# WHALING-GUARD : PHISHING DETECTION USING MACHINE LEARNING

A PROJECT REPORT
Submitted by

**ADITHYAN M S       PRP19CS006**

Under the guidance of
Mrs. KRISHNAPRIYA V J

to

the APJ Abdul Kalam Technological University in partial fulfillment of the
requirements for the award of the Degree

of

Bachelor of Technology
In
*Computer Science and Engineering*



**Department of Computer Science and Engineering**
**COLLEGE OF ENGINEERING & MANAGEMENT PUNNAPRA**
**PUNNAPRA, ALAPPUZHA**
**JUNE 2023**

# DECLARATION

I hereby declare that the project report "WHALING-GUARD : PHISHING DETECTION USING MACHINE LEARNING " , submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of Mrs. KRISHNAPRIYA V J. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Punnapra                                              ADITHYAN M S

Date: 16-06-2023

# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

# COLLEGE OF ENGINEERING AND MANAGEMENT PUNNAPRA



## CERTIFICATE

This is to certify that the project report entitled "**WHALING-GUARD : PHISHING DETECTION USING MACHINE LEARNING**" submitted by : **ADITHYAN M S (KTU ID : PRP19CS006)** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering is a bonafide record of the project work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Mrs. KRISHNAPRIYA V J          Mrs. SMITHA M JASMINE          Mrs. NEETHU SATHYAN M
Asst. Professor                Asst. Professor                Asst. Professor
Dept. of IT                    Dept. of CSE                   Dept. of CSE
(Project Guide)                (Project Co-Ordinator)         (HOD)

# CONTENTS

# ACKNOWLEDGEMENT

# ABSTRACT

Phishing is a type of cyberattack that aims to steal sensitive information such as usernames, passwords, and credit card details. Phishing attacks have become increasingly sophisticated and prevalent in today's digital world. Thus, The development of the phishing detection model is essential to combat the increasing threat of phishing attacks, protect users from fraud and data breaches, and enhance overall cybersecurity in the digital landscape. Several 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.

# LIST OF TABLES

# LIST OF FIGURES

# ABBREVIATIONS

URL            Uniform Resource Locator

LR            Logistic Regression

CM            Confusion Matrix

API            Application Programming Interface

# Chapter 1

# INTRODUCTION

Phishing detection is a crucial aspect of cybersecurity aimed at identifying and preventing phishing attacks. Phishing refers to a malicious practice where cyber-criminals 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 learn-

ing that can detect whether a website URL relates to phishing or not. • We have compared 15 different machine learning algorithms for the development of this phishing detection model. • We extracted 74 different features from about 5.5 lakh URLs. And these features where used for training and testing the model. • For comparing the models, we use python-pycaret library and found that Logistic Regression provides better accuracy than any other machine learning algorithms, thus the model is evaluated using Logistic Regression Algorithm. • 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.

## 1.1    Problem Definition

Phishing has a list of negative effects on a Business, including loss of money, loss of intellectual property, damage to reputation, and disruption of operational activities. An attack is disguised as a message from a legitimate company. Phishing is facilitated by communicating a sense of urgency in the message, which could threaten account suspension, money loss or loss of the targeted users job. According to the FBI, phishing incidents nearly doubled in frequency, from 114,702 incidents in 2019, to 241,324 incidents in 2020. Therefore, we suggest a phishing detection model based on machine learning that compares the features of the target websites mainly the URLs.

# Chapter 2

# LITERATURE SURVEY

PhishHaven—An Efficient Real-Time AI Phishing URLs Detection System [1] 2020 : Design a PhishHaven which detects and classifies a URL using three subcomponents. First subcomponent, URL Hit The second subcomponent is Features Extractor. The third subcomponent is Modelics.In this new paradigm executes ensemble-based machine learning models in parallel using multi-threading technique, and results in real-time detection by significant speed-up in the classification process.

Sufficiency of Ensemble Machine Learning Methods for Phishing Websites Detection [2] 2021 : Phishing instances are usually derived from PhishTank Other legitimate instances are from Alexa, DMOZ, and Common Crawl. Features used in phishing detection are usually extracted from URLs (protocol, domain, path, parameters) This feature selection framework achieves a remarkable 87.6% reduction in feature quantity with suffering from only a 0.1% deterioration in detecting accuracy, making it possible for up-date training and real-time detecting in a production environment.

Eth-PSD: A Machine Learning-Based Phishing Scam Detection Approach in Ethereum [3] 2021 : Detect phishing scam-related transactions using a novel machine learning-based approach. Eth-PSD tackles some of the limitations in the existing works, such as the use of imbalanced datasets, complex feature engineering, and lower detection accuracy. Proposed Eth-PSD to detect the phishing scam in Ethereum. Started with derived requirements based on the limitations of related works and other effective IDSs from previous related works.

A Systematic Literature Review on Phishing Email Detection Using Natural Language Processing Techniques [3] 2022 : The use of Natural Language Processing (NLP) techniques for detection of phishing except one that shed light on the use of NLP techniques for classification and training purposes, while exploring a

few alternatives. In this research, journal, conference, and workshop papers were carefully analysed, published between 2006 and 2022, with different techniques to investigate the trend of phishing email detection.

OFS-NN: An Effective Phishing Websites Detection Model Based on Optimal Feature Selection and Neural Network [4] 2019 : In the proposed OFS-NN, a new index, feature validity value (FVV), is first introduced to evaluate the impact of sensitive features on the phishing websites detection. Then, based on the new FVV index, an algorithm is designed to select the optimal features from the phishing websites. This algorithm could properly deal with problems of big number of phishing sensitive features and the continuous changes of features. Consequently, it can mitigate the over-fitting problem of the neural network classifier.

Comparison of Classification Algorithms for Detection of Phishing Websites [5] 2020 : Compare classic supervised machine learning algorithms on all publicly available phishing datasets with predefined features and to distinguish the best performing algorithm for solving the problem of phishing websites detection, regardless of a specific dataset design.

Phishing Detection using Random Forest, SVM and Neural Network with Backpropagation [6] 2020 : The paper explains the improved Random Forest classification method, SVM classification algorithm and Neural Network with backpropagation classification methods which have been implemented with accuracies of 97.369%, 97.451% and 97.259% respectively.

Intelligent rule-based phishing websites classification [7] 2014 : The authors shed light on the important features that distinguish phishing websites from legitimate ones and assess how good rule-based data mining classification techniques are in predicting phishing websites and which classification technique is proven to be more reliable.The features extracted automatically without any intervention from the users using computerised developed tools.

Phishing sites detection based on Url Correlation. [8] 2016 : Proposed Vulnerable Sites List and a new feature which is named URL Correlation. URL Correlation is based on the similarity of URLs with the List above that we created. A large improvement of accuracy is observed by comparing methods which use our new feature with the others which use the normal one.

Characteristics of Understanding URLs and Domain Names Features: The Detection of Phishing Websites With Machine Learning Methods [9] 2019 :The proposed method simplifies the process of feature extraction, and reduces processing overhead while going beyond analyzing on HTML, DOM, and URL based features by considering URLs, and domain names. A minimum loss in data con-

version, selecting the appropriate machine learning technique, and consistency of definitions in the data set.

Detecting phishing websites using machine learning technique [10] 2018 : The proposed framework employs RNN—LSTM to identify the properties Pm and Pl in an order to declare an URL as malicious or legitimate. The proposed method (LURL) is developed in Python 3.0 with the support of Sci—Kit Learn and NUMPY packages. Also, the existing URL detectors are constructed for evaluating the performance of LURL. LURL has produced an average of 97.4% and 96.8% for Phishtank and Crawler datasets respectively.

Web Phishing Detection Using Machine Learning [11] 2022 : Suggest a literacy-based strategy to categorize Web sites into three categories: benign, spam, and malicious. Our technology merely examines the Uniform Resource Locator (URL) itself, not the content of Web pages. As a result, it removes run-time stillness and the risk of drug users being exposed to cyber surfer-based vulnerabilities. When compared to a blacklisting service, our approach performs better on generality and content since it uses learning techniques.

Detection of Phishing Websites using Machine Learning [12] 2021 : Collect unstructured data of URLs from Phishtank website, Kaggle website and Alexa website, etc. Train the three unique classifiers and analyse their presentation based on exactness two classifiers utilized are Decision Tree and Random Forest algorithm. Each classifier is trained using training set and testing set is used to evaluate performance of classifiers. Performance of classifiers has been evaluated by calculating classifiers accuracy score.

A New Method for Detection of Phishing Websites: URL Detection [13] 2018 : Random forest algorithm is implemented using Rstudio. The parsed dataset undergoes heuristic classification where the dataset is spilt into 70% and 30%. The 70% data is considered for training and 30% for testing. In this paper, a different methodology has been proposed to detect phishing websites by using random forests as the classification algorithm with the help of Rstudio.

Detection of URL based Phishing attacks Using Machine Learning [14] 2018 : Hybrid Algorithm Approach is a mixture of different classifiers working together which gives good prediction rate and improves the accuracy of the system. This system provides us with 85.5 % of accuracy for XG Boost Classifier, 86.3% accuracy for SVM Classifier, 80.2 % accuracy for NaÃ¯ve Bayes Classifier and finally 85.6 percentage of accuracy when using Stacking Classifier.

Large-Scale Automatic Classification of Phishing Pages [15] 2010 : We describe the design and performance characteristics of a scalable machine learning

classifier we developed to detect phishing websites. We use this classifier to maintain Googles phishing blacklist automatically. Despite the noise in the training data, our classifier learns a robust model for identifying phishing pages which correctly classifies more than 90% of phishing pages several weeks after training concludes.

Table 2.1: **Comparison between Related Works**

| Sl No | Name of Paper | Paper type | Year | Description |
|---|---|---|---|---|
| 1 | PhishHaven—An Efficient Real-Time AI Phishing URLs Detection System | IEEE paper | 2020 | In this new paradigm executes ensemble-based machine learning models in parallel using multi-threading tech- nique, and results in real-time detection by significant speed-up in the classification process. |
| 2 | Sufficiency of Ensemble Machine Learning Methods for Phishing Websites Detection | IEEE paper | 2021 | This feature selection framework achieves a remarkable 87.6% reduction in feature quantity with suffering from only a 0.1% deterioration in detecting accuracy. |
| 3 | Eth-PSD: A Machine Learning-Based Phishing Scam Detection Approach in Ethereum | IEEE paper | 2021 | The project started with derived requirements based on the limitations of related works and other effective IDSs from previous related works. |
| 4 | OFS-NN: An Effective Phishing Websites Detection Model Based on Optimal Feature Selection and Neural Network | IEEE paper | 2019 | In the proposed OFS-NN, a new index, feature validity value (FVV), is first introduced to evaluate the impact of sensitive features on the phishing websites detection |

| 5 | Comparison of Classification Algorithms for Detection of Phishing Websites | IEEE paper | 2020 | The comparison results are presented in this paper, showing ensembles and neural networks outperforming other classical algorithms. |
|---|---|---|---|---|
| 6 | Phishing Detection using Random Forest, SVM and Neural Network with Backpropagation | IEEE paper | 2020 | This paper explains the existing machine learning methods that are used to detect phishing websites. |
| 7 | Intelligent rule-based phishing websites classification | IET paper | 2014 | The features extracted automatically without any intervention from the users using computerised developed tools. |
| 8 | Phishing sites detection based on Url Correlation | IEEE paper | 2016 | A large improvement of accuracy is observed by comparing methods which use our new feature with the others which use the normal one. |
| 9 | Characteristics of Understanding URLs and Domain Names Features: The Detection of Phishing Websites With Machine Learning Methods | IEEE paper | 2022 | A minimum loss in data conversion, selecting the appropriate machine learning technique, and consistency of definitions in the data set. |
| 10 | Detecting phishing websites using machine learning technique | IEEE paper | 2018 | The existing URL detectors are constructed for evaluating the performance of LURL. LURL has produced an average of 97.4% and 96.8% for Phishtank and Crawler datasets respectively. |
| 11 | Web Phishing Detection Using Machine Learning | IJITEE paper | 2022 | Each classification is performed using a training set, and the performance of the classifiers is evaluated using a testing set. The accuracy score of classifiers was calculated to assess their performance. |

| 12 | Detection of Phishing Websites using Machine Learning | IJERT paper | 2021 | Train the three unique classifiers and analyse their presentation based on exactness two classifiers utilized are Decision Tree and Random Forest algorithm. |
| 13 | A New Method for Detection of Phishing Websites: URL Detection | IEEE paper | 2018 | In this paper, a different methodology has been proposed to detect phishing websites by using random forests as the classification algorithm with the help of Rstudio. |
| 14 | Detection of URL based Phishing attacks Using Machine Learning | IJERT paper | 2018 | This system provides us with 85.5 % of accuracy for XG Boost Classifier, 86.3% accuracy for SVM Classifier, 80.2 % accuracy for NaÃ¯ve Bayes Classifier and finally 85.6 percentage of accuracy when using Stacking Classifier. |
| 15 | Large-Scale Automatic Classification of Phishing Pages | IEEE paper | 2010 | Despite the noise in the training data, our classifier learns a robust model for identifying phishing pages which correctly classifies more than 90% of phishing pages several weeks after training concludes. |

# Chapter 3

# METHODOLOGY

## 3.1 System Architecture



Figure 3.1: System Architecture

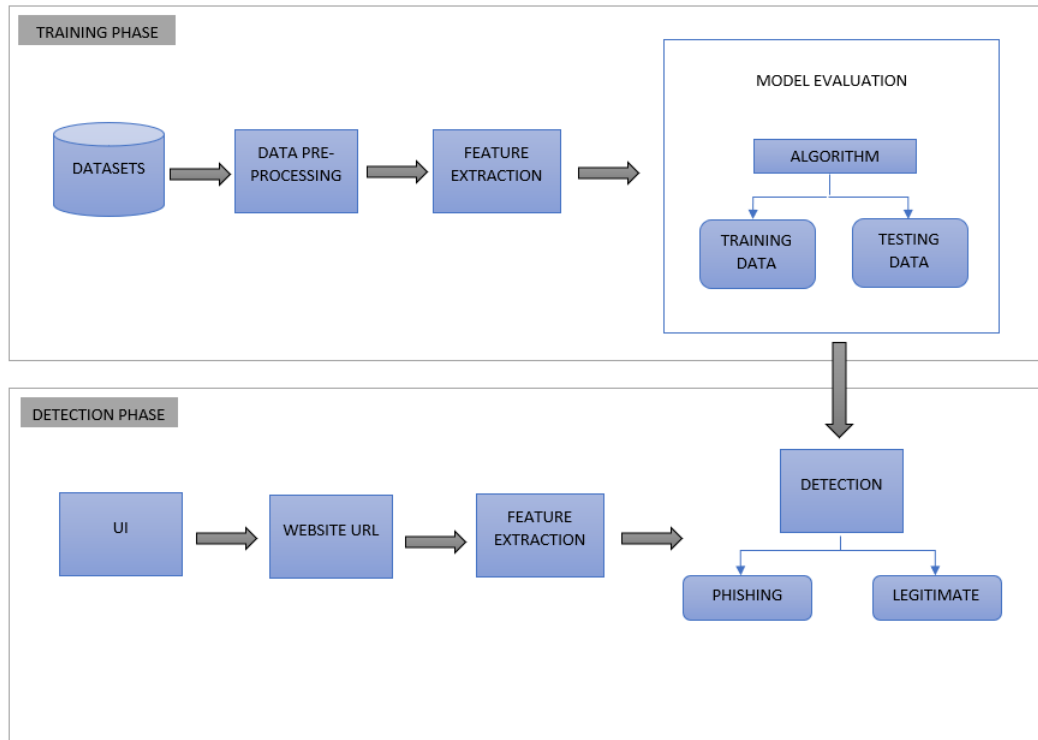In this section, we describe the phishing detection framework which consists of two major parts as shown in Fig:3.1. The first part is the Training Phase and the second part is the Detection Phase. In the first phase, datasets containing urls are prepared for the machine learning algorithm and model 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.

## 3.2 Training Phase

### 3.2.1 Dataset

A dataset is a collection of structured or unstructured data that is organized and labeled for specific tasks. Datasets serve as the foundation for training and evaluating machine learning models, conducting statistical analysis, and extracting meaningful insights from data.

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.

2. Dataset-II

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

|   | domain | label |
|---|---|---|
| 0 | nobell.it/70ffb52d079109dca5664cce6f317373782/... | 1.0 |
| 1 | www.dghjdgf.com/paypal.co.uk/cycgi-bin/webscrc... | 1.0 |
| 2 | serviciosbys.com/paypal.cgi.bin.get-into.herf.... | 1.0 |
| 3 | mail.printakid.com/www.online.americanexpress.... | 1.0 |
| 4 | thewhiskeydregs.com/wp-content/themes/widescre... | 1.0 |

Figure 3.2: Dataset-I from phishtank.com

| | Unnamed: 0 | url | label | result |
|---|---|---|---|---|
| 0 | 0 | https://www.google.com | benign | 0 |
| 1 | 1 | https://www.youtube.com | benign | 0 |
| 2 | 2 | https://www.facebook.com | benign | 0 |
| 3 | 3 | https://www.baidu.com | benign | 0 |
| 4 | 4 | https://www.wikipedia.org | benign | 0 |

Figure 3.3: Dataset-II from kaggle.com

### 3.2.2    Data Pre-Processing

Data preprocessing is a crucial step in machine learning that involves transforming raw data into a suitable format for training machine learning models. It helps to improve the quality and reliability of the data, reduces noise and inconsistencies, and enhances the performance of the models.Here are some data preprocessing techniques used in our project :

1. Data Cleaning:  This step involves handling missing values, outliers, and duplicates in the data.  Missing values can be imputed using techniques like mean, median, or mode.  Outliers can be detected and treated using statistical methods or replaced with more reasonable values. Duplicates can be removed to avoid bias in the analysis.

2. Data Reduction: Data reduction in preprocessing refers to the process of reducing the size or complexity of a dataset without losing critical information. It is often performed as an initial step in data preprocessing to address challenges such as computational limitations, storage constraints, or noise in the data. Data reduction techniques aim to retain the most relevant information while reducing redundancy and removing irrelevant or noisy data points.

3. Data Integration: Data integration in preprocessing refers to the process of combining multiple heterogeneous data sources into a unified and consistent format. It involves merging, transforming, and resolving inconsistencies in the data from different sources to create a comprehensive dataset for further analysis or modeling. Data integration is a critical step in data preprocessing when working with data from various systems, databases, or file formats.

### 3.2.3    Feature Extraction

Feature extraction in machine learning refers to the process of transforming raw input data into a set of meaningful and representative features that can be used as inputs for machine learning algorithms.  The goal of feature extraction is to capture relevant information from the data and create a compact and informative representation that facilitates the learning process and improves the performance of the models.

Statistical measures can be computed from the raw data to capture important characteristics.  These measures can include mean, median, standard deviation,

variance, skewness, kurtosis, and other moments. Statistical measures provide insights into the distribution, central tendency, and spread of the data. Exploratory data analysis, domain expertise, and experimentation are often necessary to determine the most effective feature extraction methods for a given problem.

In this project, the features are classified into 2 categories:

- Lexical Features

- Numerical Features

### 3.2.4   Algorithm

**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.

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:

$p(x) = 1 / (1 + e\hat{}(-z))$

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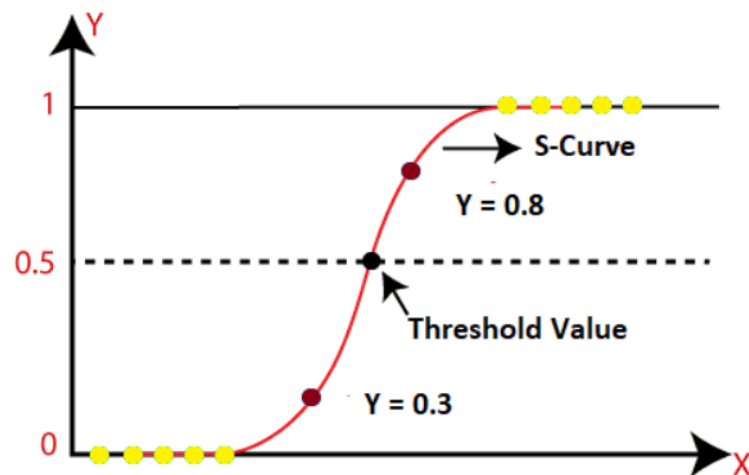


Figure 3.4: Logistic Regression

## 3.3    Detection Phase

### 3.3.1    User Interface

A user interface (UI) refers to the visual and interactive elements through which users interact with a software application or system. It provides a means for users to input commands or data and receive feedback or output from the system. The primary goal of a user interface is to create a user-friendly and intuitive experience, allowing users to efficiently and effectively interact with the software.

**Reactjs**

ReactJS, also known as React, is a popular JavaScript library for building user interfaces. It was developed by Facebook and is widely used for creating interactive and dynamic web applications. React follows a component-based architecture, where the UI is broken down into reusable and modular components, making it easier to develop and maintain complex user interfaces.

**Nodejs**

Node.js is an open-source JavaScript runtime environment built on Chrome's V8 JavaScript engine. It allows developers to run JavaScript code outside the browser, on the server-side, and build scalable and high-performance web applications. Node.js provides an event-driven, non-blocking I/O model that makes it lightweight and efficient, suitable for real-time applications and server-side programming.

Node.js can be used to build APIs (Application Programming Interfaces) for server-side development. APIs are used to define the endpoints and functionality of web services, allowing client applications to interact with the server and exchange data.

### 3.3.2    Website URL

A URL (Uniform Resource Locator) is a specific address that identifies a resource on the internet. It typically consists of several components, including the protocol, domain name, path, and possibly query parameters.
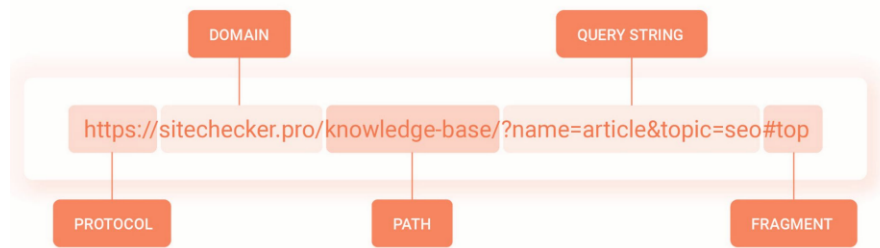
Figure 3.5: Basic structure of a URL

### 3.3.3 Detection

The input URL from the user interface is predicted either as Phishing or Non-Phishing.

- Phishing: Type of URLs that are specifically designed to deceive users by impersonating legitimate websites or services.

- Non-Phishing Type of URLs that are the genuine URLs of legitimate websites or services.

## 3.4 Objectives

- To extract features that can produce effective accuracy in model evaluation

- To analyse the accuracy level for different machine learning algorithms and implementing the best among them

- To design and Implement a Web Application to search and detect whether it is phishing or not.

- To store URLs, which are detected as either phishing or non-phishing

## 3.5 Scope

- Algorithm will analyze various blacklisted and legitimate URL features to accurately detect the phishing websites including zero-hour phishing websites.

# 3.6    System Requirements

## 3.6.1    Software Requirements

**Programming Languages:** Python, JavaScript

**Integrated Development Environment (IDE):**VisualStudioCode, Google Colab

**Libraries:** numpy, panda, scikit-learn, Matplotlib/Seaborn, pycaret, pickle, React

**Frameworks:** Anaconda, Express.js

**Web Browsers:** Google Chrome, Mozilla Firefox,Microsoft Edge

**Package Managers:** pip, npm

**Version Control:** Git, Github

## 3.6.2    Hardware Requirements

**Processor:** Intel i5 and above

**RAM:** 8gb and above

# Chapter 4

# SYSTEM IMPLEMENTATION

The implementation is divided into five parts.

1. Data Pre-processing

2. Feature Extraction

3. Model Comparison

4. Model Evaluation

5. Model Implementation

## 4.1  DATA PRE-PROCESSING

Data pre-processing is an essential step in preparing raw data for analysis and machine learning tasks. It involves transforming 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.

**Data Cleaning**

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 this datasets.

**Data Reduction**

We only need the urls and there corresponding label values(1 0r 2 representing phishing or legitimate) as the input datas, the Dataset-II contains unwanted columns so they are also dropped.

**Data Integration**

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 DataFrames with different data types for the same column name might lead to inconsistencies and unexpected behavior and merging DataFrames having different column names produce splitting of the values. So they both are handled before merging. After Merging we obtained a dataset having 546,104 datas 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 4.2 shows the Bar-Graph Classification of Phishing and Legitimate urls inside the new dataset:
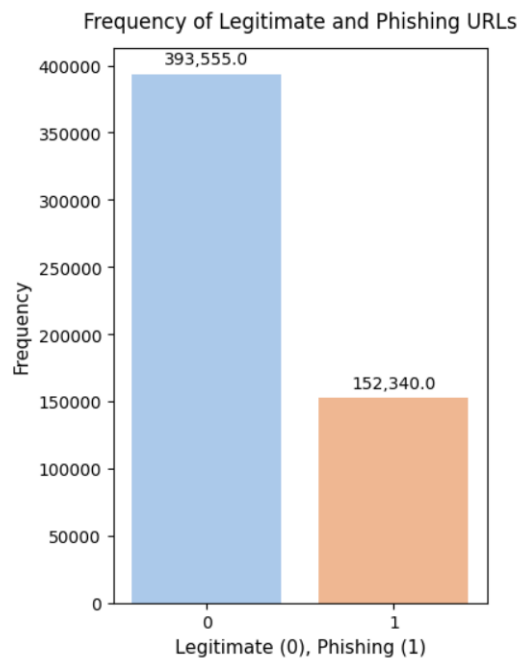


Figure 4.1: Bar-Graph Classification of Phishing and Legitimate data.

```python
import pandas as pd

df=pd.read_csv("../data/initial_urls.csv")
df2=pd.read_csv("../data/additional_urls.csv")

> def initial_read(df): ···

df.dropna(inplace=True)
df2.drop(columns = ['Unnamed: 0', 'label'], inplace=True)

df['label']=df['label'].astype(int)

df.rename(columns={"domain": "url", "label": "phishing"},inplace=True)
df2.rename(columns={"result": "phishing"},inplace=True)

df['url']= 'https://' + df['url'].astype(str)

df_final = pd.concat([df,df2])
df_final .drop_duplicates(inplace=True)

df_final.to_csv('../data/final_urls.csv', index=False)
print(df.columns)
print(df2.columns)
```

Figure 4.2: Data Pre-Processing

## 4.2 FEATURE EXTRACTION

Feature extraction is a technique used to reduce the dimensionality 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

### 4.2.1 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 $\geq= 54$ , 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 sub pages 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 webpage. 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).

```python
def getEntropy(url):
    url = url.lower()
    probs = [url.count(c) / len(url) for c in set(url)]
    entropy = -sum([p * log(p) / log(2.0) for p in probs])
    return entropy

def hasLogin(url):
    return int('login' in url.lower())

def redirection(url):
    pos = url.rfind('//')
    if pos > 6:
        if pos > 7:
            return 1
        else:
            return 0
    else:
        return 0

def lenClassify(url):
    if len(url) < 54:
        length = 0
    else:
        length = 1
    return length

def haveAtSign(url):
    if "@" in url:
        at = 1
    else:
        at = 0
    return at
```

```python
def getDepth(url):
    s = urlparse(url).path.split('/')
    depth = 0
    for j in range(len(s)):
        if len(s[j]) != 0:
            depth = depth+1
    return depth

def tinyURL(url):
    match=re.search(shortening_services,url)
    if match:
        return 1
    else:
        return 0

def isDomainIp(domain):
    domain = domain.split(':')
    pattern = r'^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$|^ \
               (?:[a-f0-9]{1,4}:){7}[a-f0-9]{1,4}$'
    match = re.match(pattern, domain[0])
    if match is not None:
        return 1
    else:
        return 0

def prefixSuffix(domain):
    if '-' in domain:
        return 1
    else:
        return 0
```

Figure 4.3: Lexical Feature Extraction

## 4.2.2 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.

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

These features are :

| | | |
|---|---|---|
| url_length | qty_hyphen_path | qty_space_query |
| qty_dot_url | qty_slash_path | qty_tilde_query |
| qty_hyphen_url | qty_equal_path | qty_comma_query |
| qty_slash_url | qty_at_path | qty_plus_query |
| qty_questionmark_url | qty_and_path | qty_asterisk_query |
| qty_equal_url | qty_exclamation_path | qty_dollar_query |
| qty_at_url | qty_space_path | qty_percent_query |
| qty_and_url | qty_tilde_path | fragment_length |
| qty_exclamation_url | qty_comma_path | qty_dot_fragment |
| qty_space_url | qty_plus_path | qty_hyphen_fragment |
| qty_tilde_url | qty_asterisk_path | qty_slash_fragment |
| qty_comma_url | qty_dollar_path | qty_questionmark_fragment |
| qty_plus_url | qty_percent_path | qty_equal_fragment |
| qty_asterisk_url | query_length | qty_and_fragment |
| qty_hashtag_url | qty_dot_query | qty_exclamation_fragment |
| qty_dollar_url | qty_hyphen_query | qty_space_fragment |
| qty_percent_url | qty_slash_query | qty_comma_fragment |
| domain_length | qty_questionmark_query | qty_asterisk_fragment |
| qty_dot_domain | qty_equal_query | qty_hashtag_fragment |
| qty_hyphen_domain | qty_at_query | qty_dollar_fragment |
| path_length | qty_and_query | qty_percent_fragment |
| qty_dot_path | qty_exclamation_query    . | |

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.

```
needed_cols = ['url', 'domain', 'path', 'query', 'fragment']
for col in needed_cols:
    df[f'{col}_length']=df[col].str.len()
    df[f'qty_dot_{col}'] = df[[col]].applymap(lambda x: str.count(x, '.'))
    df[f'qty_hyphen_{col}'] = df[[col]].applymap(lambda x: str.count(x, '-'))
    df[f'qty_slash_{col}'] = df[[col]].applymap(lambda x: str.count(x, '/'))
    df[f'qty_questionmark_{col}'] = df[[col]].applymap(lambda x: str.count(x, '?'))
    df[f'qty_equal_{col}'] = df[[col]].applymap(lambda x: str.count(x, '='))
    df[f'qty_at_{col}'] = df[[col]].applymap(lambda x: str.count(x, '@'))
    df[f'qty_and_{col}'] = df[[col]].applymap(lambda x: str.count(x, '&'))
    df[f'qty_exclamation_{col}'] = df[[col]].applymap(lambda x: str.count(x, '!'))
    df[f'qty_space_{col}'] = df[[col]].applymap(lambda x: str.count(x, ' '))
    df[f'qty_tilde_{col}'] = df[[col]].applymap(lambda x: str.count(x, '~'))
    df[f'qty_comma_{col}'] = df[[col]].applymap(lambda x: str.count(x, ','))
    df[f'qty_plus_{col}'] = df[[col]].applymap(lambda x: str.count(x, '+'))
    df[f'qty_asterisk_{col}'] = df[[col]].applymap(lambda x: str.count(x, '*'))
    df[f'qty_hashtag_{col}'] = df[[col]].applymap(lambda x: str.count(x, '#'))
    df[f'qty_dollar_{col}'] = df[[col]].applymap(lambda x: str.count(x, '$'))
    df[f'qty_percent_{col}'] = df[[col]].applymap(lambda x: str.count(x, '%'))

col_in_question = ['qty_slash_domain', 'qty_questionmark_domain','qty_equal_domain', 'qty_at_domain', 'qty_and_domain',
'qty_exclamation_domain', 'qty_space_domain', 'qty_tilde_domain','qty_comma_domain', 'qty_plus_domain',
'qty_asterisk_domain','qty_hashtag_domain', 'qty_dollar_domain', 'qty_percent_domain', 'qty_questionmark_path',
'qty_hashtag_path', 'qty_hashtag_query', 'qty_at_fragment','qty_tilde_fragment', 'qty_plus_fragment']

df.drop(columns = col_in_question, inplace=True)
```

Figure 4.4: Numerical Feature Extraction

## 4.3 MODEL COMPARISON

### 4.3.1 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 returns 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.

```
import pandas as pd
from pycaret.classification import *
import time

start_time = time.time()

df = pd.read_csv('/content/drive/MyDrive/ML-Phishing Detection Project/data/url_features.csv')
data = df.drop(columns=['url','protocol','domain','path','query','fragment'])

s = setup(data, target = 'phishing', session_id = 123)
best = compare_models()

print("\n--- pycaret check completed in %s seconds ---" % (time.time() - start_time))
```

Figure 4.5: Pycaret Comparison

## 4.4    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.

Evaluating a 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: Fiting 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.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import time
start_time = time.time()
import warnings
warnings.filterwarnings("ignore")

df = pd.read_csv("../data/url_features.csv")

x = df.drop(columns=['url','protocol','domain','path','query','fragment','phishing'])
y = df['phishing']
X_train, X_test, y_train, y_test = train_test_split(x,y,random_state = 42, stratify = y )

clf = LogisticRegression(penalty="l2",C=10,max_iter=1000, random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

acc = accuracy_score(y_test, y_pred)*100
train_accuracy = clf.score(X_train, y_train)*100
test_accuracy = clf.score(X_test, y_test)*100

print(f"LogisticRegression Accuracy: {acc}")
print("Training accuracy:", train_accuracy)
print("Test accuracy:", test_accuracy)

print("\n--- Model Evaluation ended in %s seconds ---" % (time.time() - start_time))
```

Figure 4.6: Model Evaluation

## 4.5   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 Appication 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 a 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 features. Fiting 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 results is also saved in csv file in the server, continously for each API calls.Then, at the user interface, the users can view the results whether the entered url is phishing or non-phishing.

```javascript
app.post('/api/url', (req, res) => {
    const data = req.body.url;
    try {
        parsedUrl = new URL(data);
        const dataStr = JSON.stringify(data);
        const pythonProcess = spawn('python', ['./predictor.py']);
        pythonProcess.stdin.write(dataStr);
        pythonProcess.stdin.end();
        let outputData = '';
        pythonProcess.stdout.on('data', (data) => {
            outputData += data;
        });
        pythonProcess.stderr.on('data', (data) => {
            res.status(400).json(error)
            console.error(`stderr: ${data}`);
        });
        pythonProcess.on('error', (error) => {
            res.status(400).json(error)
            console.error(`Python process error: ${error}`);
        });
        pythonProcess.on('close', (code) => {
        if (code !== 0) {
            res.status(505).json(`Python process exited with code ${code}`)
            console.error(`Python process exited with code ${code}`);
        } else {
            const result = JSON.parse(outputData);
            res.status(200).json(result.phishing_value)
        }
        });
    } catch (err) {
        res.status(400).json(err)
    }
})
```

(a) api.js

```python
with open('./model/model.pkl', 'rb') as file: #LOAD MODEL
    model = pickle.load(file)
with open(filename, 'w', newline='') as file:
    df = pd.read_csv(filename,names=['url'])
    url=[]
    data = json.loads(sys.stdin.read()) # RECEIVE INPUT
    url.append(data)
    df["url"]=url
    urls = [url for url in df['url']]
    df['protocol'],df['domain'],df['path'],df['query'],df['fragment'] = \
                        zip(*[urllib.parse.urlsplit(x) for x in urls])
    shortening_services = r"bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\ ...
    # FEATURE EXTRACTION
    def getEntropy(url): ...
    def hasLogin(url): ...
    def redirection(url): ...
    def lenClassify(url): ...
    def haveAtSign(url): ...
    def getDepth(url): ...
    def tinyURL(url): ...
    def isDomainIp(domain): ...
    def prefixSuffix(domain): ...
    def get_features(df): ...
    get_features(df)
    df.to_csv("./dataTest/test_features.csv",index=False)
    file.close()

x_test = df.drop(columns=['url','protocol','domain','path','query','fragment'])
y_pred = model.predict(x_test) #PREDICT
result = {'phishing_value': str(y_pred[0])}
sys.stdout.write(json.dumps(result)) #SEND OUTPUT
```

(b) predictor.py

# Chapter 5

# RESULTS AND DISCUSSIONS

## 5.1 Feature Analysis

### 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:
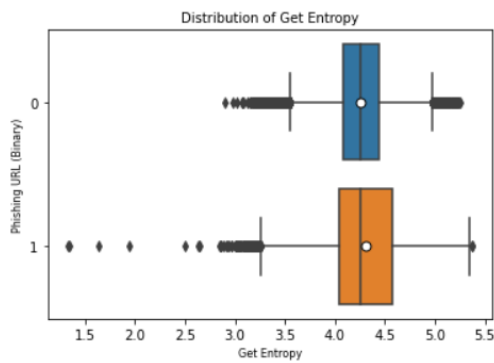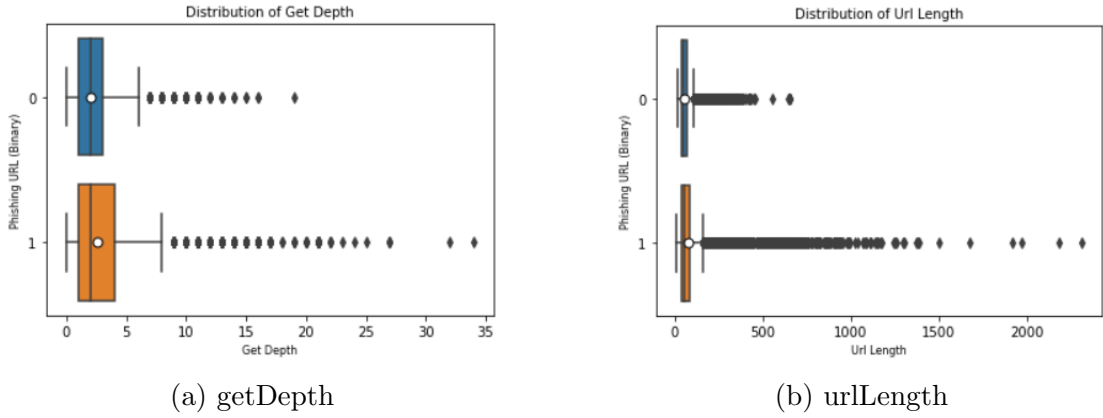


(a) getEntropy                    (b) hasLogin

(a) getDepth           (b) urlLength
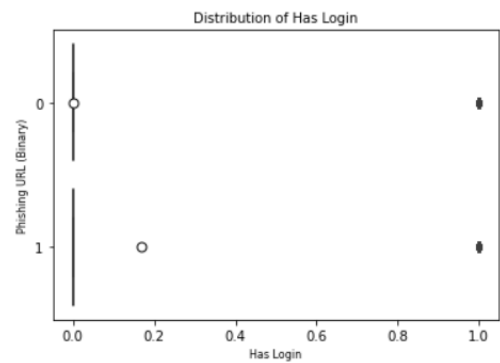
Figure 5.2: Visualization of URL features

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.

## 5.2 Pycaret Checking

| | Model | Accuracy | AUC | Recall | Prec. | F1 | Kappa | MCC | TT (Sec) |
|---|---|---|---|---|---|---|---|---|---|
| lr | Logistic Regression | 0.9445 | 0.9619 | 0.8166 | 0.9816 | 0.8915 | 0.8547 | 0.8612 | 74.4470 |
| et | Extra Trees Classifier | 0.9402 | 0.9677 | 0.8584 | 0.9221 | 0.8891 | 0.8483 | 0.8493 | 125.1360 |
| rf | Random Forest Classifier | 0.9400 | 0.9716 | 0.8526 | 0.9266 | 0.8880 | 0.8472 | 0.8485 | 114.7950 |
| xgboost | Extreme Gradient Boosting | 0.9318 | 0.9677 | 0.8028 | 0.9446 | 0.8679 | 0.8223 | 0.8273 | 156.8630 |
| qda | Quadratic Discriminant Analysis | 0.9313 | 0.9444 | 0.7752 | 0.9738 | 0.8630 | 0.8179 | 0.8276 | 5.9550 |
| lda | Linear Discriminant Analysis | 0.9284 | 0.9571 | 0.7442 | 0.9987 | 0.8529 | 0.8069 | 0.8221 | 9.6110 |
| ridge | Ridge Classifier | 0.9279 | 0.0000 | 0.7425 | 0.9988 | 0.8518 | 0.8055 | 0.8210 | 1.3870 |
| svm | SVM - Linear Kernel | 0.9271 | 0.0000 | 0.8320 | 0.9215 | 0.8679 | 0.8182 | 0.8261 | 16.7550 |
| lightgbm | Light Gradient Boosting Machine | 0.9226 | 0.9608 | 0.7715 | 0.9403 | 0.8475 | 0.7963 | 0.8033 | 9.2440 |
| dt | Decision Tree Classifier | 0.9204 | 0.8977 | 0.8463 | 0.8655 | 0.8558 | 0.8009 | 0.8010 | 7.8590 |
| knn | K Neighbors Classifier | 0.9187 | 0.9402 | 0.7856 | 0.9106 | 0.8435 | 0.7890 | 0.7929 | 264.9220 |
| gbc | Gradient Boosting Classifier | 0.9092 | 0.9379 | 0.7161 | 0.9454 | 0.8149 | 0.7563 | 0.7691 | 118.1240 |
| ada | Ada Boost Classifier | 0.9076 | 0.9321 | 0.7240 | 0.9294 | 0.8139 | 0.7537 | 0.7641 | 30.9280 |
| nb | Naive Bayes | 0.7918 | 0.8363 | 0.2822 | 0.9089 | 0.4306 | 0.3439 | 0.4324 | 2.9420 |
| dummy | Dummy Classifier | 0.7209 | 0.5000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 1.3600 |

Figure 5.3: 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:5.3 shows the comparison result.
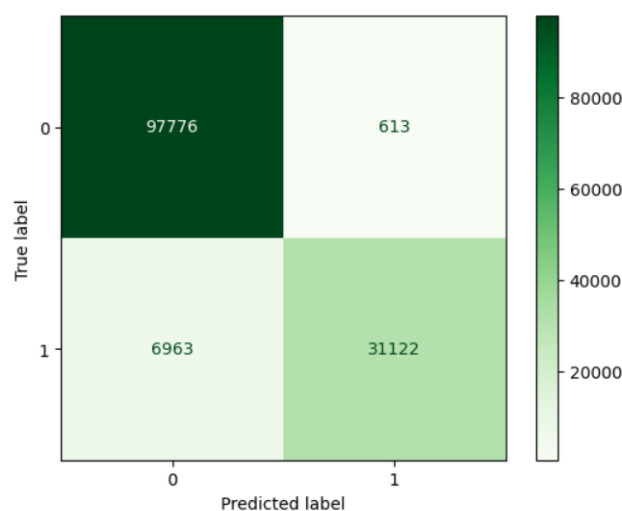
## 5.3   Model Analysis



Figure 5.4: 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:5.5

```
Accuracy:  94.4487594706684
Misclassification Rate:  5.5512405293315945
Recall:  81.71721150059078
Specificity:  99.37696287186576
Precision:  98.06837876161967
```

Figure 5.5: Evaluation metrics

Based on the Logistic Regression Model, it has assigned feature importance score to each extracted features. It helps to provide insights into which features
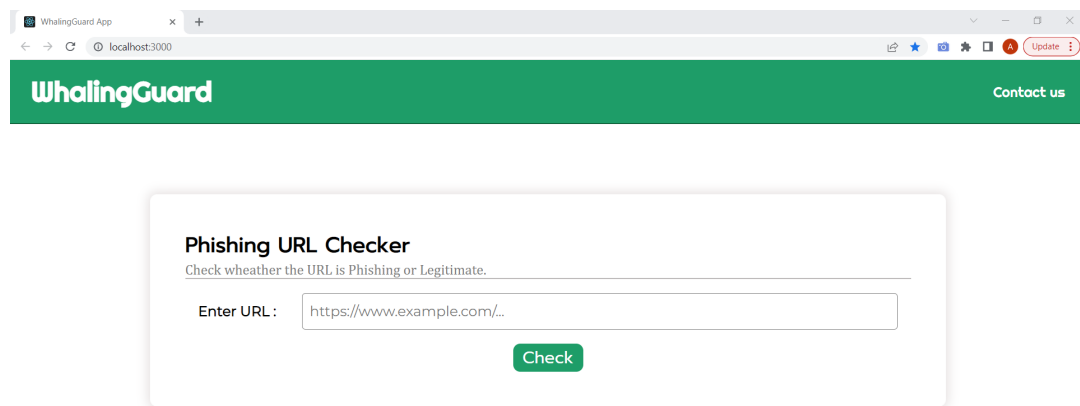
have the most influence on the model's predictions. Fig:5.6 shows the importance of some of the features in our project in the descending order.

```
Features                    Importance

qty_slash_query             19.22431794318797
qty_slash_url               19.00393575525493
qty_slash_path              17.56294918389899
has_Login                   15.496830923910235
is_Domain_Ip                12.2086835431673
fragment_length             8.95439355275234
domain_length               8.821962652886436
query_length                8.78695676875877
path_length                 8.7356989966141
url_length                  8.720265081406051
qty_questionmark_url        8.662046144542202
redirection                 4.730542172215818
qty_questionmark_query      4.37714698163631
qty_hashtag_url             2.821487228346918
qty_dot_domain              1.5784326669308384
qty_space_url               1.4523991381482824
qty_space_path              1.2964672425292696
qty_at_query                1.2930669172697076
qty_equal_query             1.2887258394991314
qty_exclamation_url         1.1228730198964438
qty_equal_path              1.108152536414815
get_Depth                   1.0793035301220977
qty_plus_url                0.9850752151579182
qty_space_fragment          0.9608811254058621
qty_exclamation_path        0.9175487760941471
qty_at_path                 0.8710959813259251
qty_equal_fragment          0.7850819986074719
prefix_Suffix               0.7489112091707167
qty_equal_url               0.7294578183660566
qty_dot_path                0.6550562709481025
qty_hashtag_fragment        0.6525086831833055
qty_dot_query               0.6436619988673965
qty_percent_fragment        0.58294145846656
```

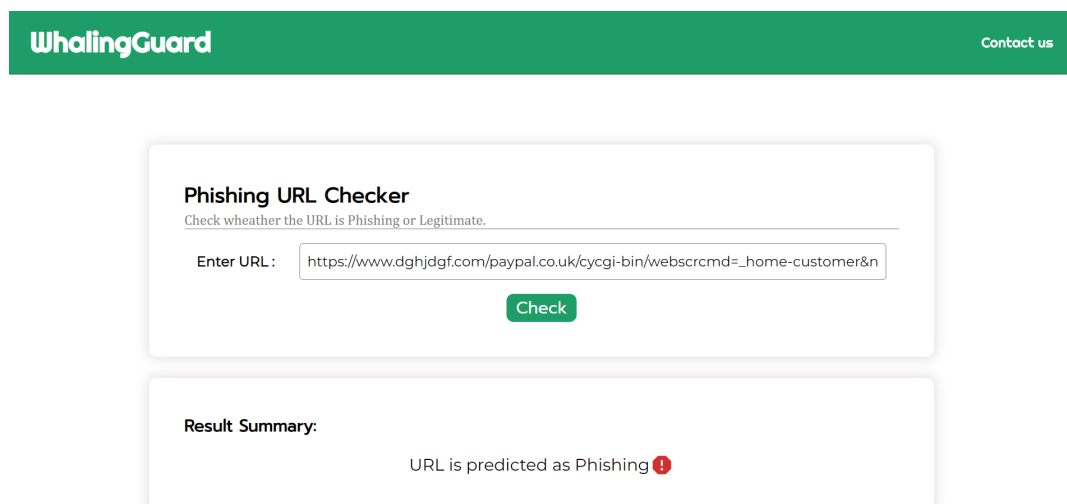Figure 5.6: Features and their importances

## 5.4   Web Application

We have developed a web application where the users can input urls to check if it is phishing or non-phishing.The UI of the application is developed using Reactjs and the API setup for the application is implemented using Nodejs. Fig:5.7 shows the User Interface of the WhalingGuard web application.

Figure 5.7: 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.

Figure 5.8: Prediction Results

# Chapter 6

# 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 machine 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] "PhishHaven—An Efficient Real-Time AI Phishing URLs Detection System" ( Maria Sameen, Kyunghyun Han, Seong Oun Hwang [2020])

[2] "Sufficiency of Ensemble Machine Learning Methods for Phishing Websites Detection" (Yi Wei, Yuji Sekiya. [2021])

[3] "Eth-PSD: A Machine Learning-Based Phishing Scam Detection Approach in Ethereum" (Arkan Hammoodi Hasan Kabla, Mohammed Anbar, Selvakumar Manickam, Shankar Karupayah . [2021])

[4] "OFS-NN: An Effective Phishing Websites Detection Model Based on Optimal Feature Selection and Neural Network" (Erzhou Zhu, Yuyang Chen, Chengcheng Ye, Xuejun Li, Feng Liu. [2019])

[5] "Comparison of Classification Algorithms for Detection of Phishing Websites" (Paulius Vaitkevicius. [2020])

[6] "Phishing Detection using Random Forest, SVM and Neural Network with Backpropagation" (Sindhu, Sunil Parameshwar Patil, Arya Sreevalsan, Faiz Rahman [2020])

[7] "Intelligent rule-based phishing websites classification" (Rami M. Mohammad, Fadi Thabtah, Lee McLusky. [2014])

[8] "Phishing sites detection based on Url Correlation" (Ying Xue, Yang Li,Yuangang Yao, Xianghui Zhao, Jianyi Liu,Ru Zhang. [2016])

[9] "Characteristics of Understanding URLs and Domain Names Features: The Detection of Phishing Websites With Machine Learning Methods" (Ilker Kara, Murathan Ok, Ahmet Ozaday . [2022])

[10] "Detecting phishing websites using machine learning technique" (Ashit Kumar Dutta, Zhihan Lv. [2018])

[11] "Web Phishing Detection Using Machine Learning " (N Kumaran, Purandhar Sri Sai, Lokesh Manikanta. [2022])

[12] "Detection of Phishing Websites using Machine Learning" (Atharva Deshpande, Omkar Pedamkar, Nachiket Chaudhary . [2021])

[13] A New Method for Detection of Phishing Websites: URL Detection" (Shraddha Parekh, Dhwanil Parikh , Srushti Kotak , Prof. Smita Sankhe. [2018])

[14] "Detection of URL based Phishing attacks Using Machine Learning" (Ms. Sophiya Shikalgar , Dr. S. D. Sawarkar , Mrs. Swati Narwane. [2018])

[15] "Large-Scale Automatic Classification of Phishing Pages " (Collin Whittaker, Brian Ryner, Maria Nasif. [2010])