A Report on Malicious URL Detection

Submitted By:
(Group IV)
Aishwarya Karki (C0903073)
Bikash Thapa Magar (C0907642)
Nirmala Regmi (C0903616)
Samar Fathima (C0908466)
Sijal Shrestha (C0910639)
Sujata Gurung (C0903143)

Submitted To:
Bhavik Gandhi

TABLE OF CONTENTS

INTRODUCTION

In the realm of cybersecurity, the identification and neutralization of malicious online entities are paramount imperatives. Amidst this digital landscape, "Malicious URL Detection" stands as a crucial necessity, offering a sophisticated system designed to differentiate between harmless and hazardous URLs with precision and efficiency. At its core, this project is dedicated to the pivotal task of classifying URLs, discerning whether they harbor malicious intent or serve as benign conduits to digital content.

The urgency for "Malicious URL Detection" is underscored by the escalating sophistication and frequency of cyber threats. Malicious URLs serve as potent vectors for cyberattacks, enabling the spread of malware, phishing attempts, and other malicious activities. Therefore, the swift and accurate identification of such URLs is essential in safeguarding digital assets, sensitive information, and organizational integrity. Moreover, in an era marked by rapid technological advancements and evolving threat landscapes, the development of robust URL detection mechanisms is paramount to outmaneuvering cyber adversaries. By harnessing comprehensive datasets and innovative methodologies, "Malicious URL Detection" empowers cybersecurity professionals with the tools and insights needed to fortify defenses and mitigate risks posed by malicious URLs.

This report presents an evaluation of various machine learning classifiers for the task of malicious URL detection. The classifiers under consideration include L1 and L2 Regularized Logistic Regression, K-Nearest Neighbors (KNN), Gaussian Naive Bayes (GaussianNB), Decision Tree Classifier, and Random Forest Classifier. The performance of each classifier is assessed in terms of accuracy, aiming to ascertain which method offers the most effective detection of malicious URLs.

DATA COLLECTION AND UNDERSTANDING

The Kaggle Dataset on Malicious URL Detection represents a pivotal resource in the realm of cybersecurity, specifically tailored to facilitate research and development in the field of threat detection. Comprising a diverse assortment of URLs, encompassing both benign and malicious entities, the dataset is accompanied by an extensive array of calculated features. These features span various dimensions of URL characteristics, including basic components, domain information, content analysis, path and query parameters, SSL certificate details, host reputation, network features, machine learning-derived features, and behavioral nuances.

By encompassing such a broad spectrum of features, the dataset offers a holistic view of URLs, empowering researchers and data scientists to gain comprehensive insights into potential threats. Moreover, the dataset serves as a foundation for the development and benchmarking of URL detection models, leveraging a plethora of machine learning and deep learning techniques. Researchers can explore novel methodologies and approaches, capitalizing on the abundance of calculated features to enhance the accuracy and robustness of URL classification systems.

Beyond its utility in research and development, the dataset fosters collaboration and knowledge sharing within the cybersecurity community. By encouraging the exchange of insights and methodologies, it promotes collective efforts towards advancing cyber vigilance and fortifying digital defenses against malicious URLs. In essence, the Kaggle Dataset on Malicious URL Detection stands as a cornerstone in the pursuit of cyber resilience, offering a wealth of data to fuel innovation and collaboration in the ongoing battle against online threats.

The link to the dataset is:
https://www.kaggle.com/datasets/pilarpieiro/tabular-dataset-ready-for-malicious-url-detection

DATA PROCESSING AND FEATURE ENGINEERING

In Data Processing and Feature Engineering, we've streamlined our system by removing all dataset columns for universal compatibility. Then, we extract essential features for model training and future applications.

- **Replace 'www.' with an empty string in the 'url' column:** This line of code uses the replace() function to remove 'www.' from the URLs in the 'url' column of the dataset. The regex=True parameter indicates that the operation should be performed using regular expressions.

- **Add a new column 'url_len' to store the length of each URL:** Here, a new column named 'url_len' is created in the dataset, storing the length of each URL. This is achieved by applying the len() function to each URL string using the apply() function.

- **Extract the top-level domain (TLD) from each URL and store it in the 'domain' column:** This code defines a function called process_tld() that extracts the top-level domain (TLD) from a given URL. The function is then applied to each URL in the 'url' column using the apply() function, and the extracted TLDs are stored in a new column named 'domain'.

- **Check if the URL contains the hostname and store the result in the 'abnormal_url' column:** This code defines a function called abnormal_url() that checks if the URL contains the hostname within it. The function returns 1 if the hostname is found within the URL and 0 otherwise. The function is applied to each URL in the 'url' column, and the results are stored in a new column named 'abnormal_url'.

- **Check if the URL contains an IP address and store the result in the 'use_of_ip_address' column:** This code defines a function called having_ip_address() that checks if the URL contains an IP address in its domain. The function returns 1 if an IP address is found in the URL and 0 otherwise. The function is applied to each URL in the 'url' column, and the results are stored in a new column named 'use_of_ip_address'.

- **Count the occurrences of special characters in each URL and store the counts in separate columns for each character:** This code defines a list of special characters and then iterates over each character to count its occurrences in each URL. The counts are stored in separate columns for each character.

- **Summarize the total count of special characters in each URL and store it in the 'sum_count_special_chars' column:** This code defines a function called sum_count_special_characters() that counts the total occurrences of special characters in

a given URL. The function is applied to each URL in the 'url' column, and the total counts are stored in a new column named 'sum_count_special_chars'.

- **Drop unwanted columns from the dataset:** This line of code removes unwanted columns from the dataset. It uses the drop() function to specify the columns to be dropped based on their index.
- **Check if the URL uses the HTTPS protocol and store the result in the 'https' column:** This code defines a function called httpSecured() that checks if the URL uses the HTTPS protocol. The function returns 1 if the URL uses HTTPS and 0 otherwise. The function is applied to each URL in the 'url' column, and the results are stored in a new column named 'https'.

- **Count the number of digits in each URL and store the counts in the 'digits' column:** This code defines a function called digit_count() that counts the number of digits in a given URL. The function is applied to each URL in the 'url' column, and the counts are stored in a new column named 'digits'.

- **Count the number of letters (alphabetic characters) in each URL and store the counts in the 'letters' column:** This code defines a function called letter_count() that counts the number of letters (alphabetic characters) in a given URL. The function is applied to each URL in the 'url' column, and the counts are stored in a new column named 'letters'.

- **Check if the URL is from a URL shortening service and store the result in the 'Shortining_Service' column:** This code defines a function called Shortining_Service() that checks if the URL is from a URL shortening service. The function returns 1 if the URL is from a shortening service and 0 otherwise. The function is applied to each URL in the 'url' column, and the results are stored in a new column named 'Shortining_Service'.

- **Check if the URL is indexed by Google and store the result in the 'google_index' column:** This code defines a function called google_index() that checks if the URL is indexed by Google. The function returns 1 if the URL is indexed and 0 otherwise. The function is applied to each URL in the 'url' column, and the results are stored in a new column named 'google_index'.

- **Remove unnecessary columns from the dataset:** This line of code drops unwanted columns from the dataset using the drop() function, specifying the columns to be dropped based on their names.

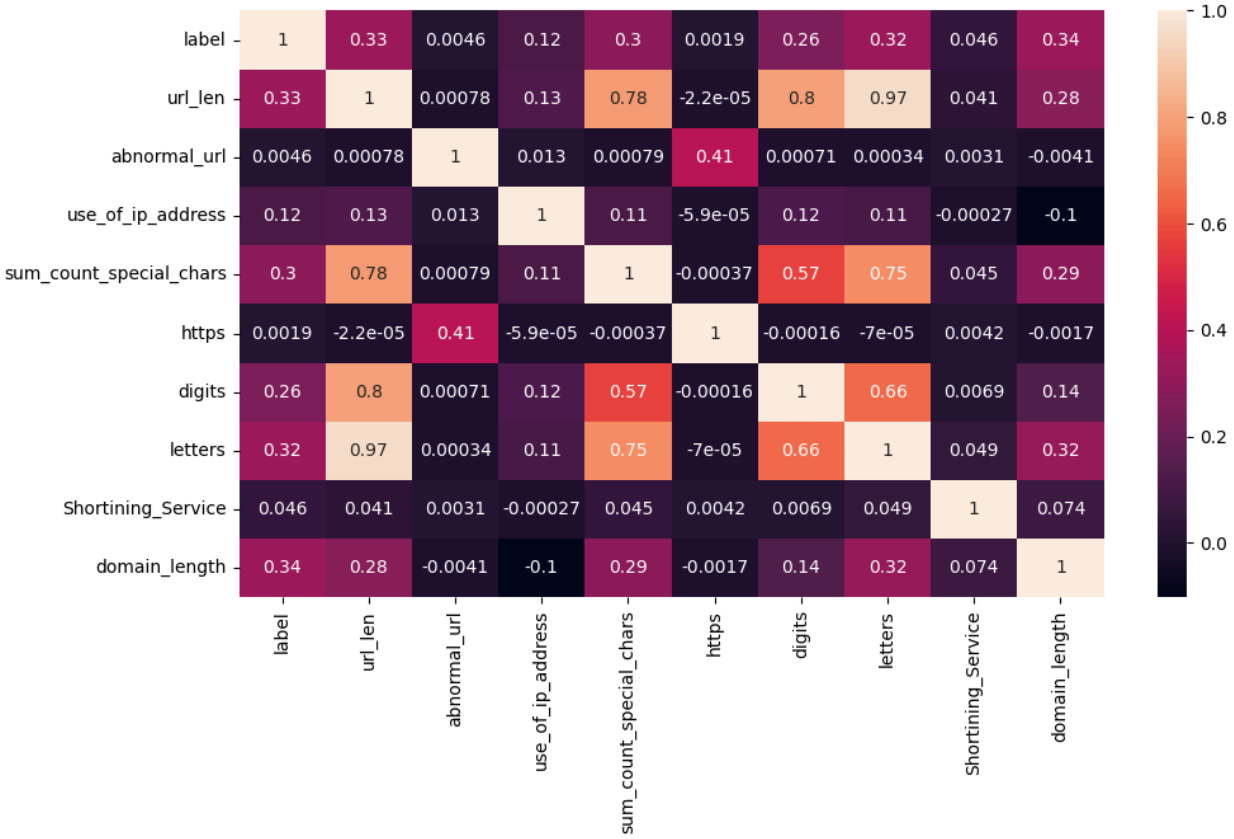
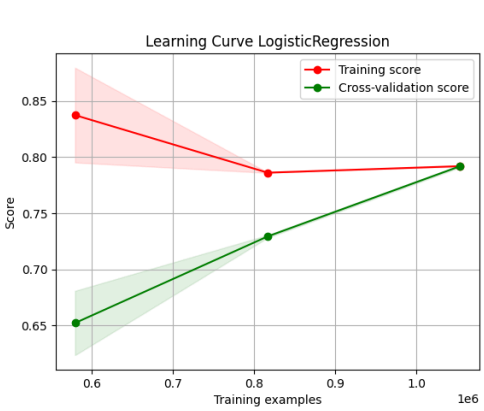
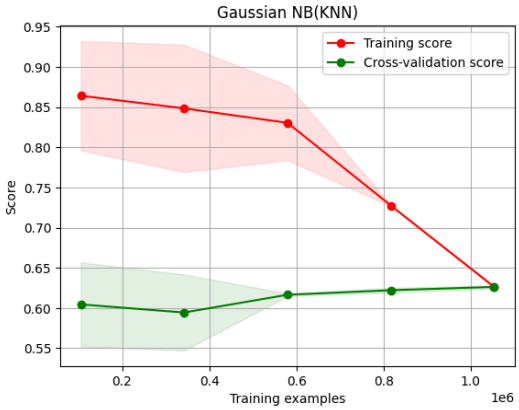
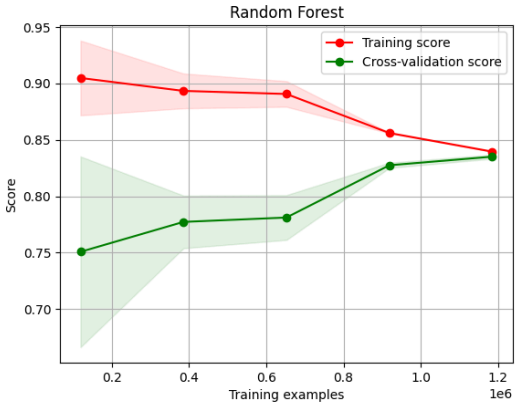


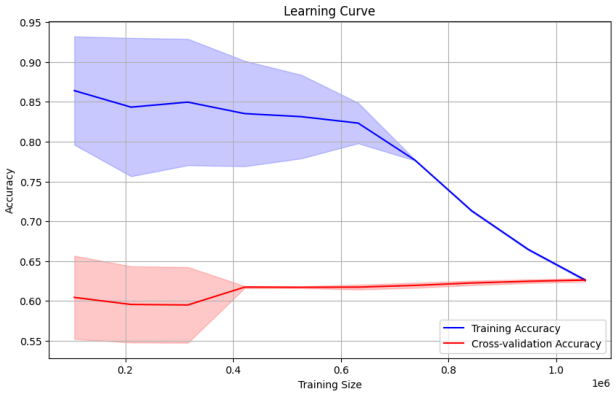
Figure: Correlation Matrix

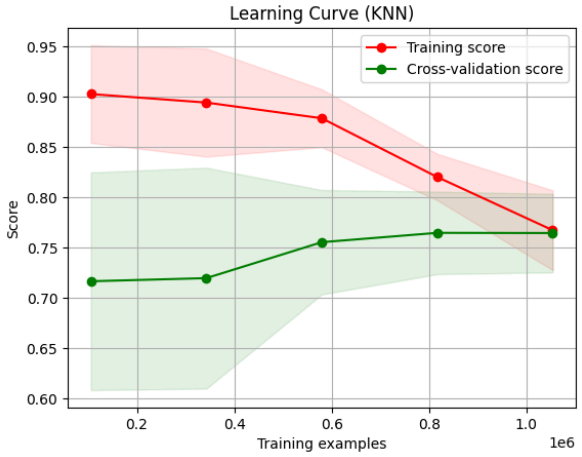
MODEL BUILDING, IMPLEMENTATION AND COMPARISON





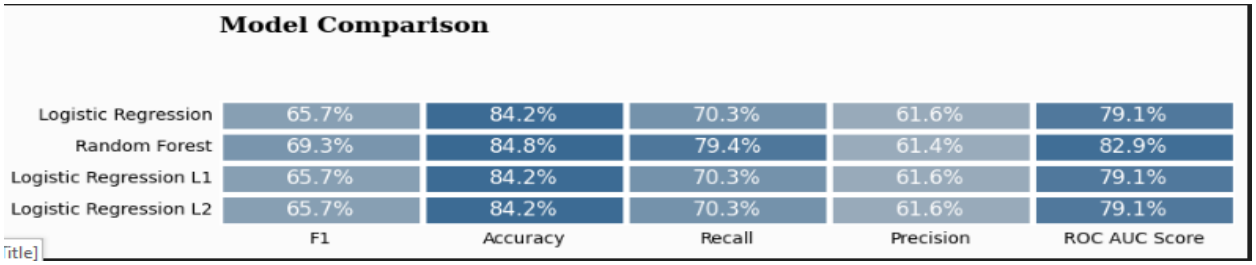






Visualizing learning curves depicts model performance evolution with increasing training data

size, aiding in assessing model effectiveness and scalability.



Model Performance Metrics:

- F1 Score: A metric that combines precision and recall, providing a balance between the two. It represents the harmonic mean of precision and recall.
- Accuracy: The percentage of correctly predicted outcomes (both positive and negative) out of all predictions made by the model.
- Recall: The proportion of actual positive cases (true positives) that were correctly identified by the model out of all positive cases in the dataset.
- Precision: The proportion of true positive predictions among all positive predictions made by the model.
- ROC AUC Score: The area under the Receiver Operating Characteristic curve, indicating the model's ability to distinguish between positive and negative cases across different thresholds.

Comparison of Models:

- Logistic Regression (LR):
  Achieves an F1 score of 65.7, indicating a balance between precision and recall.
  Has an accuracy of 84.2%, implying a high level of correctness in predictions.
  Shows a recall of 70.3%, meaning it correctly identifies around 70.3% of the actual positive cases.
  Exhibits a precision of 61.6%, indicating that around 61.6% of the predicted positive cases are true positives.
  Has a ROC AUC score of 79.1, suggesting good discriminative ability between positive and negative cases.
- Random Forest:
  Achieved a higher F1 score of 69.3 compared to Logistic Regression.
  Demonstrates slightly higher accuracy (84.8%) and significantly higher recall (79.4%), indicating better overall performance.

Exhibits a similar precision (61.4%) compared to Logistic Regression.
Shows a higher ROC AUC score of 82.9, indicating better discrimination between positive and negative cases compared to Logistic Regression.

- Logistic Regression with L1 regularization and Logistic Regression with L2 regularization:
  Both models exhibit similar performance to the standard Logistic Regression model in terms of F1 score, accuracy, recall, precision, and ROC AUC score.

Interpretation: Random Forest emerges as the preferable model in terms of overall performance, with higher F1 score, accuracy, recall, and ROC AUC score compared to Logistic Regression models.
The choice between Logistic Regression and Random Forest may depend on specific requirements such as interpretability, computational resources, and the importance of different performance metrics.
It's important to consider the trade-offs between different metrics and select the model that best aligns with the objectives of the task or application.

MECE TABLE

Category	Description	Assigned Member
Data Collection	**Methods and sources for gathering relevant data.**	
	- Data Search	**Sujata Gurung**
	- Data Collection	**Samar Fathima**
Data Preprocessing	**Techniques for cleaning, transforming, and preparing the data.**	
	- Missing value imputation	**Bikash Thapa Magar**

Feature Engineering	**Creation and selection of features to train the AI model.**	
	- Feature Addition	**Bikash Thapa Magar**
	- EDA	**Nirmala Regmi**
	- Correlation Matrix	**Bikash Thapa Magar**
	- Data visualizations	**Bikash Thapa Magar**
Model Selection	**Selection of appropriate algorithms or models for the task.**	
	- Logistic Regression	**Nirmala Regmi**
	- KNN	**Sijal Shrestha**
	- GaussianNB	**Sijal Shrestha**
	- Decision tree	**Samar Fathima**
	- Random Forest	**Aishwarya Karki**
Model Training	**Training the selected model using the prepared data.**	**Nirmala Regmi**
	- Cross-validation	
	L1 Regularization	**Sijal Shrestha**
	L2 Regularization	**Aishwarya Karki**

Model Evaluation	**Assessing the performance of the trained model.**	
	- Accuracy	**Bikash Thapa Magar**
	- Precision and recall	**Samar Fathima**
	- F1 score	**Sujata Gurung**
	- ROC curve	**Aishwarya Karki**
	Comparative Analysis	**Aishwarya Karki**
Deployment	**Deploying the trained model into production or real-world usage.**	
	- Research on Flask	**Aishwarya Karki**
	UI design and Development	**Aishwarya Karki**
Documentation	**Proposal Docmumenatation**	**Samar Fathima**
	Proposal Presentation	**Sujata Gurung**
	Final Report Documenataion	**Bikash Thapa Magar**
	Final Presentation	**Sijal Shrestha**

GitHub Link:

https://github.com/aishwaryakarki001/Malicious-URL-Detection/tree/master

Project Board Link:

https://github.com/users/aishwaryakarki001/projects/3

REFERENCES

1. "Tabular dataset ready for malicious url detection." *Kaggle*, https://www.kaggle.com/datasets/pilarpieiro/tabular-dataset-ready-for-malicious-url-detection. Accessed 15 April 2024.