверу, дивитися історію показань з датчиків та керувати пристроями.

Рисунок 4.1 – Структурна схема розподілення мереж

У якості апаратної частини може виступати будь-яка ЕОМ під керуванням ОС GNU/Linux.

Датчики та інтелектуальні пристрої – це окремі апаратні пристрої, які базуються на мікроконтролерах із вбудованим приємопередавачем.

На віддалених блоках керування виконується програма з веб-кліентом, що обслуговує запити керуючого сервера. У залежності від команд, що

надходять, кінцеві пристрої виконують ту чи іншу роботу (увімкнути/вимкнути світло, змінити режим роботи тощо). Датчики в основному знаходяться у режимі сну, тому що в них немає стаціонарного живлення та треба економити енергію батареї. Датчики можуть бути виконані не тільки як самостійний пристрій, а й входити у склад контролерів виконавчих пристроїв.

4.2 Опис алгоритму роботи серверу

Пристрій посилає на сервер HTTP-запит:

GET /192.168.59.3/device_status?LOG= 1111&PAS=2222&BT=127 HTTP/1.1

HOST: t.lazysmart.com,

де:

– GET – це метод запиту;

– device_status – контролер, який буде виконаний на сервері;

– 192.168.59.3 – IP-адреса сервера, на якому знаходиться програма керування.

Після знаку «?» ідуть параметри, які пристрій передає на сервер:

– LOG і PAS – логін і пароль для ідентифікації пристрою;

– BT – це, наприклад поточна температура.

Сервер, отримуючи такий запит запускає контролер «device_status». Цей контролер робить наступне:

– бере значення параметрів LOG і PAS і робить запит до бази даних в пошуках пристрою, зареєстрованого з такими логіном і паролем;

– якщо пристрій пройшов авторизацію, скрипт бере значення параметра BT (поточна температура, передана пристроєм) і заносить її в БД;

– потім скрипт отримує з БД команду реле для цього пристрою і формує відповідь для пристрою: «COMMAND 1 EOC», де 1 або 0 – поточна команда реле.

Взаємодія передавального пристрою з сервером показана на рисунку 4.2.

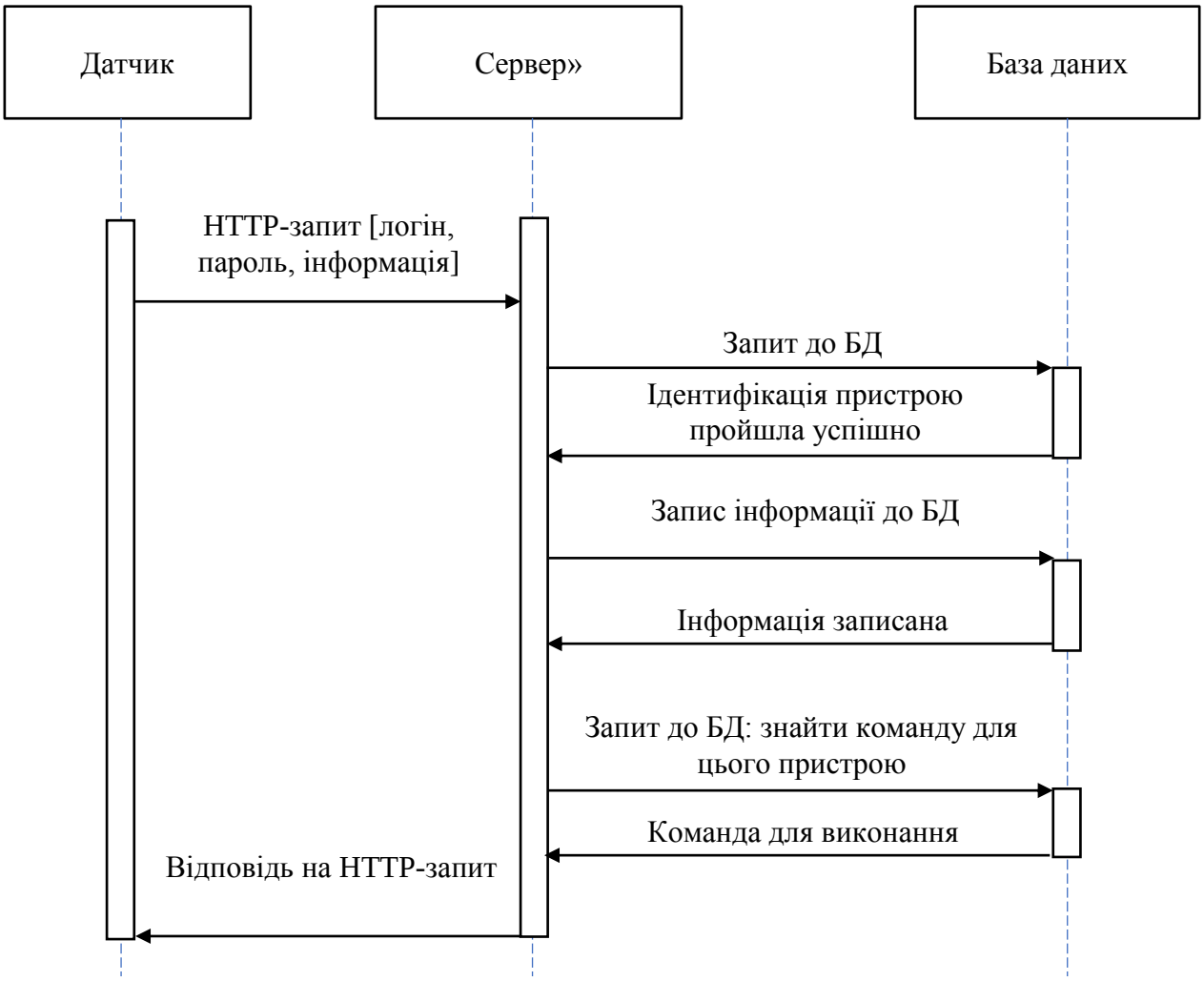


Рисунок 4.2 – Взаємодія датчика з сервером

Пристрої відсилають статистику до серверу та очікують від нього зворотних команд. В режимі очікування, сервер очікує запити від датчиків, щоб прочитати їх стан та зберегти у базі даних. Коли датчик «прокидається» і звертається до сервера, останній розбирає отримане повідомлення, декодує

його, та записує отриману інформацію до БД. На рисунку 4.3 зображено алгоритм прийому даних від датчиків.

Рисунок 4.3 – Алгоритм прийому даних з датчиків

На рисунку 4.4 показано алгоритм роботи серверу в режимі передачі команди від користувача до пристрою.

Рисунок 4.4 – Алгоритм передачі команди від користувача до пристрою

Коли оператор звертається до сервера, він може побачити історію показників датчиків за будь-який період часу. Коли користувачу треба керувати якимось пристроєм, то сервер приймає та оброблює цей запит, а потім дає команду обраному пристрою.

Таким чином, програмна частина серверу складається з інтерфейсу та контролерів. За допомогою інтерфейсу можна отримати інформацію про стан датчиків та інших пристроїв, а також керувати ними. Інтерфейс також дозволяє змінювати ці параметри [11]. Для того, щоб добавити до системи будь-який пристрій, треба увійти до системи адміністрування та запустити спеціальний інструмент для роботи з базою даних, наприклад PgAdmin.

4.3 Розробка структури тестової бази даних

Для моделювання роботи сервера IoT у хмарному сервісі AWS була створена тестова віртуальна платформа, що базується на операційній системі Linux, серверу баз даних PostgreSQL.

Зберігання отриманих від датчиків даних буде виконуватись в три таблиці:

– довідник IoT пристроїв (data_IOT);

– довідник місць розташування (data_Place);

– довідник розташування пристроїв у місцях за їх призначенням (data_Layout);

– журнал подій (data_ML).

Структура сутності data_Sensor наведена в таблиці 4.1.

Таблиця 4.1 – Довідник приміщень (data_Place)

Назва	Тип поля	Розмір поля	Первинний ключ
ID_Place	Serial	–	PK
Name_Place	Text	48	–
Notice	Text	256	–

В даній таблиці зберігаються назва місця розташування (наприклад, «Сховище автоматизоване») та примітка, що додатково характеризує місце розташування.

Для зберігання даних про IoT пристрої була запропонована наступна структура таблиці (див. табл. 4.2).

Таблиця 4.2 – Довідник IoT пристроїв

Назва	Тип поля	Розмір поля	Первинний ключ
ID_IOT	Serial	–	PK
Model	Text	48	–
SerialNumber	Text	256	–
Notice	Text	256	

В полі Model зберігається назва моделі пристрою; в полі SerialNumber – записується серійний номер (ідентифікатор) пристрою, що береться з його паспорту; в полі Notice записується примітка – додаткові дані про призначення та характеристики пристрою.

Для зберігання інформації про розташування IoT-пристроїв у місцях за їх призначенням була запропонована наступна структура таблиці (див. табл. 4.3).

Таблиця 4.3 – Інформація про розташування IoT-пристроїв у місцях за їх призначенням

Назва	Тип поля	Розмір поля	Первинний ключ
ID_Layout	Serial	–	PK
ID_Place	Integer	48	–
ID_IOT	Integer	256	–
Notice	Text	256	

В полі ID_IOT зберігається посилання на довідник IOT-пристроїв; в полі ID_Place – посилання на довідник data_Place для визначення назви місця розташування. В полі Notice записується примітка – додаткові дані про призначення та характеристики пристрою.

Для зберігання отриманих від пристроїв-IoT даних була створена таблиця data_ML. Структура даної таблиці наведена в табл. 4.4.

Таблиця 4.4 – Журнал подій з пристроями IoT

Назва	Тип поля	Розмір поля	Первинний ключ
ID_ML	Serial	–	PK
ID_IOT	Serial	48	–
Value	Text	256	–
Date_ML	Date	256	
Time_ML	Time		
NumIzm	Integer		

В полі ID_IOT зберігається посилання на довідник IOT-пристроїв; в полі Value – записуються, що були отримані від IOT-пристроїв; в полі Date_Time записується час отримання даних від пристрою; в полі NumIzm – записується порядковий номер виміру.

За допомогою інтегрованого середовища DBeaver створюємо всі зазначені вище таблиці. Приклад створення таблиці data_ML для зберігання отриманих від пристроїв-IoT даних показано на рисунку 4.5.

В результаті операції створення таблиці автоматично був сформований SQL-запит:

```
CREATE TABLE public.data_ml (
    id_ml serial NOT NULL DEFAULT nextval('data_ml_id_ml_seq'::regclass),
```

```
    id_iot int4 NULL,
    value text NULL,
    date_ml date NULL,
    time_ml time(0) NULL,
    numizm int4 NULL
).
```

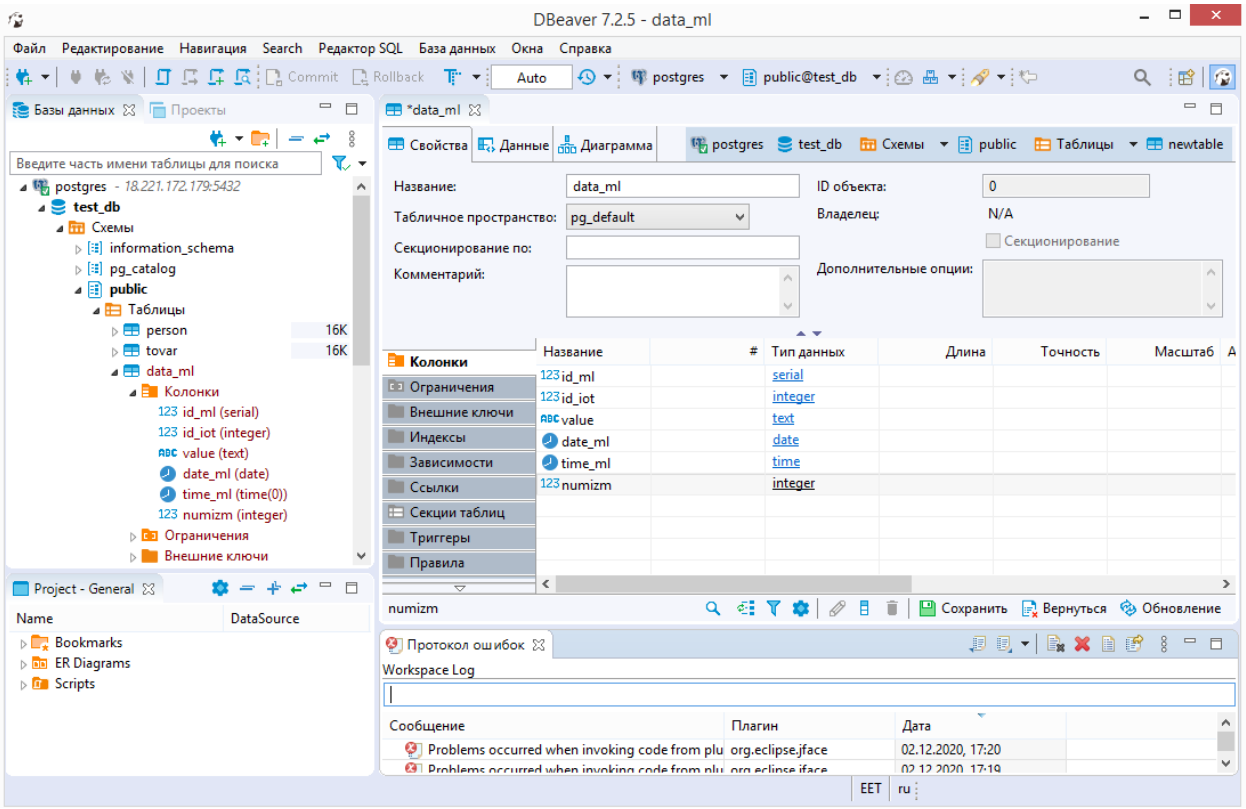


Рисунок 4.5 – Приклад створення таблиці data_ML для зберігання отриманих від пристроїв-IoT даних

4.4 Розробка програми для моделювання процесу аналізу даних, отриманих за допомогою технологією IoT.

Експериментальні дослідження будемо виконувати за допомогою програмного засобу, що спеціально розроблено для виконання прогнозування поведінки дослідного об’єкту за допомогою методу машинного навчання, використовуючи дані, отримані від IoT-пристроїв.

Програмний засіб має три робочих екрани:

– підготовка даних;

– навчання системи;

– прогнозування.

Інтерфейс програми дозволяє проводити аналіз будь-яких даних, що були отримані від різних інтелектуальних пристроїв. Для цього на першому екрані, що показано на рисунку 4.6 передбачене поле для вводу рядка підключення до бази даних.

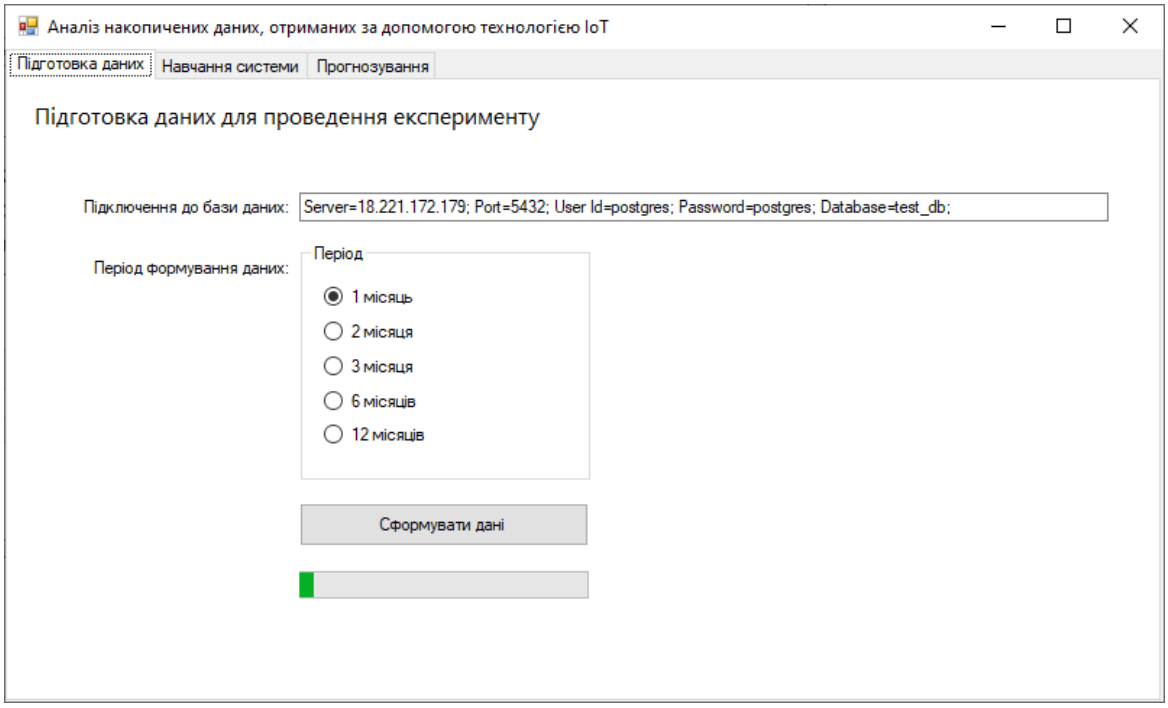


Рисунок 4.6 – Інтерфейс робочого екрану «Підготовка даних»

Для підключення до серверу, що було розроблено в попередньому розділу, використовується такий рядок: Server=18.221.172.179; Port=5432; User Id=postgres; Password=postgres; Database=test_db.

Для виконання досліджень будемо використовувати віртуальні дані, згенеровані за допомогою наступної функції:

```
if (radioButton1.Checked) {
int max = 30;
progressBar1.Minimum = 0;
```

```
progressBar1.Maximum = max + 5;
for (int i = 0; i < 30; i++) {
val = rand.Next(start_val, end_val);
sql = String.Format("Insert into data_ML (ID_IOT, Value, NumIzm, Time_ML) Values ('{0}', '{1}', '{2}', '{3}');", "12", val, i, tm);
DataBase.Exec_SQL(sql);
val = val * val;
sql = String.Format("Insert into data_ML (ID_IOT, Value, NumIzm, Time_ML) Values ('{0}', '{1}', '{2}', '{3}');", "24", val, i, tm);
DataBase.Exec_SQL(sql);
if (rang == 24) {
start_val++;
end_val++;
rang = 0;
tm = 0;   }
rang++;
tm++;
progressBar1.Value++;
Application.DoEvents();   }   }
```

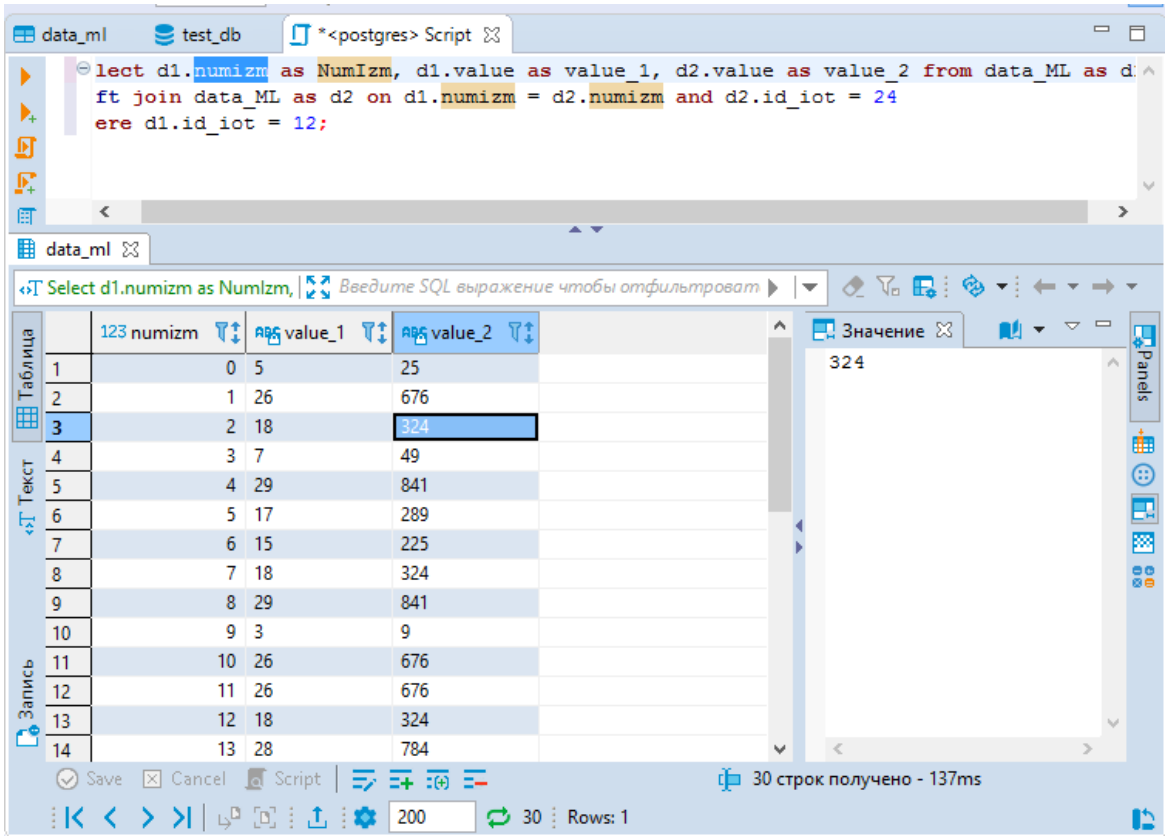
За допомогою даної функції генерується випадкові дані для двох датчиків: датчика, що підраховує кількість людей в приміщенні та датчика якості повітря а приміщенні, що вимірює рівень CO_2.

За допомогою перемикача можна задавати різний період для генерації тестових значень. Ці дані надалі будуть використовуватись для навчання алгоритму прогнозування різних поведінок досліджуваного об’єкту.

На рисунку 4.7 показано приклад даних, що були отримані на етапі підготовки. Для виводу інформації про стан двох датчиків, які збирали її протягом роботи, було написано такий запит до бази даних:

Select d1.numizm as NumIzm, d1.value as value_1, d2.value as value_2 from data_ML as d1 Left join data_ML as d2 on d1.numizm = d2.numizm and d2.id_iot = 24 where d1.id_iot = 12.

В даному запиті датчик, що підраховує кількість людей, має номер «12», а датчик якості повітря а приміщенні має номер «24».



	123 numizm	value_1	value_2
1	0	5	25
2	1	26	676
3	2	18	324
4	3	7	49
5	4	29	841
6	5	17	289
7	6	15	225
8	7	18	324
9	8	29	841
10	9	3	9
11	10	26	676
12	11	26	676
13	12	18	324
14	13	28	784

Рисунок 4.7 – Приклад даних, що були отримані на етапі підготовки

На наступному етапі виконується аналіз даних, та навчання алгоритму прогнозування. Приклад програмного інтерфейсу показано на рисунку 4.8.

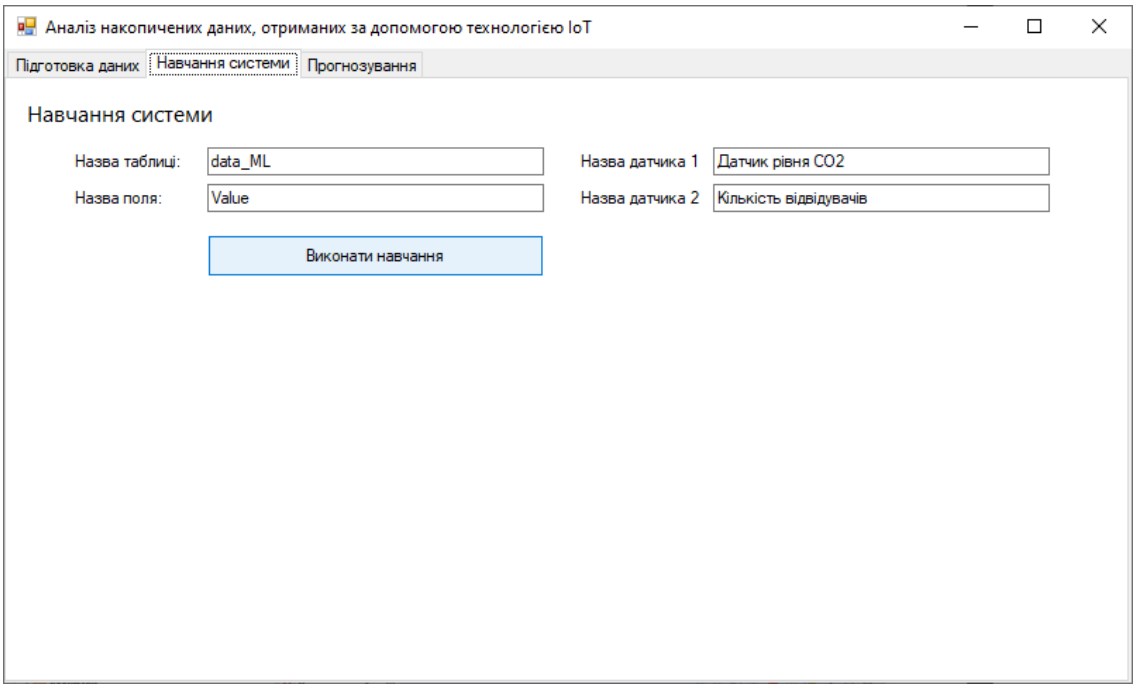


Рисунок 4.8 – Приклад програмного інтерфейсу «Навчання системи»

Для роботи з алгоритмом машинного навчання необхідно встановити бібліотеку від Microsoft. На рисунку 4.9 показано приклад додавання бібліотеки ML.NET до проекту.

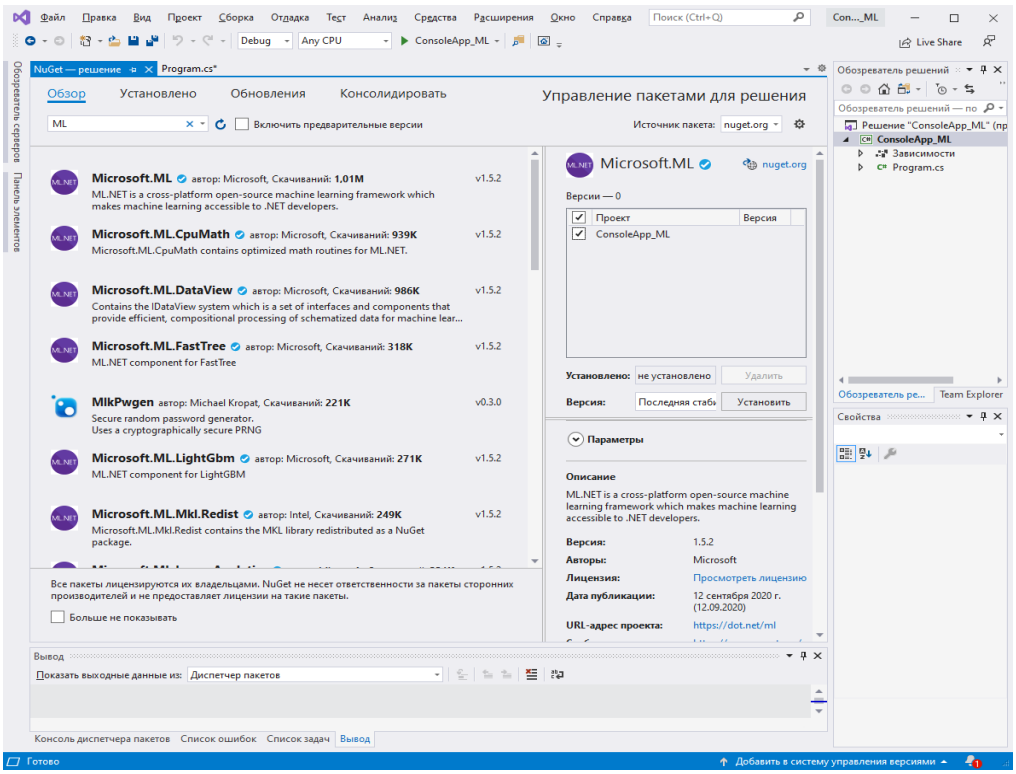


Рисунок 4.9 – Приклад додавання бібліотеки ML.NET до проекту

Для проведення аналізу та навчання необхідно обрати таблицю з даними (data_ML) та вказати назву поля, де зберігаються отримані дані (Value). Також треба обрати два типи датчиків, дані з яких потрібно аналізувати.

В даному режимі виконується функція відображення отриманих з серверу даних в програмі:

```
chart1.Series.Add("CO2");
string connString = npgsqlConStr;
NpgsqlConnection conn = new NpgsqlConnection(connString);
conn.Open();
NpgsqlDataAdapter da = new NpgsqlDataAdapter();
String sql = "Select d1.numizm as NumIzm, d1.value as value_1, d2.value as value_2 from data_ML as d1 " +
"left join data_ML as d2 on d1.numizm = d2.numizm and d2.id_iot = 24 " +
"where d1.id_iot = 12;";
NpgsqlCommand comm = new NpgsqlCommand(sql, conn);
da.SelectCommand = comm;
DataSet ds = new DataSet();
da.Fill(ds);
chart1.DataSource = ds.Tables[0];
chart1.Series[0].XValueMember = "NumIzm";
chart1.Series[0].YValueMembers = "Value_1";
chart1.Series[1].XValueMember = "NumIzm";
chart1.Series[1].YValueMembers = "Value_2";
chart1.DataBind();
conn.Close();
```

Після виконання завантаження даних вони будуть відображені на графіках для візуальної оцінки інформації. На рисунку 4.10 показано приклад інтерфейсу програмі для відображення отриманих даних.

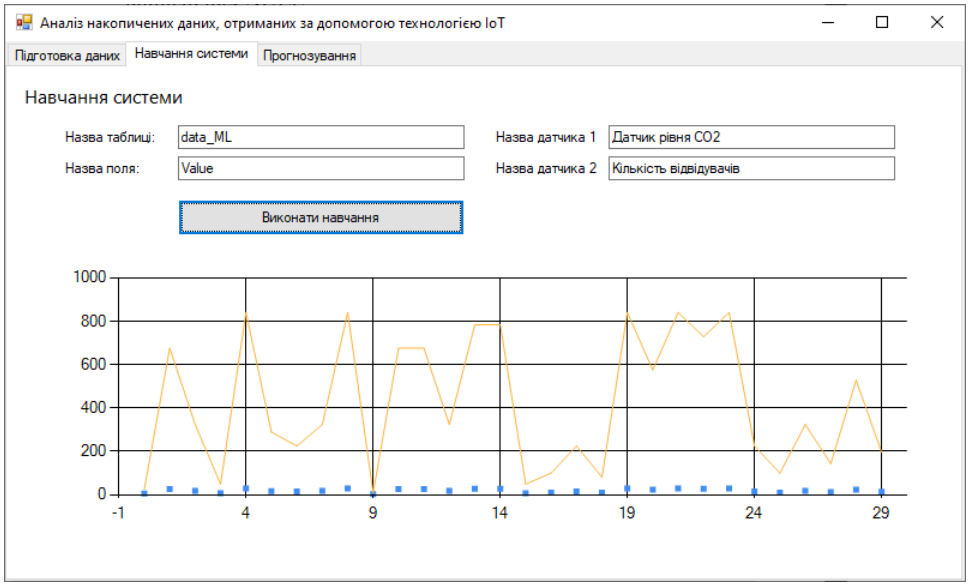


Рисунок 4.10 – Приклад інтерфейсу програмі для відображення отриманих даних

По осі X відображаються номера вимірювань, по осі Y отримані значення. Точками показані дані з першого датчика, що підраховує кількість людей. Лінією показано графік зміни значення якості повітря.

Для виконання прогнозування та аналізу поведінки автоматизованої системи (наприклад, регулювання мікроклімату) в наступному робочому вікні можна задавати потрібні значення кількості людей в приміщенні та аналізувати прогнозований стан повітря.

На рисунку 4.11 показано приклад інтерфейсу програму в режимі прогнозування.

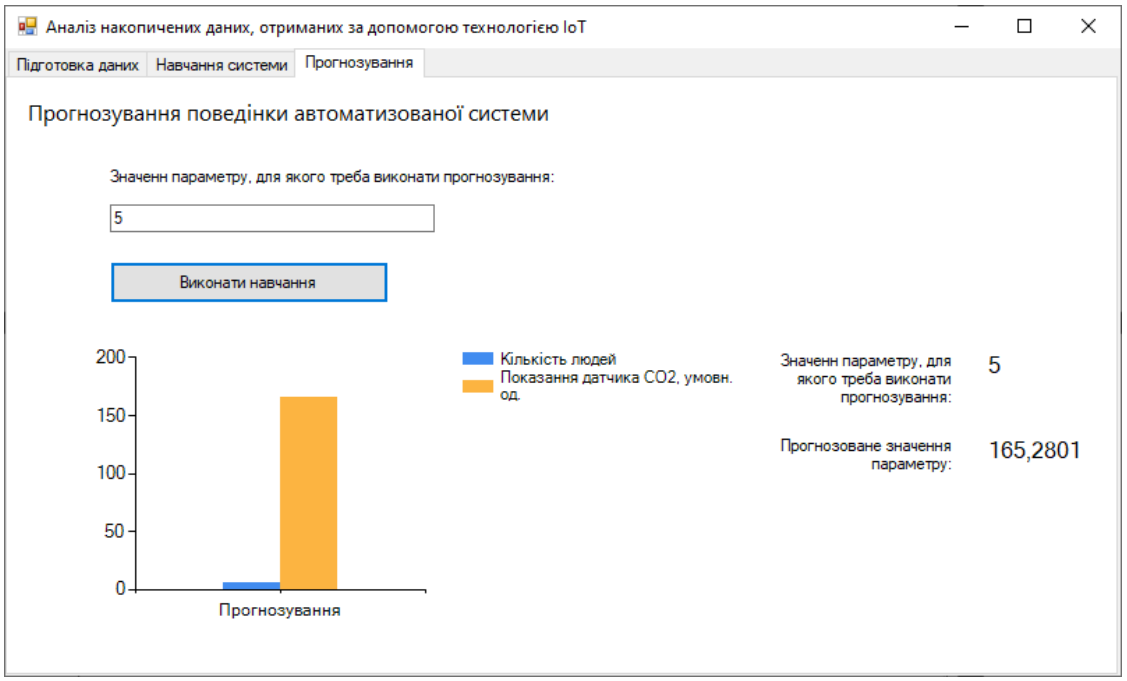


Рисунок 4.11 – Приклад інтерфейсу програму в режимі прогнозування

На графіку показано прогнозоване значення датчику CO_2 при заданій кількості людей в приміщенні.

Для виконання функції прогнозування була розроблена наступна функція:

```
String sql = "Select d1.numizm as NumIzm, d1.value as value_1, d2.value as value_2 from data_ML as d1 " +
"left join data_ML as d2 on d1.numizm = d2.numizm and d2.id_iot = 24 " +
"where d1.id_iot = 12;";
List<System.Object> rows_IOT = (List<System.Object>)DataBase.LoadList("DataAnaliz.data_ML", sql);
IOT_Data[] iotData = new IOT_Data[rows_IOT.Count];
int i = 0;
foreach (System.Object ob in rows_IOT) {
iotData[i] = new IOT_Data() { Value_1 = float.Parse((ob as DataAnaliz.data_ML).Value_1), Value_2 = float.Parse((ob as DataAnaliz.data_ML).Value_2) };
i++; }
```

```
MLContext mlContext = new MLContext();
IDataView trainingData = mlContext.Data.LoadFromEnumerable(iotData);
var pipeline = mlContext.Transforms.Concatenate("Features", new[] { "Value_1" }) .Append(mlContext.Regression.Trainers.Sdca (labelColumnName: "Value_2", maximumNumberOfIterations: 100));
var model = pipeline.Fit(trainingData);
float val = float.Parse(textBox7.Text);
var pos = new IOT_Data() { Value_1 = val };
var value = mlContext.Model.CreatePredictionEngine<IOT_Data, Prediction>(model).Predict(pos).
```

Відображення отриманих даних виконується за допомогою наступної функції:

```
chart2.Series.Clear();
var original = chart2.Series.Add("Кількість людей");
var modified = chart2.Series.Add("Показання датчика CO2, умовн. од.");
chart2.ChartAreas[0].AxisY.MajorGrid.Enabled = false;
chart2.ChartAreas[0].AxisY.MinorGrid.Enabled = false;
chart2.ChartAreas[0].AxisX.MajorGrid.Enabled = false;
chart2.ChartAreas[0].AxisX.MinorGrid.Enabled = false;
original.Points.AddXY("Прогнозування", val);
modified.Points.AddXY("Прогнозування", value.Value);
label11.Text = val.ToString();
label13.Text = value.Value.ToString().
```

4.5 Аналіз результатів дослідження

В результаті експериментального дослідження були отримані тестові дані від віртуальних пристроїв IoT, що були записані до серверу БД та за допомогою обраного методу аналізу накопичених даних виконане

прогнозування та аналізу поведінки автоматизованої системи. Отримані дані зведені в таблицю 4.5.

Таблиця 4.5 – Дані від датчиків IoT, що отримані в результаті експерименту

№ виміру	Датчик 1	Датчик 2	№ виміру	Датчик 1	Датчик 2	№ виміру	Датчик 1	Датчик 2
1	5	25	11	26	676	21	24	576
2	26	676	12	26	676	22	29	841
3	18	324	13	18	324	23	27	729
4	7	49	14	28	784	24	29	841
5	29	841	15	28	784	25	15	225
6	17	289	16	7	49	26	10	100
7	15	225	17	10	100	27	18	324
8	18	324	18	15	225	28	12	144
9	29	841	19	9	81	29	23	529
10	3	9	20	29	841	30	14	196

Для перевірки результатів роботи розробленого засобу автоматизації аналізу накопичених даних використаємо хмарний ресурс Planetcalc. Даний ресурс по введених даних будує кілька моделей регресії: лінійну, квадратичну, кубічну, ступеневу, логарифмічну, гіперболічного, показову, експонентну. Результати можна порівняти між собою по кореляції, середньої помилку апроксимації і наочно на графіку.

Дані, що наведені в таблиці 4.5, були введені у програму Planetcalc для розрахунку та в результаті її роботи були отримані наступні значення апроксимації для різних моделей. Результат аналізу наведено на рисунку 4.12.

Рисунок 4.12 – Результат аналізу отриманих даних

В таблиці 4.6 Наведені числові дані аналізу отриманих результатів за якими було побудовано графіки зміни вихідних параметрів функцій, що наведені на рисунку 4.13.

Таблиця 4.6 – Числові дані аналізу отриманих результатів

i	x	y	Лінійна регресія	Квадратична регресія	Кубічна регресія	Степенна регресія	Показникова регресія	Логарифмічна регресія	Гіперболічна регресія	Експоненціальна регресія
0.4			-234.1484	0.1600	0.1600	0.1600	22.5392	-1283.4532	-7662.9619	22.5392

Продовження таблиці 4.6

i	x	y	Лінійна регресія	Квадратична регресія	Кубічна регресія	Степенна регресія	Показникова регресія	Логарифмічна регресія	Гіперболічна регресія	Експоненціальна регресія
1	3	9	-141.4883	9.0000	9.0000	9	31.9988	-358.4216	-439.9026	31.9988
2	5	25	-70.2113	25.0000	25.0000	25	41.8993	-123.9042	4.5934	41.8993
3	7	49	1.0657	49.0000	49.0000	49	54.8631	30.5685	195.0916	54.8631
4	7	49	1.0657	49.0000	49.0000	49	54.8631	30.5685	195.0916	54.8631
5	9	81	72.3427	81.0000	81.0000	81	71.8379	145.9456	300.9240	71.8379
6	10	100	107.9812	100.0000	100.0000	100	82.2035	194.3161	337.9653	82.2035
7	10	100	107.9812	100.0000	100.0000	100	82.2035	194.3161	337.9653	82.2035
8	12	144	179.2582	144.0000	144.0000	144	107.6374	278.0190	393.5273	107.6374
9	14	196	250.5352	196.0000	196.0000	196	140.9407	348.7887	433.2145	140.9407

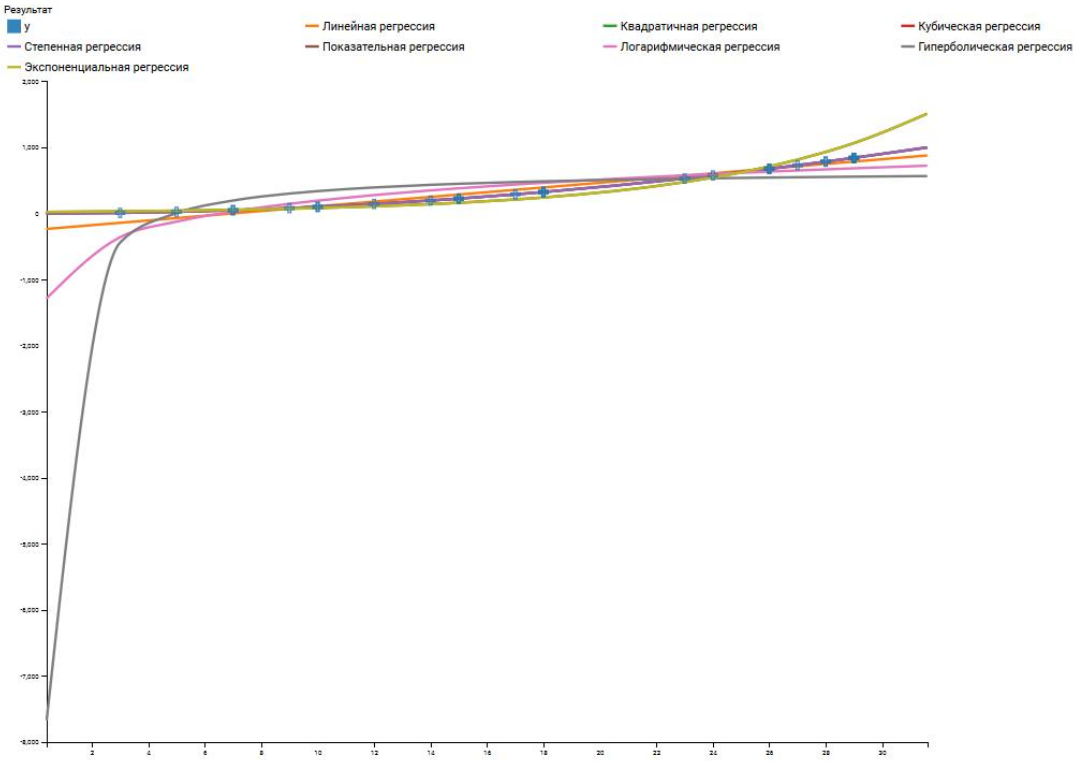


Рисунок 4.13 – Графіки зміни вихідних параметрів функцій

З наведеного рисунку можна бачити, що найбільший рівень кореляції має функція квадратичної регресії. саме така функція була використана на етапі експериментальних досліджень при генерації тестових даних (Рисунок 4.13).

```
for (int i = 0; i < 30; i++)
{
    val = rand.Next(start_val, end_val);
    sql = String.Format("Insert into data_ML (ID_IOT, Value, NumIzm, Time_ML) " +
        "Values ('{0}', '{1}', '{2}', '{3}');", "12", val, i, tm);
    DataBase.Exec_SQL(sql);

    val = val * val;
    sql = String.Format("Insert into data_ML (ID_IOT, Value, NumIzm, Time_ML) " +
        "Values ('{0}', '{1}', '{2}', '{3}');", "24", val, i, tm);
    DataBase.Exec_SQL(sql);

    if (rang == 24)
    {
        start_val++;
        end_val++;
        rang = 0;
        tm = 0;
    }
    rang++;
    tm++;
    progressBar1.Value++;
    Application.DoEvents();
}
```

Рисунок 4.12 – Генерація тестових даних

Таким чином, аналіз отриманих результатів експериментальних досліджень показав працездатність запропонованого методу аналізу накопичених даних, отриманих за допомогою технологією IoT з використанням технології машинного навчання від Microsoft ML.NET.

4.6 Розрахунок освітленості в приміщенні при виконанні наукових досліджень

За ступенем небезпеки ураження електричним струмом згідно з ПУЕ – 2011 приміщення належить до класу приміщень без підвищеної небезпеки ураження електричним струмом. Умови, які створюють підвищену і особливу небезпеку (підвищена вологість, струмопровідний пил,

струмопровідні підлоги, можливість одночасного дотику до заземлених металоконструкцій будівлі і металевих поверхонь електроприладів), відсутні [12].

З метою зниження небезпеки ураження людини електричним струмом проектом передбачається використання таких технічних засобів захисту:

– необхідно проводити контроль ізоляції відповідно до вимог ПУЕ -2011. Контроль проводити між нульовим і фазним провідниками і між фазами. Опір ізоляції не менше 500 кОм на фазу. Контроль проводити не рідше 1 разу на рік при відключеному електроживленні;

– в приміщенні використовується система живлючих провідників, трифазна, чотирипровідна з глухо заземленою нейтраллю напругою до 1000 В, тому, згідно з НПАОП 40.1-1.32-01, використовується система заземлення TN-C-S типу. Всі корпуси ПЕОМ з'єднані з глухо заземленою нейтраллю джерела живлення за допомогою нульового захисного провідника. Автомат захисту вибирається за струмом короткого замикання, час відключення 0,2 с. Додатково застосовується повторне заземлення нульового проводу з метою зниження потенціалу корпусів і напруги дотику у випадках обриву нульового проводу.

Роботи в лабораторії відносяться до робіт категорії 1а – легка фізична робота, яка виконується сидячи.

Оптимальні норми мікроклімату згідно з ДСН 3.3.6.042-99: в холодний період року – температура 22-24 °С; відносна вологість 40-60 %; швидкість руху повітря не більше 0,1 м/с. У теплий період року – температура 23-25 °С; відносна вологість 40-60 %; швидкість руху повітря не більше 0,1 м/сек. Забезпечується за допомогою загальнообмінної вентиляції.

В приміщенні використовується сумісне освітлення: природне та штучне. Згідно з ДБН В.2.5-28-2006 категорія зорових робіт, що проводяться у приміщенні – III В. Нормативні значення штучного освітлення Е=200-

500лк, природного – КПО ≥ 1,2 %. Штучне освітлення виконано як загальне, за допомогою світильників з люмінесцентними лампами [13].

Шум в приміщенні відповідає нормативним значенням – 50 дБА згідно з ДСН 3.3.6-037-99 .

Перевірочний розрахунок штучного освітлення проводиться методом коефіцієнта використання світлового потоку.

Мета перевірочного розрахунку – визначення фактичної освітленості в приміщенні.

Основна розрахункова формула методу коефіцієнта використання світлового потоку:

$$F_{св} = \frac{Eф \cdot k_з \cdot S \cdot z}{n \cdot N \cdot \eta \cdot \gamma}, \qquad (4.1)$$

де *Еф* – фактична освітленість ;

S – площа освітлюваного приміщення, S = 57,6 м^2;

z – коефіцієнт нерівномірності освітленості, z = 1,1;

$k_З$ – коефіцієнт запасу, що враховує запилення світильників і знос джерел запасу світла в процесі експлуатації, $k_З$ =1,4. Для приміщення лабораторії, освітлюваного люмінесцентними лампами та за умови чистки світильників не рідше двох разів на рік;

N – число світильників в одному ряді, N = 5;

η – коефіцієнт використання світлового потоку ламп, η = 0,52;

γ – коефіцієнт затемнення, γ = 0,8;

n – число рядів світильників, n = 2;

$n_л$ – число ламп в світильнику, 2шт.;

$F_л$ – світловий потік лампи, 2000.

З формули (4.1) потрібно визначити *Еф* і після розрахунку порівняти з *Ен*:

$$\Phi_{св} = F_{л} \cdot n_{л} = 2000 \cdot 2 = 4000.$$

Визначимо фактичну освітленість:

$$Eф = \frac{\Phi_{св} \cdot n \cdot N \cdot \eta \cdot \gamma}{k_{з} \cdot S \cdot z} = \frac{4000 \cdot 2 \cdot 5 \cdot 0,52 \cdot 0,8}{1,4 \cdot 57,6 \cdot 1,1} = 187,59 \text{ лк.}$$

Як видно з розрахунку, штучне освітлення в лабораторії – недостатнє. Для усунення даного шкідливого фактора в лабораторії слід застосовувати світильники з лампами більш високою потужністю.

У приміщенні лабораторії знаходяться тверді горючі матеріали, тому за вибухопожежної та пожежної небезпеки приміщення слід віднести до категорії В, згідно НАПБ Б.03.002-2007. Приміщення належить до класу П – IIа згідно ПУЕ-2011.

Лабораторія розташована в будівлі II ступеня вогнестійкості за ДБН В 1.1.7-2002 .

У приміщенні знаходяться ПЕОМ, які представляють собою пожежну небезпеку, тому що при підвищенні температури окремих вузлів можливо оплавлення ізоляції сполучних проводів, яке веде до замикання, що супроводжується в свою чергу іскрінням.

Причиною пожежі в лабораторії можуть бути несправність електрообладнання, руйнування ізоляції провідників, порушення правил пожежної безпеки.

Пожежна безпека в лабораторії забезпечується відповідно до ГОСТ 12.1.004-91, системою запобігання пожежі, протипожежного захисту та організаційно-технічними заходами.

Згідно ДБН В.2.5.56-2010 в приміщенні встановлено точковий димовий пожежний сповіщувач ДИП-1, який контролює площу до 86 м^2.

Згідно НАПБ Б03.001-2004 в приміщенні розміщені первинні засоби пожежогасіння – вуглекислотний вогнегасник ВВК-1,4 з розрахунку 1 вогнегасник на 3 ПК, але не менше 1 на приміщення.

Організаційні заходи: проводиться інструктаж персоналу з пожежної безпеки; розроблено заходи щодо дій адміністрації на випадок виникнення пожежі; на видному місці розміщений план евакуації при пожежі.

4.7 Висновки по четвертому розділу магістерської атестаційної роботи

В результаті виконання четвертого розділу магістерської атестаційної роботи була розроблена структура серверу IoT, що складається з серверу, датчиків та виконавчих пристроїв. Вони під’єднаються до загальної мережі, яку створює сервер та стають доступні системі керування інтелектуальними пристроями. Також було описано алгоритм роботи серверу збору даних та структура повідомлень.

Для моделювання роботи сервера IoT у хмарному сервісі AWS була створена тестова віртуальна платформа, що базується на операційній системі Linux, серверу баз даних PostgreSQL.

Експериментальні дослідження виконувалися за допомогою програмного засобу, що спеціально розроблено для виконання прогнозування поведінки дослідного об’єкту за допомогою методу машинного навчання, використовуючи дані, отримані від IoT-пристроїв. Інтерфейс програми дозволяє проводити аналіз будь-яких даних, що були отримані від різних інтелектуальних пристроїв.

Виконані дослідження показали працездатність запропонованого методу аналізу накопичених даних, отриманих за допомогою технологією IoT з використанням технології машинного навчання від Microsoft ML.NET.

ВИСНОВКИ

В результаті виконання першого розділу магістерської атестаційної роботи проведено аналіз принципів побудови додатків IoT та особливості застосування Інтернету речей на виробництві. Визначено загальні вимоги до сервера збору даних і надання послуг пристроям, які підтримують технологію Інтернету Речей (IoT).

Виконано аналіз існуючих архітектурних рішень серверів для IoT та аналіз методів обробки накопичених даних на сервері. Дано визначення терміну «Машинне навчання» та наведено приклади застосування різних математичних методів аналізу ВД.

В результаті виконання другого розділу магістерської атестаційної роботи виконана класифікація різних алгоритмов кластерного аналізу накопичених даних: алгоритми ієрархічної кластеризації, алгоритми квадратичної кластеризації, нечіткі алгоритми, алгоритми, що базуються на теорії графів, алгоритмізація виділення пов’язаних компонент, пошарова кластеризація. Наведені рекомендації щодо застосування кожного типу алгоритму.

В результаті виконання третього розділу магістерської атестаційної роботи наведені принципи використання методів машинного навчання. Наведені основні стадії підготовки та виконання аналізу даних, тренування системи та виконання прогнозу.

Проведено аналіз та вибір платформи для реалізації концепції машинного навчання та обрали платформу від AWS. Для практичного застосування обираємо інструмент, який має назву в Microsoft ML.NET. Він дозволяє виконувати виклик функцій для машинного навчання безпосередньо з програми що написані мовою програмування C# прямо з середовища Visual Studio.

В результаті виконання четвертого розділу магістерської атестаційної роботи розроблена структура серверу IoT, що складається з серверу, датчиків

та виконавчих пристроїв, які під’єднаються до загальної мережі, яку створює сервер та стають доступні системі керування інтелектуальними пристроями. Описано алгоритм роботи серверу збору даних та структура повідомлень.

Для моделювання роботи сервера IoT у хмарному сервісі AWS створена тестова віртуальна платформа, що базується на операційній системі Linux, серверу баз даних PostgreSQL.

Експериментальні дослідження виконувалися за допомогою програмного засобу, що спеціально розроблено для виконання прогнозування поведінки дослідного об’єкту за допомогою методу машинного навчання, використовуючи дані, отримані від IoT-пристроїв. Інтерфейс програми дозволяє проводити аналіз будь-яких даних, що отримані від різних інтелектуальних пристроїв.

Виконані дослідження показали працездатність запропонованого методу аналізу накопичених даних, отриманих за допомогою технологією IoT з використанням технології машинного навчання від Microsoft ML.NET.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. ДСТУ 3008–2015. Звіти у сфері науки і техніки. Структура і правила оформлення. Документація. – Введ. 2015-06-22. - К.: Держстандарт України, 2015. - 31 с.

2. Методичні вказівки з «Розробки й оформлення магістерської атестаційної роботи» для студентів другого (магістерського) рівня вищої освіти галузі знань 15 Автоматизація та приладобудування за спеціальністю 151 Автоматизація та комп'ютерно-інтегровані технології освітні програми: «Автоматизоване управління технологічними процесами», «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І.Ш. Невлюдов, В.В. Косенко, В.В. Євсєєв. – Харків: ХНУРЕ, 2019. – 55 с.

3. Офіційний сайт компанії Amazon [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/ru/iot/how-it-works/ – дата звернення 15.10.2020

4. Microsoft и Интернет Вещей [Електронний ресурс]. – Режим доступу: https://habr.com/ru/company/microsoft/blog/261367/ – дата звернення 15.10.2020

5. Технічні засоби автоматизації: Підручник / І.Ш. Невлюдов, А.О. Андрусевич, О.І. Филипенко, Н.П. Демська, С.П. Новоселов. – Кривий Ріг : Криворізький коледж НАУ, 2019. – 366 с.

6. Машинное обучение и обработка и анализ данных в Azure IoT Edgeном концепции. [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/ru-ru/azure/architecture/guide/iot-edge-vision/machine-learning – дата звернення 15.10.2020

7. Что такое ML.NET и принципы работы этой системы . [Електронний ресурс]. – режим доступу: https://docs.microsoft.com/ru-ru/dotnet/machine-learning/how-does-mldotnet-work – дата звернення 15.10.2020

8. Введение в машинное обучение и ml.net, часть 1. [Електронний ресурс]. – режим доступу: http://www.spbdev.biz/blog/vvedenie-v-mashinnoe-obuchenie-i-ml-net-chast-1 – дата звернення 15.10.2020

9. Marr B. How Big Data And The Internet Of Things Create Smarter Cities [Електронний ресурс] / Bernard Marr // Forbes – Режим доступу: http://www.forbes.com/sites/bernardmarr/2015/05/19/how-big-data-andthe-internet-of-things-create-smarter-cities/#60e178e63d8b – дата звернення 25.10.2020

10. Интеграция и взаимодействие в сети Веб [Електронний ресурс]. – Режим доступу: http://www.intuit.ru/studies/courses/485/341/lecture/8211. – дата звернення 25.10.2020

11. Невлюдов І. Ш. Людино-машинний інтерфейс в технічних засобах автоматизації: Навчальний посібник / І. Ш. Невлюдов, О. І. Филипенко, Б. О. Шостак. – Харків : «ХТМТ», 2019. – 244 с.

12. Основи наукових досліджень: Навч. посібник / І.Ш. Невлюдов, Ю.М. Олександров, А.О. Андрусевич, О.О. Чала. – Кривий Ріг: Криворізький коледж НАУ, 2019. – 396 с.

13. Sergiy Novoselov, Oksana Sychova. Intelligent Lighting Control and Management System // First International Scientific and Practical Conference «Theoretical and Applied Aspects of Device Development on Microcontrollers and FPGAs» – Kharkiv: NURE, MC&FPGA, 2019. – P. 27-28.