**Your University**

**Faculty of Commerce**

|  |
| --- |
| **Network Security Log Analysis** |
| **اسم المشروع باللغة العربية** |

**Graduation Project**

Submitted to the

Your Faculty

Supervised by:

**Dr./ Supervisor**

Cairo, Egypt

**2023/2024**

**Project Name**

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |

**Team Work: -**

**Supervisor: Dr.**

**Table of Contents**

[List of Figures](#_Toc487175731) 3

[List of Tables](#_Toc487175731) 4

[List of abbreviation](#_Toc487175731) 5

[Acknowlegdement 6](#_Toc487175731)

[Abstract](#_Toc487175731) 7

[Chapter .1 Intrduction 9](#_Toc487175731)

Chapter 2 [Tools and Programming languages 16](#_Toc487175731)

Chapter 3 Systems Development………..……………………………………………….24

[Chapter 4 Features and Implementation 40](#_Toc487175731)

Chapter 5 [Conclusions and Future Work 50](#_Toc487175731)

[Chapter 6 Reefrences 70](#_Toc487175731)

Appendix A: Terms and Difinions……………………………………………………….72

**List of Figures**

[Figure 1:Scrum framework](#_Toc487175731) 27

[Figure 2: Scrum value](#_Toc487175731) 28

[Figure 3: Welcome Screen Activity](#_Toc487175731) 52

[Figure 4: User Profile 54](#_Toc487175731)

**List of Table**

[Table 1:Scrum framework](#_Toc487175731) 27

[Table 2: Scrum value](#_Toc487175731) 28

[Table 3: Welcome Screen Activity](#_Toc487175731) 52

**List of Abbreviations**

[Table 1:Scrum framework](#_Toc487175731) 27

[Table 2: Scrum value](#_Toc487175731) 28

[Table 3: Welcome Screen Activity](#_Toc487175731) 52

**Acknowledgement**

The success of this project will be incomplete without the help of people who made their best with our team by their effort and their experience. We should mention people whose constant guidance and encouragement lead to our success. Mostly, Thanks and praises to Allah who always supports and guides us with assistance and bless.

We are grateful to **Dr………………………..,** the supervisor of our project for his valuable advice, guidance, inspiration, and constructive suggestions that was helpful in the preparation of this project. The opportunity to work with him was a precious experience, he exerts his time to help us to learn, search, and support us to do our best.

**Abstract**

In the realm of cyber security, log analysis serves as a cornerstone for identifying potential threats, detecting anomalies, and ensuring the integrity of computer networks. This project delves into the domain of log analysis, focusing on leveraging machine learning algorithms for anomaly detection within log data. The project commences with the acquisition and preprocessing of log data extracted from various sources, such as system logs, network logs, and application logs. The dataset encompasses a multitude of log attributes including timestamps, event types, user activities, and system states.

Subsequently, an array of machine learning models is trained and evaluated on the preprocessed log data to discern abnormal patterns indicative of potential security breaches or irregular system behavior. Decision trees, support vector machines, and ensemble methods are among the models employed for this purpose.

Evaluation metrics such as accuracy, precision, recall, and F1-score are employed to assess the performance of the models in accurately identifying anomalies within the log data. Additionally, techniques such as cross-validation and hyper parameter tuning are utilized to enhance model robustness and efficacy.

The project culminates in the analysis of experimental results, showcasing the efficacy and applicability of machine learning-based anomaly detection techniques in log analysis for cybersecurity purposes. Insights gained from this endeavor provide valuable contributions to the advancement of cyber security practices, particularly in the realm of log analysis and anomaly detection.

**Chapter 1**

**Introduction**

**1. Introduction**

In today's digitally driven world, cyber security stands as an essential pillar in safeguarding sensitive information, critical infrastructure, and ensuring the integrity of computer networks. With the proliferation of interconnected devices and the increasing sophistication of cyber threats, the need for robust security measures has become more imperative than ever.

Central to the realm of cyber security is the analysis of logs generated by various components of computer systems, including operating systems, network devices, and applications. Log data serves as a rich source of information, capturing a plethora of events, activities, and system states occurring within an IT environment. By scrutinizing log data, cyber security professionals can gain valuable insights into potential security breaches, anomalous behavior, and emerging threats.

Log analysis encompasses a diverse range of tasks, including log collection, preprocessing, correlation, and interpretation. However, one of the most critical aspects of log analysis is anomaly detection – the identification of abnormal patterns or deviations from expected behavior within log data. Anomalies in log data could signify potential security incidents, unauthorized access attempts, system misconfigurations, or malicious activities.

Traditionally, anomaly detection in log data has relied on rule-based methods, threshold-based techniques, and manual inspection by cybersecurity analysts. While effective to some extent, these approaches often struggle to cope with the scale, complexity, and dynamic nature of modern IT environments. As a result, there is a growing interest in leveraging machine learning algorithms for automated anomaly detection in log data.

Machine learning offers a promising avenue for anomaly detection in log data, enabling the automated identification of subtle, complex, and previously unseen patterns indicative of security threats. By training machine learning models on historical log data, organizations can develop robust anomaly detection systems capable of proactively identifying and mitigating security risks.

This project aims to explore the application of machine learning techniques for anomaly detection in log data, with a focus on enhancing cyber security practices. By leveraging the power of machine learning algorithms, we seek to develop a scalable, efficient, and accurate anomaly detection system capable of bolstering the security posture of organizations in the face of evolving cyber threats. Through empirical experimentation and analysis, we aim to gain insights into the effectiveness of machine learning-based approaches for log analysis and anomaly detection, ultimately contributing to the advancement of cyber security practices in an increasingly digitized world.

* 1. **Problem Statement :**

The problem at hand revolves around the imperative need for robust and efficient anomaly detection within log data to bolster cyber security measures. Despite the advancements in cyber security practices, organizations continue to face challenges in effectively identifying and mitigating security threats within their IT environments. Traditional methods of anomaly detection often fall short in coping with the scale, complexity, and dynamic nature of modern IT infrastructures, leading to potential security breaches and vulnerabilities.

The primary objective of this project is to develop and evaluate a machine learning-based anomaly detection system tailored for log analysis in cyber security. Specifically, we aim to address the following key aspects:

**Data Collection and Preprocessing**: Acquire and preprocess log data from diverse sources, including system logs, network logs, and application logs. This involves parsing, cleaning, and transforming raw log data into a structured format suitable for analysis.

**Feature Engineering:** Extract relevant features from the preprocessed log data that can effectively capture patterns indicative of anomalous behavior. These features may include timestamps, event types, user activities, system states, and other relevant attributes.

**Model Development:** Train and evaluate machine learning models for anomaly detection using the preprocessed log data. Explore a range of algorithms, including decision trees, support vector machines, random forests, and neural networks, to identify the most effective approach for detecting anomalies within the log data.

**Performance Evaluation:** Assess the performance of the developed anomaly detection system using appropriate evaluation metrics, such as accuracy, precision, recall, and F1-score. Conduct comprehensive experiments and cross-validation to validate the robustness and efficacy of the system in detecting anomalies across different types of log data.

**Comparative Analysis:** Compare the performance of the machine learning-based anomaly detection system with traditional rule-based methods and threshold-based techniques. Identify the strengths and limitations of each approach and provide insights into the applicability of machine learning in enhancing anomaly detection capabilities within log data.

**Practical Implications:** Discuss the practical implications of the developed anomaly detection system for real-world cyber security applications. Highlight potential use cases, deployment strategies, and recommendations for organizations looking to enhance their security posture through automated log analysis.

By addressing these key aspects, this project aims to contribute to the advancement of anomaly detection techniques in log analysis and provide valuable insights into leveraging machine learning for enhancing cybersecurity practices. Ultimately, the goal is to develop a scalable, efficient, and accurate anomaly detection system capable of proactively identifying and mitigating security threats within complex IT environments.

**1.3 Objectives:**

The objectives of the project include:

**Data Acquisition and Preprocessing:** Gather log data from various sources, preprocess it to extract relevant features, and prepare it for analysis.

**Model Development:** Train machine learning models on preprocessed log data to detect anomalies and abnormal patterns indicative of security threats.

**Model Evaluation**: Assess the performance of the developed models using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score.

**Hyperparameter Tuning:** Fine-tune the hyperparameters of the machine learning models to optimize their performance and robustness.

**Deployment and Integration:** Deploy the developed anomaly detection system in a real-world cybersecurity environment and integrate it with existing security infrastructure.

**Evaluation and Validation**: Validate the effectiveness and efficacy of the deployed system through empirical evaluation and validation against known security incidents and test scenarios.

Documentation and Knowledge Sharing: Document the development process, findings, and insights gained from the project to facilitate knowledge sharing and dissemination within the cybersecurity community.

**Chapter 2**

**Tools and Programming Languages**

* **Frontend**(HTML-CSS-JavaScript)
* **Backend**(Flask or Django-Machine Learning)

**INTRODUCTION**

**1. FrontEnd :**

* **HTML:** HyperText Markup Language is used for structuring the web page.
* **CSS:** Cascading Style Sheets are used for styling the HTML elements and adding visual enhancements.
* **JavaScript**: A scripting language used to add interactivity to web pages. In this case, JavaScript is used to handle form submission, make AJAX requests to the server, update the progress bar, and handle button clicks.
* **jQuery**: A fast, small, and feature-rich JavaScript library. It simplifies things like HTML document traversal and manipulation, event handling, and animation. In this code, jQuery is used for DOM manipulation, event handling, and AJAX requests.
* **Bootstrap**: A popular CSS framework used for designing responsive and mobile-first websites. It provides pre-designed CSS styles and JavaScript plugins for creating responsive layouts and components. In this code, Bootstrap is used for styling the form elements and adding progress bars.
* **Font Awesome**: A font and icon toolkit based on CSS and LESS. It provides scalable vector icons that can be customized with CSS. In this code, Font Awesome is used for adding icons to various elements in the web page.

**2. Back End :**

* **Flask or Django:** These are popular Python web frameworks used for building web applications. In this case, it's likely that Flask is being used due to its simplicity and lightweight nature. Flask allows you to create web applications and APIs easily.
* **Scikit-learn:** This is a widely-used Python library for machine learning. It provides simple and efficient tools for data mining and data analysis. In the context of the provided code, Scikit-learn could be used to train a machine learning model for anomaly detection based on the provided input data.
* **Pandas:** Pandas is another Python library often used in conjunction with Scikit-learn for data preprocessing and manipulation. It provides data structures andfunctionsto work with structured data, which could be helpful in preparing the input data for training the machine learning model.
* **NumPy:** NumPy is a fundamental package for scientific computing with Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. NumPy could be used for numerical computations involved in data preprocessing and feature engineering.
* **Matplotlib :** These are Python libraries used for data visualization. They provide functions to create various types of plots and charts, which could be useful for analyzing the data before and after training the machine learning model.

.

**2.2 Related works**

Related works in the field of log analysis and anomaly detection in cyber security encompass a broad range of research and practical applications. Here are some notable areas and examples of related works:

Machine Learning-Based Anomaly Detection: Numerous studies have explored the application of machine learning algorithms for anomaly detection in log data. Research papers often focus on comparing different machine learning techniques, evaluating their performance on benchmark datasets, and proposing novel algorithms for improved anomaly detection accuracy and efficiency.

Deep Learning Approaches: Deep learning techniques, such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and autoencoders, have gained attention for their ability to extract complex patterns and representations from log data. Researchers investigate the effectiveness of deep learning models for log analysis and anomaly detection, exploring architectures, training strategies, and optimization techniques.

Feature Engineering and Selection: Feature engineering plays a crucial role in log analysis and anomaly detection. Studies explore various feature extraction techniques, feature selection methods, and representation learning approaches to capture relevant information from log data and improve the performance of anomaly detection models.

Ensemble Methods and Meta-Learning: Ensemble methods, including random forests, gradient boosting, and ensemble learning techniques, have been applied to combine multiple anomaly detection models and improve overall detection performance. Meta-learning approaches aim to adaptively select and combine models based on the characteristics of the input log data and the specific detection task.

Real-Time Anomaly Detection: Real-time anomaly detection is essential for detecting and responding to security threats as they occur. Research in this area focuses on developing scalable and efficient algorithms, stream processing techniques, and distributed systems for performing anomaly detection in high-speed data streams generated by diverse sources.

Anomaly Interpretability and Explainability: Interpretable anomaly detection models are crucial for understanding the underlying causes of detected anomalies and taking appropriate remedial actions. Studies explore methods for interpreting model predictions, identifying influential features, and providing explanations to cybersecurity analysts and domain experts.

Application-Specific Anomaly Detection: Anomaly detection techniques are applied in various cybersecurity domains, including network intrusion detection, malware analysis, insider threat detection, and cloud security. Research in this area investigates domain-specific challenges, data characteristics, and detection requirements to develop tailored anomaly detection solutions.

Benchmark Datasets and Evaluation Metrics: Benchmark datasets and standardized evaluation metrics are essential for comparing the performance of different anomaly detection methods. Researchers curate datasets containing real-world log data and define evaluation metrics such as accuracy, precision, recall, F1-score, and area under the ROC curve (AUC) for assessing the effectiveness of anomaly detection models.

These are just a few examples of the diverse range of related works in the field of log analysis and anomaly detection in cybersecurity. Researchers continue to explore new algorithms, techniques, and applications to address the evolving challenges posed by cyber threats and the increasing complexity of IT environments.

**Chapter 3**

**System Development**

**Building a machine learning model for log file analysis can be a powerful approach to enhance security monitoring and threat detection. Here's a high-level overview of how you could go about it:**

**Data Collection:** Gather log data from various sources within your system, network, or application. Ensure that you have a diverse and representative dataset that includes both normal and anomalous activities.

**Data Preprocessing**: Clean and preprocess the log data to make it suitable for machine learning. This may involve parsing log entries, extracting relevant features, handling missing values, and encoding categorical variables.

**Feature Engineering**: Generate additional features from the log data that could improve the performance of your machine learning model. This could include aggregating log entries over time intervals, creating frequency-based features, or extracting specific patterns related to security incidents.

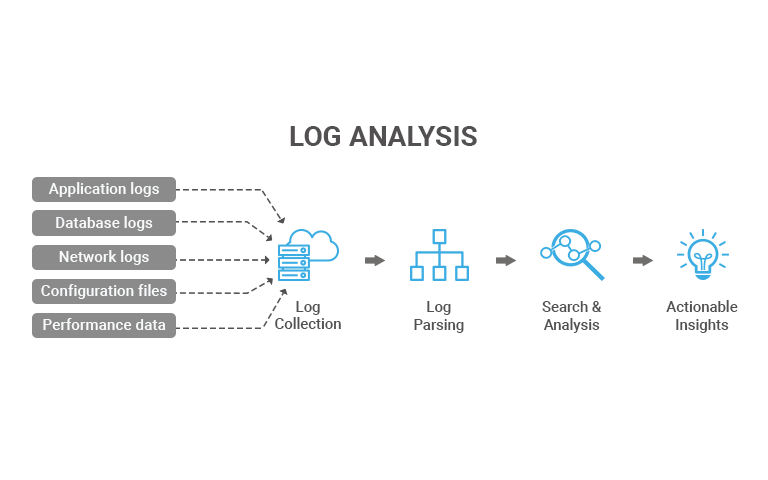
**Model Selection**: Choose appropriate machine learning algorithms for your task. For anomaly detection in log files, techniques like Isolation Forest, Local Outlier Factor (LOF), or One-Class SVM (Support Vector Machine) are commonly used. You may also consider ensemble methods or deep learning approaches depending on the complexity of your data.

**Model Training**: Split your dataset into training and testing sets. Train your chosen machine learning model on the training data and evaluate its performance using the testing data. Fine-tune hyperparameters and experiment with different algorithms to optimize performance.

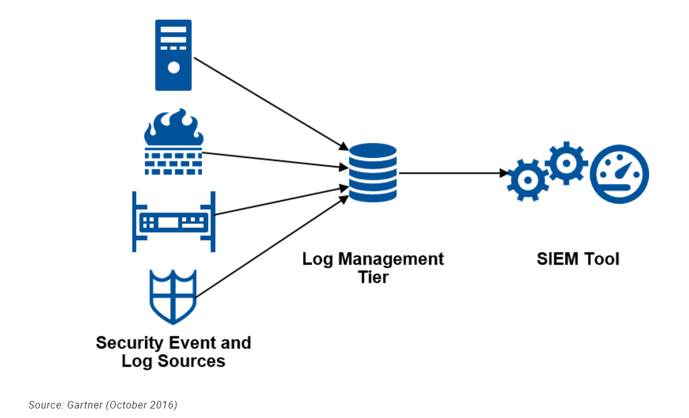
**Model Deployment**: Once you have a trained model with satisfactory performance, deploy it into your production environment. This could involve integrating it into your existing security infrastructure or developing a standalone application that takes log data as input and provides security insights or alerts.

**Monitoring and Maintenance:** Continuously monitor the performance of your machine learning model in the production environment. Periodically retrain the model with new data to ensure that it remains effective in detecting emerging threats and adapting to changes in the system or network.

**This Figure Represent Our System Development**



### First Step Data Collection:



We have provided a dataset with various attributes related to network traffic logs. Each row appears to represent a network connection, and the columns seem to describe different features of those connections. Here's a breakdown of the columns:

**duration**: Duration of the connection in seconds.

**protocol\_type**: Type of the protocol (e.g., TCP, UDP, ICMP).

**service:** Type of service or application.

**flag**: Status of the connection (e.g., normal, error, reset).

**src\_bytes**: Number of bytes sent by the source.

**dst\_bytes**: Number of bytes sent to the destination.

**land**: Indicates whether the connection is from/to the same host/port.

**wrong\_fragment**: Number of "wrong" fragments (obsolete; often zero).

**urgent**: Number of urgent packets.

**hot**: Number of "hot" indicators (suspicious actions).

**num\_failed\_logins**: Number of failed login attempts.

**logged\_in:** Indicates if the user is logged in.

**num\_compromised**: Number of compromised conditions.

**root\_shell**: Indicates if the root shell is obtained.

**su\_attempted**: Indicates if "su" command attempted.

**num\_root:** Number of "root" accesses.

**num\_file\_creations:** Number of file creation operations.

**num\_shells:** Number of shell prompts.

**num\_access\_files:** Number of access files.

**num\_outbound\_cmds**: Number of outbound commands.

**is\_host\_login**: Indicates if login is host-based.

**is\_guest\_login:** Indicates if login is guest-based.

**count**: Number of connections to the same host as the current connection.

**srv\_count**: Number of connections to the same service as the current connection.

**serror\_rate**: Error rate for connections to the same host.

**srv\_serror\_rate**: Error rate for connections to the same service.

**rerror\_rate:** Error rate for connections to the same host.

**srv\_rerror\_rate:** Error rate for connections to the same service.

**same\_srv\_rate:** Percentage of connections to the same service.

**diff\_srv\_rate**: Percentage of connections to different services.

**srv\_diff\_host\_rate**: Rate of connections to different hosts for the same service.

**dst\_host\_count**: Number of connections to the same destination host.

**dst\_host\_srv\_count**: Number of connections to the same destination service.

**dst\_host\_same\_srv\_rate**: Percentage of connections to the same service on the destination host.

**dst\_host\_diff\_srv\_rate**: Percentage of connections to different services on the destination host.

**dst\_host\_same\_src\_port\_rate**: Percentage of connections from the same source port.

**dst\_host\_srv\_diff\_host\_rate**: Rate of connections to different hosts for the same service on the destination host.

**dst\_host\_serror\_rate:** Error rate for connections to the same destination host.

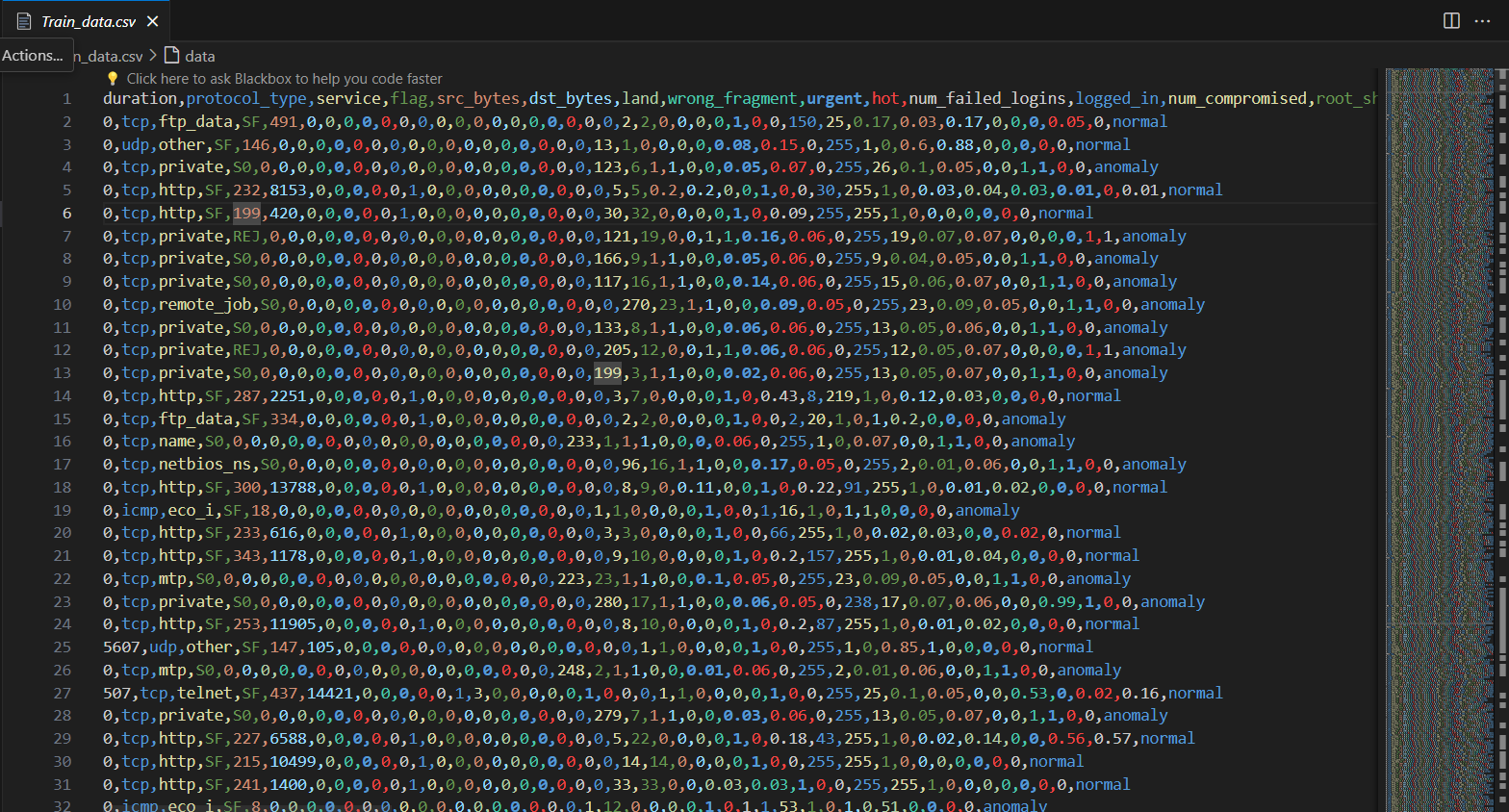
**dst\_host\_srv\_serror\_rate:** Error rate for connections to the same destination service.

**dst\_host\_rerror\_rate**: Error rate for connections to the same destination host.

**dst\_host\_srv\_rerror\_rate**: Error rate for connections to the same destination service.

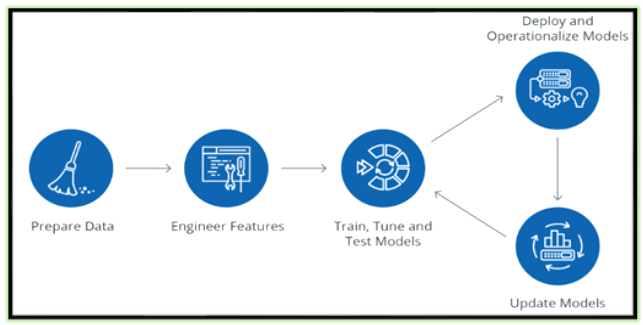
**class**: The classification of the network connection (e.g., normal, suspicious, attack).

**This Figure Represent the Train\_data.csv**

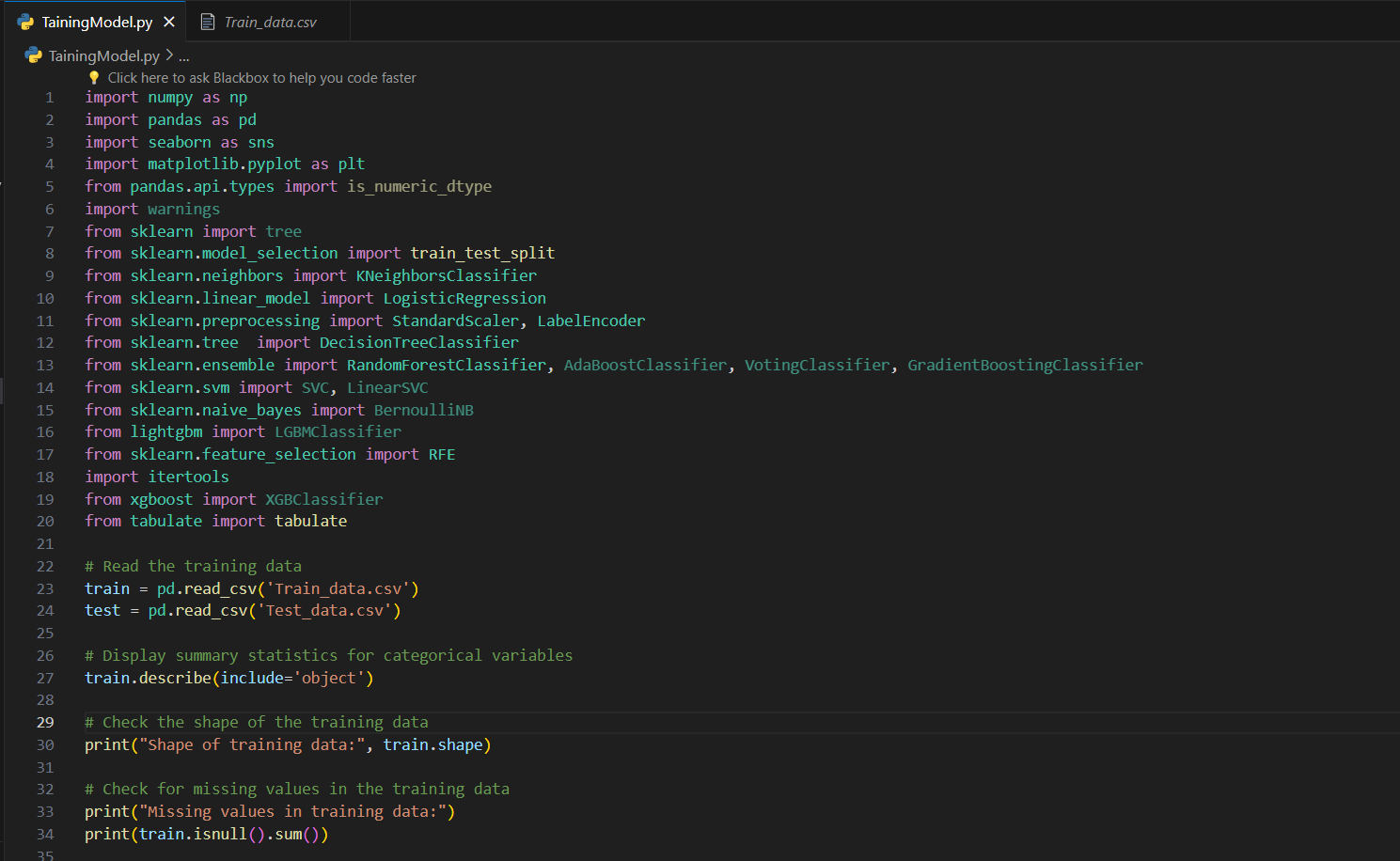


### Second Step Data Preprocessing & Feature Engineering:

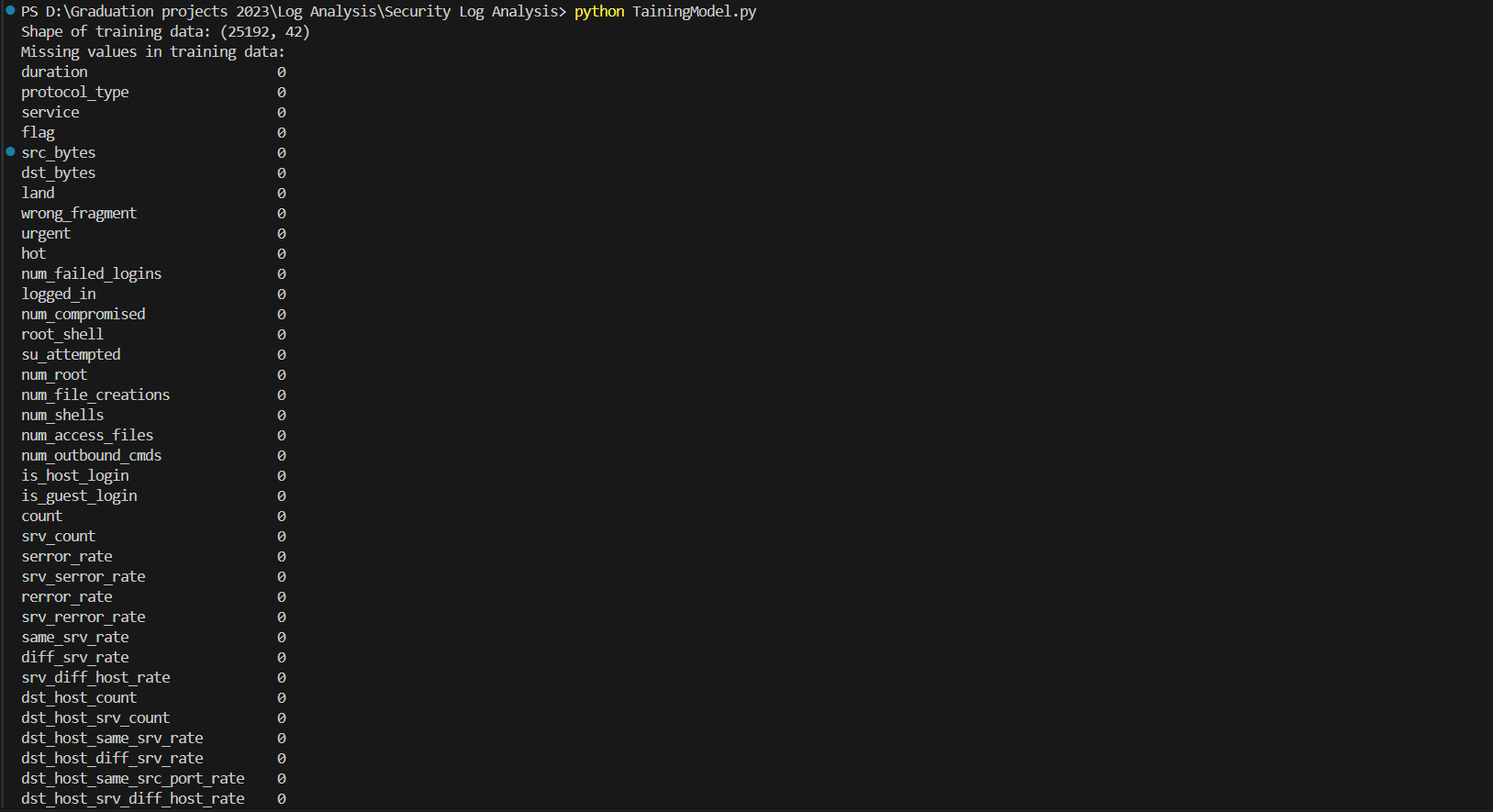
### 



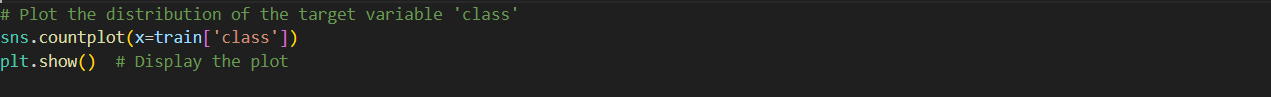
* **We have Created Python Project**
  + Importing Libraries: This imports various libraries used for data manipulation, visualization, machine learning models, and performance evaluation.

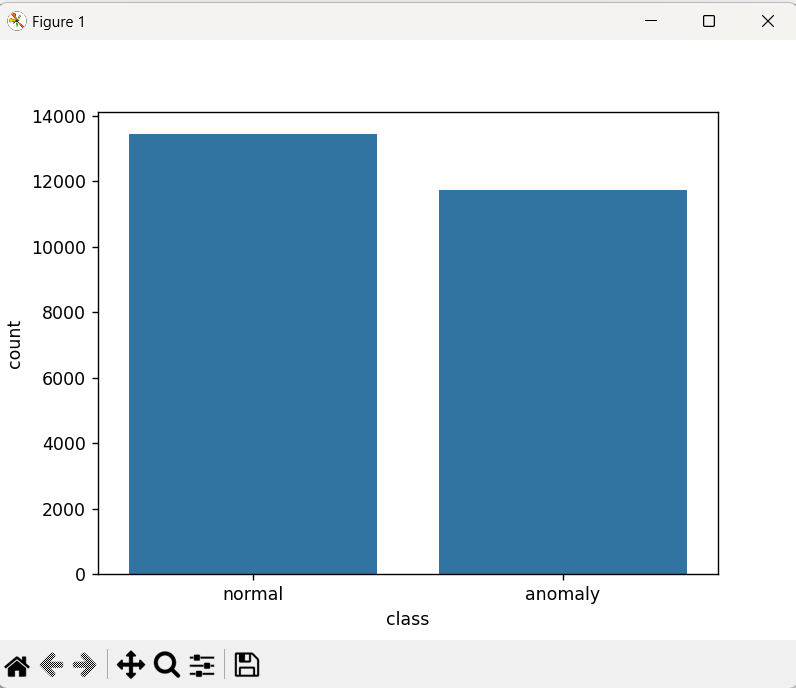
****

* **In this figure we have** 
  + Read the Training data
  + Display summary statistics for categorical variables
  + Check the shape of the training data
    - **Checking Shape**: This prints the shape (number of rows and columns) of the training dataset.
  + Check for missing values in the training data
    - **Checking Missing Values:** This checks for missing values in the training dataset and prints the sum of missing values for each column**.**

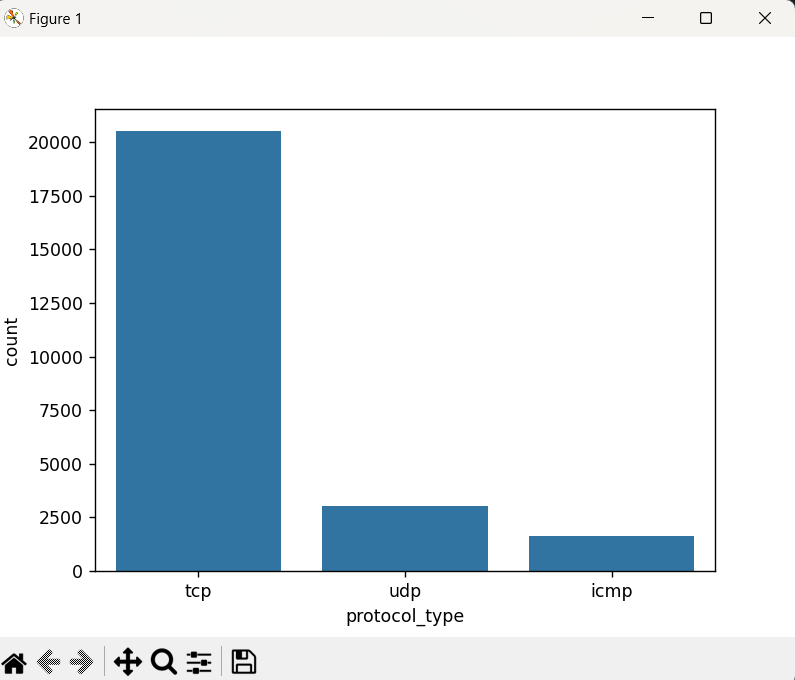


* **In this figure we have**
  + **Plot the distribution of the target variable 'class'**

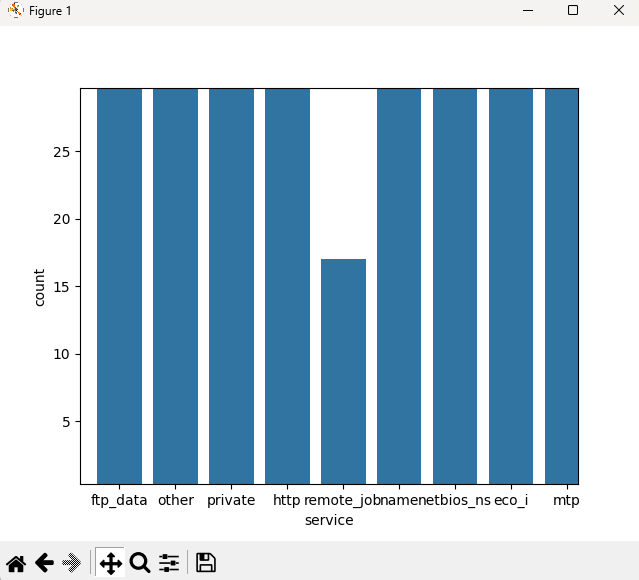




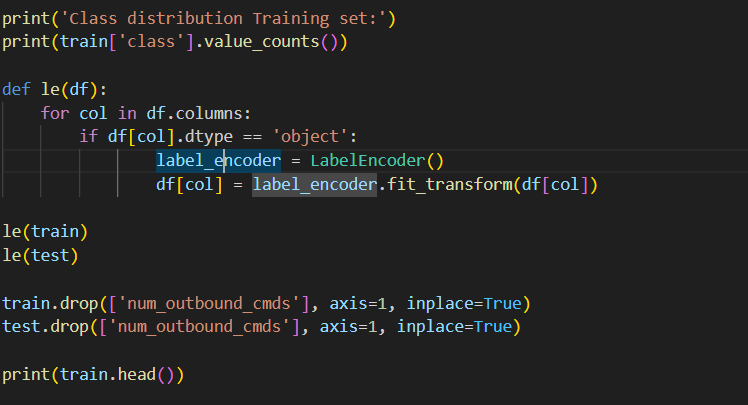
* **In this figure we have**
  + **Plot the distribution of the target variable ‘protocol type’**



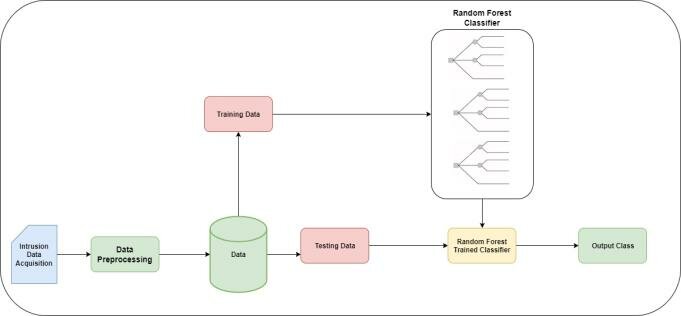
* **In this figure we have**
  + **Plot the distribution of the target variable ‘Service’**



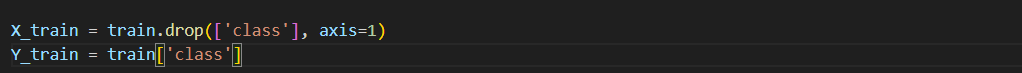
* **In this figure we have**
  + **Class Distribution**: This prints the class distribution of the target variable 'class' in the training dataset.
  + **Label Encoding**: This defines a function le() to encode categorical variables using LabelEncoder. It then applies this function to both the training and testing datasets.
  + **Removing Irrelevant Column:** This drops the column 'num\_outbound\_cmds' from both the training and testing datasets as it's considered irrelevant.
  + **Displaying First Few Rows:** This prints the first few rows of the preprocessed training dataset to verify the changes.



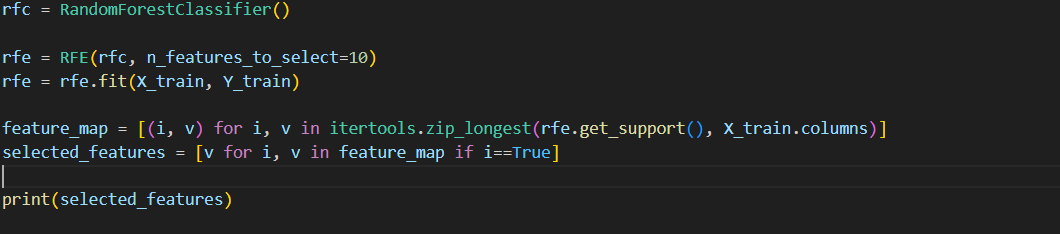
**Random Forest Classifier**



* **In this figure we have**
  + **Feature and Target Variable Separation:** This separates the features (independent variables) and the target variable 'class' from the training dataset.



* **In this figure we have**
  + **Initializing Random Forest Classifier**: This initializes a Random Forest Classifier.
  + **Feature Selection with RFE**: This initializes RFE with the Random Forest Classifier and specifies to select 10 features. It then fits RFE to the training data to select the best features.
  + **Selecting Features**: This creates feature map to map selected features to their indices, and then collects the selected features based on the RFE rankings.



* + **Printing Selected Features**: This prints the list of selected features identified by the RFE process.

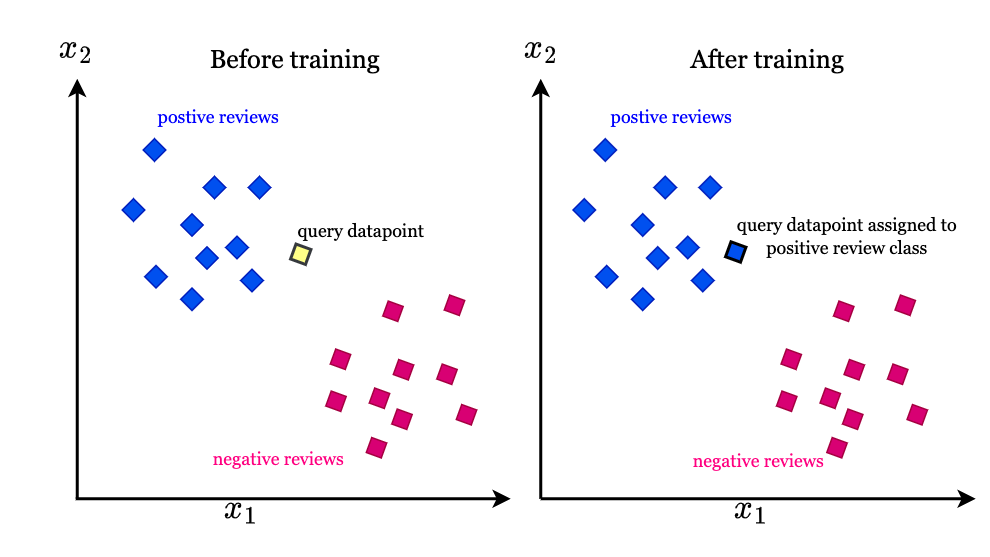




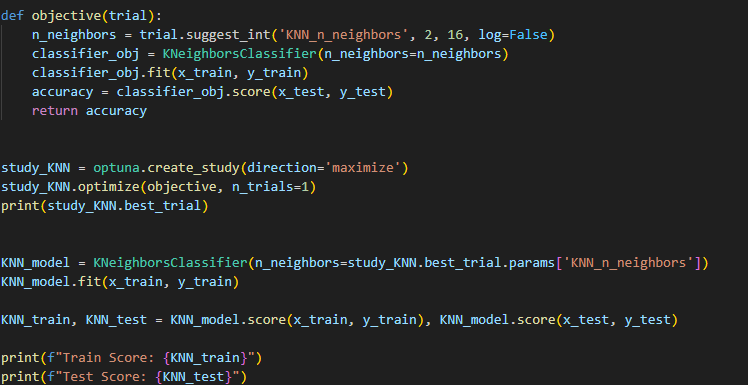
* **In this figure we have**
  + **Feature Selection and Scaling:**
    - X\_train = X\_train[selected\_features]: Selects a specific set of features from the training data.
    - scale = StandardScaler(): Initializes a StandardScaler object, which is used to scale the features.
    - X\_train = scale.fit\_transform(X\_train): Fits the scaler to the training data and then transforms it, scaling the selected features.
    - test = scale.fit\_transform(test): It's important to note that here fit\_transform() is called again on the test data. This should typically be transform() instead of fit\_transform(), as you don't want to refit the scaler on the test data
  + **Train-Test Split:**
    - train\_test\_split(): Splits the data into training and testing sets. Here, 70% of the data is allocated for training (train\_size=0.70), and the remaining 30% is allocated for testing.
    - random\_state=2: Sets the random seed for reproducibility.
  + **Printing the Shapes:**
    - print(x\_train.shape): Prints the shape (dimensions) of the training features.
    - print(x\_test.shape): Prints the shape of the testing features.
    - The printed shapes will give you an idea of how the data is split between training and testing sets.



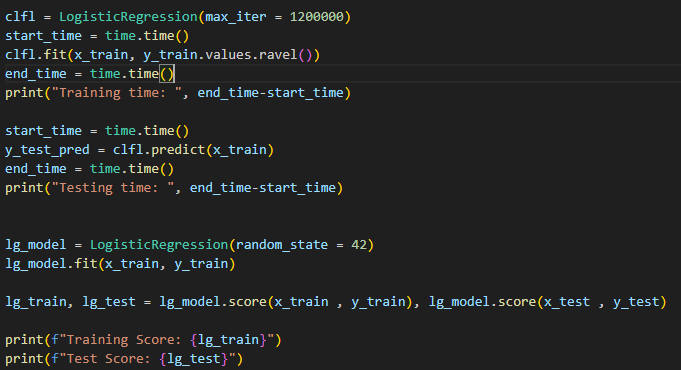
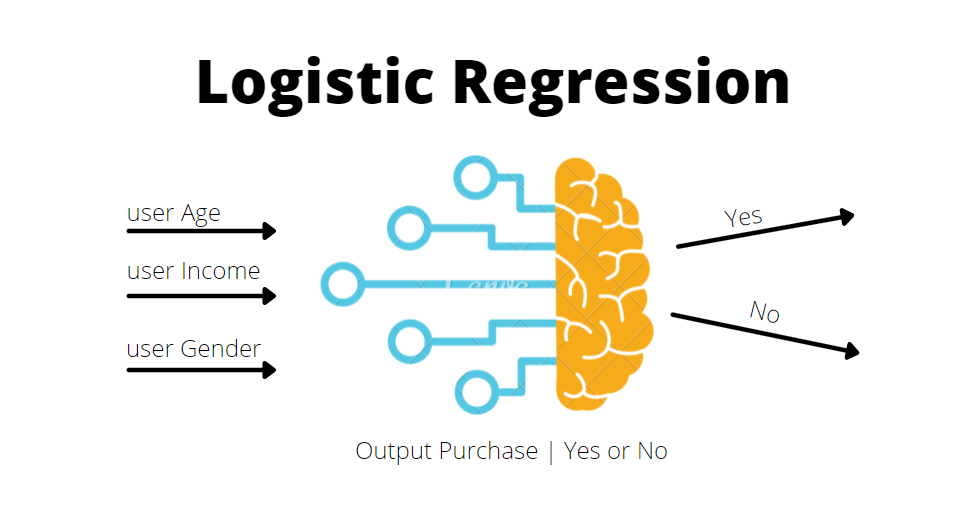
**K - nearest neighbor algorithm in machine learning**



* **In this Code we Have :** 
  + **KNN Model Training and Testing:** This code trains a K-Nearest Neighbors (KNN) model with the best hyperparameters found using Optuna and evaluates its performance on the training and testing sets
  + **Hyperparameter Tuning with Optuna for KNN:** This code uses Optuna to perform hyperparameter tuning for the K-Nearest Neighbors **(KNN) model**. It defines an objective function that takes a trial object and suggests a value for the number of neighbors parameter (**n\_neighbors**). The study object is then created to maximize the accuracy of the model, and the **optimize**() method is called to find the best hyperparameters. Finally, it prints the best trial.

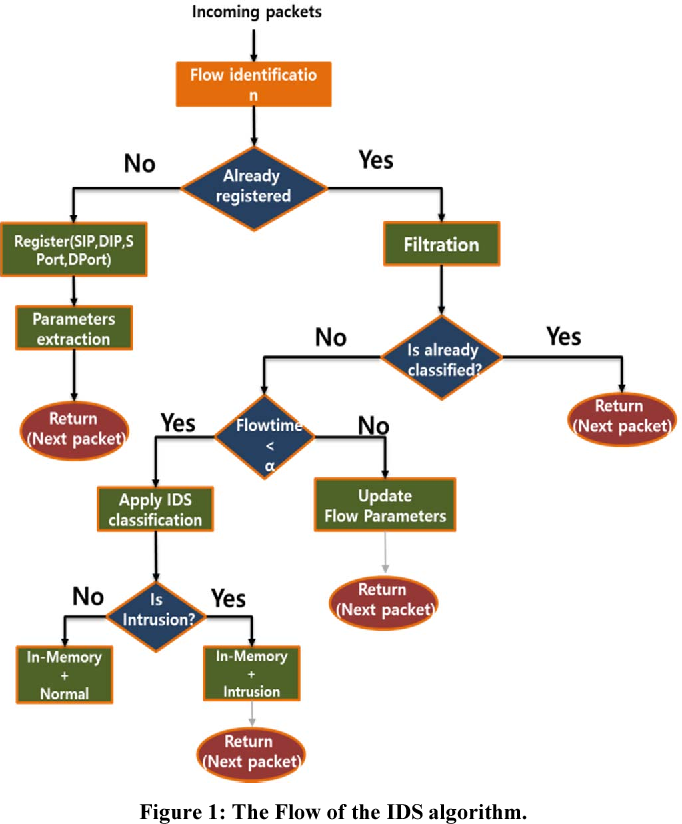


**Logistic Regression Algorithm**



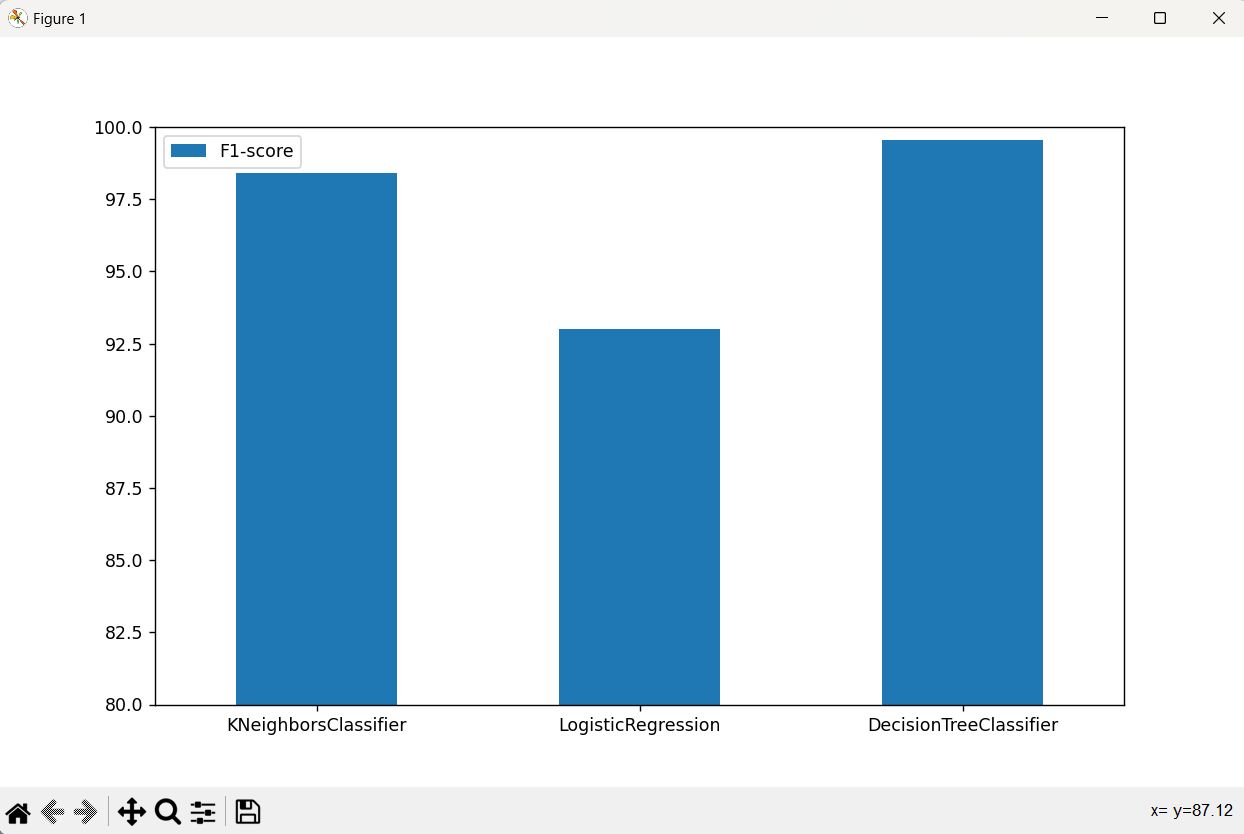
* **We have in this Code** 
  + **Logistic Regression Model Training and Testing**: This trains a Logistic Regression model and measures the training and testing time.

**Decision Tree Classifier**



* **We have in this Figure** 
  + **Model** **Comparison**: This code creates a table using the **tabulate** library to display the training and testing scores of different models (KNN, Logistic Regression, and Decision Tree) in a tabular format.





**FrontEnd UI Test WebSite**

* We have made a Flask API Using Python

**Api.py**

from flask import Flask, request, jsonify

import pandas as pd

import joblib

import webbrowser

from flask\_cors import CORS

app = Flask(\_\_name\_\_)

CORS(app, resources={r"/predict": {"origins": "\*"}})  # Allow CORS for '/predict' endpoint from all origins

# Load the trained model

model = joblib.load('DecisionTreeClassifier.pkl')

webbrowser.open\_new\_tab('index.html')

@app.route('/predict', methods=['POST'])

def predict():

    # Get the request data

    data = request.get\_json()

    # Convert the JSON data to a DataFrame with a default index

    df = pd.DataFrame(data, index=[0])

    # Encode categorical variables

    df\_encoded = pd.get\_dummies(df)  # Assuming one-hot encoding

    # Make predictions

    predictions = model.predict(df\_encoded)

    # Return the predictions as JSON

    return jsonify({'predictions': predictions.tolist()})

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True, port=5002)

**Index .html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Anomaly Detection</title>

    <!-- Bootstrap CSS -->

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

    <!-- Font Awesome for icons -->

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">

    <!-- Custom CSS for animation -->

    <style>

        body {

            background-image: url('bg.jpeg');

            color: white

        }

        .progress-bar {

            background-color: #28a745;

        }

        @keyframes fadeInOut {

            0% {

                opacity: 0;

            }

            50% {

                opacity: 1;

            }

            100% {

                opacity: 0;

            }

        }

        .animate-fade-in-out {

            animation: fadeInOut 2s infinite;

        }

        /\* Style for the prediction result \*/

        #predictionResult {

            font-size: 35px;

            text-align: center;

            font-weight: bold;

            margin-top: 20px;

        }

    </style>

</head>

<body>

    <div class="container mt-5">

        <h1 class="mb-4 text-center">Network Security Log Detection</h1>

        <div class="progress mt-2" style="display: none;">

            <div id="progressBar" class="progress-bar" role="progressbar" style="width: 0%;" aria-valuenow="0"

                aria-valuemin="0" aria-valuemax="100"></div>

        </div>

        <form id="predictionForm">

            <div class="row">

                <div class="col-md-6">

                    <div class="form-group">

                        <label for="duration">

                            <i class="fas fa-stopwatch"></i> Duration:

                        </label>

                        <input type="number" class="form-control" id="duration" name="duration" value="0" required>

                    </div>

                    <div class="form-group">

                        <label for="protocol\_type">

                            <i class="fas fa-network-wired"></i> Protocol Type:

                        </label>

                        <select class="form-control" id="protocol\_type" name="protocol\_type" required>

                            <option value="tcp">tcp</option>

                            <option value="icmp">icmp</option>

                            <option value="udp">udp</option>

                        </select>

                    </div>

                </div>

                <div class="col-md-6">

                    <div class="form-group">

                        <label for="service">

                            <i class="fas fa-server"></i> Service:

                        </label>

                        <select class="form-control" id="service" name="service" required>

                            <option value="mtp">mtp</option>

                            <option value="private">private</option>

                            <option value="ftp\_data">ftp\_data</option>

                            <option value="eco\_i">eco\_i</option>

                            <option value="telnet">telnet</option>

                            <option value="http">http</option>

                            <option value="smtp">smtp</option>

                            <option value="ldap">ldap</option>

                            <option value="pop\_3">pop\_3</option>

                            <option value="courier">courier</option>

                            <option value="imap4">imap4</option>

                            <option value="domain\_u">domain\_u</option>

                        </select>

                    </div>

                    <div class="form-group">

                        <label for="flag">

                            <i class="fas fa-flag"></i> Flag:

                        </label>

                        <select class="form-control" id="flag" name="flag" required>

                            <option value="REJ">REJ</option>

                            <option value="SF">SF</option>

                            <option value="RSTO">RSTO</option>

                            <option value="S0">S0</option>

                            <option value="RSTR">RSTR</option>

                        </select>

                    </div>

                </div>

            </div>

            <div class="row">

                <div class="col-md-6">

                    <div class="form-group">

                        <label for="src\_bytes">

                            <i class="fas fa-arrow-up"></i> Source Bytes:

                        </label>

                        <input type="number" class="form-control" id="src\_bytes" name="src\_bytes" value="0" required>

                    </div>

                    <div class="form-group">

                        <label for="dst\_bytes">

                            <i class="fas fa-arrow-down"></i> Destination Bytes:

                        </label>

                        <input type="number" class="form-control" id="dst\_bytes" name="dst\_bytes" value="0" required>

                    </div>

                </div>

                <div class="col-md-6">

                    <div class="form-group">

                        <label for="land">

                            <i class="fas fa-landmark"></i> Land:

                        </label>

                        <input type="number" class="form-control" id="land" name="land" value="0" required>

                    </div>

                    <div class="form-group">

                        <label for="wrong\_fragment">

                            <i class="fas fa-times"></i> Wrong Fragment:

                        </label>

                        <input type="number" class="form-control" id="wrong\_fragment" name="wrong\_fragment" value="0"

                            required>

                    </div>

                </div>

            </div>

            <div class="row">

                <div class="col-md-6">

                    <div class="form-group">

                        <label for="urgent">

                            <i class="fas fa-exclamation-triangle"></i> Urgent:

                        </label>

                        <input type="number" class="form-control" id="urgent" name="urgent" value="0" required>

                    </div>

                </div>

                <div class="col-md-6">

                    <div class="form-group">

                        <label for="hot">

                            <i class="fas fa-fire"></i> Hot:

                        </label>

                        <input type="number" class="form-control" id="hot" name="hot" value="0" required>

                    </div>

                </div>

            </div>

            <!-- Display prediction result here -->

            <div id="predictionResult"></div>

            <!-- Button for submitting the form -->

            <div class="text-center">

                <div class="center-button">

                    <button type="submit" id="predictButton" class="btn btn-success">

                        <i class="fas fa-robot"></i> Predict Log From AI Model

                        <span id="countdown" style="margin-left: 10px;"></span>

                    </button>

                    <button type="button" id="resetButton" class="btn btn-danger ml-3" style="display: none;">

                        <i class="fas fa-times-circle"></i> Reset

                    </button>

                </div>

            </div>

        </form>

    </div>

    <!-- Bootstrap JS -->

    <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>

    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"></script>

    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

    <!-- Your custom JavaScript -->

    <script>

        $(document).ready(function () {

            $('#predictionForm').submit(function (event) {

                event.preventDefault(); // Prevent the form from submitting normally

                // Get form data

                var duration = parseInt($('#duration').val());

                var protocol\_type = [$('#protocol\_type').val()];

                var service = [$('#service').val()];

                var flag = [$('#flag').val()];

                var src\_bytes = parseInt($('#src\_bytes').val());

                var dst\_bytes = parseInt($('#dst\_bytes').val());

                var land = parseInt($('#land').val());

                var wrong\_fragment = parseInt($('#wrong\_fragment').val());

                var urgent = parseInt($('#urgent').val());

                var hot = parseInt($('#hot').val());

                var jsonData = {

                    "duration": [duration],

                    "protocol\_type": protocol\_type,

                    "service": service,

                    "flag": flag,

                    "src\_bytes": [src\_bytes],

                    "dst\_bytes": [dst\_bytes],

                    "land": [land],

                    "wrong\_fragment": [wrong\_fragment],

                    "urgent": [urgent],

                    "hot": [hot]

                };

                console.log(JSON.stringify(jsonData));

                console.log("JSON Data: ", jsonData);

                $('#predictButton').prop('disabled', true);

                $('.progress').show();

                $('#resetButton').hide();

                var progressBar = $('#progressBar');

                var countdown = 10;

                var interval = setInterval(function () {

                    progressBar.css('width', ((10 - countdown) \* 10) + '%');

                    progressBar.attr('aria-valuenow', (10 - countdown) \* 10);

                    $('#countdown').text(countdown);

                    countdown--;

                    if (countdown < 0) {

                        clearInterval(interval);

                        progressBar.css('width', '100%');

                        $.ajax({

                            url: 'http://localhost:5002/predict',

                            type: 'POST',

                            contentType: 'application/json',

                            data: JSON.stringify(jsonData),

                            success: function (response) {

                                var resultDiv = $('#predictionResult');

                                if (response.predictions[0] === 1) {

                                    resultDiv.text('Prediction: Normal Log').addClass('text-success');

                                } else {

                                    resultDiv.text('Prediction: Anomaly Log').addClass('text-danger');

                                }

                                resultDiv.addClass('animate-fade-in-out');

                                $('#resetButton').show();

                            },

                            error: function (xhr, status, error) {

                                alert('Error: ' + error);

                            }

                        });

                    }

                }, 1000);

            });

            $('#resetButton').click(function () {

                $('.progress').hide();

                $('#predictButton').prop('disabled', false);

                $('#predictionResult').empty().removeClass('text-success text-danger animate-fade-in-out');

                $('#resetButton').hide();

            });

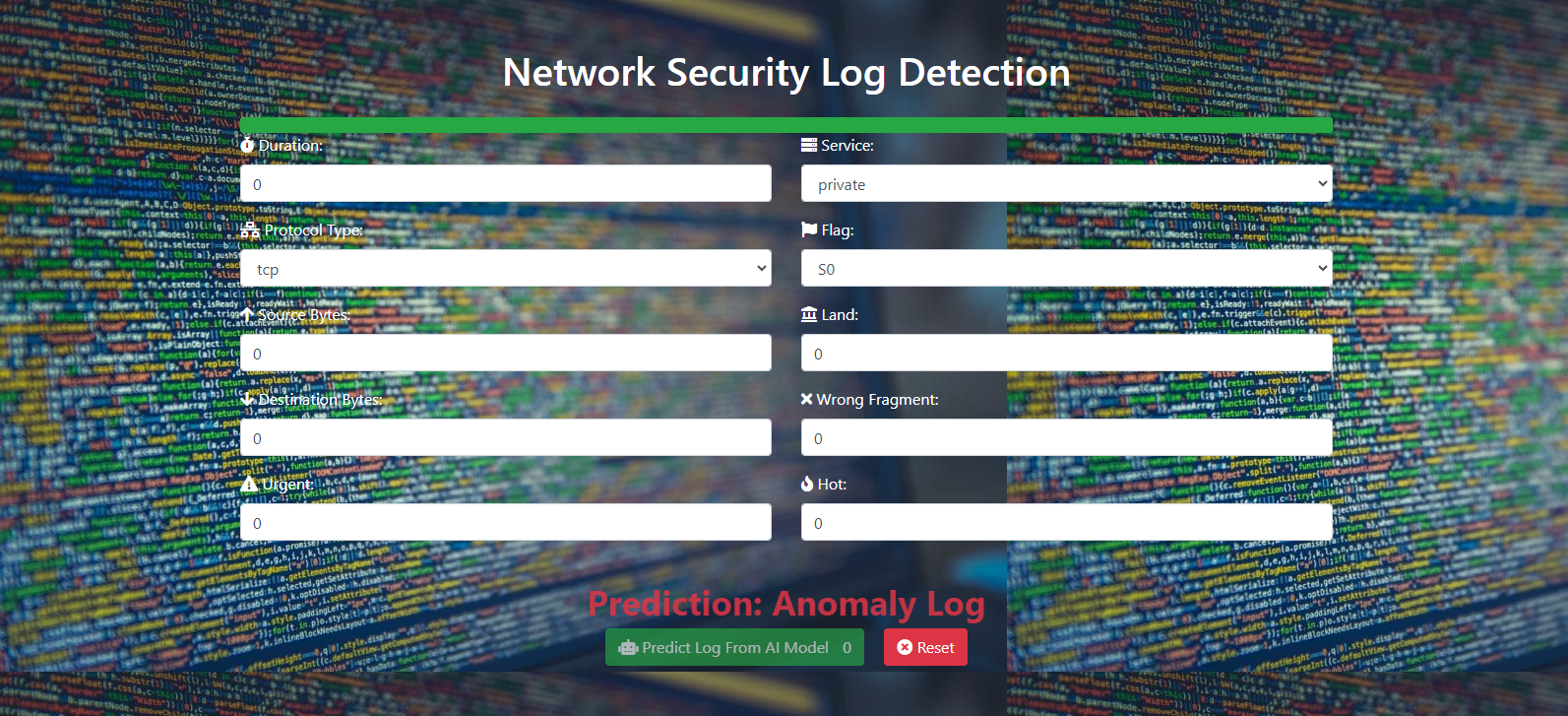
        });

    </script>

</body>

</html>

**ScreenShoot of App Testing**





**Chapter 4**

**Conclusions and**

**Future Work**

4.1 conclusions:

1. **Dataset Overview**:
   * The dataset consists of training and testing data for anomaly detection, containing both numerical and categorical features.
2. **Data Preprocessing**:
   * Categorical variables were encoded using Label Encoding to prepare the data for modeling.
   * Missing values were handled appropriately, and duplicate rows were removed from the training data.
3. **Exploratory Data Analysis (EDA)**:
   * Summary statistics and visualizations were utilized to understand the distribution of features and the imbalance in the target variable ('class').
4. **Feature Selection**:
   * Recursive Feature Elimination (RFE) with Random Forest was employed to select the top 10 features for modeling.
5. **Modeling**:
   * Three classification algorithms, namely Logistic Regression, K-Nearest Neighbors (KNN), and Decision Tree, were implemented.
   * Hyperparameter tuning was performed for the Decision Tree model using Optuna to enhance performance.
6. **Model Evaluation**:
   * Training and testing times were recorded for Logistic Regression and Decision Tree models to assess computational efficiency.
   * Accuracy scores were calculated for each model to measure overall performance.
   * Cross-validation was conducted to validate model performance using precision and recall metrics, providing insights into their effectiveness.
7. **Model Comparison**:
   * Mean precision and recall scores were visualized for each model, allowing for a comparative analysis of their performance.
   * F1-scores were computed and plotted to evaluate the balance between precision and recall.
8. **Conclusion**:
   * The report presents a comprehensive approach to anomaly detection using classification algorithms.
   * Findings indicate that while all models achieved reasonable accuracy, further fine-tuning may be necessary to enhance performance.
   * Decision Tree model optimization using Optuna demonstrated the potential for improving predictive accuracy.
   * Overall, the report provides valuable insights into the effectiveness of different algorithms for anomaly detection tasks and suggests areas for further investigation and refinement.

Haut du formulaire

4.2 Future work

1. **Improved UI/UX**: Enhance the user interface to make it more intuitive and user-friendly. This could include better form validation, error handling, and feedback to the user.
2. **Real-time Prediction**: Implement real-time prediction capabilities so that the system can analyze and predict anomalies as they occur, rather than waiting for a batch of data to be submitted.
3. **Model Tuning**: Experiment with different machine learning models and hyperparameters to improve the accuracy of anomaly detection. This could involve using more sophisticated algorithms or ensemble methods.
4. **Data Preprocessing**: Explore different data preprocessing techniques to better handle missing values, outliers, and skewed distributions. Feature scaling, normalization, and dimensionality reduction techniques could also be applied to improve model performance.
5. **Model Evaluation**: Implement a robust evaluation framework to assess the performance of the anomaly detection model. This could include metrics such as precision, recall, F1-score, and ROC-AUC curve analysis.
6. **Deployment and Scalability**: Deploy the application to a production environment using containerization technologies such as Docker and orchestration tools like Kubernetes for scalability and reliability.
7. **Integration with External Systems**: Integrate the anomaly detection system with other systems and tools commonly used in cybersecurity operations, such as SIEM (Security Information and Event Management) platforms.
8. **Security and Compliance**: Ensure that the application adheres to security best practices and compliance requirements, especially when handling sensitive data. Implement encryption, access controls, and audit trails as needed.
9. **Monitoring and Alerting**: Set up monitoring and alerting mechanisms to detect and respond to anomalies in the system's behavior. This could involve implementing thresholds, alarms, and automated notifications.
10. **User Authentication and Authorization**: Implement user authentication and authorization mechanisms to control access to the application and its features. This could involve integrating with identity providers such as LDAP or OAuth providers.

These are just a few ideas for future work to enhance and expand the anomaly detection system. Depending on the specific requirements and use cases, there may be other areas to focus on as well.

**List of**

**References**

**List of References**

https://www.kaggle.com/datasets/eliasdabbas/web-server-access-logs/code

**ملخص المشروع**

**اسم المشروع**

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |

**فريق العمل**

**المشرف: دكتور/**

|  |
| --- |
| **اسم المشروع** |
|  |

**مشروع التخرج**

**تحت إشراف:**

**دكتور/**