Report on Malicious URL Classification Using Machine Learning

Raj Vinayak 220861 **Gottupulla Venkata Aman** 220413 **Ashutosh Agarwal** 210219 **Varun Tandon** 211151

Group 2

Department of Computer Science

Indian Institute of Technology, Kanpur

1. Introduction

1.1 Background

The increase in cyber-attacks has made malicious URL detection a vital aspect of cybersecurity. Malicious URLs serve as vectors for phishing attacks, malware distribution, and other cyber threats, and they are often designed to look legitimate to deceive users and evade detection. Efficiently identifying these URLs is essential for preemptive threat mitigation.

In this project, we leverage machine learning to classify URLs as either malicious or benign and the malicious ones are further categorized into spam, phishing, defacement etc. By analyzing patterns and features within URLs, we aim to distinguish between potentially harmful URLs and safe ones. This classification process supports cybersecurity by enabling automated URL screening, thereby enhancing defenses against web-based threats.

1.2 Objective

The objective of this project is to develop a machine learning-based model to accurately classify URLs as either malicious or benign, leveraging the diverse features provided in the ISCX-URL2016_All.csv dataset. By analyzing URL characteristics such as entropy, directory names, and filenames, this project focuses on creating a reliable classification pipeline. Through thorough data preprocessing and feature selection, we aim to maximize the model's accuracy and robustness. Ultimately, this model will serve as an essential tool in cybersecurity, enabling efficient identification of harmful URLs to enhance threat detection and internet safety.

1.3 Challenges

Malicious URL detection is inherently challenging due to the adaptive tactics used by cyber attackers. Attackers frequently alter URLs, apply obfuscation techniques, and mimic legitimate web structures to evade detection. Furthermore, URLs vary widely in structure, requiring classifiers to be highly adaptable and resilient to different malicious behaviors. These challenges make machine learning particularly valuable, as it can detect subtle patterns in URL characteristics that traditional rule-based systems might miss.

2. Data Exploration

2.1 Dataset Overview

The *ISCX-URL2016_All.csv* dataset contains a set of features that describe URL attributes. The target variable, URL_Type_obf_Type, categorizes URLs as malicious or benign, allowing for supervised learning approaches to classification into the following categories.

2.1.1 Dataset Information

Using df.info(), we gathered metadata on the dataset, including column names, data types, and non-null counts. This step provided insights into potential issues with data types or missing values.

2.1.2 Dataset Description

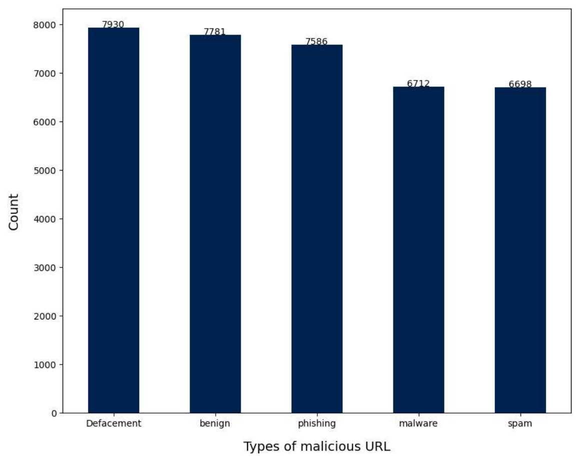
The summary statistics obtained with df.describe() offered an overview of the data distribution, including mean, standard deviation, and quartiles of the numerical columns. These statistics highlighted any irregularities, such as outliers, that might require further preprocessing.

2.2 Unique Target Values

The target column, URL_Type_obf_Type, was analyzed for its unique values using df['URL_Type_obf_Type'].unique(). This exploration revealed multiple categories within the target variable, which helped us understand the variety and balance of malicious versus benign URLs.

2.3 Frequency Distribution of Target

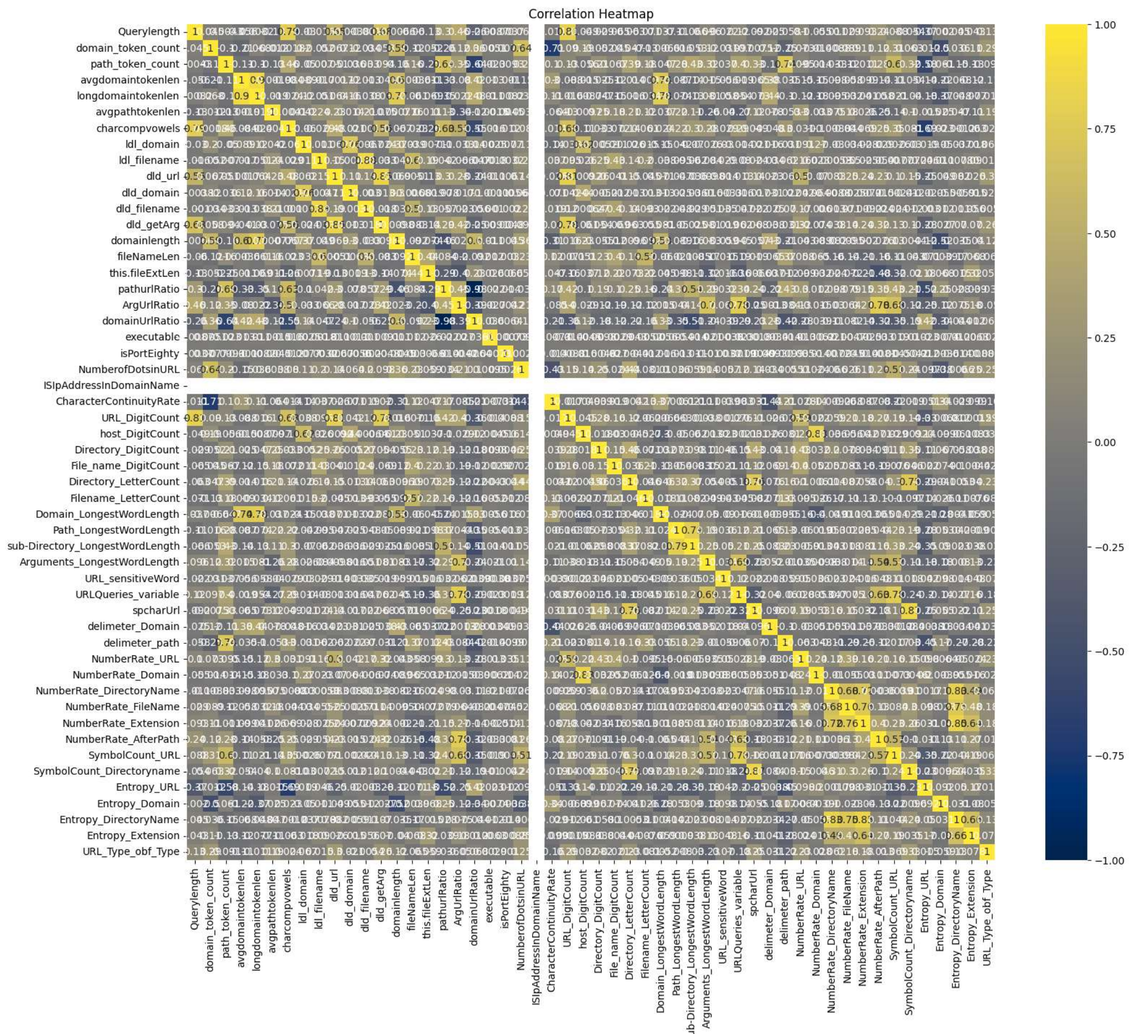
A bar chart depicting the frequency of each URL_Type_obf_Type value was generated to visualize the target distribution. This allowed us to assess the class balance, which is crucial for evaluating model performance on imbalanced data.



3. Missing Data Analysis

3.1 Heatmap of Missing Values

A heatmap of missing values was generated using sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap="cividis"). This visualization allowed us to identify columns with high rates of missing data, providing guidance on necessary data handling steps.



3.2 Handling Missing Data

To manage missing data, we filled missing values with the mean of each respective column using fillna(). This approach preserved dataset integrity and ensured compatibility with machine learning models.

4. Data Preprocessing

4.1 Feature Selection and Scaling

After calculating feature correlations, we removed highly correlated features (correlation > 0.9) to minimize redundancy. Then, we scaled all features with MinMaxScaler to place them on a uniform scale, improving model stability.

4.2 Label Encoding

The target variable URL_Type_obf_Type was label-encoded using LabelEncoder, converting it into a numerical format compatible with machine learning models.

---

5. Anomaly Detection

5.1 Isolation Forest

To eliminate outliers, we used the Isolation Forest algorithm, an anomaly detection model that excels in identifying unusual data points. This step was critical in refining both training and testing datasets.

---

6. Model Training and Hyperparameter Tuning

6.1 Initializing Models

Three machine learning models were chosen for this task: Decision Tree, Logistic Regression, and Random Forest. These models, each with unique strengths, were initialized and stored in a dictionary for easy access during training.

6.2 Hyperparameter Grid Search

A grid search with cross-validation was used to tune hyperparameters, maximizing model performance by finding the optimal settings.

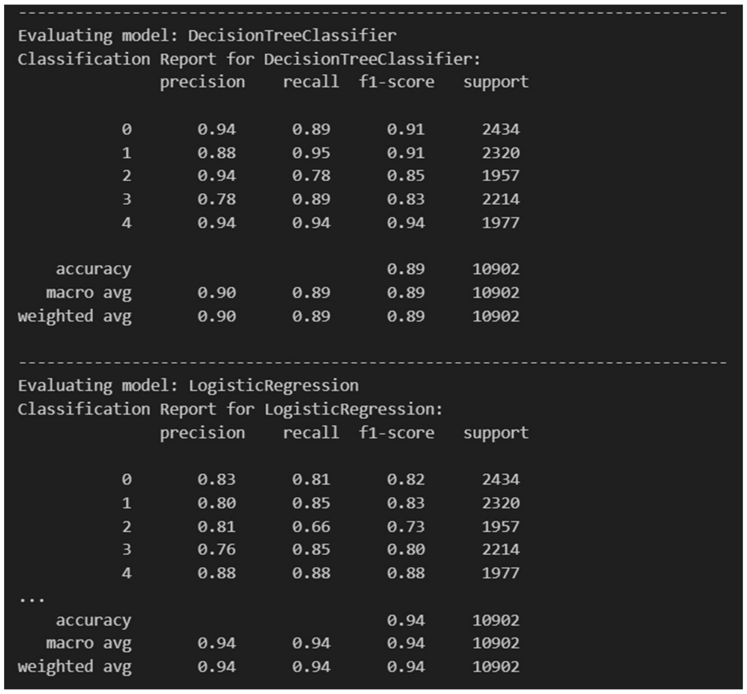
grid_search = GridSearchCV(model, param, cv=5, n_jobs=3, verbose=1, scoring='accuracy')

---

7. Model Evaluation

7.1 Evaluation Metrics

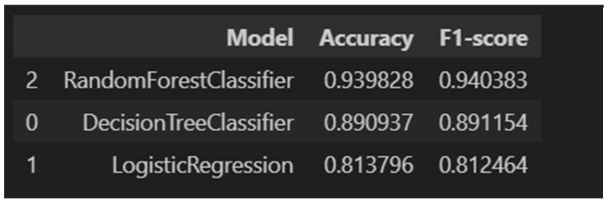
Our model evaluation relied on two primary metrics: Accuracy and Weighted F1-Score. These metrics provide insight into the classifier’s ability to handle imbalanced data while ensuring reliable results across URL types.



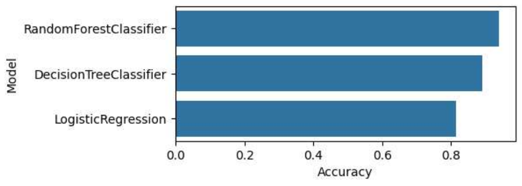
```
--------------------------------------------------------------------------
Evaluating model: DecisionTreeClassifier
Classification Report for DecisionTreeClassifier:
              precision    recall  f1-score   support

           0       0.94      0.89      0.91      2434
           1       0.88      0.95      0.91      2320
           2       0.94      0.78      0.85      1957
           3       0.78      0.89      0.83      2214
           4       0.94      0.94      0.94      1977

    accuracy                           0.89     10902
   macro avg       0.90      0.89      0.89     10902
weighted avg       0.90      0.89      0.89     10902

--------------------------------------------------------------------------
Evaluating model: LogisticRegression
Classification Report for LogisticRegression:
              precision    recall  f1-score   support

           0       0.83      0.81      0.82      2434
           1       0.80      0.85      0.83      2320
           2       0.81      0.66      0.73      1957
           3       0.76      0.85      0.80      2214
           4       0.88      0.88      0.88      1977
...
    accuracy                           0.94     10902
   macro avg       0.94      0.94      0.94     10902
weighted avg       0.94      0.94      0.94     10902
```

7.2 Results

The final results of model performance, including accuracy and F1-score, were presented in both tabular and graphical formats to facilitate comparison.



	Model	Accuracy	F1-score
2	RandomForestClassifier	0.939828	0.940383
0	DecisionTreeClassifier	0.890937	0.891154
1	LogisticRegression	0.813796	0.812464



---

8. Conclusion

This project developed a machine learning model for classifying URLs as malicious or benign. Through comprehensive preprocessing steps, including handling missing data, scaling features, and removing anomalies, we prepared the dataset for optimal model performance. The models were then fine-tuned, and the Random Forest classifier emerged as the top performer based on accuracy and F1-score.

---

9. Future Work

For further improvement, the following enhancements can be explored:

- Advanced Anomaly Detection: Implementing more sophisticated anomaly detection methods, such as robust principal component analysis, could yield better data refinement.
- Additional Classification Models: Incorporating models like Support Vector Machines (SVM) or Gradient Boosting Machines (GBM) could improve classification performance.
- Class Imbalance Techniques: Techniques like Synthetic Minority Over-sampling (SMOTE) would help mitigate class imbalance, potentially increasing classifier robustness across categories.

10. GitHub Repository & Acknowledgments

All code, documentation, and resources for this project are available on GitHub, providing open access for collaboration and further development. The repository includes the machine learning model, data preprocessing scripts, and deployment instructions, facilitating ease of use and reproducibility. We would like to acknowledge the **Canadian Institute for Cybersecurity** for providing the foundational dataset, ISCX-URL2016_All.csv, which has been crucial in enabling robust URL classification for our project.

GitHub Repository Link: [Project link]

11. Web Application and Chrome Extension Implementation

To make our model accessible and user-friendly, we have developed a web application that allows users to classify URLs in real time. Built with a **Flask backend**, the application takes user-submitted URLs, processes them using the trained machine learning model in the backend, and returns the classification result immediately. In addition, we have also dockerized the flask app and deployed it on render. This tool is also available as a chrome extension making it easy to verify the authenticity of any URL on the get-go. This interface enables easy integration of our model into practical cybersecurity applications, offering an interactive tool for URL threat assessment and further enhancing usability for non-technical users.