Simulation and Implementation of a Phishing Detection Dataset and Model

4.1 Introduction

This chapter provides a comprehensive account of the simulation of a phishing detection dataset, preprocessing steps, training and evaluation of a Logistic Regression model, and the subsequent deployment of the model as a Flask-based web application. The primary goal of this chapter is to document the creation of a reliable and practical phishing detection system that can classify SMS and email messages in real time.

4.2 Simulation of the Dataset

The simulation phase was designed to generate a realistic dataset that mimics both phishing and legitimate communications. This dataset served as the foundation for training and evaluating the phishing detection model. The process involved the following steps:

1. **Message Categorization**:
    - **Phishing Messages**: Messages in this category imitated real-world phishing tactics, such as urgency, threats, or rewards to deceive recipients into divulging sensitive information. For example:
        - "Your account has been compromised. Click here to verify your details."
        - "Congratulations! You've won a prize. Claim now by providing your bank details."
    - **Legitimate Messages**: These messages represented typical communications without malicious intent, such as transaction confirmations or customer service notifications. For example:
        - "Your recent transaction was successful. Thank you for banking with us."
        - "Reminder: Your payment is due in 5 days. Thank you for your attention."
2. **Dataset Composition**:
    - The dataset included equal distributions of phishing and legitimate messages for balance:
        - **SMS**: 3,000 phishing messages and 3,000 legitimate messages.
        - **Email**: 3,000 phishing messages and 3,000 legitimate messages.
    - A total of 12,000 messages were generated, ensuring sufficient data for training and evaluation.
3. **Data Labeling and Categorization**:
    - Each message was labeled as either:
        - "phishing" (1): Indicates a malicious message.
        - "not phishing" (0): Indicates a legitimate message.
    - Messages were also categorized as "SMS" or "Email" to provide additional context.
4. **Randomization**:

- To prevent bias during training, the dataset was shuffled after combining the phishing and legitimate messages.

5. **Dataset Storage**:
    - The final dataset was stored in CSV format for portability and ease of use during preprocessing and model training.

4.3 Preprocessing of the Dataset

Data preprocessing is a critical step in ensuring that the dataset is in a format suitable for machine learning. The following preprocessing steps were carried out:

1. **Dataset Loading**:
    - The dataset was loaded from the CSV file into a DataFrame for manipulation and analysis.
2. **Redundant Columns Removal**:
    - The "Type" column, which indicated whether the message was an SMS or Email, was dropped, as it was not required for the classification task.
3. **Label Encoding**:
    - The labels ("phishing" and "not phishing") were encoded into binary values:
        - "phishing" = 1
        - "not phishing" = 0
4. **Text Vectorization**:
    - The **Term Frequency-Inverse Document Frequency (TF-IDF)** vectorization technique was applied to convert the text messages into numerical features. This technique assigns weights to words based on their frequency across all messages while reducing the impact of commonly occurring words like "the" or "and."
5. **Data Splitting**:
    - The dataset was split into:
        - Training Set (80%): Used for model training.
        - Testing Set (20%): Used for evaluating the model's performance.

4.4 Model Training and Evaluation

A Logistic Regression model was chosen for its effectiveness in binary classification tasks. The training and evaluation process involved the following steps:

1. **Feature Transformation**:

- The training and testing sets were transformed into TF-IDF feature matrices to serve as input for the model.

2. **Model Training**:
    - The Logistic Regression model was trained on the transformed training data to identify patterns that distinguish phishing messages from legitimate ones.
3. **Model Evaluation**:
    - The model's performance was assessed on the testing data using the following metrics:
        - **Accuracy**: Measures the overall correctness of predictions.
        - **Confusion Matrix**: Provides a detailed breakdown of true positives, true negatives, false positives, and false negatives.
        - **Classification Report**: Includes precision, recall, and F1-score for each class.
4. **Results**:
    - The evaluation results demonstrated that the model effectively distinguished between phishing and legitimate messages, achieving a high accuracy score. The confusion matrix highlighted areas for further improvement, such as minimizing false positives.

4.5 Integration into a Flask-Based Web Application

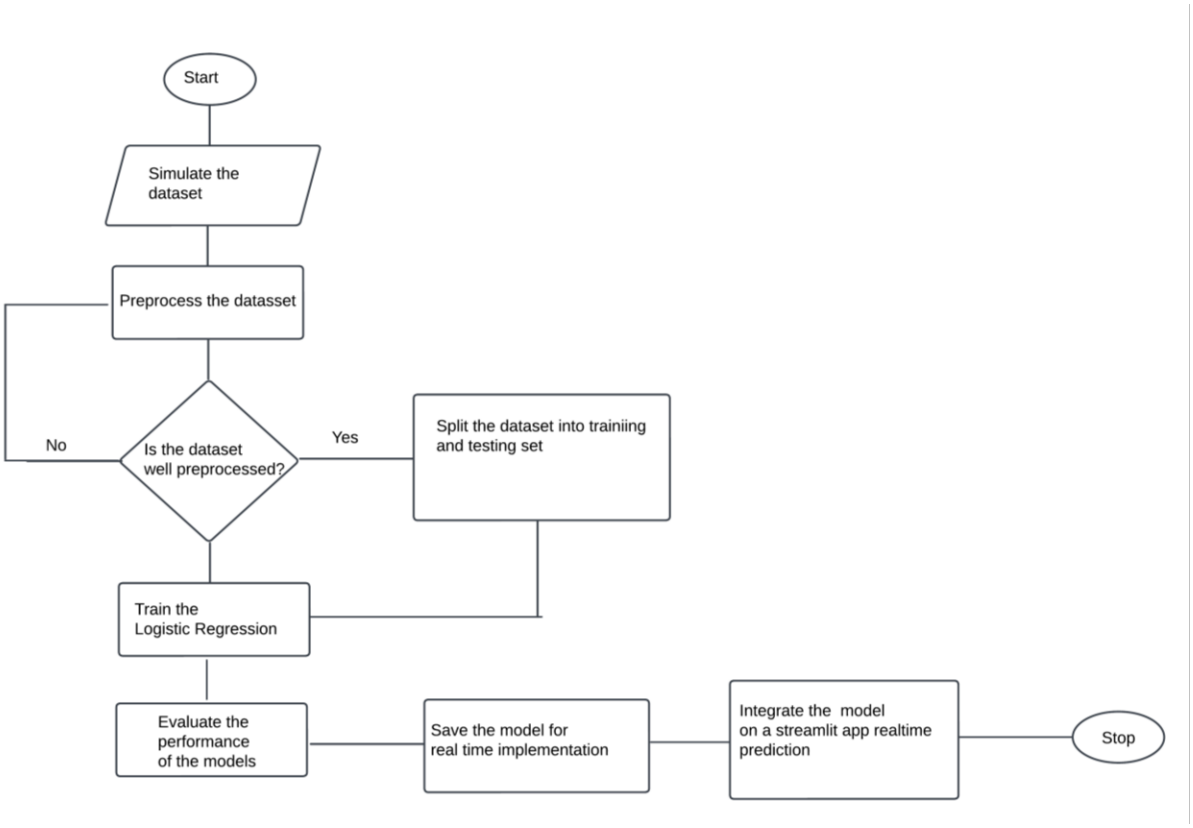
To make the phishing detection system accessible for practical use, the trained model was deployed as a web application using Flask. The integration process involved the following steps:

1. **Model Serialization**:
    - The trained Logistic Regression model and the TF-IDF vectorizer were saved using Python's pickle module. This ensured that the model and preprocessing steps could be reused without retraining.
2. **Flask Application Development**:
    - A Flask-based web application was developed to provide a user-friendly interface for real-time phishing detection. The key features of the application included:
        - **Input Form**: Users could input SMS or email messages for classification.
        - **Prediction Results**: The application displayed the classification result ("phishing" or "not phishing") along with a confidence score.
3. **Real-Time Predictions**:
    - Upon submission of a message, the application processed the text using the TF-IDF vectorizer and Logistic Regression model to provide an instant prediction.
4. **Deployment**:

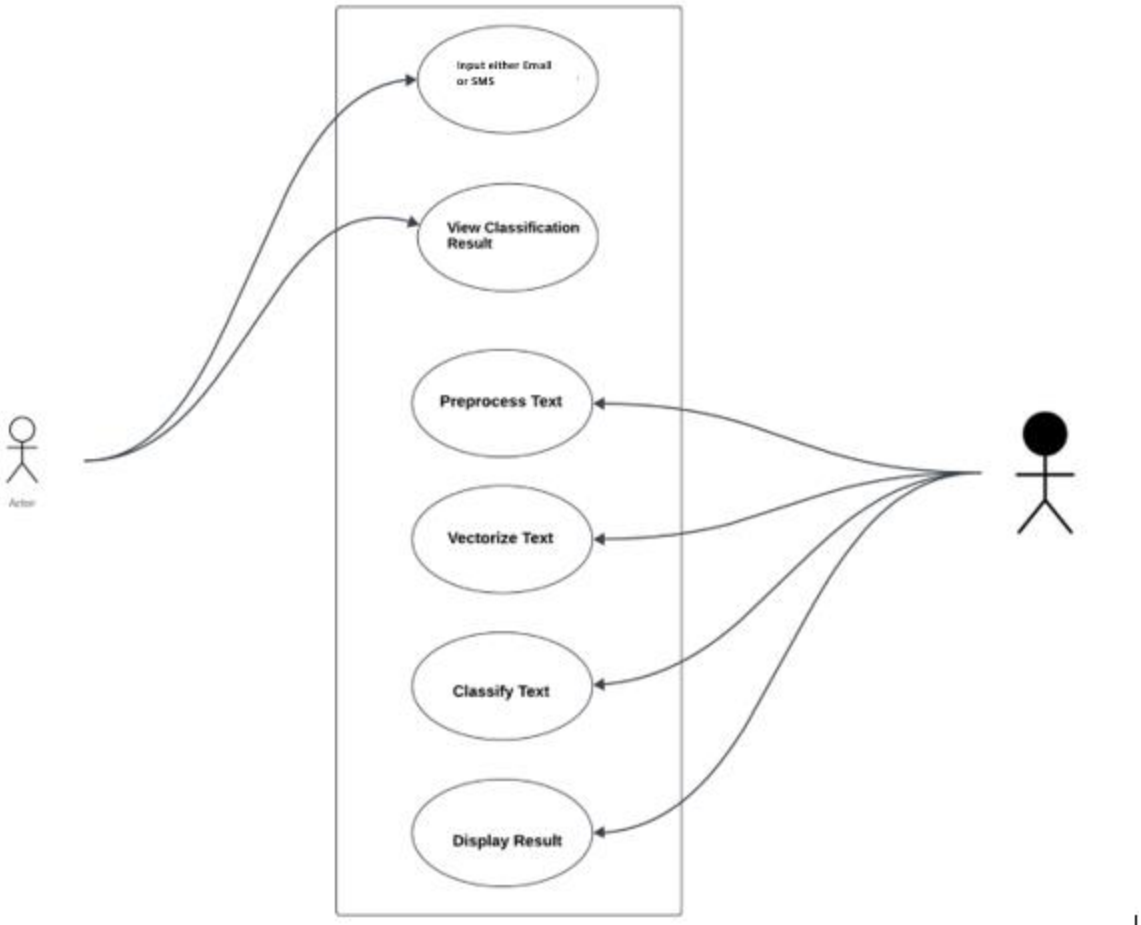
- The Flask application was deployed on a local server, with plans for future hosting on a cloud platform to ensure scalability and accessibility.

4.6 Summary

This chapter documented the end-to-end process of simulating a phishing detection dataset, preprocessing the data, training a Logistic Regression model, and deploying the model as a web application. The system demonstrates a practical approach to identifying phishing messages in real-time, combining data-driven insights with an accessible interface. By leveraging Flask for deployment, the model is now capable of serving both individual users and larger organizations, enhancing cybersecurity efforts against phishing attacks.



THE FLOWCHART OF THE DEVELOPED SYSTEM



USE CASE DIAGRAM OF THE DEVELOPED SYSTEM