Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Метод блокування незареєстрованого SQL-трафіку на бази даних PostgreSQL та MySQL

Виконав: студент IV курсу, групи СБс-42
спеціальності 125 Кібербезпека
(шифр і назва спеціальності)

_____ Гангала О.М.
(підпис) (прізвище та ініціали)

Керівник _____ Баран І.О.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Лобур Т.Б.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Загородна Н.В.
(підпис) (прізвище та ініціали)

Рецензент _____ Луцик Н.С.
(підпис) (прізвище та ініціали)

Тернопіль - 2022

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

Студенту Гангалі Олександру Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Метод блокування незареєстрованого SQL-трафіку на бази даних PostgreSQL та MySQL

Керівник роботи Баран Ігор Олегович, к.т.н., доц., декан ФІС
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 23 » 03 2022 року № 4/7-178

2. Термін подання студентом завершеної роботи 22.06.2022р.

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
1. Аналіз предметної області. 1.1 Огляд існуючих рішень. 1.2. Вибір технологій та інструментів розробки. 1.3. Архітектура програми. 2. Теоретична частина. 2.1. Клієнтська частина програми. 2.2. Формування даних для ML 2.3. Огляд методів виявлення аномалій. 3. Практична частина. 3.1. Дослідження методів виявлення аномалій на реальних даних. 3.2. Моніторинг БД та переривання запитів. 4. Безпека життєдіяльності, основи хорони праці

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титулка. 2. Актуальність. 3. Мета, задачі дослідження. 4. Порівняння існуючих аналогів. 5. Загальна схема роботи програми. 6. Клієнтська частина програми. 7 Огляд методів виявлення аномалій. 8,9. Дослідження методів детектування аномалій на реальних даних. 10. Метрики, використані для оцінки. 11. LOF (Local Outlier Factor) 12. Моніторинг БД та переривання запитів. 13. Результати проведеного дослідження

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О.Я., доцент кафедри МТ		

7. Дата видачі завдання 2021 р.

КАЛЕНДАРjНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	23.03 – 26.03	*Виконано*
2.	Підбір джерел про блокування незареєстрованого SQL-		
	трафіку на бази даних	27.03 – 09.04	*Виконано*
3.	Опрацювання джерел про блокування незареєстрованого		
	SQL трафіку на бази даних	10.04 – 16.04	*Виконано*
4.	Виконання дослідження щодо розробки методу		
	блокування незареєстрованого SQL трафіку на бази		
	даних PostgreSQL та MySQL	17.04 – 23.04	*Виконано*
5	Розроблення програмного коду	24.04 – 29.04	
6.	Оформлення розділу «Аналіз предметної області»	30.04 – 07.05	*Виконано*
7.	Оформлення розділу «Теоретична частина»	08.05 – 15.05	*Виконано*
8.	Оформлення розділу «Практична частина»	16.05 – 21.05	*Виконано*
9.	Виконання завдання до підрозділу «Безпека		
	життєдіяльності, основи охорони праці»	14.05 – 21.05	*Виконано*
10.	Оформлення кваліфікаційної роботи	22.05 – 05.06	*Виконано*
11.	Нормоконтроль	06.06 – 12.06	*Виконано*
12.	Перевірка на плагіат	10.06 – 15.06	*Виконано*
13.	Попередній захист кваліфікаційної роботи	16.06 – 19.06	*Виконано*
14.	Захист кваліфікаційної роботи	23.06	

Студент _____ Гангала О.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Баран І.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метод блокування незареєстрованого SQL-трафіку на бази даних PostgreSQL та MySQL // Гангала Олександр Михайлович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, кафедра кібербезпеки, група СБс–42 // Тернопіль, 2022 // С. – 51, рис. – 26, табл. – 13 , слайдів – 13, бібліогр. – 29.

Ключові слова: БАЗА ДАНИХ, ДЕТЕКТУВАННЯ АНОМАЛІЙ, МАШИННЕ НАВЧАННЯ, НЕЗАРЕЄСТРОВАНИЙ SQL-ТРАФІК, ПРОДУКТИВНІСТЬ ЗАПИТІВ

Кваліфікаційна робота присвячена виявленню та усуненню незареєстрованого sql-трафіку, який потенційно може завдати шкоди продуктивності базі даних, на підставі методу машинного навчання.

Проаналізована предметна область, досліджено існуючі інструменти, котрі можуть визначити потенційні збитки від виконання незареєстрованого sql - трафіку та його заблокувати. Докладно описано процес формування даних для здійснення машинного навчання. Досліджено методи виявлення аномалій. За результатами дослідження найефективнішим визначено метод Local Outlier Factor.

Спроектовано архітектуру та програмно реалізовано додаток, котрий при моніторингу БД здатний заблокувати потенційно шкідливий незареєстрований sql -трафік із використанням підготовленої моделі машинного навчання.

ANNOTATION

Method of unregistered SQL traffic locking on Databases PostgreSQL and MySQL // Hanhala Oleksandr // Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cyber Security // Ternopil, 2022 // P. - 51, Fig. - 26, Table - 13, Slides - 13, References - 29.

Keywords: ANOMALY DETECTION, DATABASE, MACHINE LEARNING, QUERY PRODUCTIVITY, UNREGISTERED SQL TRAFFIC

Thesis deals with the detection and elimination of unregistered sql-traffic, which can potentially damage the performance of the database, based on the method of machine learning.

The subject area is analyzed, the existing tools that can determine the potential losses from the execution of unregistered sql -traffic and block it are investigated. The process of data formation for machine learning is described in detail. Methods for detecting anomalies have been studied. According to the results of the study, the Local Outlier Factor method was determined to be the most effective.

The architecture was designed and the application was implemented that is able to block potentially harmful unregistered sql-traffic when monitoring the database using a prepared machine learning model.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ СКОРОЧЕНЬ І ТЕРМІНІВ

Ad-hoc запит – спеціалізований запит для одноразового вирішення певної проблеми.

GLD (Global Lexemes Dictionary) – глобальний словник лексем.

Latency метрика - метрика, що визначає час затримки відповіді між базою даних та фронтендом чи бекендом.

LLD (Local Lexemes Dictionary) - локальний словник лексемю

LOF (Local Outlier Factor) – локальний рівень викиду.

ML (Machine Learning) – машинне навчання.

OCSVM (One-Class Support Vector Machines) - метод опорних векторів для одного класу.

Production - середовище – оточення розгортання програмного забезпечення, яке використовують кінцеві користувачі.

SVM (Support Vector Machines) – метод опорних векторів.

БД – база даних.

Бекенд (Backend) - набір засобів, за допомогою яких відбувається реалізація логіки веб-сайту.

Конверсія — співвідношення покупців (або відвідувачів, які вчинили цільову дію, наприклад, проголосували, перейшли за посиланням) до відвідувачів сайту.

ПЗ – програмне забезпечення.

СУБД — система управління базами даних.

Фронтенд (Frontend) - клієнтська сторона інтерфейсу користувача або веб-сайту.

ЗМІСТ

ВСТУП

Зазвичай перед розгортанням програми в production -середовище перевіряється продуктивність кожного запиту до БД. Однак потрібно також врахувати ситуації, коли, наприклад, співробітники компанії надсилають ad-hoc запити (що є незареєстрованим трафіком), які не завжди оптимізовані, і, самі того не підозрюючи, навантажують БД, забираючи ресурси БД від звичайного sql -трафіку. Наслідком цієї проблеми може бути збільшення latency- метрики на фронтенді або бекенді, тобто час роботи запиту може збільшитися.

Але згідно з численними дослідженнями, люди не люблять чекати. Наприклад, за даними аналітичної платформи KissMetrics [1], яка досліджувала кореляцію конверсії та завантаження веб-сторінок, ми бачимо такі результати:

– "затримка завантаження web -сайту на 1 с призводить до скорочення конверсій на 7%",

– “73% користувачів мобільного Інтернету кажуть, що вони стикалися із надто повільним завантаженням веб-сайту”,

– "47% користувачів стверджують, що час завантаження веб-сторінки має бути не більше 2 с",

– “20% людей залишають веб-сайт, який завантажується понад 3 с”.

Ці цифри доводять, що якщо швидкість завантаження сайту занадто мала, тоді користувачі можуть залишити його і віддати перевагу іншій компанії, особливо якщо у них той же контент завантажується швидше. Але слід пам'ятати, що ці цифри - це комбінація різних швидкостей завантаження. Цілком можливо проблема у слабкому сервері, поганому інтернет-каналі тощо. Причин може бути безліч. Але в тому числі у ці цифри входить і робота БД як сховища. Іншими словами, однією з причин тривалого завантаження може бути і низька продуктивність запитів до БД. Вчасно виявлена аномальна активність може зберегти ресурси сервера БД, а значить репутацію, лояльність клієнтів та бюджет, який потенційно міг би втратити бізнес у разі низької продуктивності їхньої програми.

Метою роботи є розробка методу для конкретного бізнес-трафіку, який

дозволить не допускати стрибки продуктивності та одночасно блокувати такий незареєстрований трафік у БД PostgreSQL та MySQL.

В процесі виконання роботи потрібно вирішити наступні завдання:

– проаналізувати існуючі інструменти, здатні виявити, визначити потенційні збитки від виконання незареєстрованого sql -трафіку та його заблокувати;

– підготувати дані для ML та тестування роботи інструменту;

– вибрати найбільш ефективний метод ML для виявлення незареєстрованого sql –трафіку;

– розробити інструмент, який під час моніторингу БД блокуватиме потенційно шкідливий незареєстрований sql -трафік на підставі підготовленої моделі ML.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих рішень

На сьогоднішній день прямих аналогів даної програми на ринку не було знайдено, однак існують рішення, які частково виконують поставлене завдання.

Розглянемо рішення для БД PostgreSQL.

ClusterControl – інструмент для моніторингу та управління БД MySQL, MariaDB, MongoDB, PostgreSQL. Однією з його функцій є єдиний та комплексний моніторинг усієї інфраструктури БД та сервера у реальному часі. Він збирає інформацію про повільні запити, але не вбиває їх.

Команди pg_cancel_backend, pg_terminate_backend, kill дозволяють вбити вказаний запит.

Параметр конфігурації statement_timeout дозволяє знищити запит, який виконується більше заданої кількості мілісекунд.

Вручну написані скрипти. Для того, щоб відстежити повільний запит, можна звернутися до таблиці pg_stat_activity, а вбити через одну з команд, запропонованих у попередньому пункті.

Для MySQL існують такі рішення:

– інструмент pt-kill – підключається до MySQL та отримує запити від таблиці information_schema.PROCESSLIST. Можна налаштувати так, щоб убивав запити, які виконуються більше, ніж визначений час;

– самописні скрипти, що дозволяють знищити запит, який виконується більше, ніж визначений час. Відстежити повільний запит можна за допомогою таблиці information_schema.PROCESSLIST.

Для оцінки аналогів виділимо такі критерії (можливості):

– відстеження повільних запитів. Оскільки зареєстрований трафік оптимізований і має передбачуваний характер, то однією з ознак наявності незареєстрованого трафіку є тривалий час виконання. Це тягне за собою навантаження на БД, і, як наслідок, забирає ресурси БД від звичайного sql трафіку;

– переривання повільного запиту. Якщо відразу не вбити повільний запит, то навантаження так і не буде припинено;

– перевірки запиту відповідність певному паттерну (наприклад, зміст хинта). Це необхідно для того, щоб структурою відрізнити зареєстрований трафік від незареєстрованого;

– визначити потенційні збитки від виконання незареєстрованого запиту. Оскільки основна мета - це запобігання стрибкам продуктивності, а не просто вбити всі запити, що стосуються незареєстрованого трафіку.

Узагальним казане вище у формі таблиці 1.1.

Таблиця 1.1 - Порівняння існуючих аналогів

		Відстеження повільних запитів	Переривання повільного запиту	Перевірка запиту на відповідність певному патерну	Визначення потенційної шкоди від виконання незареєстрованого запиту
PostgreSQL	ClusterControl	+	–	–	–
	Команди pg_cancel_backend, pg terminate backend, kill	–	+	–	–
	Параметр конфігурації statement_timeout	+	+	–	–
	Самописні скрипти	+	+	+	–
MySQL	Інструмент pt-kill	+	+	–	–
	Самописні скрипти	+	+	+	–

Таким чином, вищезазначені інструменти не виконують основну мету, а саме: не допустити стрибки продуктивності та одразу блокувати трафік, який потенційно може навантажити БД. Вони працюють постфактум, коли latency-

метрика вже збільшилася, що сигналізує про незареєстрований трафік. Тому було прийнято рішення розробити програму, яка б задовольняла визначеній меті кваліфікаційної роботи.

1.2 Вибір технологій та інструментів розробки

Для реалізації програми була використана мова програмування Python, яка є найбільш популярною та застосовуваною в галузі ML за рахунок наявності безлічі фреймворків та готових бібліотек.

У процесі роботи застосовувалися бібліотеки pandas, numpy, matplotlib.pyplot, skleam. Для роботи з БД PostgreSQL використовувалася бібліотека psycopg2, до роботи з MySQL - mysql.connector.

Для реалізації клієнтської частини використали фреймворк з відкритим кодом Streamlit [16]. У ньому є велика бібліотека готових компонентів, які можна використовуватиме швидкої розробки найпростіших інтерфейсів.

Як середовище розробки було обрано PyCharm [17] від компанії JetBrains.

1.3 Архітектура програми

Архітектура цієї програми складається з двох частин: серверної та клієнтської. Інтерфейс для взаємодії з користувачем, наведений у п. 3.1.

Також хотілося б зазначити, що в кваліфікаційній роботі зареєстрований трафік, представлений у вигляді запитів, що містять певний патерн, наприклад, блоковий коментар, тобто коментар, записаний у стилі мови програмування С:

SELECT /*OK ...*/ value FROM table;

Коментар видаляється з вхідного потоку на початку синтаксичного аналізу та фактично замінюється пробілом. Тому він є зручним для маркування зареєстрованого трафіку. Виходячи з цього, незареєстрованим трафіком будуть запити, які не містять коментар.

Для визначення потенційної шкоди від виконання незареєстрованого

запиту буде виконуватися перевірка подібності запиту, що надійшов, із зареєстрованим трафіком. Якщо запит схожий з таким трафіком, то передбачається, що він не навантажуватиме БД і забиратиме ресурси БД від звичайного sql трафіку.

Основні логічні кроки програми:

– початкове налаштування програми. На клієнтській стороні необхідно вказати тип БД, дані для підключення, коментар, характерний для зареєстрованого трафіку, та завантажити 2 файли з прикладами зареєстрованого та незареєстрованого трафіку;

– після цього необхідно натиснути кнопку “Навчити модель” для виконання відповідної дії на представлених даних;

– потім потрібно натиснути кнопку “Почати моніторинг” і після цього програма почне моніторити БД;

– запити, що містять коментар, будуть проігноровані (оскільки це зареєстрований трафік). Інші запити надходитимуть на вхід моделі ML, яка на виході повинна дати відповідь, чи схожі запити на ті, на яких модель була навчена (тобто, чи схожі на зареєстрований трафік).

Якщо модель повернула значення -1, тобто запит не схожий на зареєстрований трафік, він вбивається.

На рис. 1.1 відображена схема роботи програми.

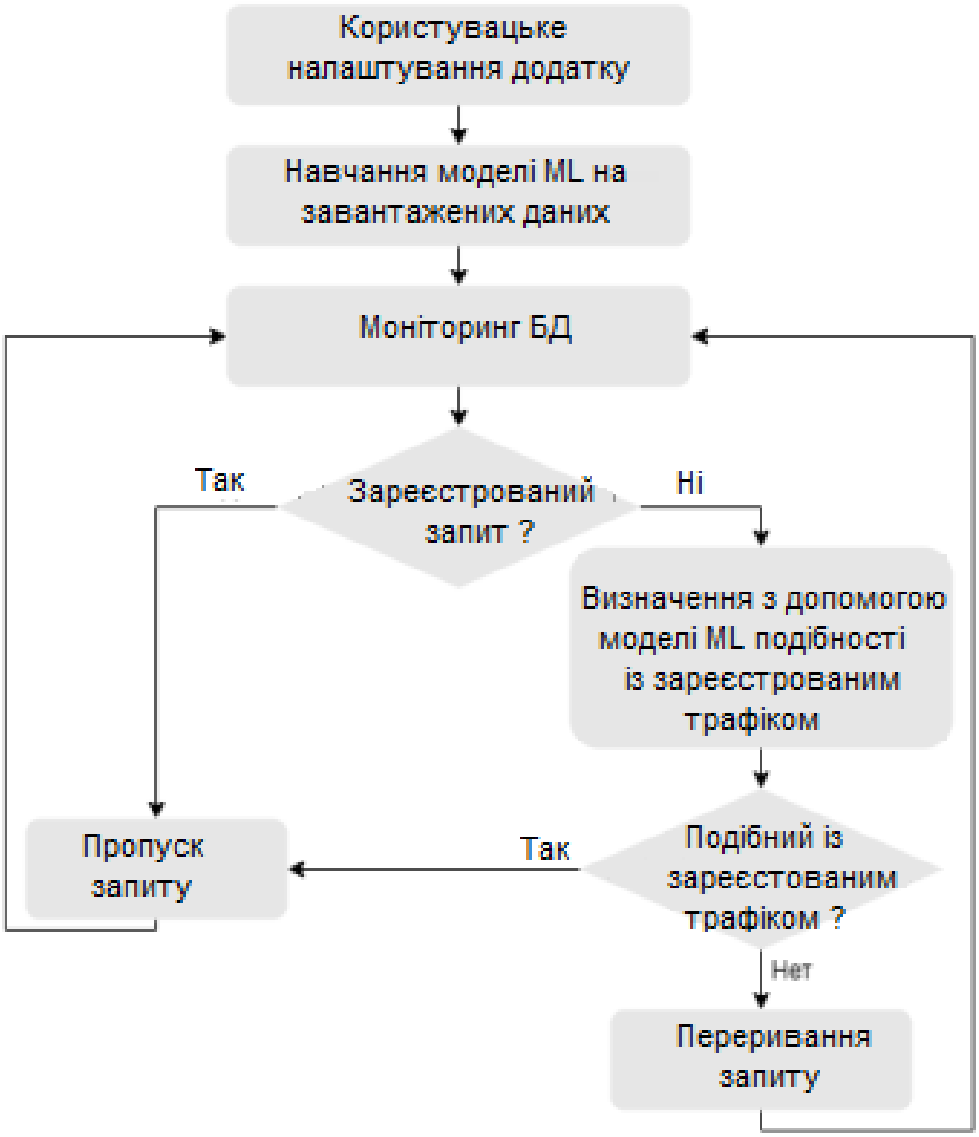

Рисунок 1.1 – Схема роботи програми

В наступному розділі буде докладно розглянуто процес підготовки даних для ML, обґрунтувано вибів моделі ML та реалізацію моніторингу БД та переривання запиту.

2 ТЕОРЕТИЧНА ЧАСТИНА

2.1 Клієнтська частина програми

Для розуміння можливостей розробленого програмного додатку варто навести скріншоти вікон для взаємодії з користувачем (рис. 2.1 – 2.4).

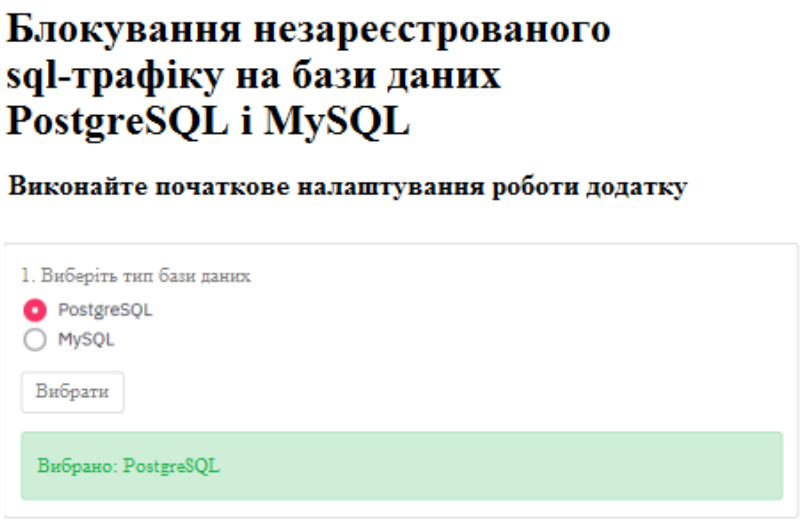

Рисунок 2.1 – Головне вікно програми

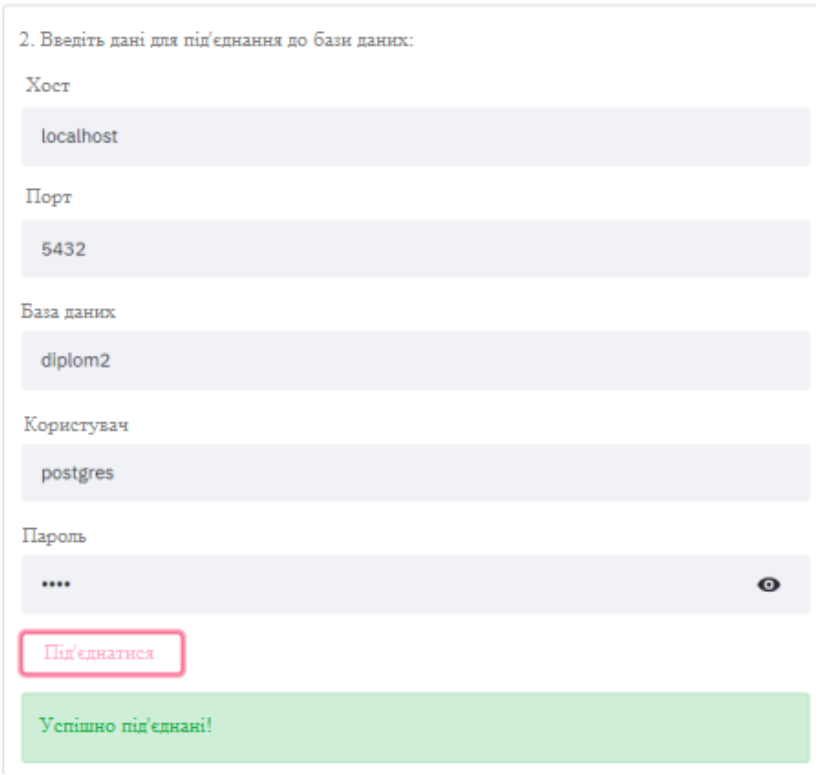

Рисунок 2.2 – Вікно вводу даних для під’єднання до БД

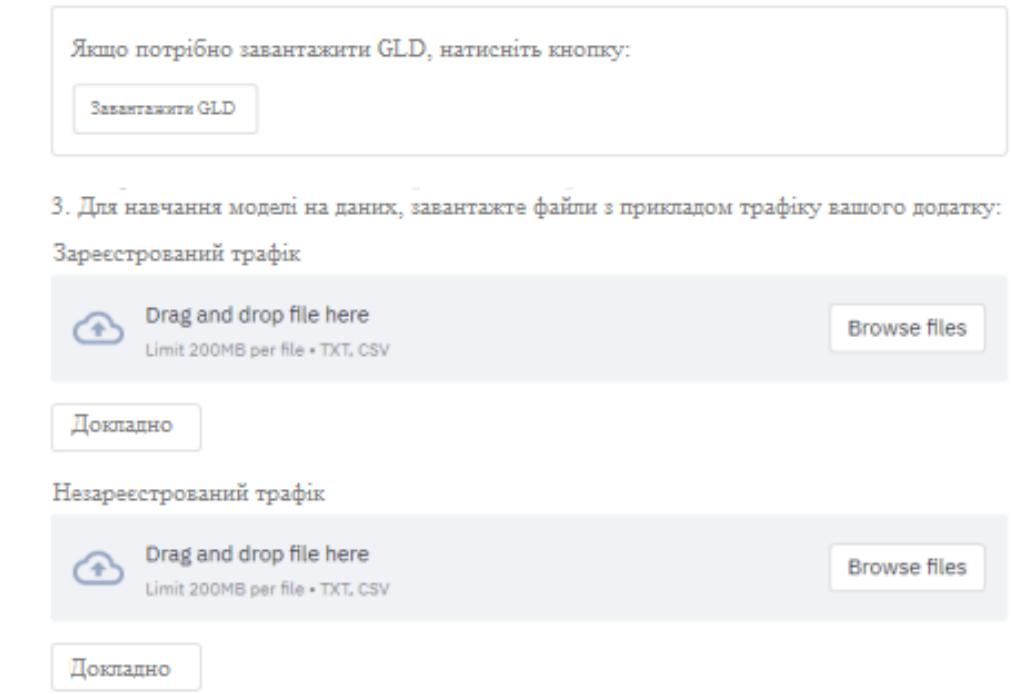

Рисунок 2.3 – Вікно для завантаження файлу з прикладом трафіку

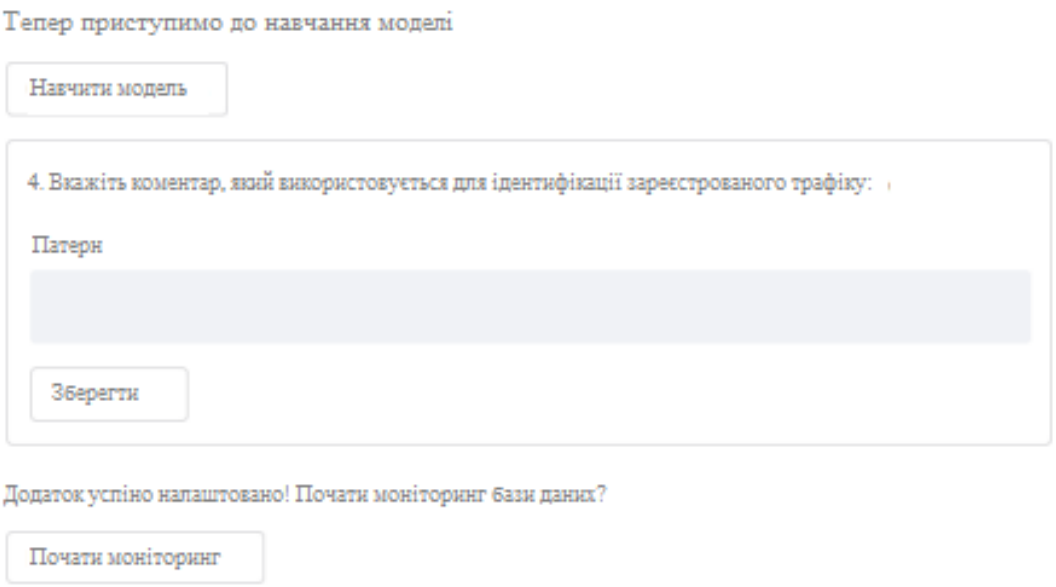

Рисунок 2.4 – Вікно для задання коментаря для ідентифікації зареєстрованого трафіку

2.2 Формування даних для ML

2.2.1 Обробка вихідних даних

Необхідно зазначити, що для якісного навчання моделі ML на вхід

програма отримує 2 журнальні файли з логами для PostgreSQL/MySQL (із зареєстрованим та незареєстрованим трафіком) (див. рис. 2.5, 2.6).

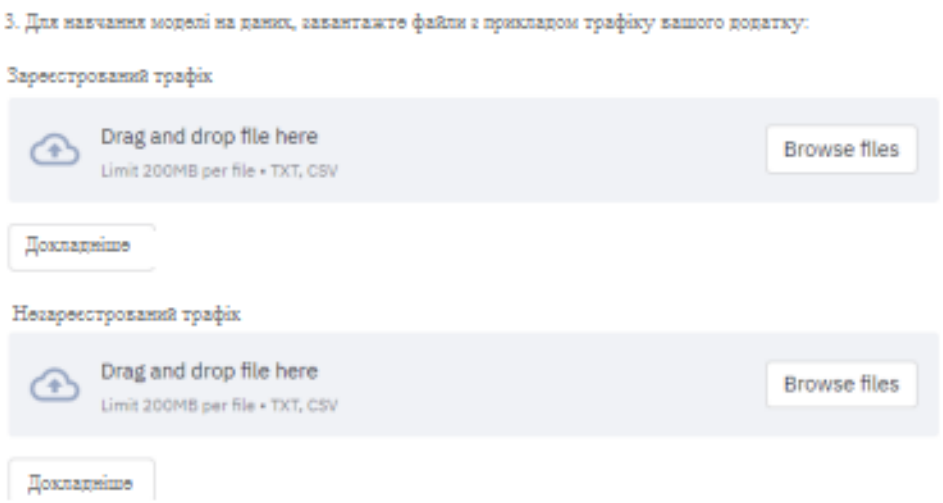

Рисунок 2.5 – Завантаження файлів за допомогою інтерфейсу користувача

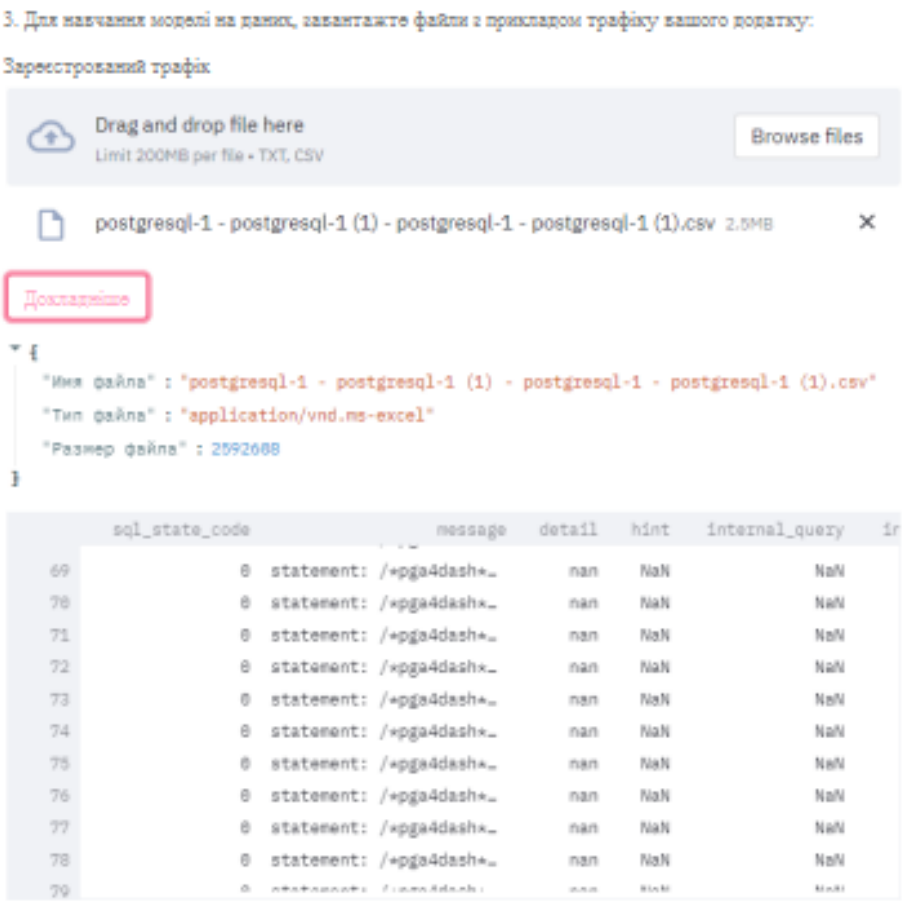

Рисунок 2.6 – Демонстрація даних після їх завантаження

Рядки журналу є значеннями, розділеними комами (CSV формат) з наступними полями для PostgreSQL (табл 2.1).

Таблиця 2.1 – Перелік значень загального журнального файлу для PostgreSQL

Назва	Тип	Опис	Приклад
log_time	timestamp(3) with time zone	штамп часу з мілісекундами	2021-04-25 17:34:34.934 MSK
user_name	text	Ім'я користувача	postgres
database_name	text	ім'я бази даних	postgres
process_id	integer	Ідентифікатор процесу	21288
connection from	text	клієнтський вузол: номер порту	::1:60397
session_id	text	ідентифікатор сесії	60857dfa.5328
session_line_num	bigint	номер рядка кожної сесії	2
command_tag	text	тег команди	
session_start_time	timestamp with time zone	час початку сесії	2021-04-25 17:34:34 MSK
virtual transaction id	text	Віртуальний ідентифікатор транзакції	8/11354
transaction_id	bigint	Ідентифікатор транзакції	0
error_severity	text	рівень важливості помилки	LOG
sql_state_code	text	код помилки SQLSTATE	0
message	text	повідомлення про помилку/текст запиту	"statement: SELECT ..."
detail	text	подробиці до повідомлення про помилку	"parameters: $1 = '1', $2 = '8'"
hint	text	підказка до повідомлення про помилку	
internal_query	text	внутрішній запит, що спричинив помилку (якщо є)	
internal_query_pos	integer	номер символу внутрішнього запиту, де сталася помилка	
context	text	контекст помилки	

Продовження таблиці 2.1

query	text	запит користувача, що спричинив помилку (якщо є і	
query_pos	integer	номер символу у запиті користувача, де сталася	
location	text	розташування помилки у вихідному коді PostgreSQL (якщо log_error_verbosity встановлений у verbose)	
application_name	text	ім'я програми	pgAdmin 4 - CONN:8048076

Для MySQL (табл. 2.2).

Таблиця 2.2 – Перелік значень загального журнального файлу для MySQL

Назва	Тип	Опис	Приклад
event_time	timestamp(6)	час, коли було виконано запит	2020-05-26T08:01:39.429740Z
thread_id	int(11)	Ідентифікатор потоку	1
server_id	int(10)	Ідентифікатор сервера	17
command_type	varchar(64)	тип команди	Query
argument	mediumtext	текст запиту	INSERT INTO rental VALUES (1,2005-05-24 22:53:30')

Для отримання SQL запиту в PostgreSQL треба розглядати поле message, в MySQL – поле argument .

2.2.2 Переклад запитів у векторну форму

Далі необхідно описати в термінах вектора ознак кожен запит SQL . За основу використовуватимемо LLD та GLD.

GLD– це всі слова, які є ключовими у стандарті в PostgreSQL. Тобто слова, що мають фіксоване значення у мові SQL, наприклад, INSERT, UPDATE,

SELECT, DELETE тощо.

LLD формується на основі GLD, але в нього входять ті ключові слова, які використовуються для складання SQL запитів конкретного дампа даних, або, як в даному випадку, найбільш значущі. Тому що знаходження подібності між парами всіх векторів – це обчислювально складне завдання [4]. Тому необхідно залишити значні ознаки. Для PostgreSQL ключові слова можна взяти у документації [2]. Однак наведена там таблиця містить зарезервовані та незарезервовані слова. За стандартом PostgreSQL, зарезервовані ключові слова є єдиними справжніми ключовими словами. Тому при складанні LLD розглядатимемо лише їх. Для MySQL звернімося до документації [3] і залишимо для розгляду лише зарезервовані ключові слова.

2.2.3 Побудова частотної матриці

Для побудови частотної матриці описуємо кожен SQL запит у термінах вектора ознак де індекс вектора відповідає індексу LLD. Елементом вектора ознак має бути число в інтервалі від [0,1] що показує частоту вживання (рівна кількості згадок лексеми поділеному на кількість лексем) лексеми всередині SQL запиту.

На виході отримуємо матрицю M розмірністю (m x n), де m – кількість SQL запитів, n – розмірність LLD словника.

Наприклад:

SQL_1 = {SELECT * FROM a INNER JOIN b ON a.id = b.id LEFT JOIN c ON c.id = a.id WHERE a.c2 = :c2 AND b.c1 = :c1}

Усього лексем = 10.

У таблиці 2.3 наведено отримані частотності кожної з лексем.

Таблиця 2.3 – Частотності лексем для SQL_1

	SELECT	FROM	INNER	ON	RIGHT	…	LEFT	JOIN	WHERE	AND
SQL_1	1/10	1/10	1/10	2/10	0	…	1/10	2/10	1/10	1/10

SQL_2 = {SELECT * FROM a RIGHT JOIN b ON a.id = b.id}

Всього лексем = 5.

Отримуємо частотність кожної з лексем (табл .2.4).

Таблиця 2.4 – Частотності лексем для SQL_1, SQL_2

	SELECT	FROM	INNER	ON	RIGHT	…	LEFT	JOIN	WHERE	AND
SQL_1	1/10	1/10	1/10	2/10	0	…	1/10	2/10	1/10	1/10
SQL_2	1/10	1/10	0	1/10	1/10	…	0	1/10	0	0

$SQL_n = \{…\}$

Таблиця 2.5 – Частотності лексем для SQL_1, SQL_2, …, SQL_n

	SELECT	FROM	INNER	ON	RIGHT	…	LEFT	JOIN	WHERE	AND
SQL_1	1/10	1/10	1/10	2/10	0	…	1/10	2/10	1/10	1/10
SQL_2	1/10	1/10	0	1/10	1/10	…	0	1/10	0	0
…	…	…	…	…	…	…	…	…	…	…
SQL_n	-	-	-	-	-	-	-	-	-	-

Програмно підхід до побудови частотної матриці виглядає так.

1. Послідовно переглядаються всі запити;

2. Текст запиту подається у вигляді масиву слів, кожне з яких порівнюється з LLD для отримання масиву, що складається лише з лексем. Далі розраховується частота вживання лексеми всередині SQL запиту та зберігається у змінну типу словник dict (лістинг 2.1).

Лістинг 2.1 – Частота вживання лексеми всередині SQL запиту

```
def create_lld_and_safe_into_db(self, message, lld_tablename, counter_tablename,
                                dbname, user, password, host, port):
    gld = gldService.get_gld_values_from_db(dbname, user, password, host, port)
    lld = []
    for i in message.split():
        if i.upper() in gld:
            lld.append(i)
    # print(lld): ['SELECT', 'DISTINCT', 'FROM']
    size = len(lld)
    dict = {i: lld.count(i) for i in lld}
    # print(dict): {'SELECT': 1, 'DISTINCT': 1, 'FROM': 1}
    for key, value in dict.items():
        dict[key] = value / size
    # print(dict): {'SELECT': 0.33333333, 'DISTINCT': 0.33333333, 'FROM': 0.33333333}
    self.safe_into_lld_table(dict, lld_tablename, counter_tablename,
                             dbname, user, password, host, port)
```

Так як модель ML з певною періодичністю буде повторно навчатися на оновлених даних, то ми зберігаємо вектора в БД, а не використовуємо, наприклад, масив словників. Отриманий словник (dict) зберігається залежно від типу даних у таблицю unregistered або registered вертикально (лістинг 2.2). Тобто в стовпець key записується лексема, в стовпець value частота вживання, в id_query локальний для цієї таблиці ідентифікатор запиту.

Лістинг 2.2 – Код збереження словника (dict)

```
def safe_into_lld_table(self, dict, lld_tablename, counter_tablename,
                        dbname, user, password, host, port):
    self.__connect__(dbname, user, password, host, port)
    sql = '''INSERT INTO ''' + lld_tablename + '''(key, value, id_query) VALUES ('''
    self.cur.execute("SELECT value FROM " + counter_tablename + " WHERE id = 1")
    count = self.cur.fetchone()
    count = count[0] + 1
    for key, value in dict.items():
        sql1 = sql + '''\'''' + key + '''\',''' + str(value) + ''',''' + str(count) + ''')'''
        self.cur.execute(sql1)
        self.cur.execute('''UPDATE ''' + counter_tablename + ''' SET value = ''' +
                         str(count) + ''' WHERE id = 1''')
    self.con.commit()
    self.__disconnect__()
```

Існує ще один варіант зберігання векторів - це створення таблиці, стовпцями якої були значення LLD. Тоді було б простіше зберігати та

отримувати вектор, проте PostgreSQL має обмеження кількості стовпців – максимум 1600 [5], а у MySQL ліміт 4096 стовпців [6]. При цьому, максимальна кількість стовпців таблиці додатково зменшується у зв'язку з тим, що кортеж, що зберігається, має займати одну сторінку розміром 8192 байта для PostgreSQL.

MySQL містить вимоги для ліміту стовпців та власне максимального розміру рядка для таблиці, оскільки загальна довжина всіх стовпців властиво не може перевищувати цей розмір. Тому у разі збільшення LLD та перевищення зазначених лімітів дана схема зберігання не актуальна.

Після збереження всіх запитів у таблицю БД, викликається метод get_matrix(), який дозволяє на основі значень таблиці lld_table побудувати матрицю у вигляді DataFrame (лістинг 2.3).

Лістинг 2.3 – Код виклику методуу get_matrix()

```
def get_matrix(self, tablename, dbname, user, password, host, port):
    values = self.get_lld_values_from_db(tablename, dbname, user, password, host, port)
    gld = gldService.get_gld_values_from_db(dbname, user, password, host, port)
    df = pd.DataFrame(columns=gld)
    # группируем по id_query
    get_attr = operator.attrgetter('id_query')
    new_list = [list(g) for k, g in itertools.groupby(sorted(values, key=get_attr), get_attr)]
    # для каждого запроса формируем dict и записываем в DataFrame
    for x in new_list:
        val = {}
        for y in x:
            val[y.key.upper()] = y.value
        df = df.append(val, ignore_index=True)
    # Замена Nan на 0.0
    df = df.fillna(0)
    return df
```

Дані представлені у необхідному форматі, далі розглянемо методи ML, щоб обрати найоптимальніший для поставленого завдання.

2.3 Огляд методів виявлення аномалій

Оскільки на вхід програма отримує 2 журнальні файли з логами: із зареєстрованим та незареєстрованим трафіком, то ми маємо один датасет, який не забруднений викидами, а інший складатиметься лише з них. Також очевидно,

що випадків незареєстрованого трафіку буде в рази менше, ніж зареєстрованого.

Тому це завдання було вирішено як завдання детектування аномалій (anomaly detection) - один із видів завдань ML без вчителя, який ділиться на 2 напрямки:

– детектування викидів (Outlier Detection);
– детектування "новизни" (Novelty Detection).

"Новизна" - це об'єкти, які відрізняються за властивостями від об'єктів навчальної вибірки. Прикладом завдання на детектування «новизни» є ситуація, коли в самій вибірці викиду ще немає і у разі його необхідно виявити.

Виявлення аномалій в завданні кваліфікаційної роботи - це процес ідентифікації незареєстрованого трафіку, шляхом виявлення нехарактерного трафіку для програми.

Для виявлення аномалій використовується ряд інструментів та методів, від простих статистичних методів до складних алгоритмів ML, залежно від складності даних та необхідної складності.

Статистичний метод – основа практично всіх моделей. Він застосовується до кожної ознаки і виявляє його екстремальні значення. Приклад такого методу є Z-score. Але, зазвичай, аномалія характеризується екстремальними значеннями окремих ознак. Тому було б помилково набувати максимального або мінімального значення за аномалію. Тому цей метод не розглядатиметься.

Метричний метод є також одним із найпопулярніших, якщо судити за кількістю публікацій [11]. Але для великих наборів даних висока вартість прогнозування. Це пов'язано з тим, що у великих наборах даних вартість обчислення відстані між новою точкою та кожною існуючою точкою стає вищою. Тому також не розглядатиметься.

Далі розглянемо популярні методи ML, що застосовуються виявлення аномалій.

2.3.1 OCSVM

Це алгоритм навчання без вчителя, звичайний SVM, який відокремлює вибірку від початку координат.

Основна ідея SVM – розділити класи гіперплощиною, щоб максимізувати відстань між ними. Спочатку алгоритм працював тільки з лінійно розділеними класами, але в 1992 був впроваджений "Kernel Trick", який дозволив працювати і з лінійно неподільними даними. Ідея, що лежить в основі "Kernel Trick", полягає в тому, що класи, які лінійно нероздільні в поточному просторі ознак, можуть стати розділеними у просторах вищої розмірності (рис. 2.7).

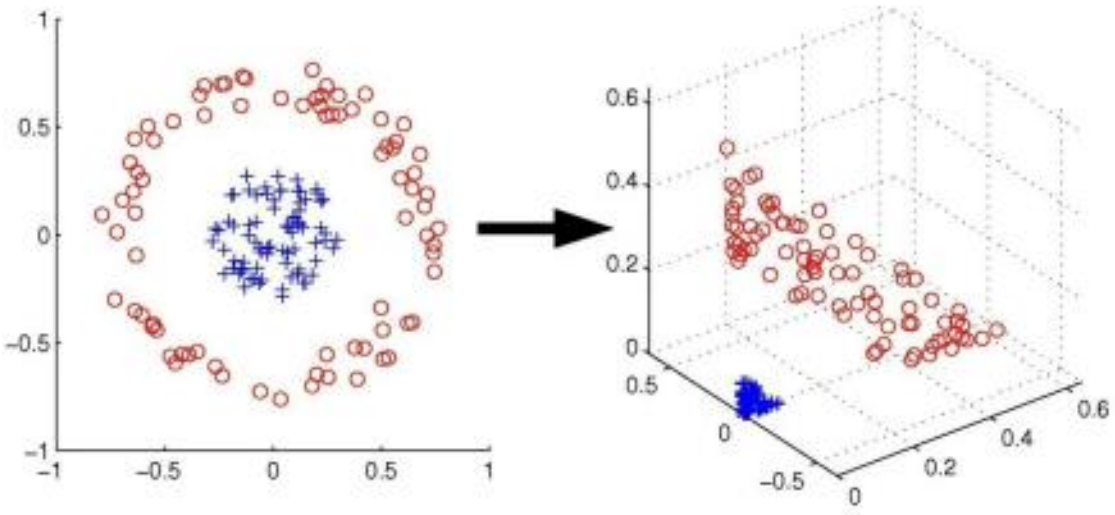

Рисунок 2.7 – Наочний приклад ідеї "Kernel Trick"

Стандартний алгоритм SVM є двокласовим алгоритмом, це означає, що потрібні позитивні та негативні приклади. В OCSVM модель опорних векторів навчається на даних, які мають лише один клас – «нормальний» клас. Він виводить властивості нормальних випадків і основі цих властивостей може передбачити, які приклади від нормальних прикладів. Це корисно для виявлення аномалій, оскільки, зазвичай, дані не збалансовані, оскільки прикладів аномальної поведінки зустрічається у рази менше, ніж нормальної.

Основна ідея алгоритму полягає в тому, щоб перетворити простір ознак і провести роздільну гіперплощину так, щоб спостереження знаходилися якнайдалі від початку координат (рис. 2.8). Тобто аномальними вважаються ті об'єкти у вибірці, для яких відповідні вектори описів близькі до початку координат у просторі ознак.

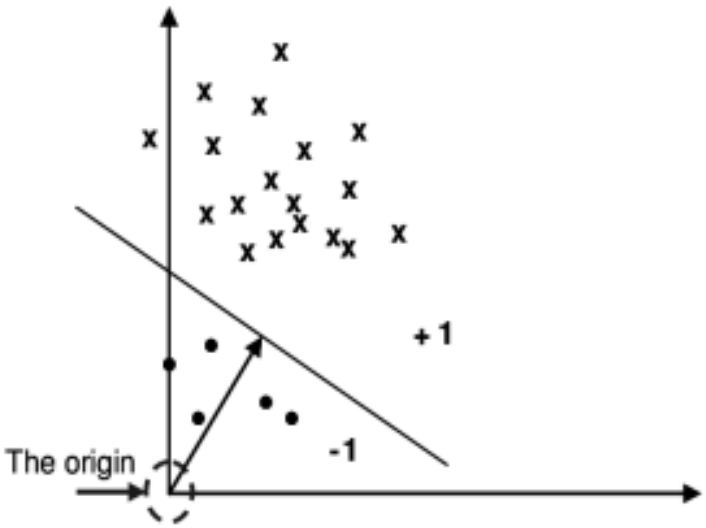

Рисунок 2.8 – Схематичне представлення алгоритму OCSVM

В результаті ми отримуємо кордон, по один бік якого спостереження з нашої чистої тренувальної вибірки упаковані максимально щільно, а по другий будуть знаходитися аномальні значення, несхожі на ті, на яких навчався алгоритм.

Слід зазначити, що на відміну стандартного методу опорних векторів, тільки радіальні базисні функції (rbf) підходять як ядра у разі, оскільки інші нелінійні ядра працюють гірше.

Переваги:

– наявність параметра *nu* на відміну від SVM, який контролює чутливість опорних векторів. Повинен бути налаштований на приблизне співвідношення викидів даних;

– модель здатна проводити нелінійні розділяючі межі, завдяки “Kernel trick”;

– зручність роботи з незбалансованими даними (у разі, якщо дані недостатньо "поганих" спостережень для використання стандартного підходу навчання з учителем — бінарної класифікації).

Слід зауважити, що OCSVM - це швидше алгоритм пошуку новизни, аніж викидів, тому що «заточується» під навчальну вибірку. Також має бути достовірно відомо, що дані, на яких навчатиметься модель, не містять викидів,

інакше алгоритм розглядатиме їх як нормальні спостереження.

2.3.2 Ізолюючий ліс (IsolationForest)

Це різновид ідеї випадкового лісу (Random Forest) (рис. 2.5):

– ліс складається із дерев;

– кожне дерево будується до того часу, доки закінчиться вибірка;

– заснований на принципі Монте-Карло: випадкова ознака та випадкове розщеплення вибираються для побудови розгалуження у дереві. Для кожного об'єкта мірою його нормальності є середнє арифметичне глибин листя, в яке він потрапив (ізолювався).

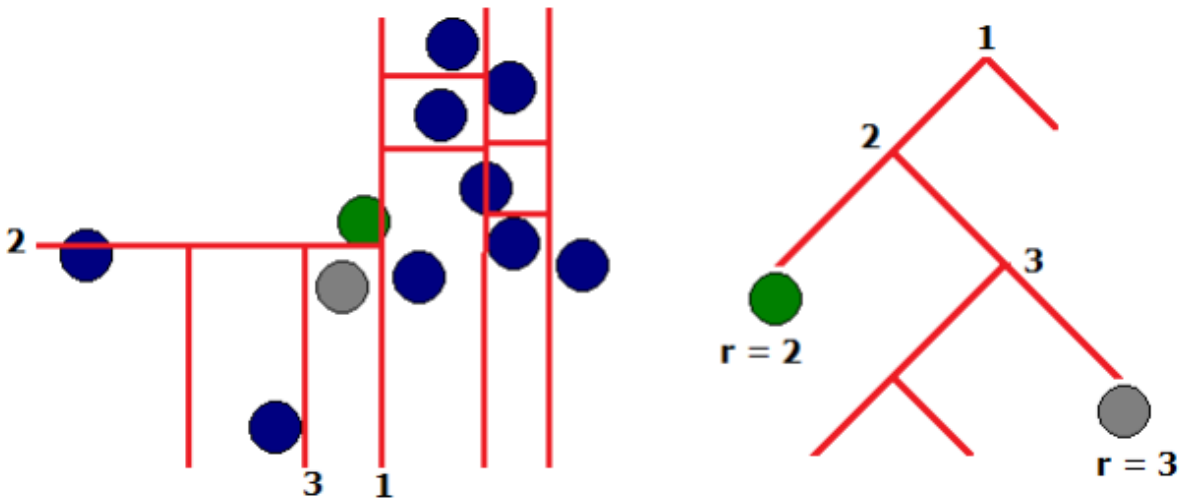

Рисунок 2.9 – Розрахунок оцінки аномальності в ізолюючому лісі

Ізолюючий ліс (Isolation forest) - ансамблевий метод навчання для класифікації, регресії та інших завдань, що виконуються шляхом побудови множини вирішувальних дерев під час навчання та виведення класу, що є середнім прогнозом (регресія) окремих дерев чи модою класу (класифікація) [7, 12]. При використанні великої кількості вирішувальних дерев результат буде точнішим, у той час як кожне окреме дерево дає низьку якість детектування аномалій.

Ідея методу у побудові випадкового двійкового дерева рішень, коренем якого буде ознаковий простір. У кожному вузлі вибираються випадкові ознаки та

поріг розбиття. Дерево формується доти, доки кожен об'єкт не опиниться в окремому аркуші. Тому викиди потраплять у листя на початкових етапах (власне на невеликій глибині дерева), та їх простіше властиво «ізолювати». Отже, об'єкти з кластерів невеликих розмірів, які потенційно є аномаліями, матимуть нижчий anomaly_score, ніж із кластерів нормальних даних.

Переваги алгоритму:

– ефективність у порівнянні з більшістю інших алгоритмів через складність O(n log n);

– розпізнає і ізольовані точки з низькою локальною густиною, і кластери аномалій малих розмірів;

– відсутні параметри, що потребують підбору;

– інваріантність до масштабування ознак;

– не вимагає значних витрат пам'яті, як, наприклад, метричні методи, котрим, переважно, необхідно побудова матриці попарних відстаней.

– стійкий до прокляття розмірності, пов'язаного з експоненційним зростанням обсягу даних через збільшення розмірності простору (прикладами рішень можуть бути метод основних компонентів, здійснення відбору ознак, використання алгоритму обчислення оцінок).

2.3.3 Еліпсоідальна апроксимація даних (EllipticEnvelope)

У еліпсоїдальної апроксимації даних, як випливає з назви, хмара точок моделюється як нутрощі еліпсоїда. Значення, які у цю область попадають, вважаються нормальними даними, проте, що перебуває поза - розглядаються як викиди (рис. 2.10).

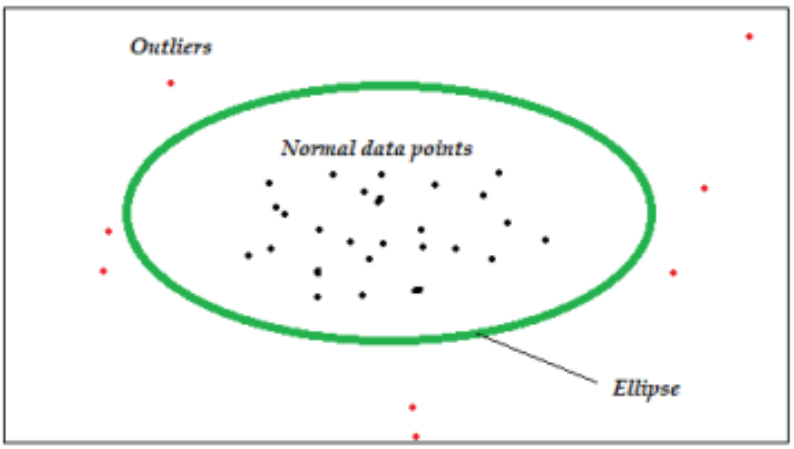

Рисунок 2.10 – Схематичне представлення алгоритму EllipticEnvelope

Цей метод дає добрі результати тільки власне на одномодальних даних, а найкраще, якщо дані мають нормальний розподіл, який також називається розподілом Гауса. Ступінь новизни визначається за відстанню Махалонобіса

2.3.4 LOF

Це алгоритм виявлення аномалій без вчителя, який обчислює локальне відхилення щільності даної точки даних стосовно її сусідів (рис. 2.11). Тому викидами вважаються зразки, які мають значно меншу щільність, ніж їхні сусіди.

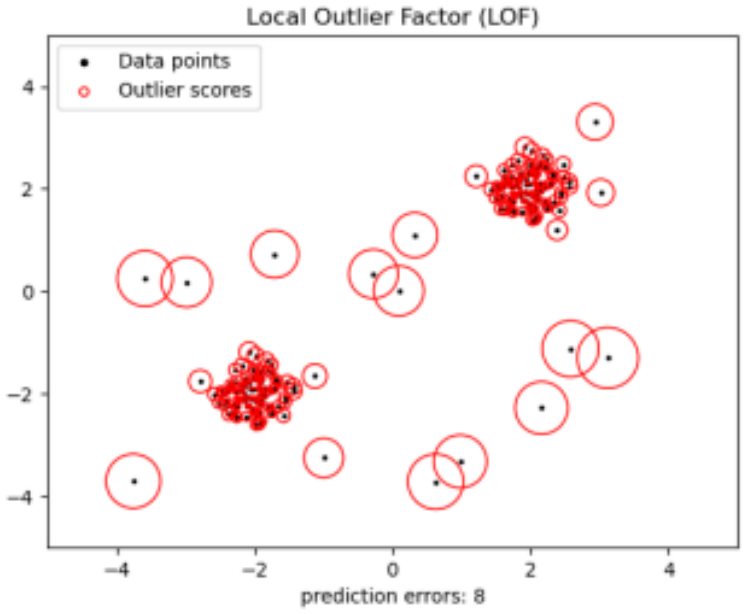

Рисунок 2.11 – Схематичне представлення алгоритму LOF

Переваги:

– складність – O(n*n);

– враховано недоліки метричних методів. Алгоритм враховує середню досяжність точки та її найближчих сусідів. У представників нормальних даних буде мала не лише оцінка локальної щільності, а й вона незначно відрізнятиметься від такої ж оцінки для найближчих сусідів.

3 ПРАКТИЧНА ЧАСТИНА

3.1 Дослідження методів детектування аномалій на реальних даних

Після розгляду основних методів виявлення аномалій необхідно перевірити їх на реальних даних для вибору моделі з найкращими показниками. Приклад було протестовано для БД PostgreSQL.

Для цього було сформовано 2 датасети:

– із зареєстрованими запитами (5928 рядків та 100 стовпців);

– із незареєстрованими запитами (14 рядків та 100 стовпців).

Результат їх порівняння наведено на рис. 3.1.

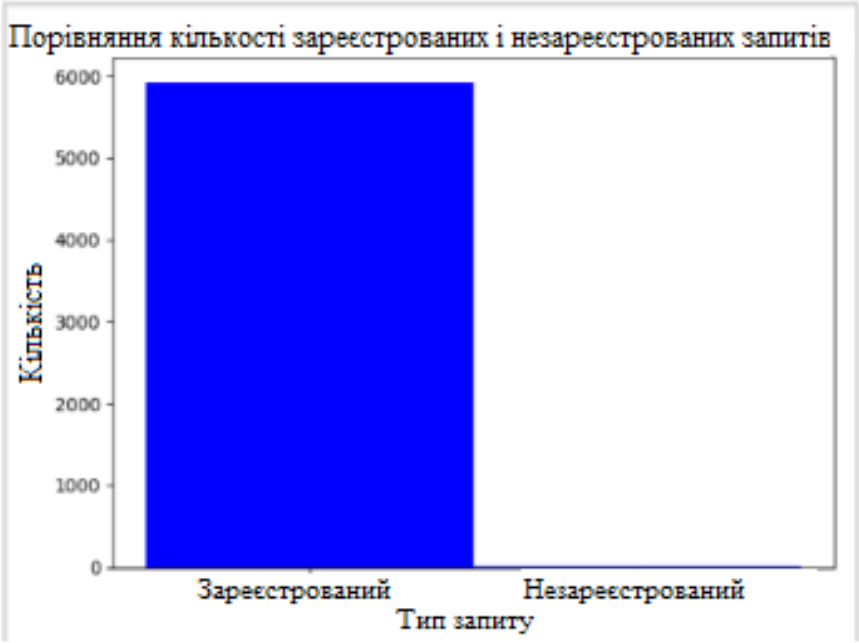

Рисунок 3.1 – Порівняння кількості зареєстрованих та незареєстрованих запитів

Кожен об'єкт – ознаковий опис SQL запиту. Як ознаки використовувалися 100 вихідних значень, за основу яких були взяті значення GLD. Для навчання моделі ми маємо лише приклади звичайних, неаномальних об'єктів, а прикладів аномальних настільки мало, що, як говорилося раніше, було прийнято рішення скористатися методами навчання без вчителя. Тому моделі були навчені лише з

прикладів зареєстрованого трафіку.

Якість моделей було оцінено за Recall, Precision та їхньою prodigy (harmonic mean) - оцінкою F1.

Recall (повнота) – це здатність алгоритму виявляти цей клас. Precision (точність) – здатність відрізняти цей клас від інших класів.

Для наочного представлення результатів роботи кожного класифікатора була побудована матриця неточностей (англ. Confusion matrix). Вона будується на основі підрахунку числа випадків, коли система прийняла правильне і неправильне рішення відносно справжнього значення класу належності:

– TP - істино-позитивне рішення;
– TN - істинно-негативне рішення;
– FP - хибно-позитивне рішення;
– FN - хибно-негативне рішення.

Значення Precision та Recall розраховується за формулами (3.1):

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN} \tag{3.1}$$

F1 - це метрика, що поєднує в собі інформацію про Precision і Recall алгоритму, через що набагато простіше прийняти рішення, про те чи є зміни в алгоритмі на краще чи ні. Значення F1 розраховується за формулою (3.2):

$$F1 = 2\frac{Precision \times Recall}{Precision + Recall} \tag{3.2}$$

Також для кращого значення F1 було оптимізовано гіперпараметри кожної моделі.

Розглянемо метрику Accuracy. Її значення розраховується за формулою (3.3):

$$Accuracy = P/N , \quad (3.3)$$

де P - число об'єктів, за якими класифікатор прийняв правильне рішення, а N - розмір навчальної вибірки.

Оскільки дані сильно незбалансовані та частка незареєстрованих запитів лише 0,23%, то accuracy не є гарною метрикою якості, тому що будь-яка модель, яка промаркує всі запити як зареєстровані, отримає все одно високий бал.

Перед побудовою моделей на основі вихідних датасетів були складені навчальна та тестова вибірки:

– датасет із зареєстрованим трафіком був розділений на навчальну та тестову вибірки так, щоб на 70% даних відбувалося навчання, а на 30% контроль;

– до тестової вибірки також було додано датасет із незареєстрованим трафіком.

Лістинг 3.1. Реалізація на Python

```
X_train, X_inliers = train_test_split(register, test_size=0.30, random_state=1)
X_outliers = unregister[:]
X_test = np.r_[X_inliers, X_outliers]
y_test = np.concatenate([np.zeros(len(X_inliers)), np.ones(len(unregister))])
```

У аналізі методів ML стосовно завдання виявлення незареєстрованого трафіку у роботі використовувалися такі методи:

– EllipticEnvelope;

– LOF;

– Isolation Forest;

– OCSVM.

Elliptic Envelope. Основні параметри реалізації sklearn.covariance. EllipticEnvelope:

– store_precision – true, якщо зберігається розрахункова точність;

– assume_centered - якщо False, надійне розташування та коваріація обчислюються безпосередньо за допомогою алгоритму FastMCD без додаткової

обробки. Якщо True, то обчислюється підтримка надійних оцінок місцезнаходження та підступності, і на її основі перераховується оцінка підступності без центрування даних;

– support_fraction - частка точок, які будуть включені на підтримку необробленої оцінки MCD. За замовчуванням - None , через що в алгоритмі використовуватиметься мінімальне значення support_fraction: [n_sample + n_features + 1] / 2;

– contamination – ступінь забруднення набору даних, тобто частка викидів у наборі даних.

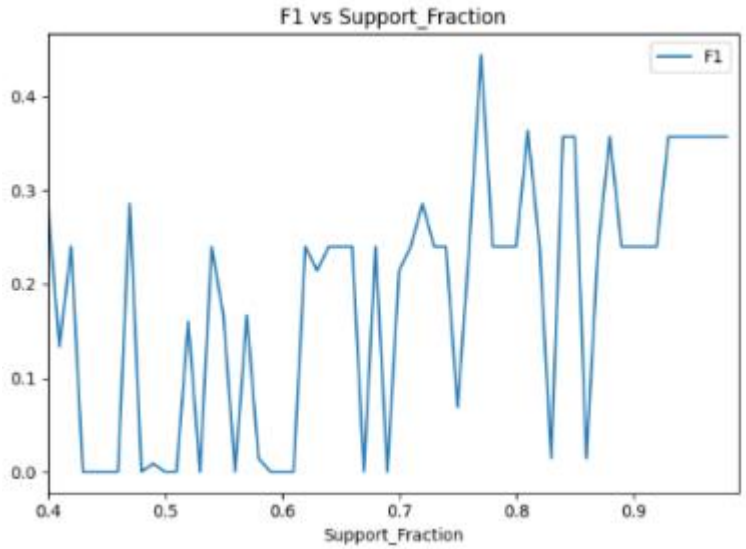

Рисунок 3.2 – Графік залежності F1 від параметра support_fraction

```
cov = EllipticEnvelope(support_fraction=0.774, contamination=PercFraud,
                       store_precision=True, assume_centered=False)
```

Таблиця 3.1 – Значення метрик точності, повноти та F - заходи

Precision	Recall	F1 score on Test
0.4615	0.4286	0.4444

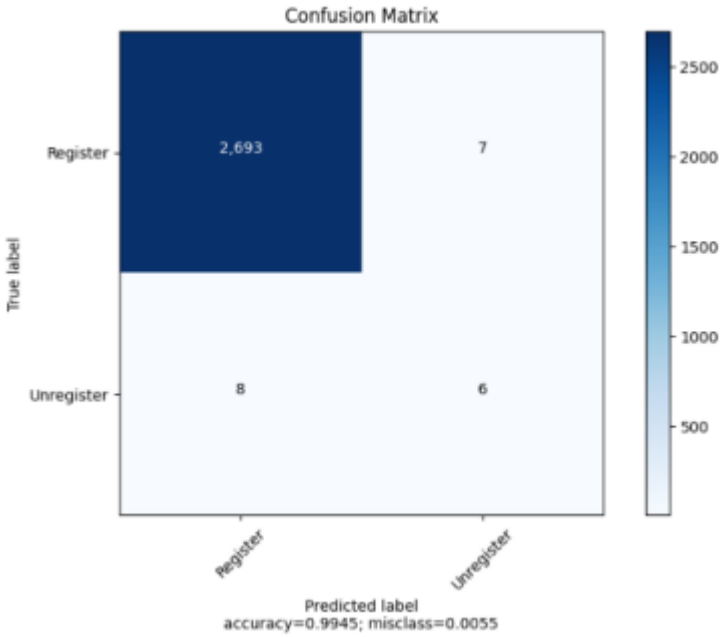

Рисунок 3.3 – Матриця неточностей для Elliptic Envelope

LOF. Основні параметри реалізації sklearn.neighbors.LOF:

– n_neighbors - кількість сусідів, які використовуються за замовчуванням для kneighbors запитів;

– algorithm - алгоритм, що використовується для обчислення найближчих сусідів ('ball_tree' буде використовувати BallTree, 'kd_tree' – KDTree, 'brute' використовуватиме пошук методом перебору, 'auto' спробує вибрати найбільш вдалий алгоритм з урахуванням значень, переданих в fit -метод);

– leaf_size - розмір листа;

– metric - метрика, що використовується для обчислення відстані. Можна використовувати будь-яку метрику із scikit-learn або scipy.spatial.distance;

– p - параметр для 'minkowski' метрики;

– metric_params – додаткові аргументи ключового слова для метричної функції;

– contamination - частка викидів у наборі даних.

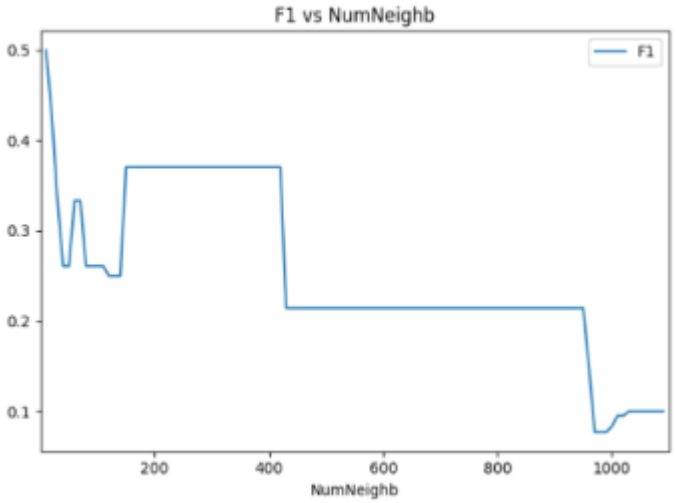

Рисунок 3.4 – Графік залежності F1 від параметра n_neighbors

```
clf = LocalOutlierFactor(n_neighbors=10, contamination=PercFraud, algorithm='auto',
                         leaf_size=30, metric='minkowski', p=2, metric_params=None)
```

Таблиця 3.2 – Значення метрик точності, повноти та F - заходи

Precision	Recall	F1 score on Test
0.6429	0.6429	0.6429

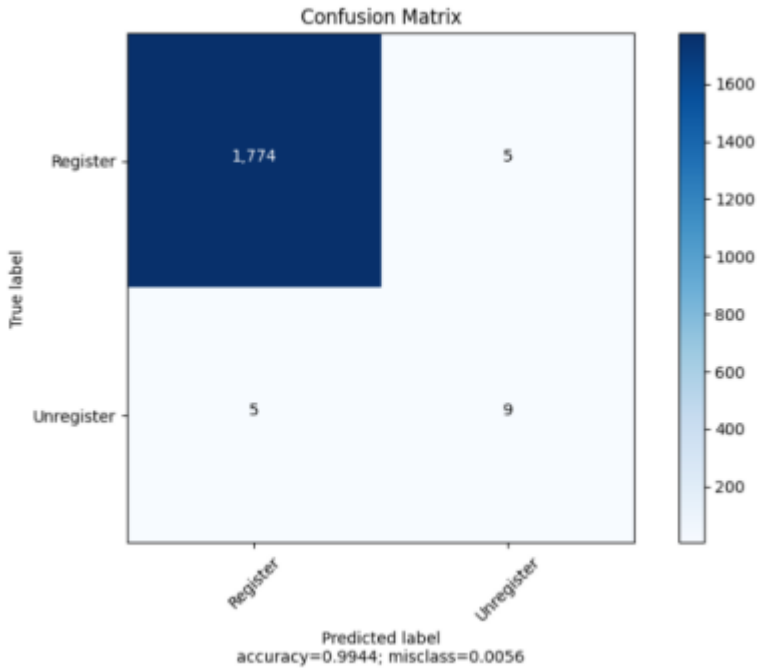

Рисунок 3.5 – Матриця неточностей для LOF

Isolation Forest. Основні параметри реалізації sklearn.ensemble. IsolationForest:

– n_estimators - кількість базових оцінювачів в ансамблі, інакше кажучи, кількість

дерев;

– max_samples - обсяг вибірки для побудови одного дерева;

– contamination - частка викидів у вибірці;

– max_features - кількість ознак, що використовуються при побудові одного дерева;

– bootstrap - якщо True, окремі дерева підходять для випадкових підмножин навчальних даних, вибраних із заміною. Якщо False, вибірка виконується без заміни;

– random_state - управляє псевдовипадковістю вибору функції та значень поділу для кожного кроку розгалуження та кожного дерева в лісі;

– verbose – управляє деталізацією процесу побудови дерева.

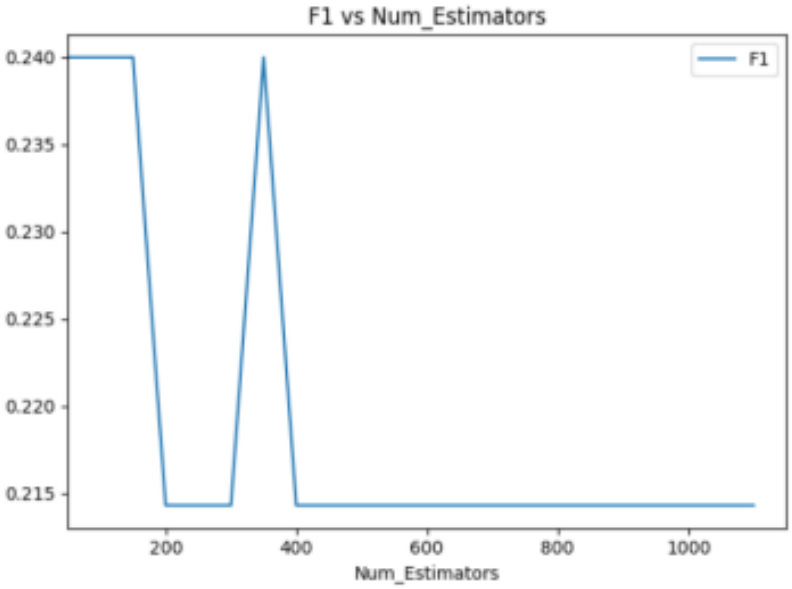

Рисунок 3.6 – Графік залежності F1 від параметра n_estimators

```
isf = IsolationForest(n_estimators=50, max_features=1.0, max_samples=1.0,
                      bootstrap=False, random_state=22, contamination=PercFraud,
                      verbose=0)
```

Таблиця 3.3 – Значення метрик точності, повноти та F - заходи

Precision	Recall	F1 score on Test
0.3571	0.3571	0.3571

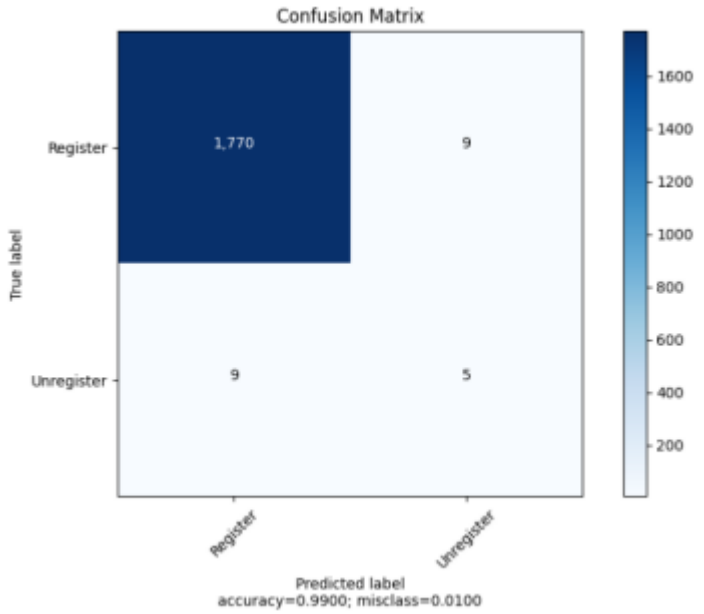

Рисунок 3.7 – Матриця неточностей для Isolation Forest

OCSVM. Основні параметри реалізації sklearn.svm.OCSVM:

– kernel - задає тип ядра, який використовуватиметься в алгоритмі (linear - лінійне, poly - поліноміальне, rbf - радіальні базисні функції, sigmoid - сигмоїдальне, своє задане);

– nu - верхня межа на % помилок та нижня на % опорних векторів (за замовчуванням 0.5);

– degree - ступінь поліноміальної ядерної функції;

– gamma - коефіцієнт ядра для rbf, poly, sigmoid (за замовчуванням 1/n_features). Визначає згладжування контурних ліній;

– max_iter - обмеження кількості ітерацій в solver. Якщо необхідно без обмеження, потрібно виставити значення -1.

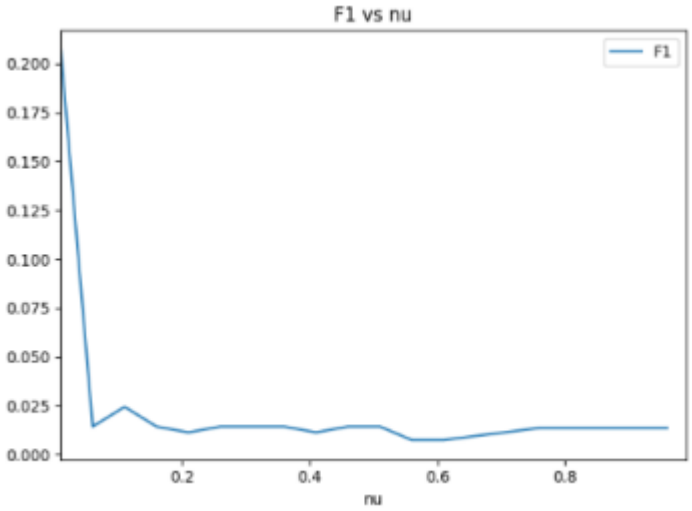

Рисунок 3.8 – Графік залежності F1 від параметра nu

```
OneSVM = OneClassSVM(kernel='rbf', nu=0.01, degree=3, gamma=0.1, max_iter=-1)
```

Таблиця 3.4 – Значення метрик точності, повноти та F - заходи

Precision	Recall	F1 score on Test
0.2	0.2143	0.2069

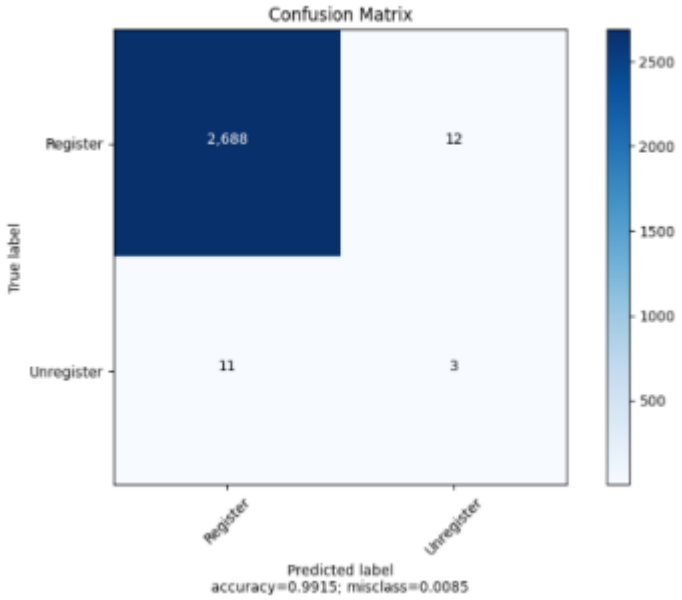

Рисунок 3.9 – Матриця неточностей для OCSVM

У таблиці 3.5 наведено порівняння роботи алгоритмів.

Таблиця 3.5 – Порівняння метрик якості алгоритмів

	precision	recall	F1-score	Оптимізація
Local Outlier Factor	0.6429	0.6429	0.6429	number of neighbors
Elliptic Envelope	0.4615	0.4286	0.4444	support fraction
Isolation Forest	0.3571	0.3571	0.3571	num estimators
One Class SVM	0.2	0.2143	0.2069	nu

За результатами дослідження методів виявлення аномалій реальних даних, найкращий результат показує LOF. Отже, у додатку для виявлення незареєстрованого трафіку буде використовуватися саме цей метод.

3.2 Моніторинг БД та переривання запитів

У PostgreSQL всі активні запити будемо отримувати за допомогою інструмента pg_stat_activity - це системне представлення, що дозволяє відстежувати процеси БД у режимі реального часу. Розглянемо докладніше стовпці pid, state та query, які є найважливішими для реалізації:

Таблиця 3.6 – Подання pg_stat_activity (стовпці pid, state та query)

Стовпець	Тип	Опис
pid	integer	Ідентифікатор процесу серверного процесу
state	text	Загальний стан серверного процесу. Можливі значення: active, idle, idle in transaction, idle in transaction (aborted), fastpath function call, disabled.
query	text	Текст останнього запиту серверного процесу

Тобто ми опитуємо представлення pg_stat_activity кожну секунду та

отримуємо інформацію про запущені процеси (розглядаємо ті, у кого state = 'active '). Потім з отриманої інформації отримуємо query, представляємо його у векторній формі і передаємо навченої моделі LOF для того, щоб отримати передбачення: якщо запит відноситься до зареєстрованого трафіку, метод predict() поверне 1, якщо є аномалією і не характерний для зареєстрованого трафіку , то буде повернуто -1, у разі ми вбиваємо запит з допомогою методу pg_cancel_backend().

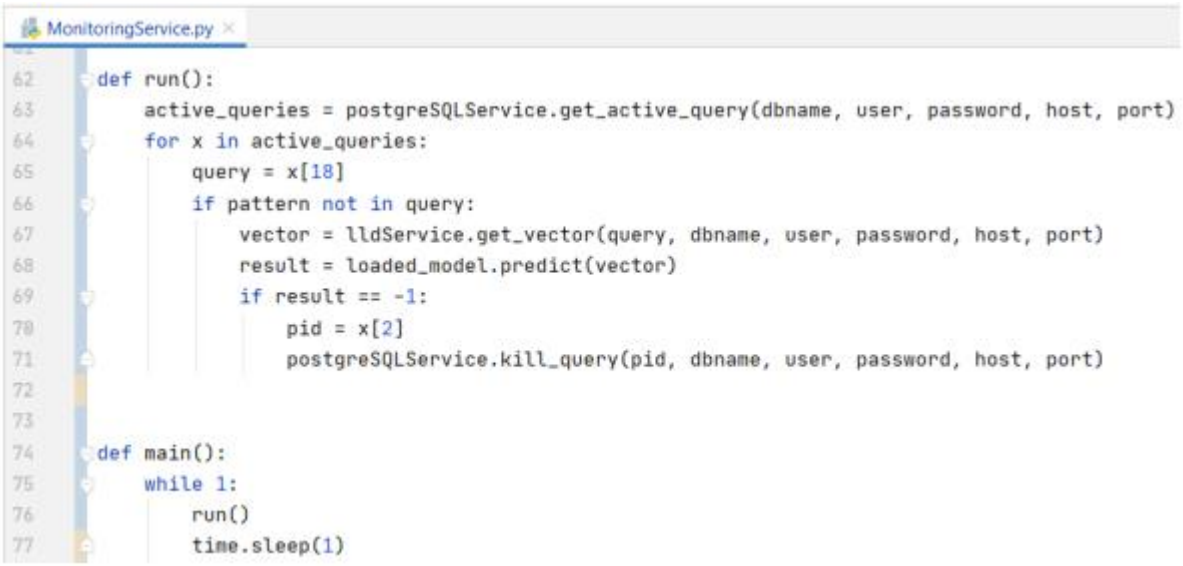

Рисунок 3.9 – Фрагмент коду, що демонструє моніторинг БД та подальші дії, залежно від типу трафіку

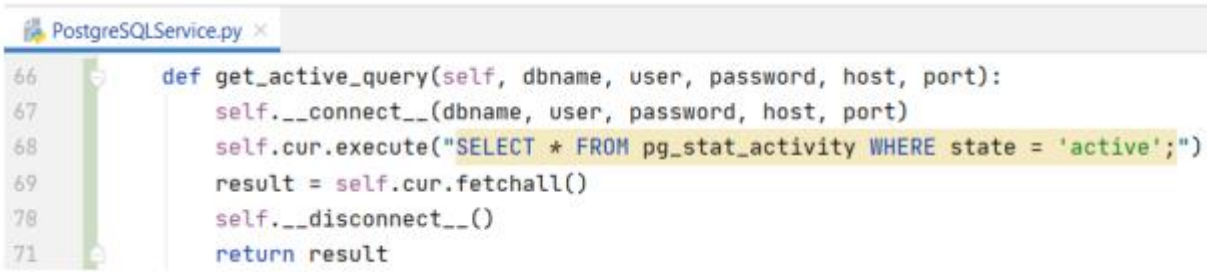

Рисунок 3.10 – Фрагмент коду, який демонструє роботу з pg_stat_activity

Рисунок 3.11 – Фрагмент коду, що демонструє переривання запиту

У MySQL всі активні запити будемо отримувати за допомогою таблиці INFORMATION_SCHEMA.PROCESSLIST та запиту *“show processlist”*. Цей запит повертає потоки, які виконуються в даний час, тому немає необхідності фільтрувати значення стовпця state. Розглянемо докладніше стовпці id та info, які є найбільш важливими для реалізації

Таблиця 3.7 – Інформація про стовпці id та info таблиці

Стовбець	Опис
Id	Ідентифікатор підключення
Info	Оператор, який виконується потоком, або NULL, якщо він не виконує жодних операторів

Загальна концепція реалізації аналогічна, що і для PostgreSQL, тільки для переривання запиту буде використовуватися команда kill().

MonitoringService.py

```
14      def run(self, dbname, user, password, host, port, pattern):
15          active_queries = mySQLService.get_active_query(dbname, user, password, host, port)
16          for x in active_queries:
17              query = x[7]
18              if pattern not in query:
19                  vector = lldService.get_vector(query, dbname, user, password, host, port)
20                  result = loaded_model.predict(vector)
21                  if result == -1:
22                      id = x[0]
23                      mySQLService.kill_query(id, dbname, user, password, host, port)
24
25
26      def main(self, dbname, user, password, host, port, pattern):
27          while 1:
28              self.run(dbname, user, password, host, port, pattern)
29              time.sleep(1)
```

Рисунок 3.12 – Частина коду, що демонструє моніторинг БД та подальші дії, залежно від типу трафіку

MySQLService.py

```
46      def get_active_query(self, dbname, user, password, host, port):
47          result = []
48          try:
49              conn = mysql.connector.connect(host=host, database=dbname, user=user,
50                                             password=password, port=port)
51              cursor = conn.cursor()
52              cursor.execute("show processlist")
53              result = cursor.fetchall()
54              conn.commit()
55          except Error as e:
56              print(e)
57          finally:
58              cursor.close()
59              conn.close()
60          return result
```

Рисунок 3.13 – Фрагмент коду, що демонструє роботу з таблицею processlist

MySQLService.py

```
62      def kill_query(self, id, dbname, user, password, host, port):
63          try:
64              conn = mysql.connector.connect(host=host, database=dbname, user=user,
65                                             password=password, port=port)
66              cursor = conn.cursor()
67              sql = '''kill ''' + id
68              cursor.execute(sql)
69              conn.commit()
70          except Error as e:
71              print(e)
72          finally:
73              cursor.close()
74              conn.close()
```

Рисунок 3.14 – Частина коду, що демонструє переривання запиту

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Стихійні лиха та їх класифікація

Стихійні дії сил природи, поки що не повною мірою підвладні людині та щорічно завдають державі і населенню величезних збитків. Стихійні лиха - це такі явища природи, що викликають екстремальні ситуації, порушують нормальну життєдіяльність населення, роботу безлічі об'єктів. Стихійні лиха є трагедією для будь-якої держави.Через стихійні лиха страждає економіка країни, бо при цьому руйнуються виробничі підприємства, знищуються матеріальні цінності, гинуть люди.

Стихійні лиха - небезпечні природні явища, як правило раптового походження, хоча іноді і прогнозовані за допомогою метеорогогії, але на інтенсивність яких люди впливати не можуть. Їх можна класифікувати: за швидкістю переміщення - землетруси, зсуви, цунамі, снігопади, ожеледі - швидкі; підвищення рівня води в ріках через інтенсивні опади або танення снігу, льоду (повіні), звільнення внутрішньої енергії Землі, виверження вулканів - повільні. Часто виникають потужні, високошвидкісні потоки повітря через швидкий перепад значень атмосферного тиску (урагани, смерчі і т.п.). Стихійні лиха речовинного характеру можуть ініціювати виникнення різноманітних полів, які негативно впливають на здоров'я, самопочуття людини. [26].

Стихійні явища часто виникають в комплексі, що значно посилює їх негативний вплив. Небезпечні природні явища визначаються трьома основними групами процесів - ендогенні, екзогенні та гідрометеорологічні.

Стихійні лиха, які характерні для України, за структурою можна поділити на прості, що включають один елемент - наприклад, сильний вітер, зсув або землетрус та складні. Вони складаються з декількох процесів однієї групи або кількох груп. Найбільші збитки спричиняють повені - 40%, на другому місці - циклони (20%), на третьому - посухи та землетруси (15%). Деякі стихійні лиха (пожежі, обвали, зсуви і навіть землетруси) можуть виникати в результаті дій самих людей, тобто мають антропогенне походження, але наслідки їх завжди є

діями сил природи. Для кожного стихійного лиха характерна наявність властивих йому вражаючих чинників, що несприятливо впливають на стан здоров'я, життя людини [27].

Причинами стихійних лих можуть бути:

– швидке переміщення речовини (землетрусу, зсуви);

– вивільнення внутриземної енергії (вулканічна діяльність, землетруси);

– підвищення рівня вод річок, ставків і морів (повені, цунамі);

– вплив надзвичайно сильного вітру (урагани, торнадо, циклони).

Важливо своєчасно провести роботи, спрямовані на локалізацію природного лиха, щоб зменшити зони руйнувань, звести до мінімуму кількість загиблих та постраждалих.

В Україні найчастіше спостерігаються такі надзвичайні ситуації природного характеру:

– небезпечні геологічні явища (зсуви, обвали, осипки, просадки земної поверхні);

– небезпечні метеорологічні явища (зливи, урагани, сильні снігопади, сильний град, ожеледь);

– небезпечні гідрологічні явища (повені, паводки);

– природні пожежі лісових та торф’яних масивів;

– масові інфекції та хвороби людей, тварин, рослин.

В останні роки кількість стихійних лих в Україні та в світі в цілому значно збільшилася. Найчастіше в Україні виникають такі природні катастрофи як землетруси, повені, посухи (на Півдні України), лісові пожежі в літню пору року, снігові замети, зсуви поверхні.

Є серйозні підстави вважати, що масштабність впливу лиха й катастроф на соціальні, економічні, політичні та інших процесів сучасного нашого суспільства та їх драматизм вже перевищили такий рівень, який дозволяв ставитися до них як до локальних збоїв у розміреному функціонуванні державних та громадських структур [26].

Отже, перед людиною та громадськістю в XXI в. вимальовується нова мета - глобальна безпека. Досягти цього можна, в першу чергу, за допомогою зміни

світогляду людини, а також покращення системи профілактичних заходів у боротьбі зі стихійними лихами, а саме: вдосконалення рятувальних служб та рятувальної техніки, проведення попереджувальних заходів та пропагандистської роботи з громадянами щодо правил поведінки та дій під час стихійних лих. Це допоможе в майбутньому зменшити кількість загиблих та постраждалих від природних катастроф, а також зменшить матеріальні збитки, що були завдані стихійним лихом.

Природні лиха з часом нікуди не зникнуть. Будуть виникати землетруси в геологічно активних районах, будуть виникати повені, а штормові припливи стануть, раз у раз затопляти морські узбережжя, не обійдеться і пожеж. Людина безсила запобігти природним процесам, але тільки в наших силах зменшити кількість жертв і матеріальних втрат.

4.2 Соціальне значення охорони праці

Соціальне значення охорони праці полягає в сприянні росту ефективності суспільного виробництва шляхом безперервного вдосконалення і поліпшення умов праці, підвищення їх безпеки, зниження виробничого травматизму і профзахворювань [28]. Соціальне значення охорони праці проявляється в зростанні продуктивності праці, збереженні трудових ресурсів і збільшенні сукупного національного продукту.

Охорона праці полягає в сприянні росту ефективності виробництва, яке досягається шляхом безперервного вдосконалення і поліпшення умов праці, підвищення їх безпеки, зниження виробничого травматизму і профзахворювань.

Зростання продуктивності праці відбувається в результаті збільшення фонду робочого часу завдяки скороченню внутрішньо-змінних простоїв шляхом ліквідації мікротравм або зниження їх кількості, а також завдяки запобіганню передчасного стомлення шляхом раціоналізації і покращення умов праці та введенню оптимальних режимів праці і відпочинку та інших заходів, які сприяють підвищенню ефективності використання робочого часу.

Важливим питанням є зростання продуктивності праці, яка відбувається в

результаті збільшення фонду робочого часу завдяки скороченню внутрішньозмінних простоїв шляхом ліквідації мікротравм або зниження їх кількості, а також завдяки запобіганню передчасного стомлення шляхом раціоналізації і покращення умов праці та введенню оптимальних режимів праці і відпочинку та інших заходів, які сприяють підвищенню ефективності використання робочого часу [28].

Особливої уваги заслуговує те, що збереження трудових ресурсів і підвищення професійної активності працюючих відбувається завдяки покращенню стану здоров'я і подовженню середньої тривалості життя шляхом покращення умов праці, що супроводжується високою трудовою активністю і підвищенням виробничого стажу. Підвищується професійний рівень також завдяки зростанню кваліфікації і майстерності. Відповідно і збільшення сукупного національного продукту відбувається завдяки покращенню вищеперелічених показників та їх складових компонентів [29]. Збереження трудових ресурсів і підвищення професійної активності працюючих відбувається завдяки покращенню стану здоров'я і подовженню середньої тривалості життя шляхом покращення умов праці, що супроводжується високою трудовою активністю і підвищенням виробничого стажу. Підвищується професійний рівень також завдяки зростанню кваліфікації і майстерності. Збільшення сукупного національного продукту відбувається завдяки покращенню вищеперелічених показників та їх складових компонентів. Крім того, соціальне значення охорони праці проявляється в зростанні продуктивності праці, збереженні трудових ресурсів.

Комплекс заходів з поліпшення умов праці може забезпечити приріст продуктивності праці на 15-20%. Так, нормалізація освітлення робочих місць збільшує продуктивність на 6-13% та скорочує брак на 25%. Раціональна організація робочого місця підвищує продуктивність праці на 21%, раціональне фарбування робочих приміщень – на 25% [29]. Збільшення ефективного фонду робочого часу може бути досягнуто за рахунок скорочення тимчасової непрацездатності працівників внаслідок хвороб та виробничого травматизму.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було розроблено інструмент для визначеного бізнес-трафіку, який дозволить не допускати стрибки продуктивності та одразу блокувати такий незареєстрований трафік у БД PostgreSQL та MySQL.

Для цього було виконано такі завдання:

– проаналізовано аналоги;

– підготовлено дані для ML та тестування роботи інструменту;

– реалізовано подання запитів у векторній формі;

– вивчено та порівняно сучасні методи виявлення аномалій, обрано метод із найкращим результатом - LOF;

– спроектовано архітектуру, програмно реалізовано додаток блокування незареєстрованого трафіку, який є потенційно шкідливим для продуктивності БД.

Таким чином, всі поставлені у кваліфікаційній роботі завдання були успішно виконані.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. How Loading Time Affects Your Bottom Line [Електронний ресурс]. - Режим доступа: https://neilpatel.com/blog/loading-time/ (дата звернення 22.01.2022).

2. Ключові слова SQL [Електронний ресурс]. - Режим доступа: https:// https://w3schoolsua.github.io/sql/sql_ref_keywords.html (дата звернення 20.02.2022).

3. Keywords and Reserved Words [Електронний ресурс]. - Режим доступа: https://dev.mysql.com/ doc/refman/ 8.0/ en/keywords .html (дата звернення 20.02.2022).

4. Векторні моделі [Електронний ресурс]. - Режим доступа: https://uk.education-wiki.com/1309569-types-of-machine-learning (дата звернення 07.02.2022).

5. Обмеження PostgreSQL [Електронний ресурс]. - Режим доступа: https://postgrespro.ru/docs/postgresql/12/limits (дата звернення 03.03.2022).

6. Limits on Table Column Count and Row Size [Електронний ресурс]. - Режим доступа: https://dev.mysql.com/doc/refman/8.0/en/column-count-limit.html (дата звернення 03.03.2022).

7. Liu, T., Ting, KM, і Zhou, Z. Isolation Forest // Proceeding of the IEEE International Conference on Data Mining. - 2008. - Pp. 413-422.

8. Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, Jörg Sander. LOF: identifying density-based local outliers // ACM sigmod record. – ACM, 2000. – Т. 29. – №. 2. – С. 93-104.

9. Hodge, V. and Austin, J. A survey of outlier detection methodologies // Artificial Intelligence Review. - 2004. - Vol. 22, no. 2. - Pp. 85-126.

10. Omar, S., Ngadi, A., і Jebur, H. Machine Learning Techniques for Anomaly Detection: An Overview // International Journal of Computer Applications. - 2013. - Vol. 79, no. 2. - Pp. 33-41.

11. Aggarwal, CC: Outlier Analysis. SpringerVerlag, New York (2013). doi: 10.1007/ 978-1-4614-6396-2

12. Ho, Tin Kam. Random Decision Forests // Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC. - 1995. - Pp. 278-282.

13. R. Schalkoff, Pattern Recognition, Statistical, Structural and Neural Approach, John Wiley, New York, NY, USA, 1991

14. T.Kohonen, "Self-Organizing Maps", 2nd.Ed.,Springer, 1997

15. Knorr EM, Ng RT, Tucakov V. Distance-based outliers: algorithms and applications // The VLDB Journal—The International Journal on Very Large Data Bases. – 2000. – Т. 8. – №. 3-4. – С. 237-253.

16. Streamlit [Електронний ресурс] // Офіційний сайт Streamlit – Режим доступа: https://streamlit.io (дата звернення 03.05.2022)

17. PyCharm [Електронний ресурс] // Офіційний сайт PyCharm - Режим доступа : https://wwwjetbrains.com/ru-ru/pycharm (дата звернення 03.05.2022).

18. He Z. та ін. Fp-outlier: Frequent pattern based outlier detection // Computer Science and Information Systems. – 2005. – Т. 2. – №. 1. – С. 103-118

19. Метрики у завданнях машинного навчання [Електронний ресурс] - Режим доступа: https://habr.com/ru/company/ods/blog/328372/ (дата звернення 03.05.2022).

20. Підготовка даних для алгоритмів машинного навчання [Електронний ресурс] - Режим доступа: http:// http://specials.kunsht.com.ua/machinelearning2 (дата звернення 13.04.2022).

21. А. І. Мурзіна, “Про метод машинного навчання виявлення аномалій в SQL -запросах”, ПДМ. Додаток, 2017 № 10, 121-122

22. Моніторимо активні сесії PostgreSQL 10, як у Oracle [Електронний ресурс] - Режим доступа: https://habr.com/ru/post/413411/ (дата звернення 25.04.2022).

23. System Administration Functions [Електронний ресурс] - Режим доступа: https://www.postgresql.org/docs/12/functions-admin.html (дата звернення 02.05.2022).

24. Accessing the Process List [Електронний ресурс] - Режим доступа: https://dev.mysql.com/doc/refman/8.0/en/processlist-access.html (дата звернення

05.05.2022).

25. Show Processlist Statement [Електронний ресурс] - Режим доступа: https://dev.mysql.com/doc/refman/8.0/en/show-processlist.html (дата звернення 05.05.2022).

26. Стеблюк М.І. Цивільна оборона: Підручник. – Знання, 2006. – 487 с.

27. Толок А.О. Крюковська О.А. Безпека життєдіяльності: Навч. посібник. – 2011. – 215 с.

28. Агєєв Є .Я. Основи охорони праці: Навчально-методичний посібник для самостійної роботи по вивченню дисципліни – Львів: «Новий Світ – 2000», 2009. – 404 с.

29. Основи охорони праці: Підручник.; 3-те видання / За ред. К. Н Ткачука. – К.: Основа, 2011. – 480 с.