WhalingGuard : Phishing Detection Using Machine Learning

Adithyan M S *Department of CSE CEM-Punnapra* Alappuzha, India [adith](mailto:adithyan2ms@gmail.com)[yan2ms@gmail.com](mailto:yan2ms@gmail.com)

Harikrishnan K B *Department of CSE CEM-Punnapra* Alappuzha, India [baluoger91@gmail.com](mailto:baluoger91@gmail.com)

Krishnapriya V J

*Asst. Professor, Department of IT CEM-Punnapra* Alappuzha, India

[krishnapriya](mailto:krishnapriyavj@gmail.com)[vj@gmail.com](mailto:vj@gmail.com)

***Abstract*—Phishing is a type of cyberattack that aims to steal sensitive information such as usernames, passwords, and credit card details. Machine learning algorithms have been proposed to detect phishing websites by analyzing various features.**

**In this research paper, we compare several machine learning algorithms to detect phishing websites using a dataset of 545,895 samples which is created by pre-processing and merging two separate datasets. We use 74 features to train and evaluate the algorithms, and we found that Logistic Regression (LR) achieved the highest accuracy of 94.44% after comparing models. The trained LR model is integrated into our WhalingGuard Web Application for predicting whether the input URL from the user interface is phishing or non-phishing.**

***Index Terms*—Phishing, Machine Learning, Logistic Regression,**

1. Introduction

Phishing detection is a crucial aspect of cybersecurity aimed at identifying and preventing phishing attacks. Phishing refers to a malicious practice where cybercriminals impersonate legitimate entities or organizations to deceive individuals into sharing sensitive information such as passwords, financial details, or personal data. These attackers often use email, text messages, or deceptive websites to trick unsuspecting users.

Phishing attacks can have severe consequences, including identity theft, financial loss, and compromise of sensitive data. To combat this threat, various techniques and technologies have been developed to detect and mitigate phishing attempts. Phishing detection involves the use of sophisticated algorithms, machine learning models, and behavioral analysis to identify patterns and indicators that differentiate legitimate communications from phishing attempts. 36% of all data breaches involved phishing according to Verizon’s 2022 report. It was estimated that by 2022 a ransomware or phishing attack will occur every 11 seconds.

Website phishing, also known as phishing websites or fake websites, refers to the creation of fraudulent websites that mimic legitimate websites to deceive users into revealing sensitive information or performing malicious actions. These phishing websites are designed to look and feel like the real ones, often imitating the branding, layout, and functionality of trusted organizations, businesses, or online services. The Phishing statistics suggests that compare to malware sites, phishing sites are 75% higher in presence. It was identified

that 61% of subjects in a study conducted could not differentiate between a real and a fake Amazon login page. So, a process/system for identifying and mitigating phishing attacks that occur through fraudulent websites should be implemented Website, which is termed as website phishing detection.

In this context, we suggest a phishing detection model based on machine learning that can detect whether a website URL relates to phishing or not. We have compared several machine learning algorithms in the proposed phishing detection model. We extracted multiple features from about 5.5 lakh URLs. And these features where used for training and testing the model. After comparing models, the results indicate that Logistic Regression provides better accuracy than any other machine learning algorithms. We have developed a web application that accepts input URLs from the user to predict whether it is phishing or legitimate based on the evaluated model.

1. PHISHING DETECTION MODEL

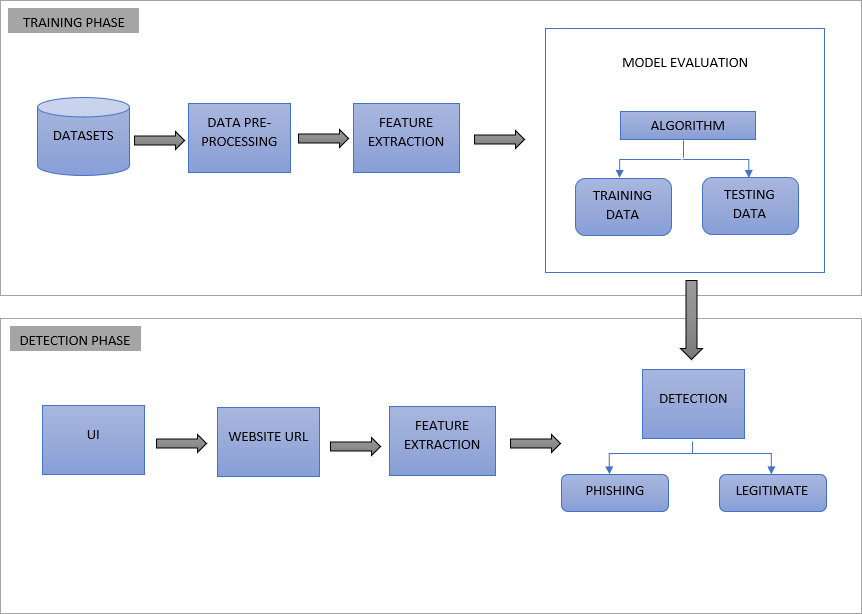


Fig. 1: System Architecture

In this section, we describe the phishing detection frame- work which consists of two major parts as shown in Fig:1. The first part is the Training Phase and the second part is the Detection Phase. In the first phase, datasets containing URLs are prepared for the machine learning algorithm and model is evaluated. In the second phase, an input URL is received from

the user and based on the evaluated model, the URL is classified into phishing or legitimate.

*DATASET:*

The dataset is created by merging two different datasets to improve the efficiency of prediction.

1. Dataset-I

This dataset is collected from phishtank.com which contains 96,020 data with 50% phishing and 50% of legitimate URLs. This dataset contains columns - domain and label, where domain represents the URLs and label denotes whether the URL is phishing or legitimate by representing 1 and 0 respectively.

1. Dataset-II

This dataset is collected from Kaggle.com which contains 450,176 data, having URL, label and result as columns.



Fig. 2: Dataset-I from phishtank.com

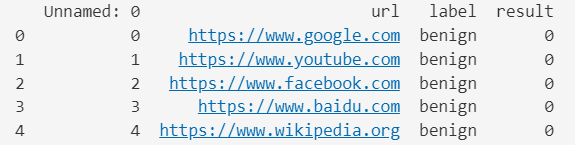


Fig. 3: Dataset-II from kaggle.com

1. IMPLEMENTATION The implementation is divided into five parts.
2. Data Pre-processing
3. Feature Extraction
4. Model Comparison
5. Model Evaluation
6. Model Implementation
7. *DATA PRE-PROCESSING*

Data pre-processing is an essential step in preparing raw data for analysis and machine learning tasks. It involves trans- forming and cleaning the data to ensure its quality, consistency, and suitability for further analysis. We obtained two separate datasets from different sources and merged them to create a larger dataset for training and testing our machine learning models.

In the Dataset-I (URL and label as columns), we found 92 URLs without any corresponding label values i.e., having null values. and these rows were dropped from these datasets. We only need the URLs and their corresponding label values (1 0r 2 representing phishing or legitimate) as the input data, the Dataset-II contains unwanted columns so they are also

dropped. Next, we will be merging our 2 datasets, for that we will be considering the Datatypes and names of the columns to avoid damage of the resultant dataset after merging. Merging the DataFrames with different data types for the same column name might lead to inconsistencies and unexpected behavior and merging DataFrames having different column names pro- duce splitting of the values. So, they both are handled before merging. After Merging we obtained a dataset having 546,104 data and saved in a new csv file.

After merging there are chances of having duplicate data inside the resultant dataset. There were 194 Duplicate URLs in our resultant dataset. And these were dropped. Fig 4. shows the Bar-Graph Classification of Phishing and Legitimate URLs inside the new dataset:

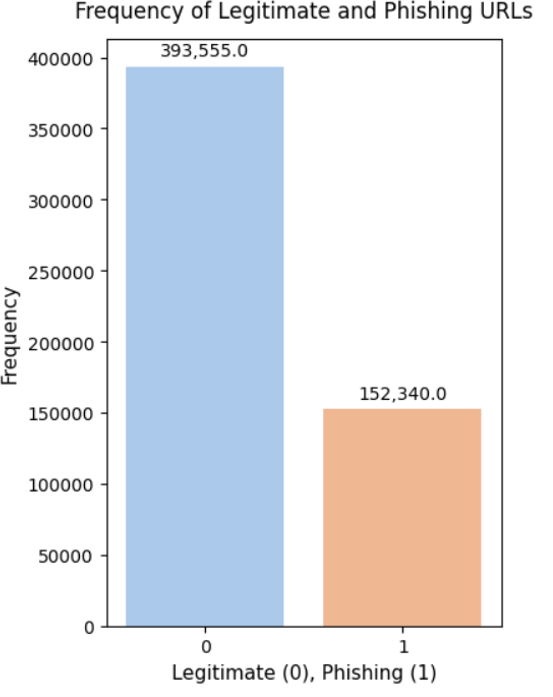


Fig. 4: Bar-Graph Classification of Phishing and Legitimate data.

1. *FEATURE EXTRACTION*

Feature extraction is a technique used to reduce the dimensionalities of data by transforming the raw input into a set of derived features that capture the essential information. It aims to highlight the most relevant aspects of the data and discard irrelevant or redundant information.

In our project, the features are generally classified into two:

* 1. Lexical Features
  2. Numerical Features

*Lexical Features*

Lexical features in URL feature extraction involve analyzing the textual components and patterns within a URL. These features capture characteristics related to the structure, keywords, symbols, and other textual elements of a URL.

* getEntropy :- It is known that DGA(Domain Generation Algorithm) domains have a greater level of disorderliness in their alphabetic distribution. Legitimate domains tend to have well-defined names that speak to a brand or a

product so tend to be less disorganized. Thus, measuring the entropy of URL strings tells us which domain names are ‘not-so-real.

* hasLogin :- Check if the URL contains specific keyword ”login,” .We can capture the fact that certain ‘red flag’ keywords appear in a URL string. These keywords may relate to keywords attackers use when trying to spoof a legitimate page or keywords that relate to popular nomenclature of security settings on a website that a hacker will try to manipulate. So, If the URL string contains “login” keyword, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).
* Redirection :- Phishing attackers often employ redirection techniques to hide the true destination of a malicious URL. By redirecting users through multiple URLs or using URL shorteners, they can make it more difficult for users and security systems to identify the final destination. Checks the presence of ‘//’ in the URL. The existence of “// within the URL path means that the user will be redirected to another website. (avoiding the ‘//’ after the http/https)
* lenClassify :- Computes the length of the URL. Phishers can use long URL to hide the doubtful part in the address bar. In this project, if the length of the URL is greater than or equal 54 characters then the URL classified as phishing otherwise legitimate. If the length of URL >= 54, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).
* haveAtSign :- Checks for the presence of ’@’ symbol in the URL. Using “@” symbol in the URL leads the browser to ignore everything preceding the “@” symbol and the real address often follows the “@” symbol. And, also used to mimic or spoof well-known websites or services. If the URL has ‘@’ symbol, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).
* getDepth :- The depth of a URL refers to the number of hierarchical levels or directories in the URL’s path. It represents how nested a specific resource is within a website’s directory structure. This feature calculates the number of subpages in the given URL based on the ‘/’.
* tinyURL :- URL shortening is a method in which a URL are shortened and can still lead to the required web- page. And this is accomplished by most of the phishing websites. If, the URL is using Shortening Services, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).
* isDomainIp :-Some Phishing URLs represents the domain name by IP address instead of hostname. Checks for the presence of IP address in the URL. URLs may have IP address instead of domain name. If an IP address is used as an alternative of the domain name in the URL, we can be sure that someone is trying to steal personal information with this URL. If the domain part of URL has IP address, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).
* prefixSuffix :- Checking the presence of ’-’ in the domain

part of URL. The dash symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separated by (-) to the domain name so that users feel that they are dealing with a legitimate webpage. If, the URL has ‘-’ symbol in the domain part of the URL, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

*Numerical Features*

Numerical value-based features are features that represent quantitative or continuous values. These features can take on a wide range of numeric values and are often used in various machine learning algorithms and statistical analyses.

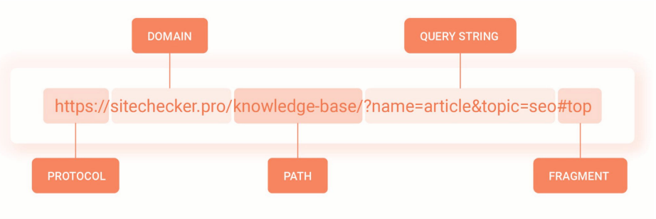


Fig. 5: Basic structure of a URL

Features of a URL are basically classified into - protocol, domain, path, query, fragment. The length of this features and the count of the different special characters in that specific feature are extracted. The special characters like ‘.’ ‘-’ ‘/’ ‘?’ ‘=’ ‘@’ ‘&’ ‘!’ ‘~’ ‘ ’ ‘,’ ‘+’ ‘\*’ ‘#’ ‘$’ ‘%’ .

These feature are : • url length, • qty dot url,

* qty hyphen url, • qty slash url, • qty questionmark url,
* qty equal url, • qty at url, • qty and url,
* qty exclamation url, • qty space url, • qty tilde url,
* qty comma url, • qty plus url, • qty asterisk url,
* qty hashtag url, • qty dollar url, • qty percent url,
* domain length, • qty dot domain, • qty hyphen domain,
* path length, • qty dot path, • qty hyphen path,
* qty slash path, • qty equal path • qty at path,
* qty and path, • qty exclamation path, • qty space path,
* qty tilde path, • qty comma path, • qty plus path,
* qty asterisk path, • qty dollar path, • qty percent path,
* query length, • qty dot query, • qty hyphen query,
* qty slash query, • qty questionmark query,
* qty equal query, • qty at query, • qty and query,
* qty exclamation query, • qty space query,
* qty tilde query, • qty comma query, • qty plus query,
* qty asterisk query, • qty dollar query,
* qty percent query, • fragment length, • qty dot fragment,
* qty hyphen fragment, • qty slash fragment,
* qty questionmark fragment, • qty equal fragment,
* qty and fragment, • qty exclamation fragment,
* qty space fragment, • qty comma fragment,
* qty asterisk fragment, • qty hashtag fragment,
* qty dollar fragment, • qty percent fragment.

Finally, a total of 74 features are extracted in this project. After extracting both the Lexical and Numerical Features, they are saved to a new csv file for the Model Evaluation.

1. *MODEL COMPARISON*

*Pycaret:* PyCaret is a Python library that provides an easy- to-use interface for training and comparing multiple machine learning models. It offers a variety of functions and tools to streamline the model development process and make it efficient. We use this library to compare the different machine learning models by using the extracted features and it will return a table of model performance metrics sorted by specified evaluation metrics - accuracy, AUC, recall, precision, F1, Kappa and MCC.

15 different algorithms were compared and Logistic Regression (94.45%) gave more accuracy among them. Thus, Logistic Regression model is used for testing and training the features.

1. *MODEL EVALUATION*

Model evaluation is a crucial step in machine learning to assess the performance and effectiveness of a trained model. It involves quantitatively measuring how well the model generalizes to new, unseen data and how accurately it predicts the target variable. Based on the model comparison, we evaluated the Logistic Regression (LR) model in this project.

*Logistic Regression*

Logistic Regression is a statistical modeling technique used for binary classification problems, where the goal is to predict the probability of an event or the likelihood of an outcome falling into one of two classes. Despite its name, Logistic Regression is a classification algorithm rather than a regression algorithm. It is based on the assumption that the relationship between the input variables (also known as independent or predictor variables) and the log-odds of the binary outcome (dependent or response variable) can be approximated by a linear relationship.

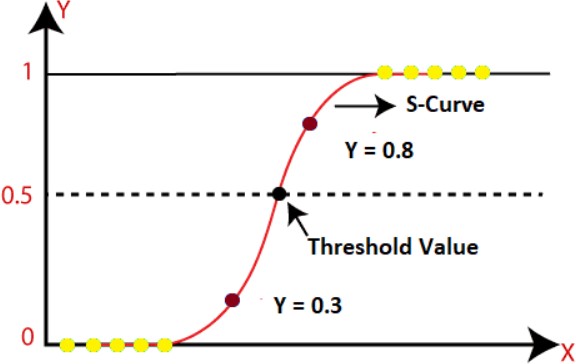


Fig. 6: Logistic Regression

The logistic function, also called the sigmoid function, is used to map the linear combination of input variables to a range between 0 and 1, representing the probability of belonging to the positive class. The logistic function is given by:

1

variables.

Evaluating the LR model includes the following steps:

* Splitting the dataset: Dividing dataset into training and testing sets, ensuring that both sets have a representative distribution of the target variable.
* Training the LR model: Fitting the Logistic Regression model to the training data using a suitable library such as scikit-learn.
* Make predictions: Use the trained model to predict the target variable for the test dataset.
* Calculating evaluation metrics: Comparing the predicted values with the actual values from the test dataset and calculating the evaluation metrics such as accuracy, precision, recall, F1-score, AUC-ROC, and confusion matrix. Libraries like scikit-learn provide functions to calculate these metrics.

1. *MODEL IMPLEMENTATION*

This phase involves deploying and integrating the trained machine learning model into a production environment where it can be used to make predictions on new, unseen data. This includes the following processes:

* Exporting the Model: Saving the trained model in a serialized format that can be easily loaded and used for predictions. We will be using python pickle module for exporting the trained LR model, which is a built-in module that provides a way to serialize and deserialize Python objects.
* Model Integration: Integrating the model into the target application which is the WhalingGuard Web Application where it will be used for predictions by API call.
* Model Loading: Loading the serialized model into the implementation environment, making it ready for use. Again, the pickle module is used to deserialize the model.
* Input Data Feeding: Providing the new data, which is the URL to the model for prediction. This is done by passing the input data through an API call from the WhalingGuard User-Interface.
* Feature Extraction: Extract the 74 different features from the input URL that where extracted during training phase. And saving that features to a new file for predicting.
* Prediction Generation: Utilize the loaded model to generate predictions on the extracted feature. Fitting the loaded model to the extracted features.
* Output Delivery: The prediction result is then passed to the API and the API then sends the results as a response to the frontend. The prediction result is also saved in csv file in the server, continuously for each API calls. Then, at the user interface, the users can view the results whether the entered URL is phishing or non-phishing.

1. RESULTS AND DISCUSSION

*p*(*x*) =

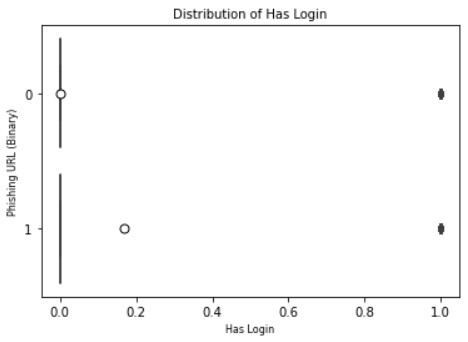
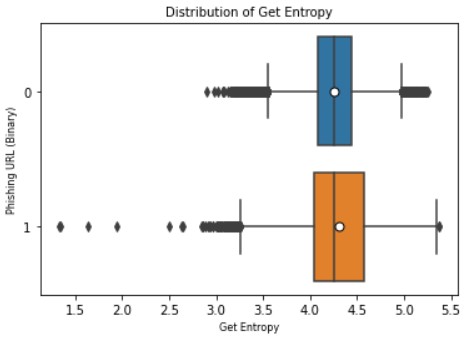
1 + *e−z*

* 1. *Feature Analysis*

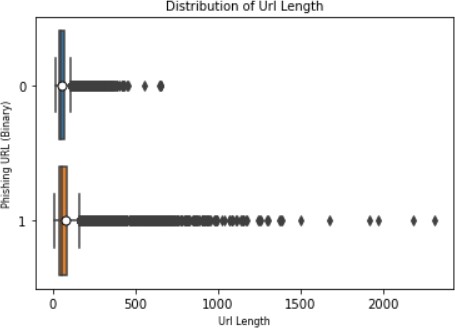
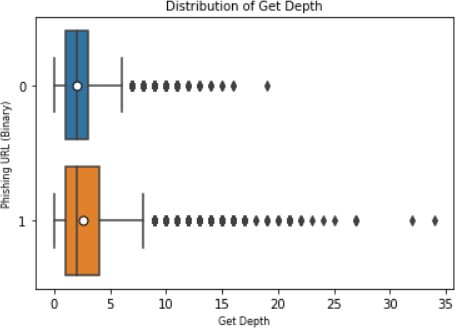
where p(x) is the predicted probability, x is the input variables, and z is the linear combination of the input

*Boxplot Graph:* The boxplot graph is a useful visualization for displaying the distribution of a numerical feature or

variable. It is employed to explore and understand the distribution of feature values during the analysis of the dataset. By, creating the boxplot graph, we could gain insights into the distribution of feature values, including measures of central tendency, dispersion, mean and identification of outliers. Visualization of some features are shown below:



(a) getEntropy (b) hasLogin



(c) getDepth (d) URLLength

Fig. 7: Visualization of URL features

Visually, we see that legitimate URLs have higher entropy and are generally longer than phishing URLs. Similarly, we can capture the fact that ‘red flag’ keywords appear in a URL string which is the keyword ‘login’, is more seen in the phishing URLs. The depth of the URL i.e., number of hierarchical levels or directories in the URL’s path for phishing URLs is seen greater than the legitimate ones. It is also seen that the phishing URLs have higher URL length than the legitimate URLs.

* 1. *Pycaret Analysis*

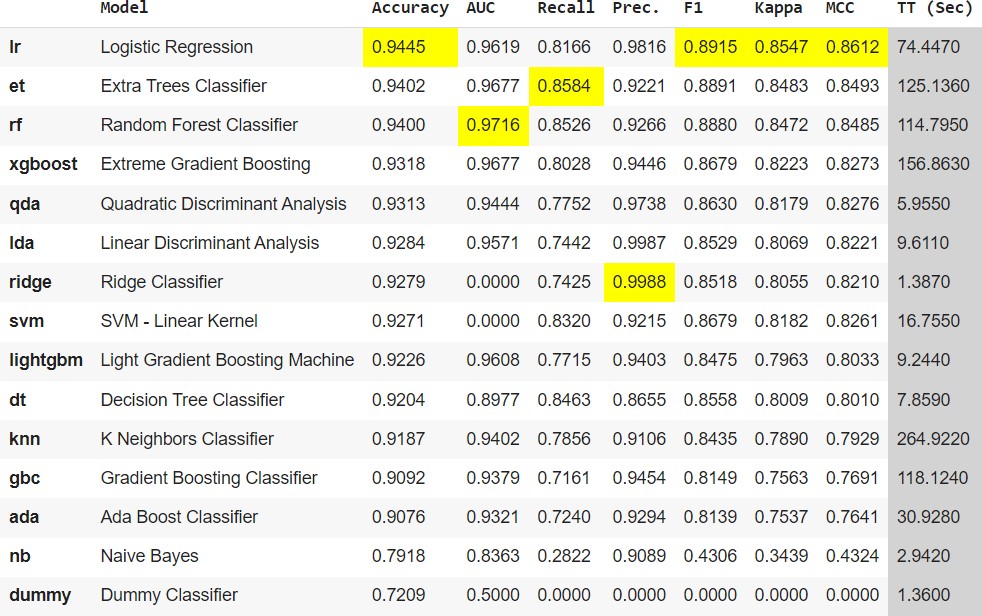


Fig. 8: Pycaret Comparison

When using PyCaret for model comparison, the results obtained can provide valuable insights into the performance of different machine learning algorithms on the given dataset. The comparison results obtained using PyCaret can guide in selecting the most suitable machine learning model(s) for this project, taking into account performance metrics, statistical significance, feature importance, and other relevant factors. Fig:8 shows the comparison result.

* 1. *Model Analysis*

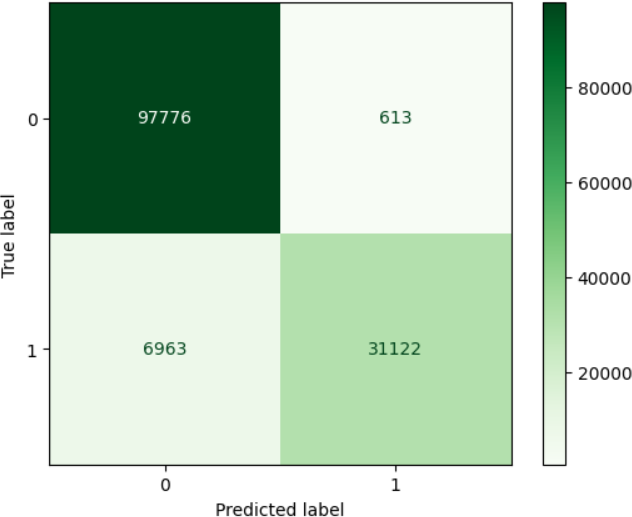


Fig. 9: Confusion Matrix

A confusion matrix is a table that is used to evaluate the performance of a classification model. It shows the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) by comparing the predicted labels with the true labels of a dataset. The confusion matrix provides a more detailed understanding of the model’s performance beyond simple accuracy. From the confusion matrix, some of the evaluation metrics calculated in our project are shown in fig:10

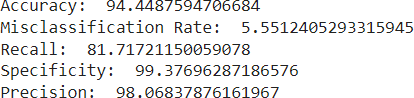


Fig. 10: Evaluation metrics

Based on the Logistic Regression Model, it has assigned feature importance score to each extracted feature. It helps to provide insights into which features have the most influence on the model’s predictions. Fig:11 shows the importance of some of the features in our project in the descending order.

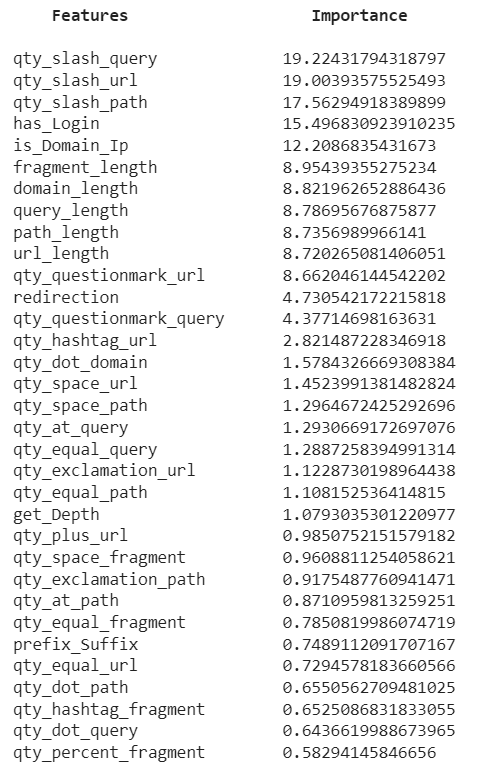
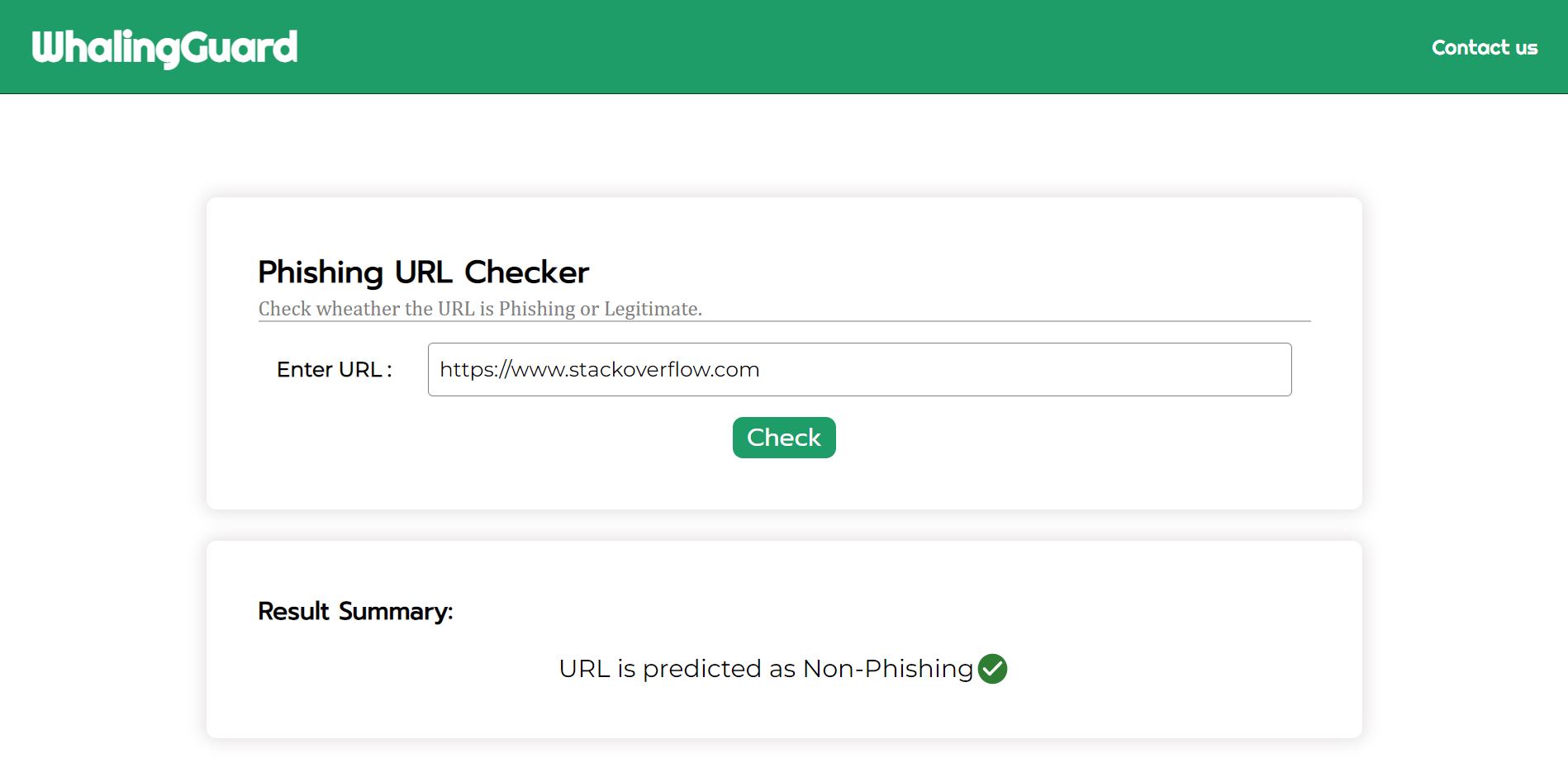
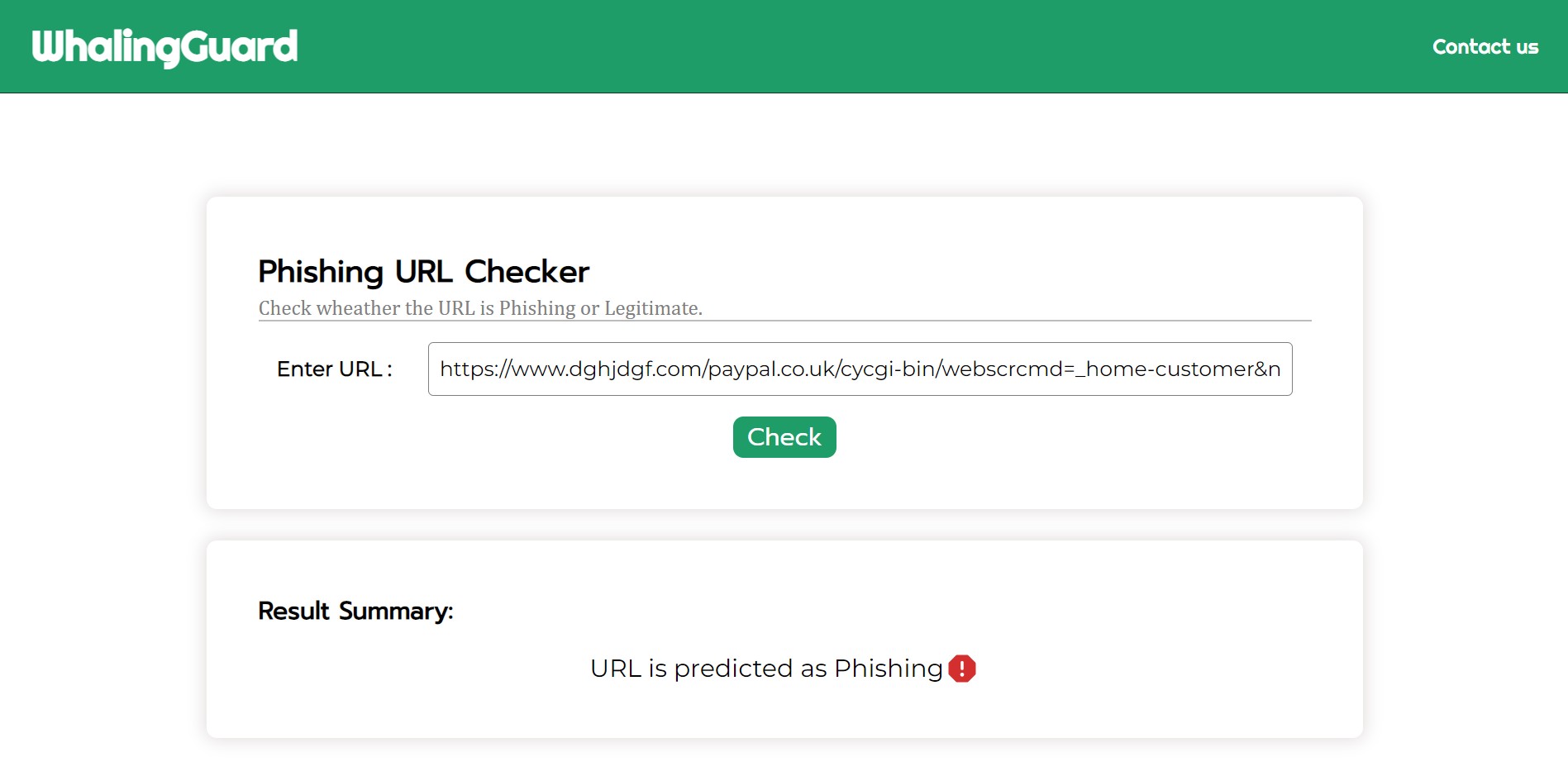
 

Fig. 11: Features and their importance

* 1. *Web Application*

We have developed a web application where the users can input URLs to check if it is phishing or non-phishing. UI of the application is developed using Reactjs and the API setup for the application is implemented using Nodejs. Fig:12 shows the User Interface of the WhalingGuard web application.

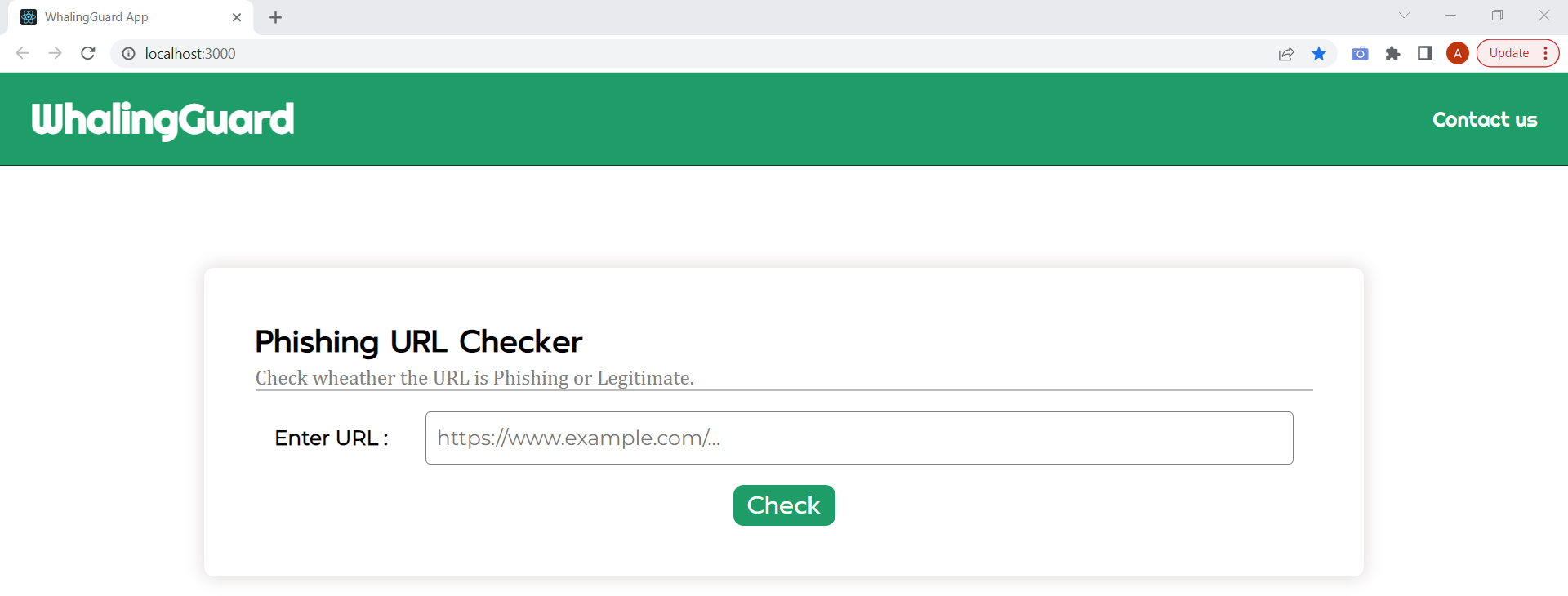


Fig. 12: User Interface

The URL entered by the user is received by the server and it is predicted by the evaluated LR model. Based, on the prediction, the summary of the result is shown as output in the user-interface. The following figures shows the 2 different output produced.

Fig. 13: Prediction Results

1. CONCLUSION

In particular, phishing has become more common and has begun to raise significant issues. There needs to design phishing detection method to affectively detect if the website is phishing or non-phishing. Considering the significance of phishing detection, in this study, we extracted 74 different features from the website URL. After comparing different ma- chine learning algorithms, we found that Logistic Regression gives higher accuracy. So, we used Logistic Regression model for phishing prediction. In future we are planning to design a framework for identifying and preventing phishing attacks that are delivered through email messages.

References

1. Ms. Sophiya Shikalgar, Dr. S. D. Sawarkar, Mrs. Swati Narwane, “Detection of URL based Phishing attacks Using Machine Learning,” 2018.
2. Yi Wei, Yuji Sekiya, “Sufficiency of Ensemble Machine Learning Methods for Phishing Websites Detection,” 2021.
3. Arkan Hammoodi Hasan Kabla, Mohammed Anbar, Selvakumar Man- ickam, Shankar Karupayah, “Eth-PSD: A Machine Learning-Based Phishing Scam Detection Approach in Ethereum,” 2021.
4. Erzhou Zhu, Yuyang Chen, Chengcheng Ye, Xuejun Li, Feng Liu, “OFS-NN: An Effective Phishing Websites Detection Model Based on Optimal Feature Selection and Neural Network,” 2019.
5. Paulius Vaitkevicius, “Comparison of Classification Algorithms for Detection of Phishing Websites,” 2020.
6. Sindhu, Sunil Parameshwar Patil, Arya Sreevalsan, Faiz Rahman, “Phishing Detection using Random Forest, SVM and Neural Network with Backpropagation,” 2020.
7. Ying Xue, Yang Li, Yuangang Yao, Xianghui Zhao, Jianyi Liu,Ru Zhang, “Phishing sites detection based on URL Correlation,” 2016.
8. Ilker Kara, Murathan Ok, Ahmet Ozaday, “Characteristics of Under- standing URLs and Domain Names Features: The Detection of Phishing Websites with Machine Learning Methods,” 2022.
9. Atharva Deshpande, Omkar Pedamkar, Nachiket Chaudhary, “Detection of Phishing Websites using Machine Learning,” 2021.
10. Ashit Kumar Dutta, Zhihan Lv, “Detecting phishing websites using machine learning technique,” 2018.