Malicious URL Detector Project

Student:

BADEREDDINE Haitham

May 20, 2025

Contents

1 Introduction

In the rapidly evolving digital world, cyberattacks have become a major concern. One of the most common forms of cyberattack is through malicious URLs that host malware, phishing pages, or defaced websites. These URLs are crafted to trick users into providing sensitive information or compromising their devices.

The goal of this project is to detect malicious URLs by analyzing their lexical structure (the text of the URL) without needing to access the actual websites, which makes the detection faster and safer. It is a machine learning and deep learning-based project designed to classify URLs into four categories:

- **Benign** – Safe and legitimate websites.
- **Phishing** – Fraudulent websites designed to steal sensitive information.
- **Defacement** – Websites that have been hacked and altered.
- **Malware** – URLs hosting malicious software.

This report provides a detailed analysis of the preprocessing techniques, feature engineering, model training, evaluation, and deployment of the system.

2 Dataset Overview

Source: `https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset`

The dataset consists of URLs labeled into four classes:

`Benign, Phishing, Defacement, Malware`

Each record contains:

- A URL string.
- A label indicating the type of URL.

3 Feature Engineering & Preprocessing

The project extracts lexical and statistical features from URLs to train ML & DL models. Below is a breakdown of the key features:

3.1 IP Address in URL (use_of_ip)

Checks if the URL contains an IP address (common in phishing/malware URLs).

Observation: Malicious URLs often use IP addresses instead of domain names.

3.2 Abnormal URL (abnormal_url)

Checks if the hostname matches the URL structure.

Observation: Legitimate URLs usually have a clear domain structure.

3.3 Google Index (google_index)

Determines if the URL is indexed by Google (legitimate sites are usually indexed).

Observation: Many malicious URLs are not indexed.

3.4 Count of Dots (count.)

Counts the number of dots (.) in the URL.

Observation: More dots may indicate subdomain manipulation (common in phishing).

3.5 Count of WWW (count-www)

Counts occurrences of www.

Observation: Legitimate URLs usually have one www, while malicious ones may have zero or multiple.

3.6 Count of @ (count@)

Checks for @ in URLs (used to hide the real domain).

Observation: Phishing URLs often use @ to deceive users.

3.7 Count of Directories (count_dir)

Counts the number of / in the path.

Observation: More directories may indicate obfuscation.

3.8 Count of Embedded Domains (count_embed_domain)

Checks for // (used in redirection attacks).

Observation: Malicious URLs may have multiple embedded domains.

3.9 Suspicious Words (sus_url)

Detects keywords like login, bank, PayPal, update, etc.

Observation: Phishing URLs often contain such keywords.

3.10 Shortened URL (short_url)

Detects URL shortening services (bit.ly, goo.gl).

Observation: Attackers use short URLs to hide malicious links.

3.11 HTTPS/HTTP Count (count_https, count_http)

Checks the presence of secure (https) and insecure (http) protocols.

Observation: Malicious sites often use http or misuse https.

3.12 Special Characters (count%, count?, count-, count=)

Counts occurrences of %, ?, -, = (common in malicious URLs).

Observation: Excessive special characters indicate obfuscation.

3.13 URL & Hostname Length (url_length, hostname_length)

Measures the length of the full URL and hostname.

Observation: Malicious URLs are often longer to hide the true domain.

3.14 First Directory Length (fd_length)

Measures the length of the first directory after the domain.

Observation: Longer directories may indicate malicious intent.

3.15 Top-Level Domain (TLD) Length (tld_length)

Checks the length of TLD (e.g., .com, .org).

Observation: Safe URLs usually have TLDs of length 2-3.

3.16 Count of Digits & Letters (count_digits, count_letters)

Counts digits and letters in the URL.

Observation: Malicious URLs often have more digits and letters to appear complex.

4 Machine Learning Models

The following models were trained and evaluated:

4.1 Traditional ML Models

4.1.1 Logistic Regression

Simple, interpretable, but may struggle with complex patterns.

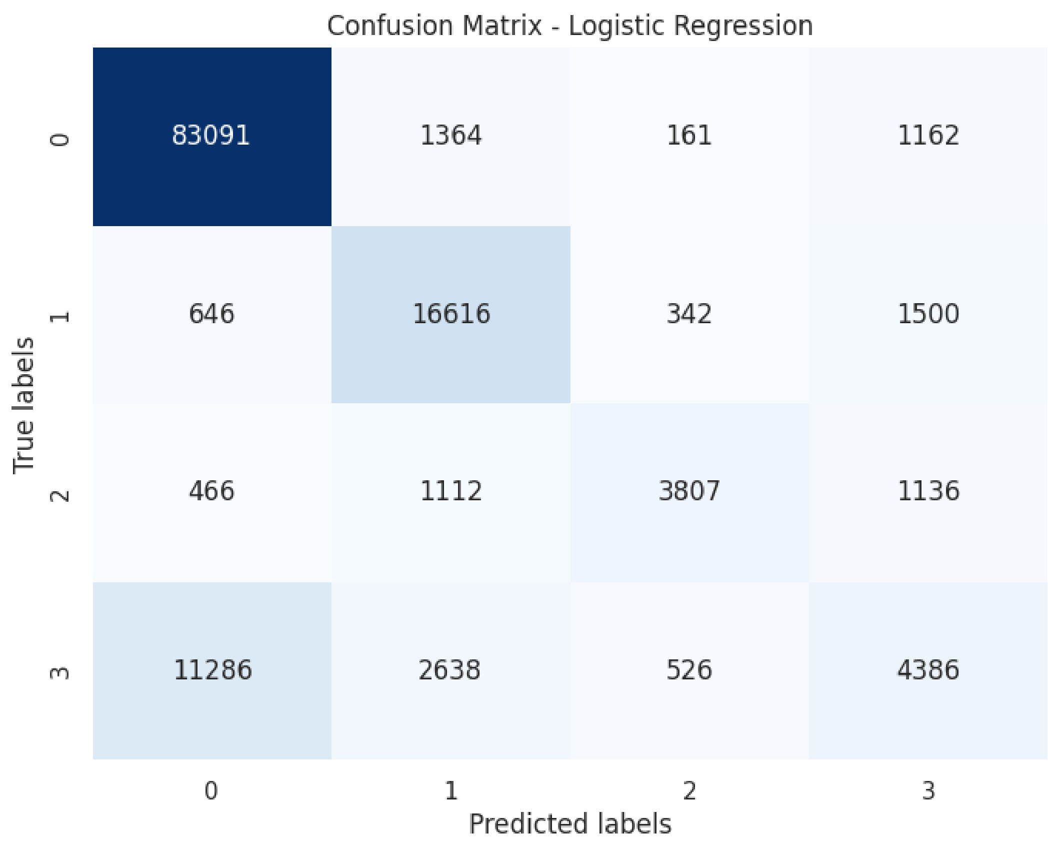

Figure 1: Logistic Regression

Table 1: Logistic Regression Performance Metrics

Metric	**Training**	**Testing**
Accuracy	0.8292	0.8285
Precision	0.8034	0.8022
Recall	0.8292	0.8285
F1 Score	0.8047	0.8037

4.1.2 Decision Tree

Prone to overfitting but useful for feature importance.

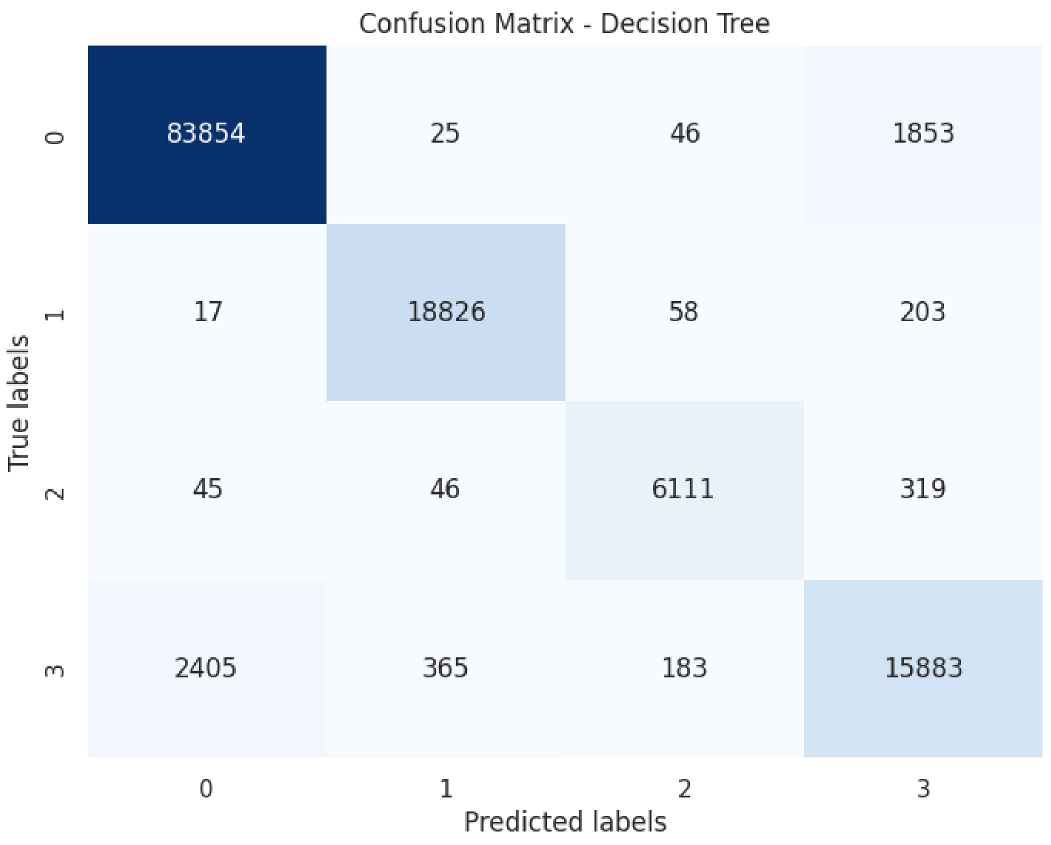

Figure 2: Decision Tree

Table 2: Decision Tree Performance Metrics

Metric	**Training**	**Testing**
Accuracy	0.9862	0.9573
Precision	0.9861	0.9568
Recall	0.9862	0.9573
F1 Score	0.9861	0.9570

4.1.3 Random Forest

Ensemble method, reduces overfitting.

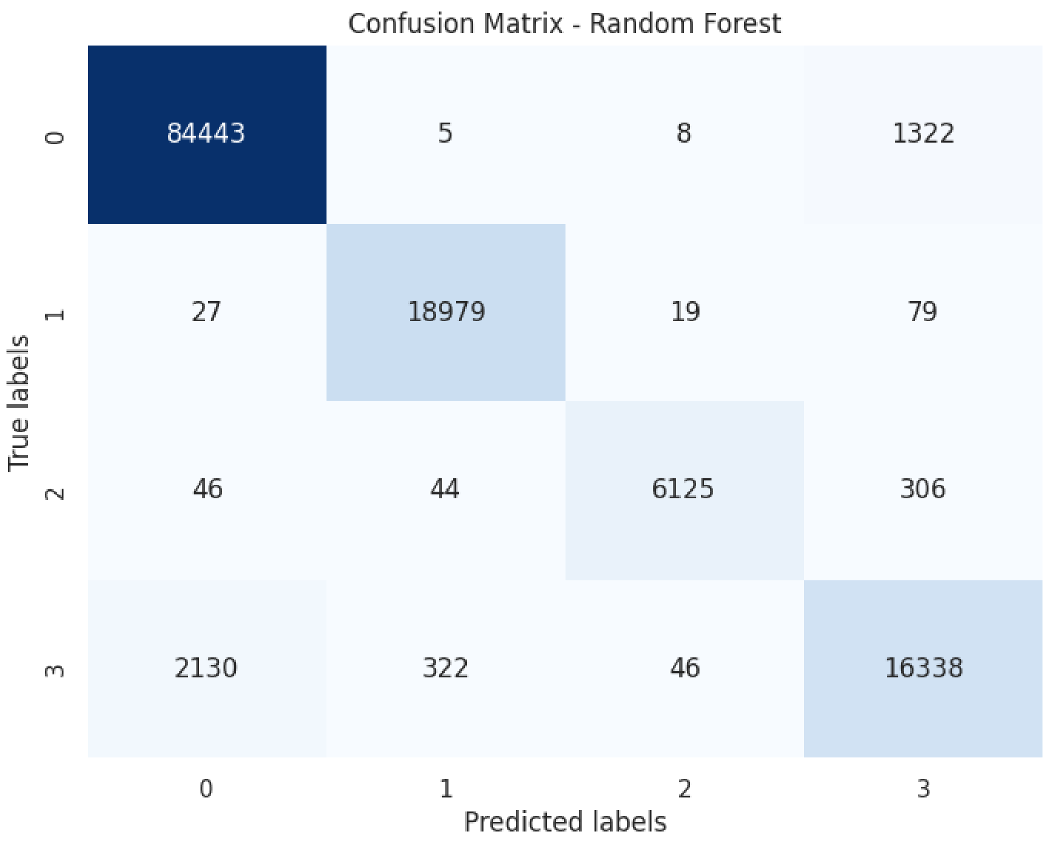

Figure 3: Random Forest

Table 3: Random Forest Performance Metrics

Metric	**Training**	**Testing**
Accuracy	0.98615	0.96657
Precision	0.98608	0.96617
Recall	0.98615	0.96657
F1 Score	0.98609	0.96626

4.1.4 XGBoost

Gradient boosting, high accuracy, handles imbalanced data well.

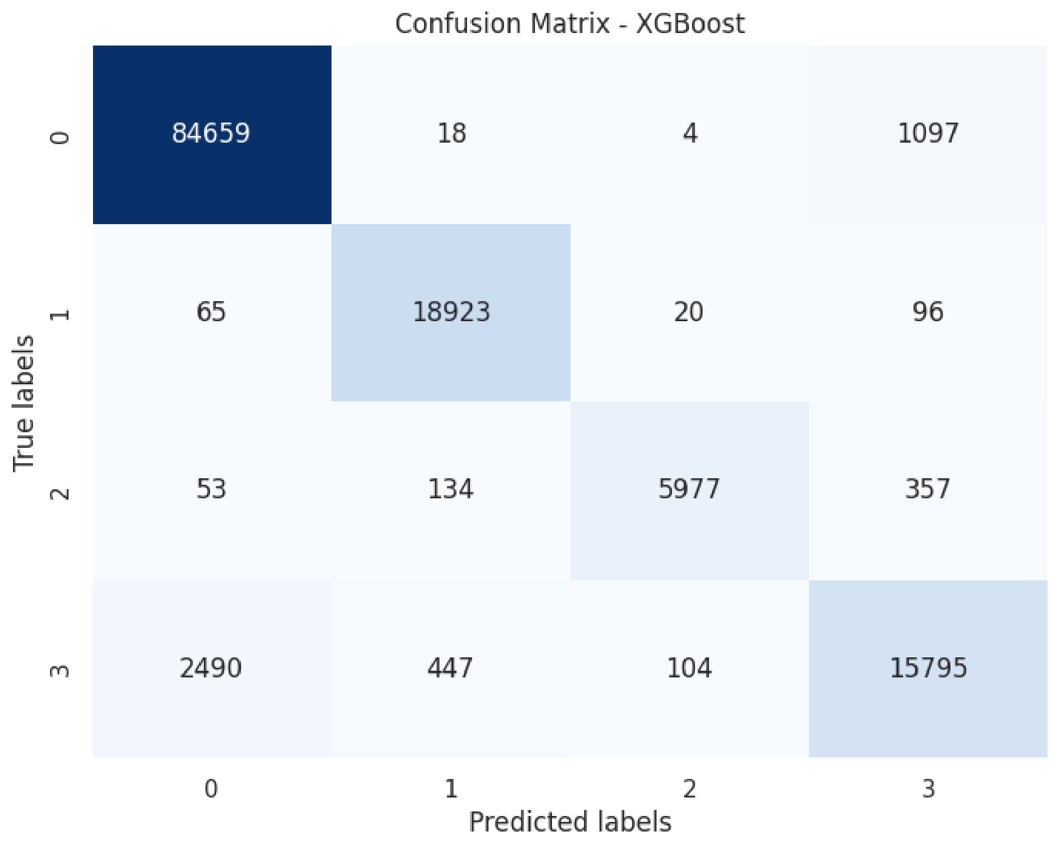

Figure 4: XGBoost

Table 4: XGBoost Performance Metrics

Metric	**Training**	**Testing**
Accuracy	0.96426	0.96249
Precision	0.96366	0.96184
Recall	0.96426	0.96249
F1 Score	0.96361	0.96184

4.2 Deep Learning Models

4.2.1 Simple RNN

Basic recurrent network for sequence modeling.

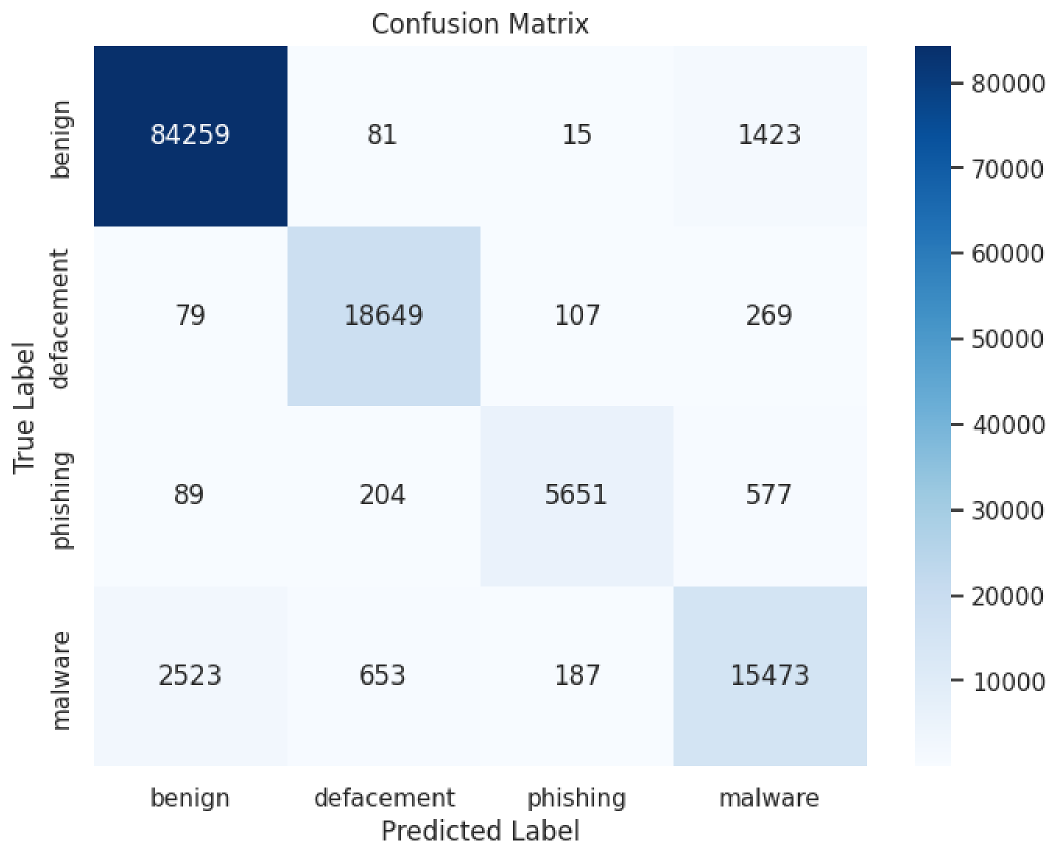

Figure 5: Simple RNN

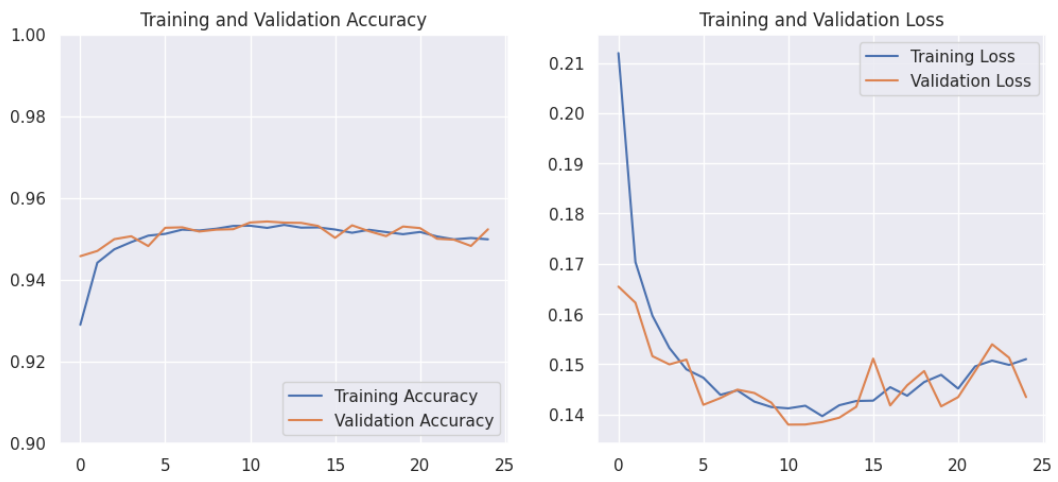

Figure 6: Simple RNN

4.2.2 LSTM RNN

Long Short-Term Memory, better at capturing long-term dependencies.

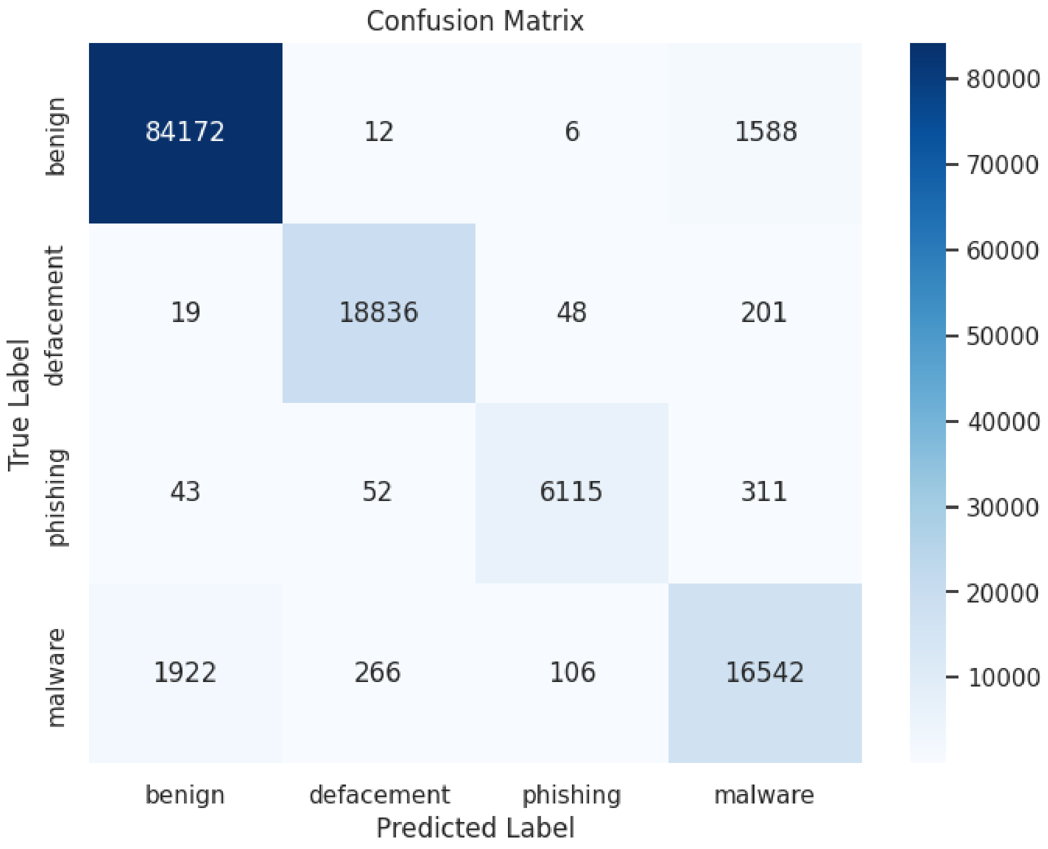

Figure 7: LSTM RNN

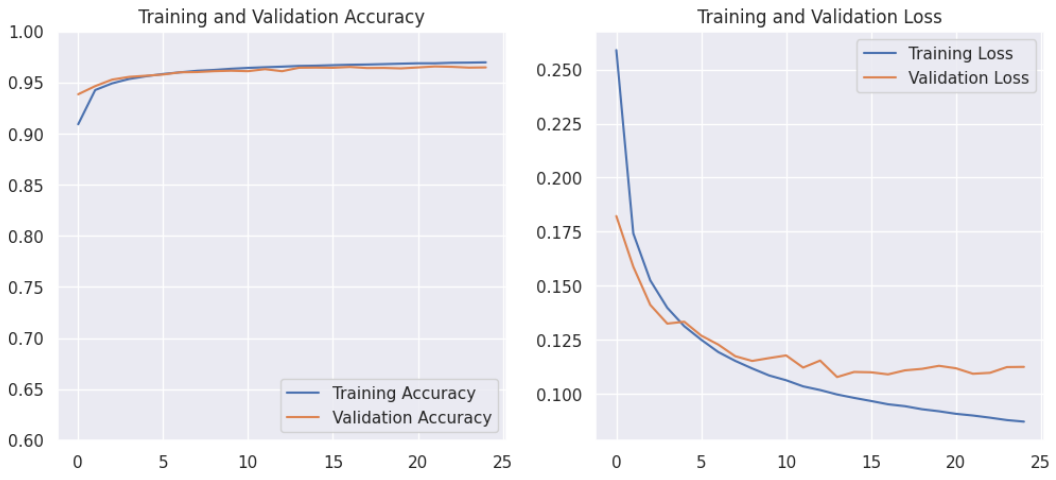

Figure 8: LSTM RNN

4.2.3 GRU RNN

Gated Recurrent Unit, faster than LSTM but slightly less accurate.

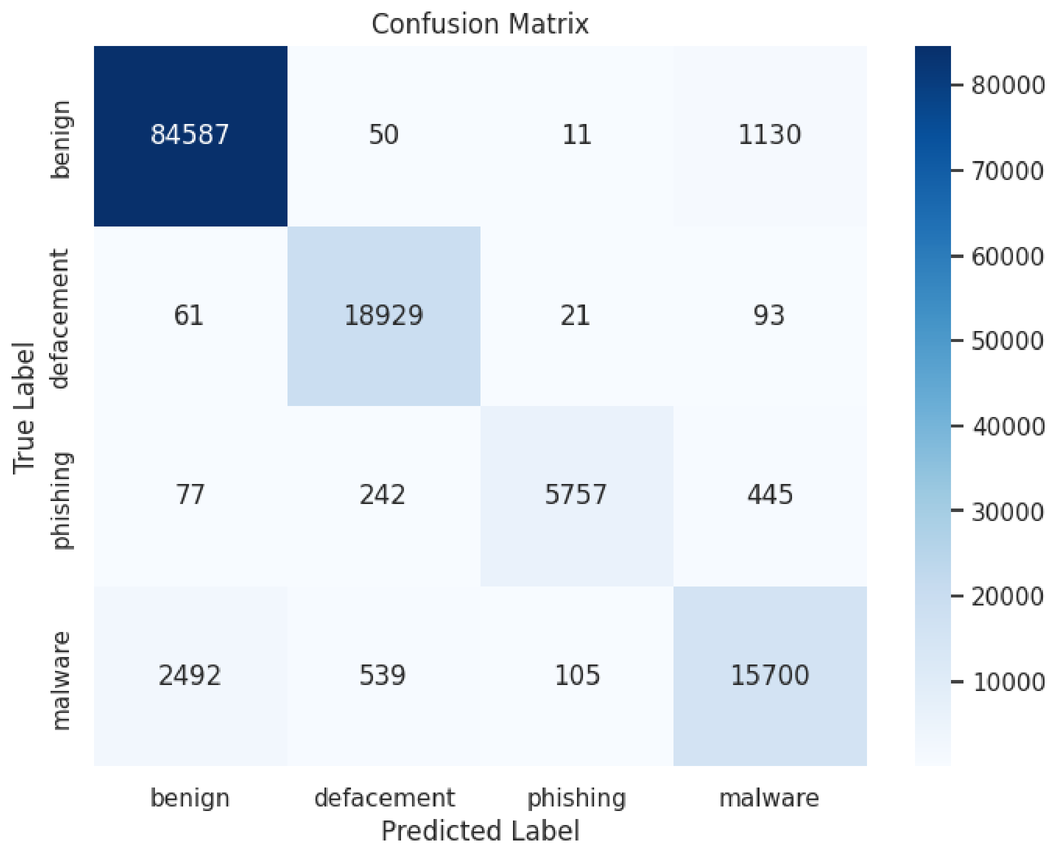

Figure 9: GRU RNN

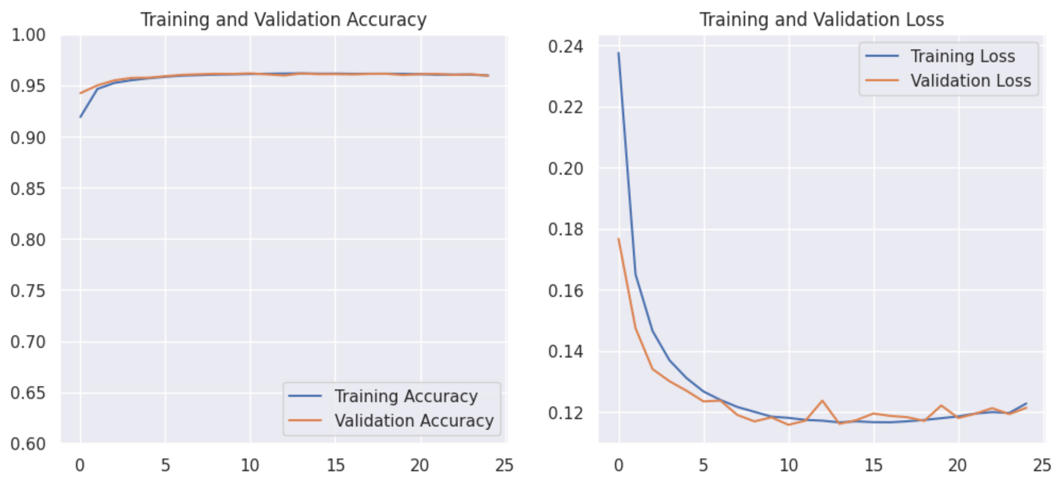

Figure 10: GRU RNN

4.3 Model Selection (Best Model: LSTM)

Why LSTM?

- Excels in sequence-based classification (URLs are sequential data).
- Handles long-term dependencies better than Simple RNN/GRU.
- Achieved the highest precision, recall, and F1-score on test data.

5 Deployment (Web Application)

The LSTM model was chosen for deployment due to its superior performance. The web application:

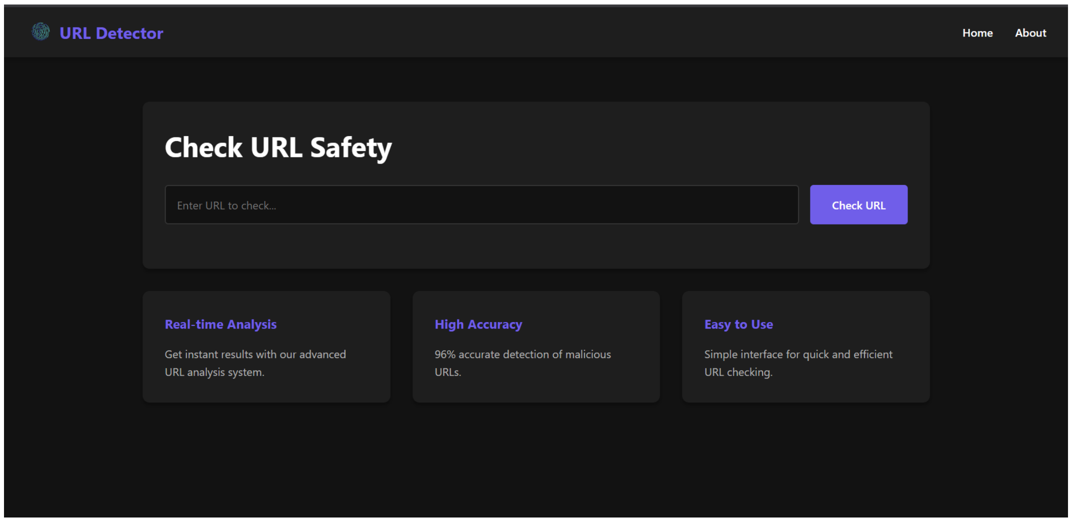
URL Detector
Home
About
Check URL Safety
Enter URL to check...
Check URL
Real-time Analysis
Get instant results with our advanced URL analysis system.
High Accuracy
96% accurate detection of malicious URLs.
Easy to Use
Simple interface for quick and efficient URL checking.

Figure 11: Malicious URL Detector Web Application

- **Input:** User enters a URL.
- **Processing:** The URL is preprocessed, and features are extracted.
- **Prediction:** The LSTM model classifies the URL into one of four categories.
- **Output:** Displays the result (Benign/Phishing/Defacement/Malware) with confidence.

Tech Stack for Deployment:

- **Frontend:** HTML, CSS, JavaScript (for user interaction).
- **Backend:** Flask/Django (Python-based web framework).
- **Model Serving:** TensorFlow/Keras for LSTM inference.

6 Conclusion

The Malicious URL Detector successfully classifies URLs into benign, phishing, defacement, and malware categories using lexical features and LSTM-based deep learning. The system is deployable as a web application for real-time detection, providing a robust defense against malicious URLs.