PROJECT REPORT ON

# PISHSHILD: MACHINE LEARNING<br>FOR<br>PHISHING DETECTION

Submitted in partial fulfillment of the requirements of the degree of bachelor's in engineering

by

| | |
|---|---|
| KARTIK PATEL | BE5 - 22 |
| PRATHAM BHADRA | BE6 - 36 |
| ASHISH JHA | BE5 - 9 |

Under the guidance of
Prof Chintal Gala (Guide)

DEPARTMENT OF INFORMATION TECHNOLOGY
SHAH & ANCHOR KUTCHHI ENGINEERING COLLEGE CHEMBUR,
MUMBAI-400088.
2023-2024

Mahavir Education Trust's

# SHAH & ANCHOR KUTCHHI ENGINEERINGCOLLEGE

**Mahavir Education Trust Chowk, W.T. Patil Marg, Chembur, Mumbai 400 088**
Affiliated to University of Mumbai, Approved by D.T.E. & A.I.C.T.E.
# UG Prgrams accredited by NBA for 3years w.e.f.1st July, 2022

## Certificate

*This is to certify that the report of the project entitled*

**"PISHSHELD: MACHINE LEARNING FOR PHISHING DETECTION"**

*is a bonafide work of*

| | |
|---|---|
| KARTIK PATEL | BE5 – 22 |
| PRATHAM BHADRA | BE6 - 36 |
| ASHISH JHA | BE6 - 41 |

Submitted to the

**UNIVERSITY OF MUMBAI**

during semester VIII in partial fulfilment of the requirement for the award of the degree

**BACHELOR OF ENGINEERING**

in

**INFORMATION TECHNOLOGY**

(Ms. Chintal Gala)
Guide

(Ms. Swati Nadkarni )
Head of Department

(Dr. Bhavesh Patel)
Principal

## Approval for Project Report for B. E. semester VIII

This project report entitled "**PhishShield: Machine Learning For Phishing Detection"** by Pratham Bhadra, Kartik Patel, Ashish Jha is approved for semester

VIII in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering.

Guide:

1.____________________________

2.____________________________

Examiners:

1.____________________________

2.____________________________

Date:

Place: Mumbai

# DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

**KARTIK PATEL (BE5 – 22)** ________________

**PRATHAM BHADRA (BE6 - 36)** ________________

**ASHISH JHA (BE5 – 9)** ________________

**(Name and Roll No)** **(Signature)**

Date:

# TABLE OF CONTENTS

# TABLE OF CONTENT

# List of Figures

# ABSTRACT

This document presents a proposal for a chrome browser plugin that can detect and alert users about phishing websites in real-time. The detection mechanism is based on the random forest classifier, which has been found to be more effective than other techniques for identifying phishing websites.

Traditionally, the classification process is performed on a server, with the plugin requesting the server for the results. However, this project takes a different approach by conducting the classification directly within the browser itself.

Classifying in the client-side browser offers several advantages, including enhanced privacy as the user's browsing data does not need to leave their machine. Additionally, the detection process is not affected by network latency.

The main objective of this project is to implement the methodology described in the aforementioned paper using JavaScript, enabling it to function as a browser plugin. Since JavaScript lacks extensive support for machine learning libraries and considering the limited processing power of client machines, it is crucial to develop a lightweight approach. This involves training the random forest classifier on a dataset of phishing websites using Python's scikit-learn library. Subsequently, the learned model parameters must be exported into a portable format for utilization in JavaScript.

# CHAPTER 1 - INTRODUCTION

## Problem Statement: -

Phishing is the act of fraudulently attempting to acquire sensitive information, such as usernames, passwords, and credit card details, often for malicious purposes. This is typically done through email spoofing or instant messaging, where users are directed to enter personal information on a counterfeit website that closely resembles a legitimate one, with the only difference being the URL. Phishing attempts often disguise themselves as communications from social media platforms, online auction sites, banks, or online payment processors in order to deceive victims. These phishing emails may also contain links to websites that distribute malware.

Detecting phishing websites often involves searching through a directory of known malicious sites. However, since phishing websites are often short-lived, it is challenging for the directory to keep track of all of them, including new ones. To address this issue, machine learning techniques can be employed to improve the detection of phishing websites. Among various machine learning techniques, the random forest classifier has shown promising performance.

The most effective way for end users to benefit from this is by implementing a browser plugin that can detect phishing sites in real time as the user browses the internet. However, browser extensions have certain limitations, such as being written only in JavaScript and having limited access to page URLs and resources.

Existing plugins typically send the URL to a server for classification, which raises concerns about user privacy and may result in delays due to network latency. Moreover, there is a risk that the plugin may fail to warn the user in a timely manner. Considering the importance of this security issue and the privacy implications, we have decided to develop a Chrome browser plugin that can perform the classification without relying on an external server.

## Objective:

Our objective is to create a browser plugin that will promptly notify the user upon visiting a phishing website. It is essential that the plugin does not establish any connections with external web services, as this could compromise the user's browsing data. The detection process should be instantaneous, enabling the user to be alerted before providing any sensitive information on the phishing website

## Scope:

In 2021, a notable 76% of organizations reported encountering phishing attacks, aligning with the trend observed in 2017. Nearly half of the surveyed information security professionals acknowledged that the frequency of these attacks has consistently escalated since 2016. Over a three-month period in the first half of 2021, an alarming 93,570 phishing incidents targeted businesses and residents in Qatar. Considering the continuously growing number of internet users and the ongoing menace of phishing attacks, the demand for robust security solutions to protect Chrome users is pressing.

# CHAPTER 2 - Literature Review

This research paper, conducted in 2017, explores the realm of phishing detection, a critical cybersecurity concern wherein fraudulent websites mimic legitimate ones to obtain sensitive user information. Despite numerous proposed solutions, no single method proves entirely effective against this threat. Employing data mining techniques, particularly focusing on various classifiers, the study evaluates their performance in distinguishing between legitimate and phishing websites. Among the classifiers assessed, Random Forest emerges as the most effective, boasting an accuracy of 97.36% and an AUC of 0.996. The study underscores the importance of phishing detection in maintaining online security and highlights Random Forest as a powerful tool in combatting this menace. [1]

The research paper titled "Phishing Detection using Random Forest, SVM and Neural Network with Backpropagation" conducted by Smita Sindhu, Sunil Parameshwar Patil, Arya Sreevalsan, and Faiz Rahman from B.M.S. College of Engineering in Bengaluru, India, presents an in-depth investigation into phishing detection methods using machine learning algorithms. Phishing, a prevalent cyber-attack, aims to obtain sensitive information by mimicking legitimate websites. The researchers explore the efficacy of three machine learning approaches: Random Forest classification, SVM classification, and Neural Network with backpropagation. Each method achieved high accuracies of 97.369%, 97.451%, and 97.259%, respectively. The study emphasizes the importance of distinguishing between phishing and legitimate websites to prevent data breaches. Leveraging lexical feature extraction, the team developed a Chrome extension employing SVM as the final classifier, yielding an accuracy of 97.451%. The paper underscores the significance of robust phishing detection systems in safeguarding users against cyber threats.[2]

The COVID-19 pandemic saw a surge in cyberattacks, including phishing. Existing anti-phishing solutions often lacked accuracy and adaptability. A new intelligent model based on URL characteristics was created, using machine learning. The result is a Chrome extension for phishing detection and prevention, aiming to reduce cybercrime and safeguard users from fraudulent websites.[3]

"Phishing Attacks and Protection Against Them" by Ivanov et al. discusses the pervasive threat of phishing in cybersecurity. The authors outline various forms of phishing, including spear phishing and vishing, which exploit social engineering tactics. They detail the stages of phishing attacks, from data collection to achieving the desired outcome, often financial gain or malware distribution. Examples of phishing attacks using ransomware and malicious PDF files are provided. The paper emphasizes user vigilance and recommends precautionary measures such as regular software updates and two-factor authentication to mitigate risks. Overall, it underscores the importance of a proactive approach to cybersecurity to combat phishing effectively.[4]

The paper presents a comprehensive study on detecting phishing attacks using machine learning techniques with multiple datasets. By evaluating six classifiers over three datasets and employing feature reduction using the Information Gain algorithm, the study assesses performance both with and without feature selection. Results show that RandomForest consistently outperforms other classifiers, particularly after feature reduction, with accuracies of 98% and 93.66% on two datasets, while J48 excels with 89.66% accuracy on the third dataset. Feature reduction slightly affects performance, but RandomForest remains robust. The study underscores the importance of tailored feature selection for effective phishing detection using machine learning.[5]

# CHAPTER 3 - REQUIREMENTS ANALYSIS

## Functional Requirements: -

- ✓ The plugin must have a fast response time to prevent users from submitting sensitive information to phishing websites.
- ✓ The plugin must not rely on any external web service or API that could potentially expose the user's browsing habits.
- ✓ The plugin must be capable of detecting newly created phishing websites.
- ✓ The plugin must have a self-updating mechanism to stay up-to-date with emerging phishing techniques.

## Non-Functional Requirements: -

- **User interface**: -

  The user interface should be designed to be intuitive and user-friendly, allowing users to easily identify phishing websites. The input for the system should be automatically extracted from the current webpage in the active tab, ensuring a seamless experience for the user. The output should be presented in a clear and easily understandable manner, enabling users to quickly determine if a website is a phishing attempt. Additionally, the user should be promptly alerted or interrupted in the event of encountering a phishing website, ensuring their safety and preventing any potential harm.
- **Hardware**: -

  The system can be successfully implemented without the need for any specific hardware interface.
- **Software**: -

  1. Python
  2. chrome browser

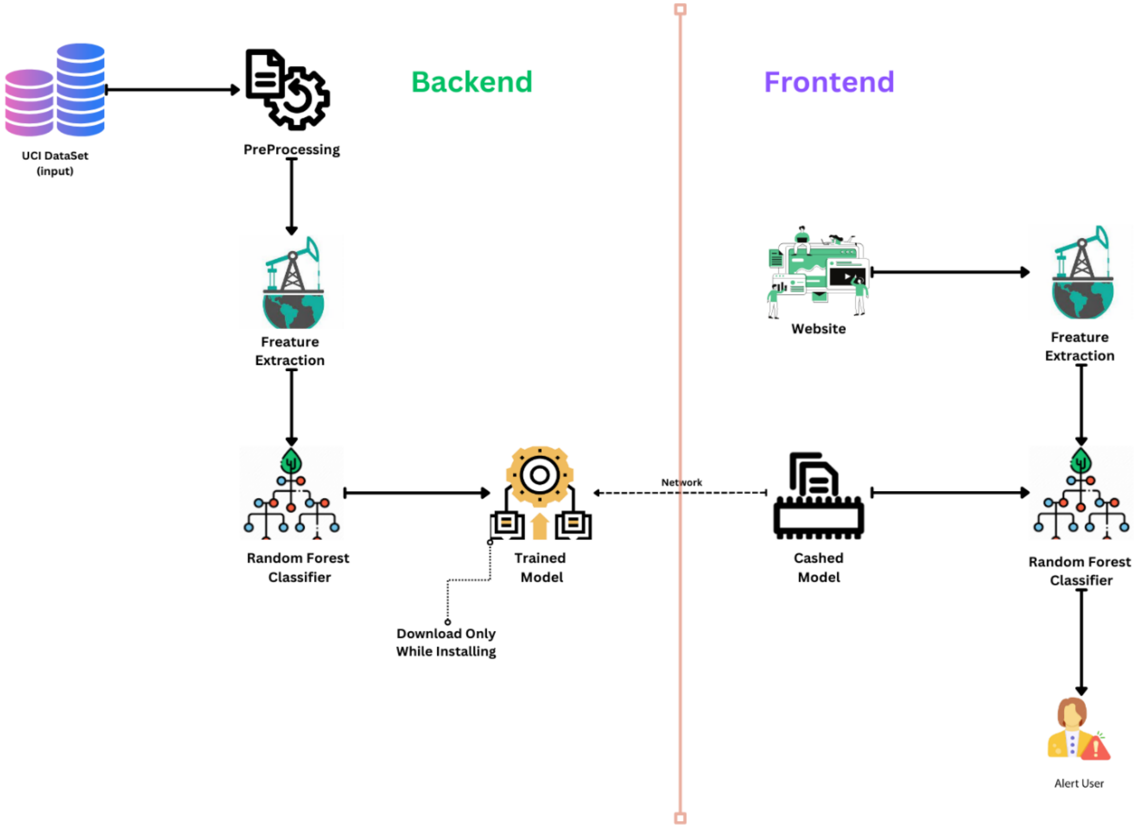
- **Performance**: -

  The plugin should be consistently accessible and capable of swiftly detecting issues with minimal false negatives.

# CHAPTER 4 - SYSTEM DESIGN

## System Architecture: -

Figure displays the block diagram of the entire system. The system utilizes a Random Forest classifier, which is trained on a dataset of phishing sites using python scikit-learn. To represent the classifier, a JSON format has been created and the learned classifier is exported to this format. A browser script has been implemented to classify the website being loaded in the active browser tab using the exported model JSON. The system's primary objective is to alert the user in the event of phishing. The Random Forest classifier uses 17 features of a website to classify whether the site is phishing or legitimate. The dataset arff file is loaded using python arff library and 17 features are chosen from the existing 30 features. The features are selected based on their ability to be extracted completely offline on the client side without being dependent on a web service or third party. The dataset with the chosen features is then separated for training and testing. The Random Forest is trained on the training data and exported to the JSON format mentioned above. The JSON file is hosted on a URL. The client-side chrome plugin executes a script on each page load and begins to extract and encode the selected features. Once the features are encoded, the plugin checks for the exported model JSON in cache and downloads it again if it is not present in the cache. With the encoded feature vector and model JSON, the script can run the classification. Then a warning is displayed to the user, in case the website is classified as phishing. The entire system is designed lightweight so that the detection will be rapid.



**4.1 Architecture diagram**

## User Interface:-

An easy-to-use User Interface has been designed for the plugin using HTML and CSS. The UI features a large circle that displays the percentage of website legitimacy in the active tab, with its color changing based on the classification output (Green for legitimate and Red for phishing). The analysis results, which include extracted features, are displayed below the circle in the following color code:

Green – Legitimate

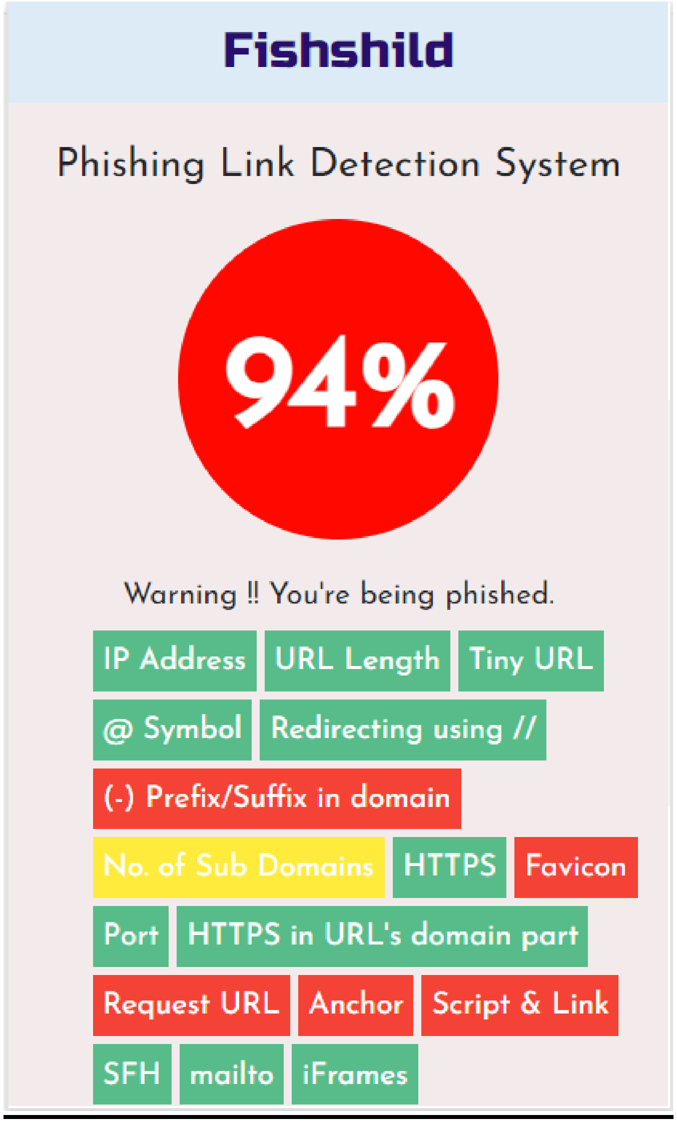
Yellow - Suspicious

Light Red - Phishing.

Additionally, the plugin alerts the user in case of phishing to prevent them from entering sensitive information on the website. The test results, such as precision, recall, and accuracy, are displayed on a separate screen. The UI can be viewed in the figure

## Classic Diagram: -

The class diagram of the entire Machine Translation system is depicted in figure, illustrating the functions of various modules in the system. Additionally, it demonstrates the interaction between these modules, offering a clear understanding for implementation.



Fishshild

Phishing Link Detection System

94%

Warning !! You're being phished.

**4.2 UI Design of Plugin**

# CHAPTER 5 - MODULE DESIGN

## Preprocessing: -

The dataset is obtained from the UCI repository and imported into a numpy array. It comprises of 30 features, which require reduction in order to facilitate extraction on the browser. Each feature is experimented with on the browser to ensure extraction without reliance on external web services or third-party tools. Through these experiments, 17 features have been selected out of the original 30, resulting in minimal loss in accuracy on the test data. Increasing the number of features enhances accuracy, but diminishes the ability to rapidly detect considering the time required for feature extraction. Consequently, a subset of features is chosen to strike a balance between these factors.

Subsequently, the dataset is divided into training and testing sets, with 30% allocated for testing purposes. Both the training and testing data are then saved to disk.

## Training: -

The training data is loaded from the disk after preprocessing. Using the scikit-learn library, a random forest classifier is trained on the data. To leverage the benefits of ensemble learning, an ensemble of 10 decision tree estimators is utilized. Each decision tree follows the CART algorithm, aiming to minimize the gini impurity.

$$Gini(E) = 1 - c\Sigma j = 1 p 2$$

Additionally, the cross-validation score is computed on the training data, while the F1 score is calculated on the testing data. Finally, the trained model is exported to JSON format using the subsequent module.

## Feature extraction: -

The 17 aforementioned features must be extracted and encoded for each webpage in real-time during the page loading process. To achieve this, a content script is utilized to access the Document Object Model (DOM) of the webpage. This content script is automatically injected into every page as it loads. Its primary responsibility is to gather the features and subsequently transmit them to the plugin.

The primary objective of this endeavor is to avoid reliance on any external web service. Additionally, it is crucial that the features remain unaffected by network latency, ensuring swift extraction. These considerations are carefully addressed during the development of the feature extraction techniques.

Upon successful extraction, each feature is encoded using the following notation:

{-1, 0, 1}. These values represent the following categories:

-1 - Legitimate

0 - Suspicious

1 - Phishing.

The resulting feature vector, consisting of 17 encoded values, is then passed from the content script to the plugin.

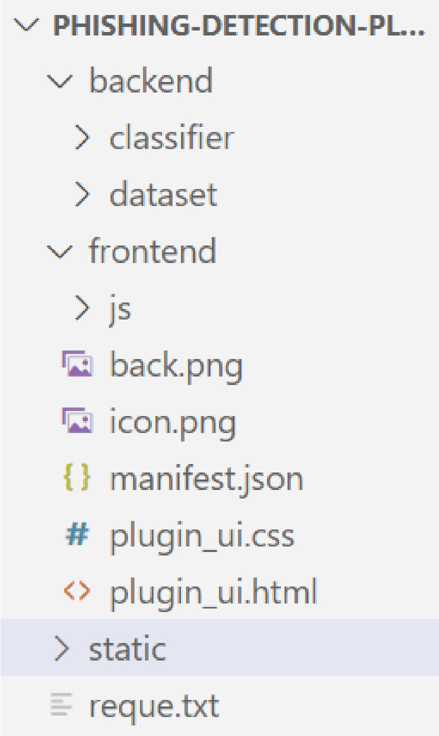
## Classification: -

The feature vector obtained from the content script undergoes classification using the Random Forest algorithm. The Random Forest parameters JSON file is downloaded and cached on the disk. The script attempts to load the JSON from the disk cache, and in the event of a cache miss, the JSON is downloaded again.

A JavaScript library has been developed to emulate the behavior of the Random Forest. This library compares the feature vector with the threshold values of the nodes in the JSON file. The binary classification output is determined based on the values of the leaf nodes. If the webpage is classified as phishing, the user is notified [2]

# CHAPTER 6 - SYSTEM DEVELOPMENT

To provide a comprehensive overview of the system's organization, it is important to note that it is divided into two main components: the backend and the plugin. The backend is responsible for dataset preprocessing and training modules, while the plugin serves as the frontend and includes javascript files for content and background scripts, as well as the Random Forest script. The plugin also contains HTML and CSS files for the user interface. A visual representation of the code's organization, including these various modules, can be found in the figure provided



```
v PHISHING-DETECTION-PL...
  v backend
    > classifier
    > dataset
  v frontend
    > js
    back.png
    icon.png
    manifest.json
    plugin_ui.css
    plugin_ui.html
  > static
  reque.txt
```

**6.1 Directory of project**

## Pipeline: -

The description of the input and output for each module of the system is provided in this section.

- **Preprocessing**

  The first module, known as Preprocessing, is responsible for taking the downloaded dataset in arff format and generating four new files. These files include training features, training class labels, testing features, and testing class labels.

- **Training**

  This module utilizes the four output files from the preprocessor to generate trained Random Forest object, accompanied by the cross validation score on the training set.

- **Exporting model**

  This module generates the JSON representation of the learned Random Forest classifier object and writes it to a file on disk.

- **Plugin Feature Extraction**

  This module accepts a webpage as input and produces a feature vector containing 17 encoded features.

- **Classification:**

  This module takes the feature vector from feature extraction module and the JSON format from the Exporting model module and then gives a boolean output which denotes whether the webpage is legitimate or phishing

# CHAPTER 7 - RESULTS AND DISCUSSION

## Dataset For Testing:

The test set comprises data points that are segregated from the dataset at a ratio of 70:30. Additionally, the plugin undergoes testing with websites enlisted in phishTank. Any newly discovered phishing sites are promptly incorporated into PhishTank. It is important to highlight that the plugin possesses the capability to identify new phishing sites. The outcomes of both module testing and the comprehensive system testing are summarized below.

## Output In Various Stages:

The results obtained from module testing are displayed in this section

- **Preprocessing:**

The figure displays the output of the preprocessing module.

**The dataset has 11055 datapoints with 30 features**

**Features:**['having_IP_Address', 'URL_Length', 'Shortining_Service', 'having_At_Symbol', 'double_slash_redirecting', 'Prefix_Suffix', 'having_Sub_Domain', 'SSLfinal_State', 'Domain_registeration_length', 'Favicon', 'port', 'HTTPS_token', 'Request_URL', 'URL_of_Anchor', 'Links_in_tags', 'SFH', 'Submitting_to_email', 'Abnormal_URL', 'Redirect', 'on_mouseover', 'RightClick', 'popUpWidnow', 'Iframe', 'age_of_domain', 'DNSRecord', 'web_traffic', 'Page_Rank', 'Google_Index', 'Links_pointing_to_page', 'Statistical_report', 'Result']

**Before spliting**

X:(11055, 17), y:(11055,)

**After spliting**

X_train:(7738, 17), y_train:(7738,), X_test:(3317, 17), y_test:(3317,)

**Saved!**

Test Data written to testdata.json

- **Training:**

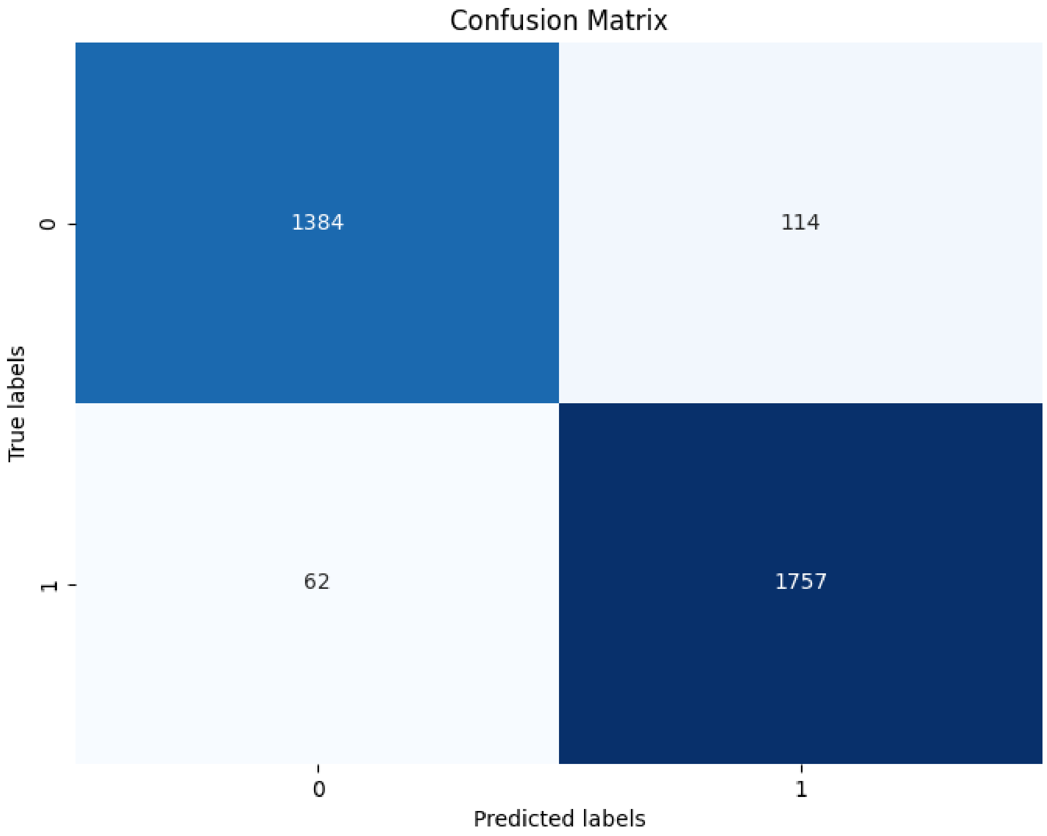
The training module's output is depicted in the figure.

```
PS D:\PhishShild\PhishShild_master\backend\classifier> python .\training.py
X_train:(7738, 17), y_train:(7738,)
Cross Validation Score: 0.9466257843029104
Accuracy: 0.9481459149834187
Number of trees in the random forest: 100
```

- **Confusion Matrix Analysis:**

The confusion matrix is a fundamental tool for evaluating the performance of a classification model, offering deeper insights into how well the model is distinguishing between different classes. Let's delve deeper into the analysis of the confusion matrix provided.

The confusion matrix provides a detailed breakdown of the model's predictions



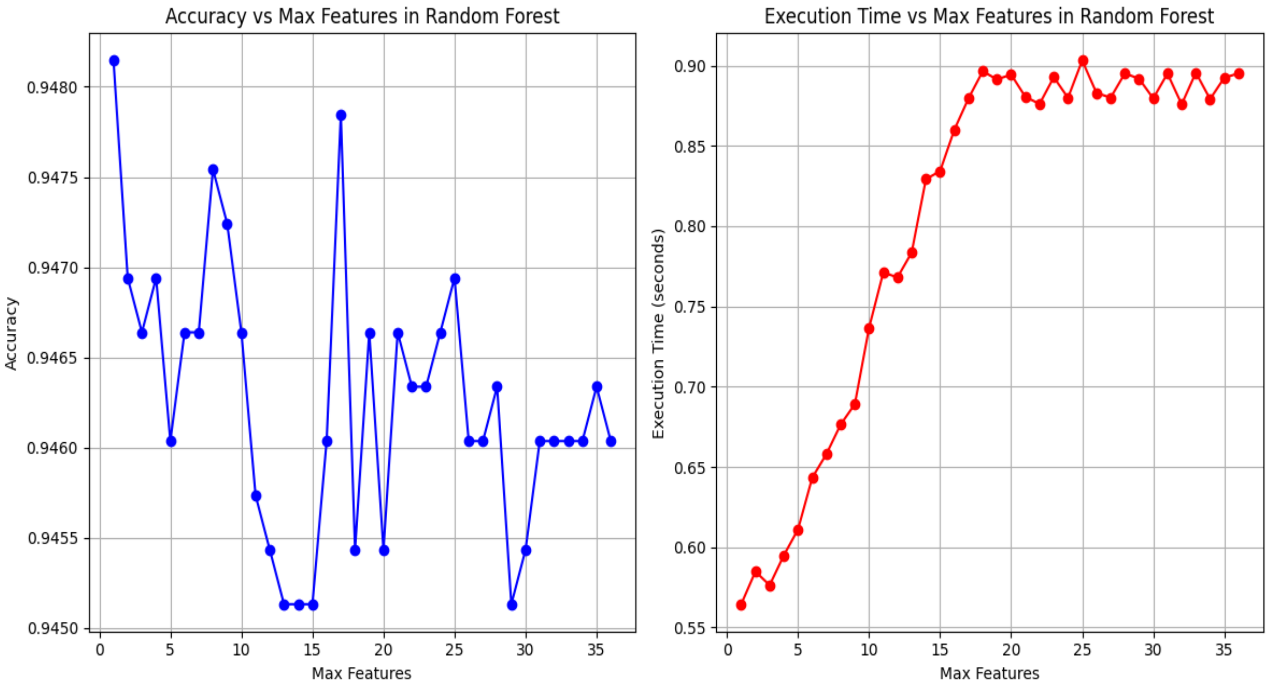
**7.1 Confusion Matrix**

- **True Positives (TP):** The model correctly identified 1757 instances belonging to class 1.
- **True Negatives (TN):** 1384 instances were accurately classified as belonging to class -1.
- **False Positives (FP):** 114 instances were incorrectly classified as class 1 when the actual class was -1.
- **False Negatives (FN):** 62 instances were wrongly classified as class -1 when they belonged to class 1.

- **Feature Count Impact: Accuracy and Execution Time Analysis:**

The analysis of the phishing detection system's performance across various configurations yielded significant insights. Notably, after thorough experimentation, it was observed that an optimal balance between accuracy and efficiency emerged with the utilization of 17 carefully selected features. This particular configuration demonstrated the highest level of accuracy while managing to maintain a reasonable execution time. Such findings underscore the critical importance of feature selection in the design and deployment of phishing detection mechanisms. By striking the right balance between accuracy and efficiency, the effectiveness of these systems can be substantially enhanced, thus fortifying cybersecurity defenses against evolving threats.

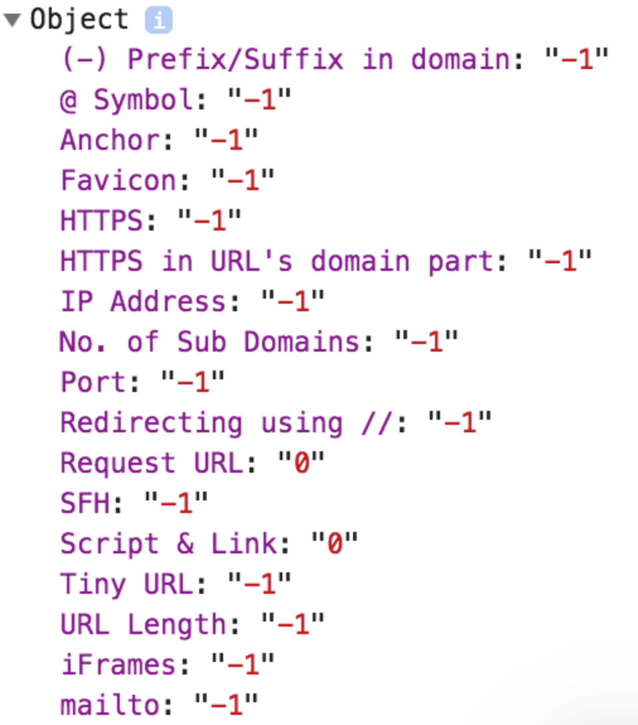
This optimal configuration serves as a testament to the intricate interplay between accuracy, feature count, and execution time in the realm of phishing detection. As phishing attacks continue to grow in sophistication, it becomes increasingly imperative for detection systems to adapt and evolve accordingly. The findings of this analysis not only shed light on the ideal configuration for maximizing detection efficacy but also pave the way for future research and development endeavors in this domain.



**7.2 Accuracy and Execution Time Analysis**

- **Feature Extraction:**

The webpage at thetechcache.science has undergone feature extraction, resulting in 17 features. These features have been logged into the console, as depicted in figure 6.4. They are stored as key-value pairs, with the values being encoded within the range of -1 to 1, as previously mentioned.

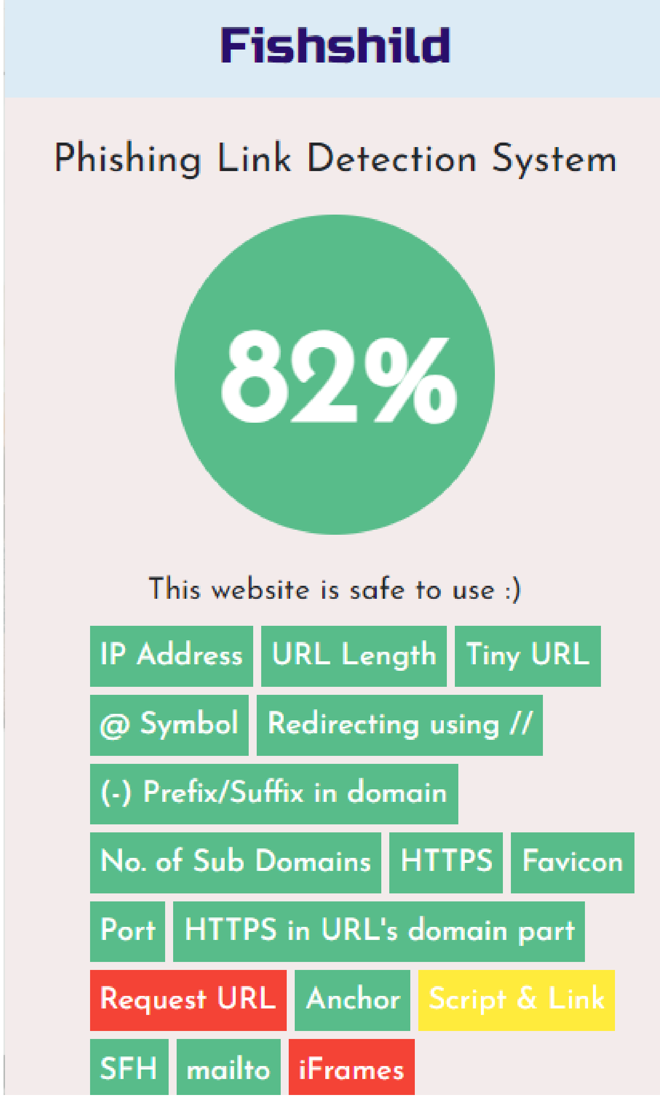


```
▼ Object
    (-) Prefix/Suffix in domain: "-1"
    @ Symbol: "-1"
    Anchor: "-1"
    Favicon: "-1"
    HTTPS: "-1"
    HTTPS in URL's domain part: "-1"
    IP Address: "-1"
    No. of Sub Domains: "-1"
    Port: "-1"
    Redirecting using //: "-1"
    Request URL: "0"
    SFH: "-1"
    Script & Link: "0"
    Tiny URL: "-1"
    URL Length: "-1"
    iFrames: "-1"
    mailto: "-1"
```

**7.3 Number of Feature**

- **Classification:**

The classification output is displayed directly in the Plugin UI, with a green circle representing legitimate sites and light red indicating phishing.



Fishshild

Phishing Link Detection System

82%

This website is safe to use :)

**7.4 Classification Diagram**

**PERFORMANCE**

The evaluation of the entire system's performance is conducted based on the standard parameters outlined below.

## Cross Validation score

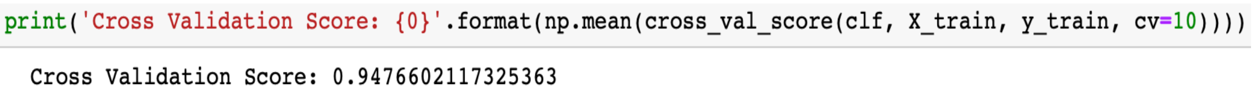
Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called overfitting.

To solve this problem, another part of the dataset can be held out as a validation set. Training proceeds on the training set, and then evaluation is done on the validation set. If the experiment seems successful, final evaluation can be done on the test set.

However, by partitioning the available data into three sets, we drastically reduce the number of samples that can be used for learning the model, and the results can depend on a particular random choice for the pair of train and validation sets.

A solution to this problem is a procedure called cross-validation (CV for short). A test set should still be held out for final evaluation, but the validation set is no longer needed when doing CV. In the basic approach, called k-fold CV, the training set is split into k smaller sets. For each of the k "folds," a model is trained using k-1 of the folds as training data, and the resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy).

The score on 10-fold cross-validation is as follows:



```
print('Cross Validation Score: {0}'.format(np.mean(cross_val_score(clf, X_train, y_train, cv=10))))
```

```
  Cross Validation Score: 0.9476602117325363
```

**7.5 Cross Validation score**

## F1 score:

The accuracy of a test can be measured using the F1 score, which takes into account both precision and recall. Precision is determined by dividing the number of correct positive results by the total number of positive results returned by the classifier. On the other hand, recall is calculated by dividing the number of correct positive results by the total number of relevant samples, which includes all samples that should have been identified as positive. The F1 score is then obtained by taking the harmonic average of precision and recall. A perfect F1 score of 1 indicates perfect precision and recall, while a score of 0 represents the worst performance.

The F1 score can also be seen as a weighted average of precision and recall, with equal contributions from both. Its best value is 1 and worst value is 0. The formula for calculating the F1 score is as follows:

$$F1 = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall}).$$

$$F1 = \frac{2 \text{ x } (0.8217636022514071 \text{ X } 0.9631665750412315)}{(0.8217636022514071 + 0.9631665750412315)}$$

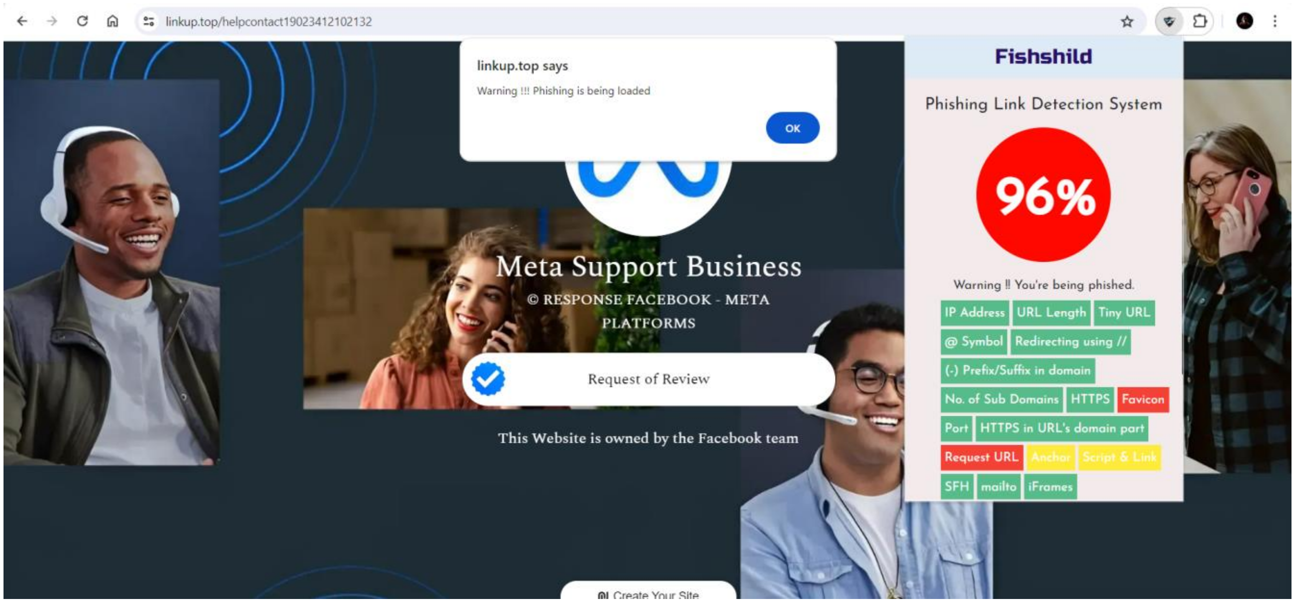
$$F1 = 0.8868640850417616$$

In the case of the phishing classifier, the precision, recall, and F1 score are manually calculated using JavaScript on the test data set. The results of these calculations are presented in the accompanying figure.

# CHAPTER – 8 SAMPLE SCREENSHOTS DURING TESTING PHISH

- **WEBSITE**

The plugin's functionality includes the ability to detect phishing sites listed on PhishTank. Upon visiting such a site, the plugin generates a warning message and assigns a low trust value to the site. In addition, a light red circle is displayed to indicate the high likelihood of phishing activity. Please refer to the accompanying figure for a visual representation of the plugin's output.



linkup.top/helpcontact19023412102132

linkup.top says

Warning !!! Phishing is being loaded

Fishshild

Phishing Link Detection System

96%

Warning ‼ You're being phished.

Meta Support Business

© RESPONSE FACEBOOK - META PLATFORMS

This Website is owned by the Facebook team

8.1 Screenshot of phished

# CHAPTER – 9 CONCLUSIONS

The system has the potential to become an even more effective solution in the ever-evolving landscape of phishing detection, offering users a robust defense against online threats while maintaining their privacy. Further enhancements could be achieved by expanding the feature set for training and implementing result caching for frequently visited sites, optimizing performance without compromising security. Moreover, the utilization of technologies like WorkerThreads could further expedite the classification process, ensuring prompt and efficient detection.

## Summary:

`This system is designed to detect phishing websites by focusing on client-side implementation and providing rapid detection to warn users before they fall victim to phishing attacks. The key implementation involves porting a Random Forest classifier to JavaScript. Unlike similar systems, which rely on extracting webpage features that are not feasible on the client side, this system only uses features that can be extracted on the client side. As a result, it is able to offer rapid detection and better privacy. Although using fewer features may slightly decrease accuracy, it significantly improves the system's usability. Through this work, a subset of webpage features that can be implemented on the client side without compromising accuracy has been identified. The porting of the Random Forest classifier from Python to JavaScript, along with the system's own implementation, has further enhanced rapid detection. The JSON representation of the model and the classification script have been designed with time complexity in mind. As a result, the plugin is capable of detecting phishing attempts even before the page finishes loading. The F1 score, calculated on the client side using the test set, is 0.886.

## Criticisms:

The system's accuracy may not be on par with the state-of-the-art, but it boasts better usability and effectively balances accuracy with rapid detection. The chrome extension API restrictions have a minimal impact on the plugin, although it cannot prevent malicious javascript code from executing since features are extracted in the content script injected during page load. It is worth investigating the accuracy reduction from 0.94 to 0.886 when porting from python to javascript. Unfortunately, Javascript does not support multithreading, and browsers only execute javascript, making it impossible to speed up classification using parallel threads. Currently, the plugin does not cache results, resulting in repeated computation even for frequently visited sites.

## Future Works:

The classifier is currently trained on 17 features, which can potentially be increased without compromising the speed of detection or compromising privacy. To optimize computation, the extension can be modified to cache results of frequently visited sites. However, caution must be exercised as this may leave the system vulnerable to undetected pharming attacks. Therefore, a solution must be devised to enable result caching while maintaining the ability to detect pharming. Additionally, the classification in javascript can be enhanced by utilizing WorkerThreads, which can potentially improve the classification time. Consequently, this system offers numerous possibilities for improvements and enhancements, ultimately providing a more effective solution in the realm of phishing detection

# CHAPERT 10 - ACKNOWLEDGEMENT

We are sincerely thankful to our Guide Ms. Chintal for her exemplary guidance, monitoring, and constant encouragement throughout the course of this project. We owe a special acknowledgement to her for giving us a lot of time during the period of preparing this project. We could never have done it without her support, technical advice and constant suggestions to improve our work. The blessing, help and guidance given by Ms. Chintal Gala will carry with us on our journey of life which we are going to embark.

We convey our special acknowledgements and gratitude to our review committee whose invaluable suggestions helped us in improving our project.

We are also grateful to Ms. Swati Nadkarni, the HOD of the Information Technology Department for guiding and supporting.

We would also like to thank our honourable Principal Dr. Bhavesh Patel, for encouraging us to go ahead with this project.

# CHAPTER 11 – REFERENCES

[1] "Intelligent phishing website detection using random forest classifier," *IEEE Conference Publication | IEEE Xplore*, Nov. 01, 2017. https://ieeexplore.ieee.org/abstract/document/8252051

[2] "Phishing Detection using Random Forest, SVM and Neural Network with Backpropagation," *IEEE Conference Publication | IEEE Xplore*, Oct. 09, 2020. https://ieeexplore.ieee.org/document/9277256

[3] "Phishing Detection and Prevention using Chrome Extension," *IEEE Conference Publication | IEEE Xplore*, Jun. 06, 2022. https://ieeexplore.ieee.org/document/9800826

[4] "Phishing attacks and protection against them," *IEEE Conference Publication | IEEE Xplore*, Jan. 26, 2021. https://ieeexplore.ieee.org/document/9396693

[5] A. H. Aljammal, S. Taamneh, A. Qawasmeh, and H. Bani-Salameh, "Machine learning based phishing attacks detection using multiple datasets," *International Journal of Interactive Mobile Technologies*, vol. 17, no. 05, pp. 71–83, Mar. 2023, doi: 10.3991/ijim.v17i05.37575.