Phishing Detection

A Project Report

Submitted by

SHAMBHVI SAMRIDHI	**20201CSE0495**
MUKTHI	**20201CSE0480**
MANYTHA RAJ	**20201CSE0472**
PRANAV BHATT	**20201CSE0488**
VISHAL SRIVASTAVA	**20201CSE0496**

Under the guidance of,

Ms. SREELTHA P.K

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

At

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

PRESIDENCY UNIVERSITY

BENGALURU

JANUARY 2024

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project report **"PHISHING DETECTION** "Shambhvi Samridhi, Mukthi K, Manyatha Raj, Pranav Bhatt, Vishal Srivatsava" 20201CSE0495, 20201CSE0480, 20201CSE0472, 20201CSE0488, 20201CSE0496" in partial fulfillment of requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

Ms. SREELTHA P.K
GUIDE
School of CSE&IS
Presidency University

Dr. PALLAVI R
PROFESSOR& HoD
School of CSE&IS
Presidency University

Dr. C. KALAIARASAN
Associate Dean
School of CSE&IS
Presidency University

Dr. L. SHAKKEERA
Associate Dean
School of CSE&IS
Presidency University

Dr. SAMEERUDDIN KHAN
Dean
School of CSE&IS
Presidency University

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled "**PHISHING DETECTION"** in partial fulfilment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of our own investigations carried under the guidance **of MS.SREELATHA , Asst. Professor, School of Computer Science and Engineering , Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

SHAMBHVI SAMRIDHI	**20201CSE0495**
MUKTHI	**20201CSE0480**
MANYTHA RAJ	**20201CSE0472**
PRANAV BHATT	**20201CSE0488**
VISHAL SRIVASTAVA	**20201CSE0496**

ABSTRACT

This document proposes a phishing[1] detection plugin for chrome browser that can detect and warn the user about phishing web sites in real-time using random forest classifier. Based on the IEEE paper, Intelligent phishing website detection using random forest classifier2, the random forest classifier seems to outperform other techniques in detecting phishing websites.

One common approach is to make the classification in a server and then let the plugin to request the server for result. Unlike the old approach, this project aims to run the classification in the browser itself. The advantage of classifying in the client side browser has advantages like, better privacy (the user's browsing data need not leave his machine), detection is independent of network latency.

This project is mainly of implementing the above mentioned paper in Javascript for it to run as a browser plugin. Since javascript doesn't have much ML libraries support and considering the processing power of the client machines, the approach needs to be made lightweight. The random forest classifier needs to be trained on the phishing websites dataset[3] using python scikit-learn and then the learned model parameters need to be exported into a portable format for using in javascript.

1 Phishing is mimicking a creditable company's website private information of a user.

2 https://ieeexplore.ieee.org/abstract/document/8252051/

3 https://archive.ics.uci.edu/ml/datasets/phishing+websites

ACKNOWLEDGEMENT

First of all, we are indebted to the **GOD ALMIGHTY** for allowing me to excel in our efforts to complete this project on time.

.We express our sincere thanks to our respected dean **Md. Sameeruddin Khan**, Dean, School of Computer Science And Engineering , Presidency University for getting us permission to undergo the project.

We record our heartfelt gratitude to our beloved Associate Deans **Dr. Kalaiarasan C and Dr. Shakkeera L**, School of Computer Science And Engineering, Presidency University **and Dr. Pallavi R**, Head of the Department, School of Computer Science And Engineering, Presidency University for rendering timely help for the successful completion of this project.

We are greatly indebted to our guide **Ms. SREELATHA** We would like to convey our gratitude and heartfelt thanks to the University Project-II **Coordinators Mr. Zia Ur Rahman, Mr. Peniel John Whistely** and also the department Project Coordinators **Dr. Sanjeev P Kaulgud**.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project..

SHAMBHVI SAMRIDHI	**20201CSE0495**
MUKTHI	**20201CSE0480**
MANYTHA RAJ	**20201CSE0472**
PRANAV BHATT	**20201CSE0488**
VISHAL SRIVASTAVA	**20201CSE0496**

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER -1

INTRODUCTION

- ## PROBLEM AREA

Phishing is a deceptive scheme designed to get sensitive information, such as passwords, credit card numbers, and financial information, usually with malevolent intent. This fraudulent activity is often carried out using instant messaging or email spoofing, and it usually involves people disclosing personal information on phony websites that look real. The website URL is the only difference. Requests are often made by chat groups, retailers, banks, and online payment systems to trick gullible people. Links in phishing emails may lead to websites that spread malicious malware. Finding the name of the malicious website is frequently necessary in order to detect a phishing website. Since most phishing sites are brief, it is impossible to keep an eye on this list everywhere, including on newly created phishing sites. As a result, machine learning can address the issue more effectively. This fraudulent activity is often carried out using instant messaging or email spoofing, and it usually involves people disclosing personal information on phony websites that look real. The website URL is the only difference. Requests are often made by chat groups, retailers, banks, and online payment systems to trick gullible people. Links in phishing emails may lead to websites that spread malicious malware. Finding the name of the malicious website is frequently necessary in order to detect a phishing website. Since most phishing sites are brief, it is impossible to keep an eye on this list everywhere, including on newly created phishing sites. As a result, the issue of phishing website detection can be more effectively resolved by machine learning. Random Forest classifier is used to compare various learning machines functions more effectively. The only way for end users to take use of this is through browser add-ons. As a result, when users visit phishing websites, they can instantly receive warnings. Nevertheless, browser extensions have certain restrictions, such the fact that they may only be created with javascript and have restricted access to resources and page URLs. In order to send the URL to the server and receive the results back, the plugin now transmits the URL to the server. This method could lead to concerns regarding user privacy, delayed detection because of network failures, and delayed user notification because of plugin malfunctions. We chose to utilize a Chrome browser extension that permits sharing without requiring authentication because this is a serious security concern and we value privacy. Problem Synopsis Make a plugin for the browser that notifies users when they land on a phishing website. To obtain user management information for this purpose, the plugin doesn't make any calls to any other network services. Users should receive an immediate detection alert on phishing websites before they access critical information. Encouragement Wikipedia reports that phishing assaults affected 76% of enterprises in 2017. A poll conducted among security specialists revealed that there were more attacks than in 2016. Over 93,570 phishing instances affected businesses and citizens in Qatar in the first half of 2017. after three months. As the number of people using the Internet rises, so does the demand for defenses against scams like phishing. For Chrome users, this makes the plugin a helpful tool. Affiliate Scheme. This is the initial online browser phishing utilizing a browser plugin. No external services required. To do this, it is necessary to leverage current phishing detection efforts in a way that is advantageous to the user. This entails translating the current Random Forest Python to JavaScript. With the plugin, you can quickly submit your training model to your website after loading it just once. The process requires creating these patterns (random forests) via JavaScript because the browser add-on only supports that language. As a result, this project will enable us to obtain information on phishing more quickly and to keep information better.

- **SWOT ANALYSIS**

STRENGTHS	**WEAKNESSES**
• Enables user privacy. • Rapid detection of phishing. • Can detect new phishing sites too. • Can interrupt the user incase of phishing.	• Javascript limits functionality. • Cannot use features that needs a external service such as SSL, DNS, page ranks. • No library support.
OPPORTUNITIES	**THREATS**
• Everyone conscious of privacy and security can use this plugin. • Non technical people who do business transactions are vulnerable to phishing and they are potential end users for this.	• Server side classification plugins may perform better than this and users without privacy concerns may opt of those. • Chrome Plugin API will be continuously changed.

Table 1.1 SWOT analysis

Political Pestle Analysis

Politics is rarely in charge of the initiative. One scenario is when the government implements policies to stop phishing, in which case the plugin's functionality will be lost.
EconomyPlugin was developed for open Chrome plugin API and public data. As a result, economic variables have no power over it.

Social
The user experience is the social component of this plugin. Users can identify phishing websites with the aid of the phishing detection system. Users must have at least some knowledge of phishing to install this plugin. Science and Technology This plugin could become outdated and vulnerable to threats from technological breakthroughs or advances in phishing detection. Legislation pertaining to user privacy, like GDPR, will improve this plugin's potential.

CHAPTER 2

RELATED WORKS

An overview of potential methods for identifying phishing websites is provided in this chapter. This survey aids in identifying the several approaches now in use and identifies their shortcomings. The challenge with the majority of ideas is that they are not put into practice quickly enough for the end user to profit from them.

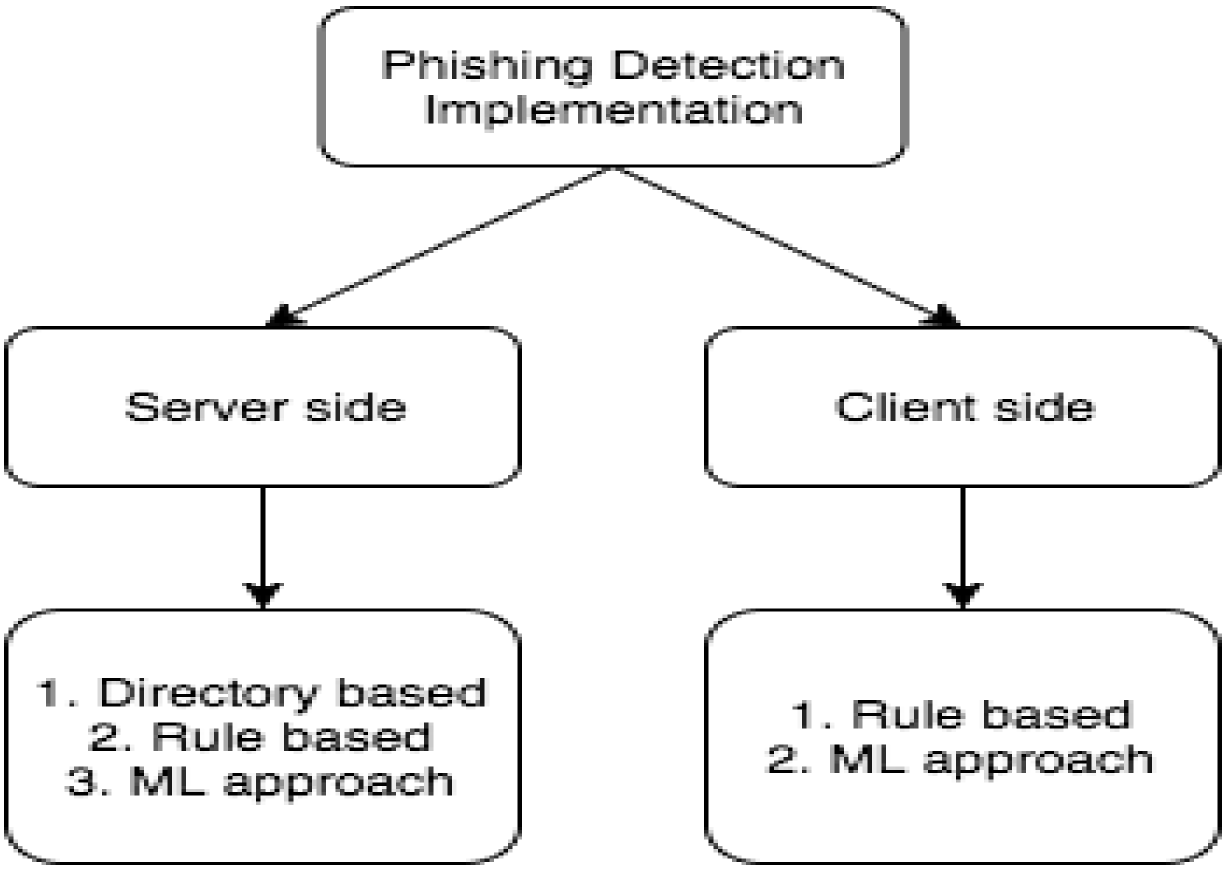


Figure 2.1 Approaches to phishing detection

DIRECTORY BASED APPROACHES

PhishTank is one of the most widely used. This is a public clearinghouse where data and information about Internet phishing are gathered, according to Phish-Tank4. Furthermore, PhishTank offers academics and developers a free open API to incorporate anti-phishing data into their apps. In order to enable developers to utilize its API to identify phishing sites, PhishTank is a list of all phishing sites that users have found and reported online.

Another grid-based method is used by Google's Google Safe Browsing API, which also offers an open API akin to PhishTank.

Since new phishing websites are being formed on a regular basis and this list cannot be maintained eternally, it is evident that this idea is ineffective. When the PhishTank API receives URLs, This influences how users browse as well.

RULE BASED APPROACHES

PhishDetector5, an existing Chrome addon, detects phishing without requiring an external service by using a rules-based methodology. While rule-based approaches are more user-friendly, machine learning techniques yield higher accuracy. Similar research on phishing attacks was conducted by Shreeram V. using the Genetic method 6 method.

making use of genetic algorithm-generated rules.

4 http://phishtank.com/

5 https://chrome.google.com/webstore/detail/phishdetector-true-phishi/kgecldbalfgmgelepbblod-foogmjdgmj

6 https://ieeexplore.ieee.org/document/5670593/

One technique for predictive blacklisting is PhishNet. TLDs match a variety of data, including IP addresses, HTTP header fields, and domain names.

Similar restrictions are applied by Stanford's SpoofGuard7 Chrome extension, which considers DNS, URLs, pictures, and links.

Phishwish: A Rule-Based Stateless Filter The authors, Michael Daniluk, Vijay K. Gurbani, and Debra L. Cook, have created a phishing filter that offers the following benefits over pre-existing filters: There is no need for security measures or training. whitebase or blacklists to reference. To ascertain whether a website is authentic, they just consider 11 criteria.

ML BASED APPROACHES

The authors of Intelligent Phishing Detection Using Random Forest Classifiers (IEEE-2017), Touseef J. Chaudhery, Esraa Molah, Abdulhamit Subaşı, and Fatin Almkallawi, talk about the use of random forest classifiers in phishing detection. Among categorization techniques, random forest performed the best, achieving an accuracy rate of up to 97.36%.

A typical paradigm for phishing detection is discussed in PhishBox: Phishing Detection and Analysis Methods (IEEE-2017), by Jhen-Hao Li and Sheng-De Wang. Consequently, there was an average 43.7% decrease in the unfavorable rate of phishing searches.

A feature that relies only on website URLs is discussed in Abdulghani Ali Ahmed and Nurul Amirah Abdullah's paper, Instant Detection of Phishing Websites (IEEE-2016). They might have a detection engine built in that can quickly and affordably identify various phishing attack types.

Nuttapong Sannglerdsinlapachai makes use of machine learning characteristics to compare domain homepages learning-based detection of phishing,

[7] https://crypto.stanford.edu/SpoofGuard/

A study on the use of a concept feature to identify web phishing issues is presented by Arnon Rungsawang. They added more domain top-page similarity features to a phishing detection system that was based on machine learning. The evaluation's f-measure result was up to 0.9250, with an error rate of 7.50%.

- One well-liked server-side prediction-based phishing detection plugin for Chrome is called Netcraft8.

DRAWBACKS

It is evident from the aforementioned connected publications that the plugins either employ a rule-based approach or a server-side machine learning technique. In contrast to machine learning (ML) approaches, rule-based approaches don't seem to perform as well. Additionally, ML-based approaches require library support, which prevents their implementation in client-side plugins. The target URL is sent by each and every plugin currently in use to an outside web server for classification. By implementing the same in a browser plugin, this project seeks to improve user privacy and do away with the requirement for an external online service.

8 https://toolbar.netcraft.com

CHAPTER 3

REQUIREMENTS ANALYSIS

- **FUNCTIONAL REQUIREMENTS**

When a user visits a phishing website, the plugin will alert them.

The following guidelines must be followed by plugins:

- In order to stop users from transferring private information to phishing websites, plugins need to be quick enough.
- Plugins ought not ever make use of third-party services or APIs that can expose user browsing habits.
- The plugin must be able to detect new phishing websites.
- Plugins must have a plan in place to keep up with new phishing techniques.

- **NON FUNCTIONAL REQUIREMENTS**

User Interface

The user must be able to recognize the phishing website with ease thanks to a straightforward and user-friendly interface. The current tab's webpage should immediately provide the input, and the result should be easily recognizable. In addition, the user ought to be notified in the event of phishing.

Hardware

There is no specific hardware interface needed for the system to be installed successfully.

Software

Python for model training

Chrome web browser

Performance

The plugin must to be constantly accessible and provide quick detection with few false negatives.

CONSTRAINTS AND ASSUMPTIONS

Constraints

• A few technologies are used in SSL, page rank, etc. This data cannot be obtained via the plugin client without an external API. As a result, prediction cannot be made using these features.

• Heavy approaches cannot be employed due to the user's machine's processing power and the site's loading time.

`Python, machine learning libraries support JavaScript less.

Assumptions

In the Chrome environment, the plugin is granted the necessary rights.

The user is familiar with the fundamentals of extensions and phishing.

SYSTEM MODELS

Use Case Diagram

Figure 3.1 displays the system's comprehensive use case diagram. After installing the plugin, the user can carry on with his regular browsing. This plugin alerts the user to potential phishing attempts by automatically scanning the pages they are browsing.

Pre condition: The user visits a website and have plugin installed.

After condition: The user gets alerted in case the webpage is a phishing attempt.

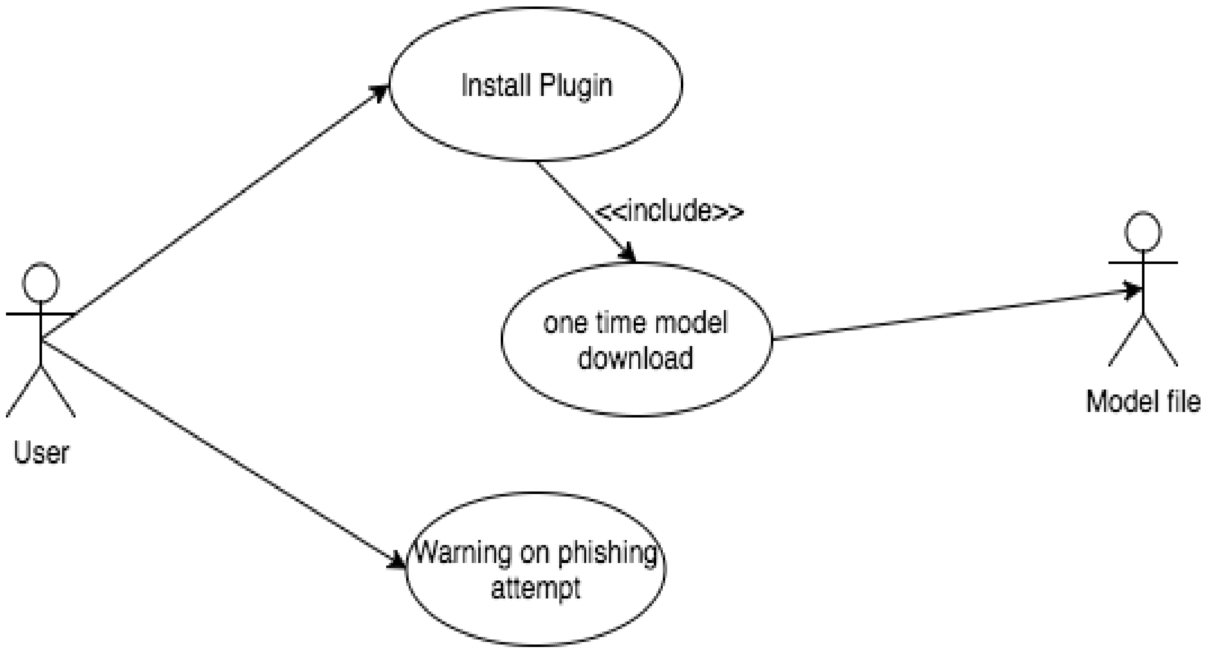


Figure 3.1 Use case diagram of the system

Sequence diagram

Figure 3.2 depicts the flow of interactions between the user and the plugin.

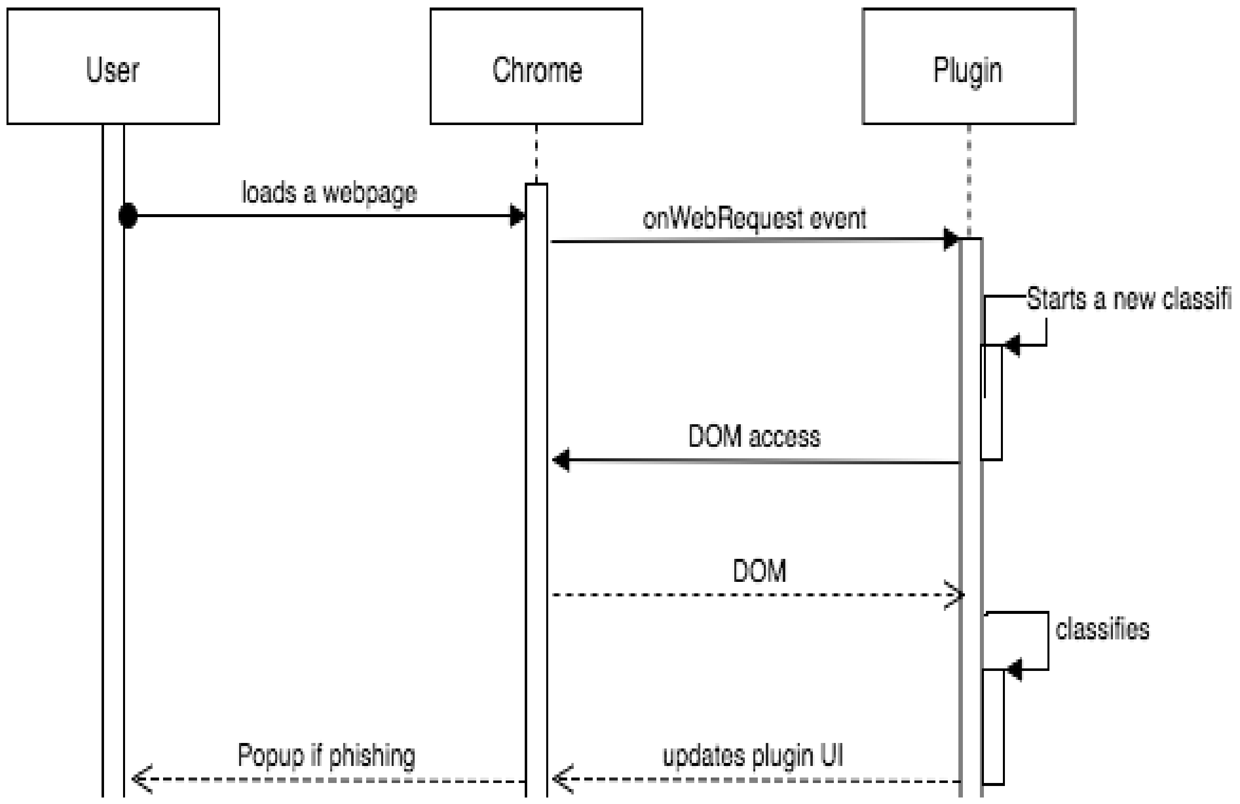


Figure 3.2 Sequence diagram

CHAPTER 4

SYSTEM DESIGN

- ## SYSTEM ARCHITECTURE

Figure 4.1 displays the block diagram of the entire system. Using Python's scikit-learn, a Random Forest classifier is trained on a dataset of phishing sites. The learnt classifier is exported to a JSON format, which has been developed to represent the random forest classifier. A browser script has been put into place that classifies the website that is loading in the tab that is now open in the browser using the exported model JSON.

The purpose of the system is to alert the user when phishing occurs. To determine if a website is phishing or authentic, the Random Forest classifier is used to 17 aspects of the website. Using the Python Arff package, the dataset Arff file is loaded, and 17 features are selected from a list of 30 features. Qualities include

chosen because they may be downloaded entirely offline from the client, independent of third parties or online services. Classification is applied to data with specific attributes for testing and training. Next, using the shown data, the random forest is trained and exported to the JSON format mentioned above. The URL hosts the JSON archive.

Customer-side Every time the page loads, the Chrome extension is utilized to generate a script that begins extracting and gaining access to the functionalities that were previously selected. After coding the property, the plugin looks for the exported JSON format in the cache and downloads it again if it isn't there.

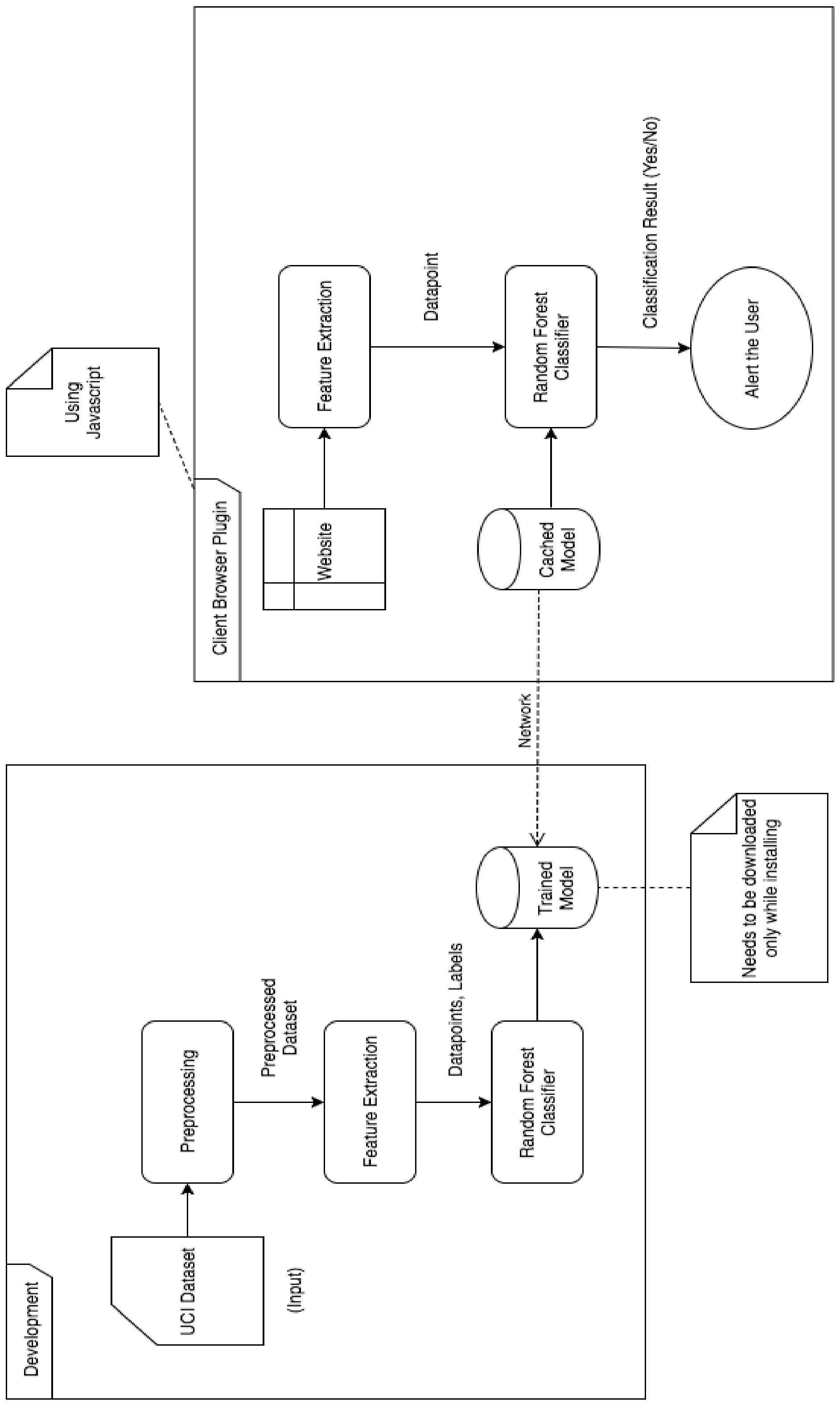


Figure 4.1 System Architecture

- The script may perform the classification using the encoded feature vector and the model JSON. The user is then presented with a warning in the event that the website is identified as phishing. Lightweight design throughout the system ensures quick detection.

- UI DESIGN

- Using HTML and CSS, a straightforward and user-friendly user interface was made for the plugin. A sizable circle on the user interface indicates how much space is occupied by the Active tab. Additionally, the circle's color changes according on classification (green for trustworthy websites, red for phishing).

- The values shown at the bottom of the circle and other analysis results are shown in the following color codes.

- Phishing - Green - Legal Yellow - Suspicious Red
- In order to stop phishing, the plugin additionally displays an alert when it happens. On other screens are test results like as accuracy, recall, and precision. The figure 4.2 displays the user interface.

CLASS DIAGRAM

Figure 4.3 displays the class diagram for the complete machine translation system. The different modules of the system are clearly shown in this diagram. It also illustrates how the system's modules interact with one another, offering a clear implementation concept.

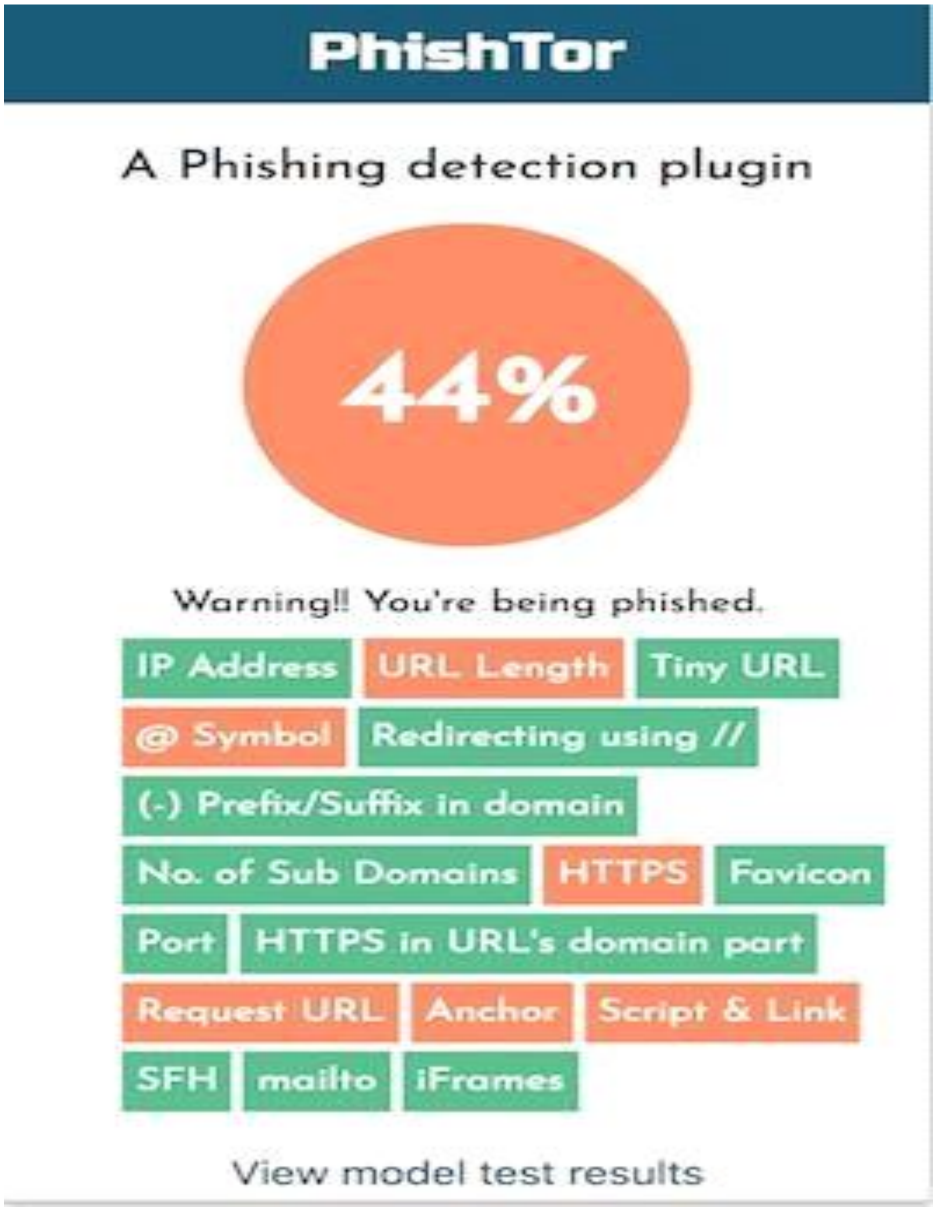

PhishTor
A Phishing detection plugin
44%
Warning!! You're being phished.
IP Address
URL Length
Tiny URL
@ Symbol
Redirecting using //
(-) Prefix/Suffix in domain
No. of Sub Domains
HTTPS
Favicon
Port
HTTPS in URL's domain part
Request URL
Anchor
Script & Link
SFH
mailto
iFrames
View model test results

Figure 4.2 UI Design

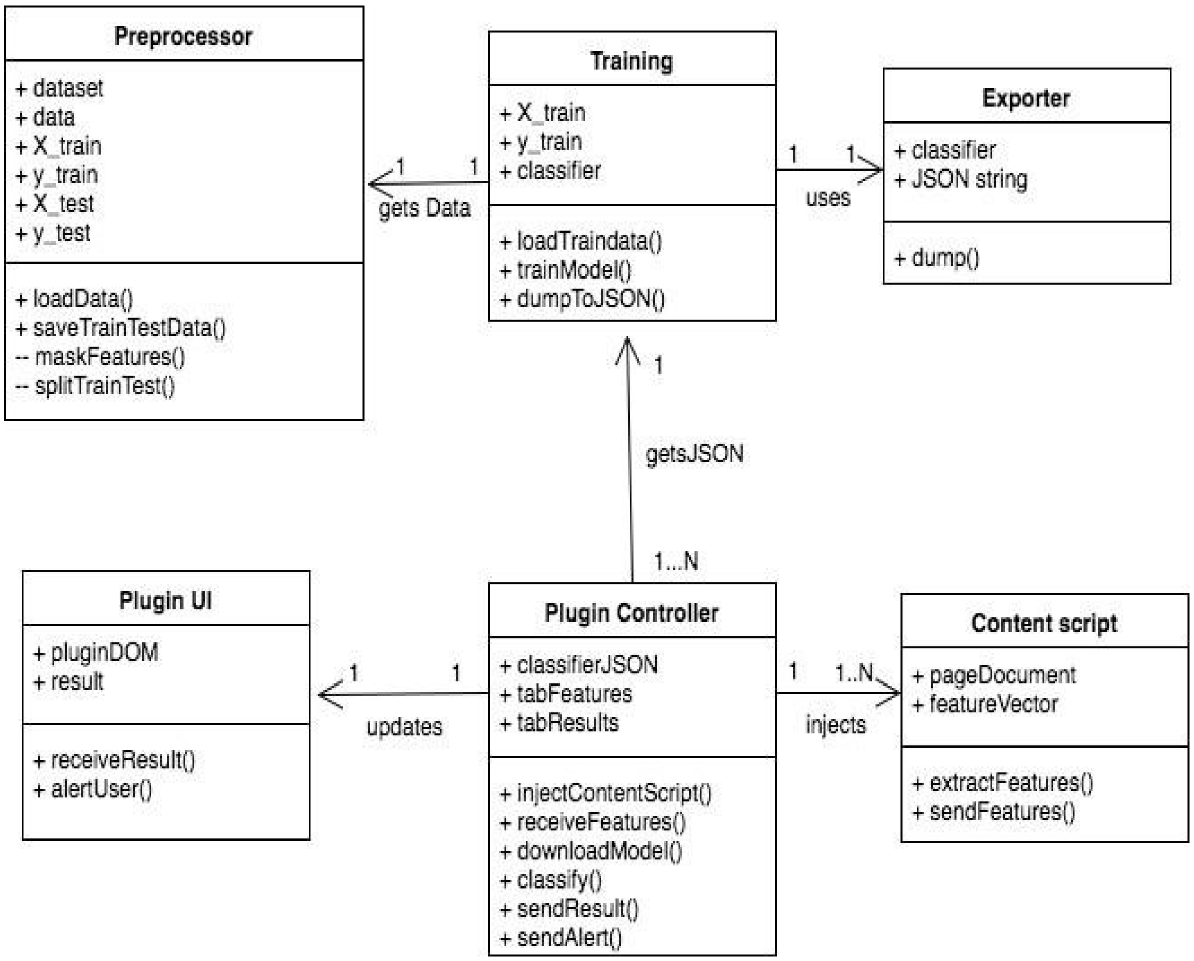


Figure 4.3 Class Diagram

- **MODULE DESIGN**

- **Preprocessing**

After downloading the dataset from the UCI repository, it is loaded into a numpy array. Thirty features make up the dataset; these must be reduced in order for the browser to extract them. Every feature is tested in the browser to ensure that it can be extracted without using any external web service or third party. Out of the thirty features, 17 have been selected based on the trials with little loss in accuracy on the test data. While having more features improves accuracy, having more features also slows down detection speed because of feature extraction time. As a result, feature subsets are chosen in an equitable manner.

IP address	Degree of subdomain	Anchor tag href domains
URL length	HTTPS	Script & link tag domains
URL shortener	Favicon domain	Empty server form handler
@' in URL	TCP Port	Use of mailto
Redirection with '//'	HTTPS in domain name	Use of iFrame
-' in domain	Cross domain requests	

Table 4.1 Webpage Features

- After then, 30% of the dataset is designated for testing and the remaining portion for training. The data used for testing and training is stored on disk.

- **Training**

The preprocessing module loads training data from disk. A random forest distribution can be trained on data by using the scikit-learn library. Using a set of ten decision tree forecasts, Random Forest is an ensemble learning technique. Every decision tree strives to minimize Gini impurity and adheres to the CART algorithm.

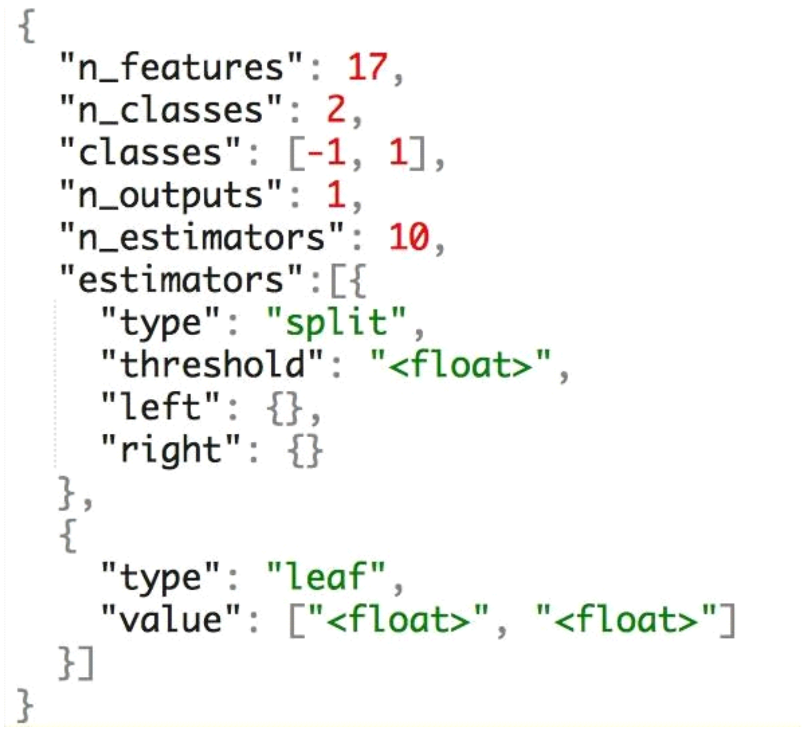
Gini(E) = 1-summation from j=1 to c*(p^2)

Scores from cross-validation are also given as data points. Test results are used to compute F1 scores. Next, export the training model to JSON using the extension model.

Exporting Model

During training, every machine learning algorithm discovers its values. Every decision tree in a random forest is a separate research that learns the starting point and leaf of the learning category outcome. As a result, a format for the random forest in JSON needs to be developed.

The number, class number, etc. of the estimator is contained in every JSON structure. Contains keys such as. Additionally, there is a series



```
{
  "n_features": 17,
  "n_classes": 2,
  "classes": [-1, 1],
  "n_outputs": 1,
  "n_estimators": 10,
  "estimators":[{
    "type": "split",
    "threshold": "<float>",
    "left": {},
    "right": {}
  },
  {
    "type": "leaf",
    "value": ["<float>", "<float>"]
  }]
}
```

Figure 4.4 Random Forest JSON structure

This is a JSON representation of an estimator for each value. Every decision tree is represented as a JSON tree with nested objects that contain the threshold for that node as well as recursively stored left and right node objects.

- **Plugin Feature Extraction**

For every webpage, the aforementioned 17 features must be extracted and encoded in real time while the page loads. It uses a tent script to gain access to the webpage's DOM. Every page has the content script automatically inserted into it as it loads. The features must be gathered by the content script before being sent to the plugin. The primary goal of this effort is to avoid using any external online services. Additionally, features must be fast to extract and unaffected by network delay. When developing methods for feature extraction, all of these are ensured. Once a feature is extracted it is encoded into values {-1, 0, 1} based on the following notation.

-1	Legitimate
0	Suspicious
-1	Phishing

The feature vector containing 17 encoded values is passed on to the plugin from the content script.

- **Classification**

Random forest classification was used to categorize feature vectors that were derived from labels. Get the JSON file for the forest random parameter and save it to a cache. If the cache is empty, the script redownloads the JSON and tries to load it from disk. developed a JavaScript package to compare feature vectors with thresholds in order to generate random forest character nodes using JSON. The page's values determine the binary classification result, which alerts the user whether the website is flagged as phishing.

COMPLEXITY ANALYSIS

- ### Time Complexity

The time complexity of each module of the system is shown in Table 4.2

S.No	Module	Complexity
1	Preprocessing	O(n)
2	Training	O(E * v * nLog(n))
3	Exporting model	O(E * nLog(n))
4	Plugin feature extraction	O(v)
5	Classification	O(E * nLog(n))

Table 4.2 Time Complexity of various modules

- 'n' denotes number of data points.
- 'E' denotes number of ensembles (decision trees).
- 'v' denotes number of features.

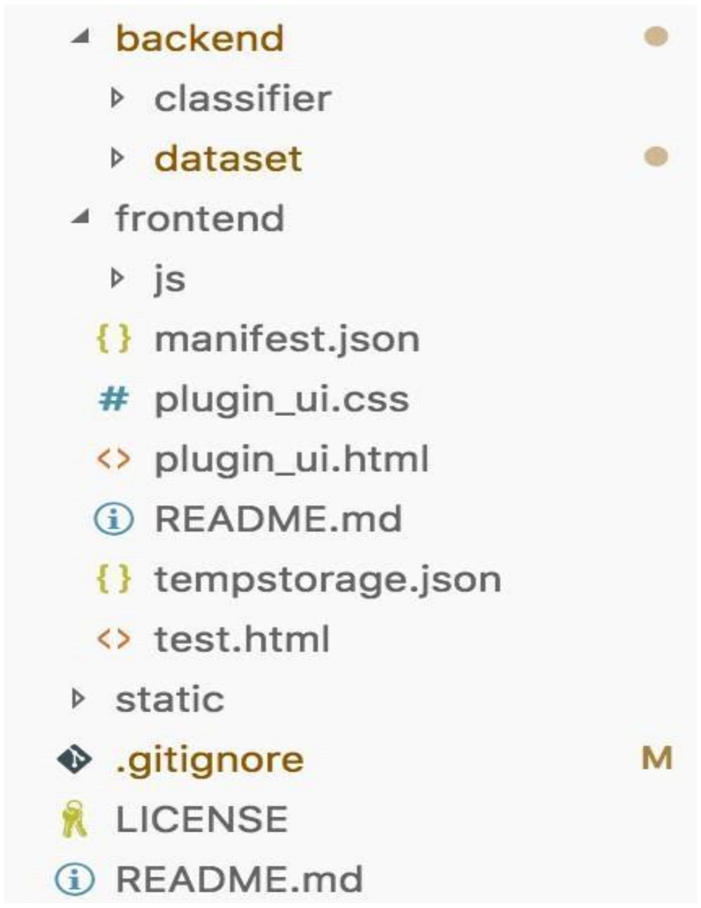
Complexity of the project

• This project's difficulty is finding the right balance between quick detection and accuracy. Accuracy can rise by choosing a subset of features without falling.
Convert Python Scikit-learn components to JavaScript using a format. For example JSON.
• Reproducing random forest behavior in Javascript marginally affects accuracy.
• A lot of things require the use of outside services to be extracted. The use of outside services will impact call duration once more.
Maintaining a high level of security is crucial because the system needs to identify phishing attempts before the user transmits critical data.

CHAPTER 5

SYSTEM DEVELOPMENT

Generally speaking, the system is split into plugins and the backend. Training modules and preconfigured data are contained in the backend. Javascript files for background scripts (such as random forest scripts) and content scripts are found on the plugin's front end. The user interface's HTML and CSS files are also included with the plugin. Figure 5.1 provides a general overview of how these various structures are organized.



```
◢ backend                ●
    ▷ classifier
    ▷ dataset            ●
◢ frontend
    ▷ js
    { } manifest.json
    #  plugin_ui.css
    <> plugin_ui.html
    ⓘ README.md
    { } tempstorage.json
    <> test.html
▷ static
.gitignore               M
LICENSE
ⓘ README.md
```

Figure 5.1 Code Overview

- **PROTOTYPE ACROSS THE MODULES**
- **This section describes the input and output to every module in the system.**

 Preprocessing: This module uses the Arff format of the downloaded dataset.

 format and generates four additional files with the names training features, testing features, and testing class labels.

 Instruction: This section uses the four output files from the previous

 processor and outputs the cross validation score on the training set in addition to a trained Random Forest object.

 Model for exporting: This section uses the acquired Random Forest

 classifier object, and then builds its JSON representation recursively, writing it to a disk file.

 Extraction of Plugin Features: This module accepts a webpage as input.

 and produces a feature vector containing 17 features that have been encoded.

 Classification: The picture vectors are retrieved by this model from

- the Export Model module's JSON format is retrieved by the Feature Extraction module, which then outputs a Boolean message indicating if the page is phishing or authentic.

- **EXPORTING ALGORITHM**

The algorithm used to export Random Forest model as JSON is as follows.

TREE_TO_JSON(NODE):

```
•    tree_json  {}
•    if (node has threshold) then
•        tree_json[“type”]  “split”
•        tree_json[“threshold”]  node.threshold
•        tree_json[“left”]  TREE_TO_JSON(node.left)
•        tree_json[“right”] ← TREE_TO_JSON(node.right)
•    else
•       tree_json[“type”] □ “leaf”
•       tree_json[“values”] □ node.values
•    return tree_json
```

RANDOM_FOREST_TO_JSON(RF):

```
•    forest_json □ {}
•    forest_json['n_features'] □ rf.n_features_
•    forest_json['n_classes'] □ rf.n_classes_
•    forest_json['classes'] □ rf.classes_
•    forest_json['n_outputs'] □ rf.n_outputs_
•    forest_json['n_estimators'] □ rf.n_estimators
•    forest_json['estimators'] □ []
•    e □ rf.estimators
•    for (i □ 0 to rf.n_estimators)
•       forest_json[‘estimators’][i] □ TREE_TO_JSON(e[i])
•    return forest_json
```

• DEPLOYMENT DETAILS

- **Python 3 is needed for the backend, and Github is used to deliver the Classifier JSON and Test set over HTTP. The plugin needs the Chrome browser to function and is supplied as a single file. To be distributed, the frontend plugin is compressed into a crx file.**

CHAPTER 6

RESULTS AND DISCUSSION

- **DATASET FOR TESTING**

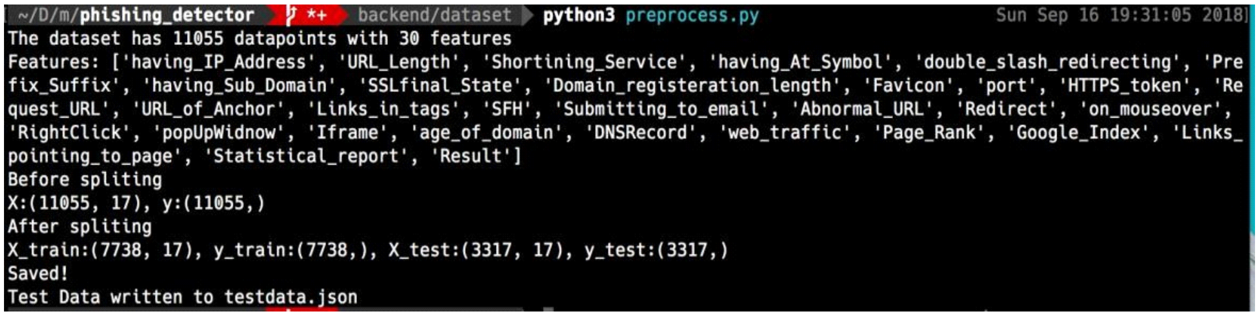
- The data points in the test system are divided in a 70:30 ratio from the data collection. The plugin additionally attempts to use phishTank-listed websites. As soon as new phishing websites are found, they are also uploaded to PhishTank. It's important to note that the plugin has the ability to identify fresh phishing websites. Below is a summary of the evaluation's findings as well as the entire procedure.

- **OUTPUT OBTAINED IN VARIOUS STAGES**

This section shows the results obtained during module testing.

- **Preprocessing**

The output the preprocessing module is shown in figure 6.1.

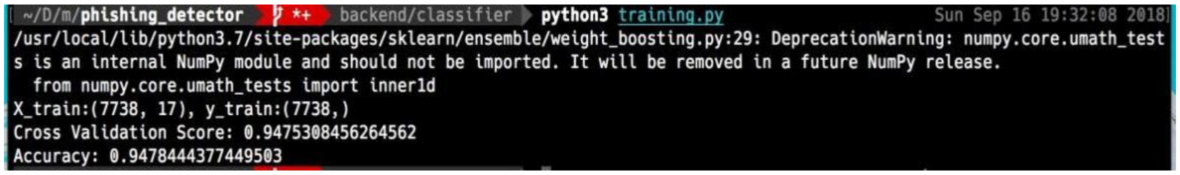


```
~/D/m/phishing_detector  *+  backend/dataset  python3 preprocess.py                    Sun Sep 16 19:31:05 2018]
The dataset has 11055 datapoints with 30 features
Features: ['having_IP_Address', 'URL_Length', 'Shortining_Service', 'having_At_Symbol', 'double_slash_redirecting', 'Pre
fix_Suffix', 'having_Sub_Domain', 'SSLfinal_State', 'Domain_registeration_length', 'Favicon', 'port', 'HTTPS_token', 'Re
quest_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
```

Figure 6.1 Preprocessing output

- **Training**

The output the training module is shown in figure 6.2.

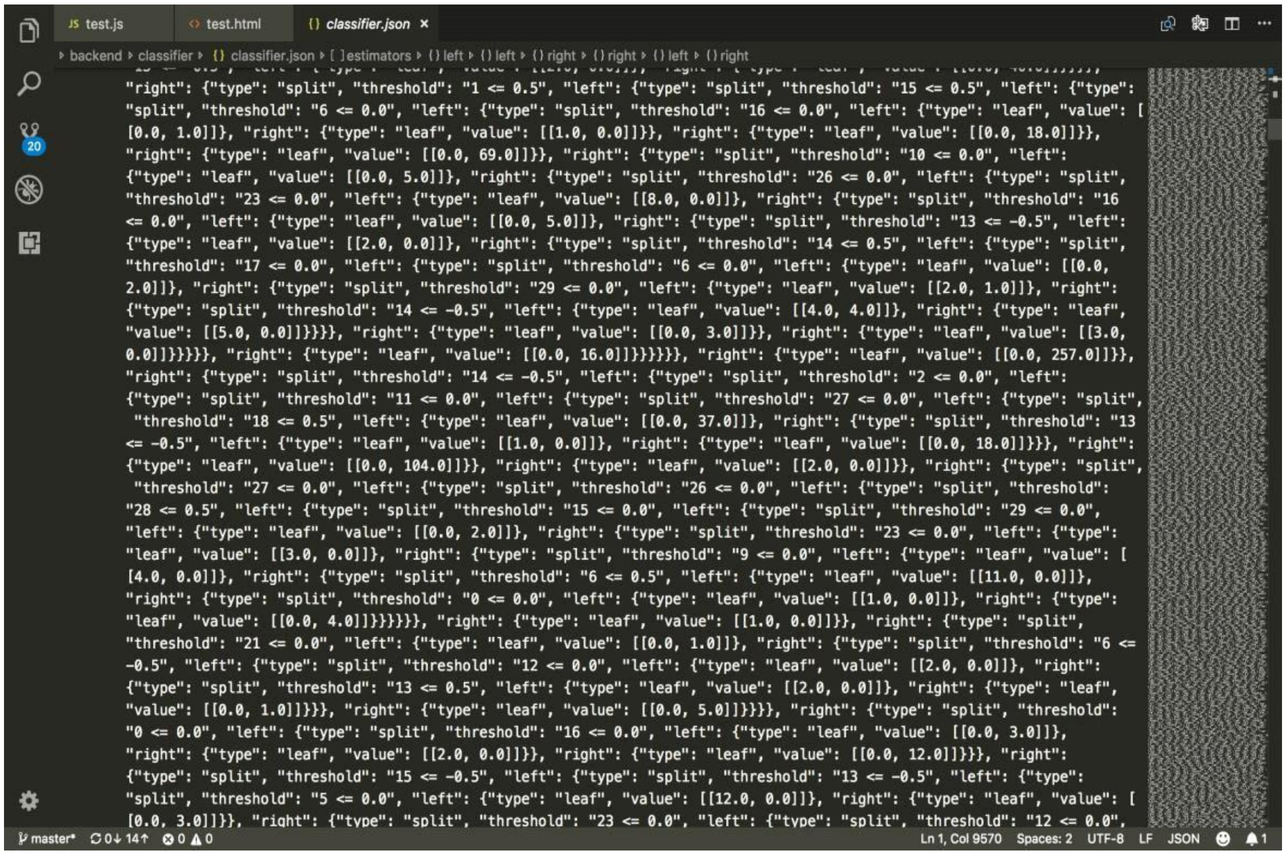


```
[ ~/D/m/phishing_detector  ✗ *+  backend/classifier  python3 training.py                Sun Sep 16 19:32:08 2018]
/usr/local/lib/python3.7/site-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarning: numpy.core.umath_test
s is an internal NumPy module and should not be imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
X_train:(7738, 17), y_train:(7738,)
Cross Validation Score: 0.9475308456264562
Accuracy: 0.9478444377449503
```

Figure 6.2 Training output

• Exporting model

The output the export module is shown in figure 6.3. It outputs a JSON file representing the Random Forest parameters.

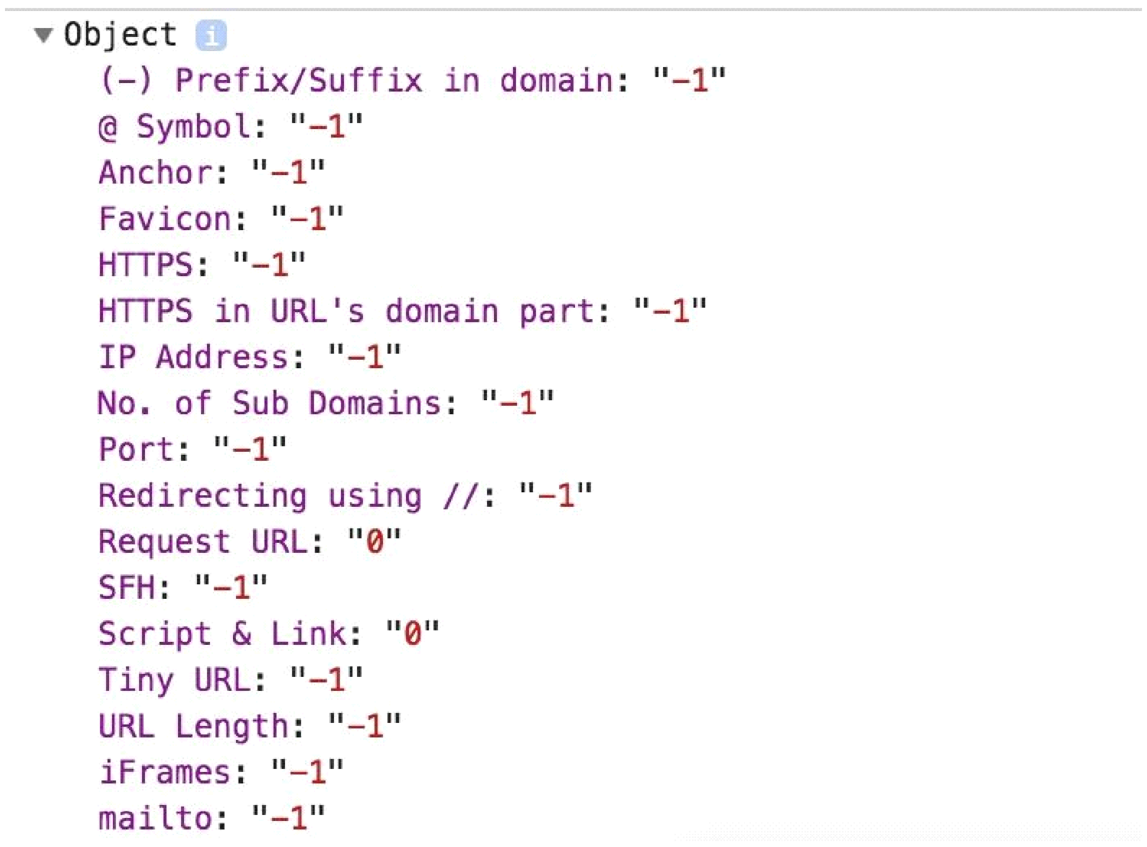

test.js
test.html
classifier.json
backend › classifier › classifier.json › estimators › left › left › right › right › left › right
master*
0↓ 14↑
0 0
Ln 1, Col 9570 Spaces: 2 UTF-8 LF JSON 1

```
"right": {"type": "split", "threshold": "1 <= 0.5", "left": {"type": "split", "threshold": "15 <= 0.5", "left": {"type":
"split", "threshold": "6 <= 0.0", "left": {"type": "split", "threshold": "16 <= 0.0", "left": {"type": "leaf", "value": [
[0.0, 1.0]]}, "right": {"type": "leaf", "value": [[1.0, 0.0]]}}, "right": {"type": "leaf", "value": [[0.0, 18.0]]}},
"right": {"type": "leaf", "value": [[0.0, 69.0]]}}, "right": {"type": "split", "threshold": "10 <= 0.0", "left":
{"type": "leaf", "value": [[0.0, 5.0]]}, "right": {"type": "split", "threshold": "26 <= 0.0", "left": {"type": "split",
"threshold": "23 <= 0.0", "left": {"type": "leaf", "value": [[8.0, 0.0]]}, "right": {"type": "split", "threshold": "16
<= 0.0", "left": {"type": "leaf", "value": [[0.0, 5.0]]}, "right": {"type": "split", "threshold": "13 <= -0.5", "left":
{"type": "leaf", "value": [[2.0, 0.0]]}, "right": {"type": "split", "threshold": "14 <= 0.5", "left": {"type": "split",
"threshold": "17 <= 0.0", "left": {"type": "split", "threshold": "6 <= 0.0", "left": {"type": "leaf", "value": [[0.0,
2.0]]}, "right": {"type": "split", "threshold": "29 <= 0.0", "left": {"type": "leaf", "value": [[2.0, 1.0]]}, "right":
{"type": "split", "threshold": "14 <= -0.5", "left": {"type": "leaf", "value": [[4.0, 4.0]]}, "right": {"type": "leaf",
"value": [[5.0, 0.0]]}}}}, "right": {"type": "leaf", "value": [[0.0, 3.0]]}}, "right": {"type": "leaf", "value": [[3.0,
0.0]]}}}}}, "right": {"type": "leaf", "value": [[0.0, 16.0]]}}}}}}, "right": {"type": "leaf", "value": [[0.0, 257.0]]}},
"right": {"type": "split", "threshold": "14 <= -0.5", "left": {"type": "split", "threshold": "2 <= 0.0", "left":
{"type": "split", "threshold": "11 <= 0.0", "left": {"type": "split", "threshold": "27 <= 0.0", "left": {"type": "split",
 "threshold": "18 <= 0.5", "left": {"type": "leaf", "value": [[0.0, 37.0]]}, "right": {"type": "split", "threshold": "13
<= -0.5", "left": {"type": "leaf", "value": [[1.0, 0.0]]}, "right": {"type": "leaf", "value": [[0.0, 18.0]]}}}, "right":
{"type": "leaf", "value": [[0.0, 104.0]]}}, "right": {"type": "leaf", "value": [[2.0, 0.0]]}}, "right": {"type": "split",
 "threshold": "27 <= 0.0", "left": {"type": "split", "threshold": "26 <= 0.0", "left": {"type": "split", "threshold":
"28 <= 0.5", "left": {"type": "split", "threshold": "15 <= 0.0", "left": {"type": "split", "threshold": "29 <= 0.0",
"left": {"type": "leaf", "value": [[0.0, 2.0]]}, "right": {"type": "split", "threshold": "23 <= 0.0", "left": {"type":
"leaf", "value": [[3.0, 0.0]]}, "right": {"type": "split", "threshold": "9 <= 0.0", "left": {"type": "leaf", "value": [
[4.0, 0.0]]}, "right": {"type": "split", "threshold": "6 <= 0.5", "left": {"type": "leaf", "value": [[11.0, 0.0]]},
"right": {"type": "split", "threshold": "0 <= 0.0", "left": {"type": "leaf", "value": [[1.0, 0.0]]}, "right": {"type":
"leaf", "value": [[0.0, 4.0]]}}}}}}, "right": {"type": "leaf", "value": [[1.0, 0.0]]}}, "right": {"type": "split",
"threshold": "21 <= 0.0", "left": {"type": "leaf", "value": [[0.0, 1.0]]}, "right": {"type": "split", "threshold": "6 <=
-0.5", "left": {"type": "split", "threshold": "12 <= 0.0", "left": {"type": "leaf", "value": [[2.0, 0.0]]}, "right":
{"type": "split", "threshold": "13 <= 0.5", "left": {"type": "leaf", "value": [[2.0, 0.0]]}, "right": {"type": "leaf",
"value": [[0.0, 1.0]]}}}, "right": {"type": "leaf", "value": [[0.0, 5.0]]}}}}, "right": {"type": "split", "threshold":
"0 <= 0.0", "left": {"type": "split", "threshold": "16 <= 0.0", "left": {"type": "leaf", "value": [[0.0, 3.0]]},
"right": {"type": "leaf", "value": [[2.0, 0.0]]}}, "right": {"type": "leaf", "value": [[0.0, 12.0]]}}}, "right":
{"type": "split", "threshold": "15 <= -0.5", "left": {"type": "split", "threshold": "13 <= -0.5", "left": {"type":
"split", "threshold": "5 <= 0.0", "left": {"type": "leaf", "value": [[12.0, 0.0]]}, "right": {"type": "leaf", "value": [
[0.0, 3.0]]}}, "right": {"type": "split", "threshold": "23 <= 0.0", "left": {"type": "split", "threshold": "12 <= 0.0",
```

Figure 6.3 Model JSON

- ### Plugin Feature Extraction

The 17 features extracted for the webpage at thetechcache.science are logged in to the console which is shown in figure 6.4. The features are stored as key value pairs and the values are encoded from -1 to 1 as dis- cussed above.



```
▼ 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"
```

Figure 6.4 Webpage features

- ### Classification

The Plugin UI displays the classification's outcome directly. A green circle denotes a trustworthy website, while light red signifies phishing.

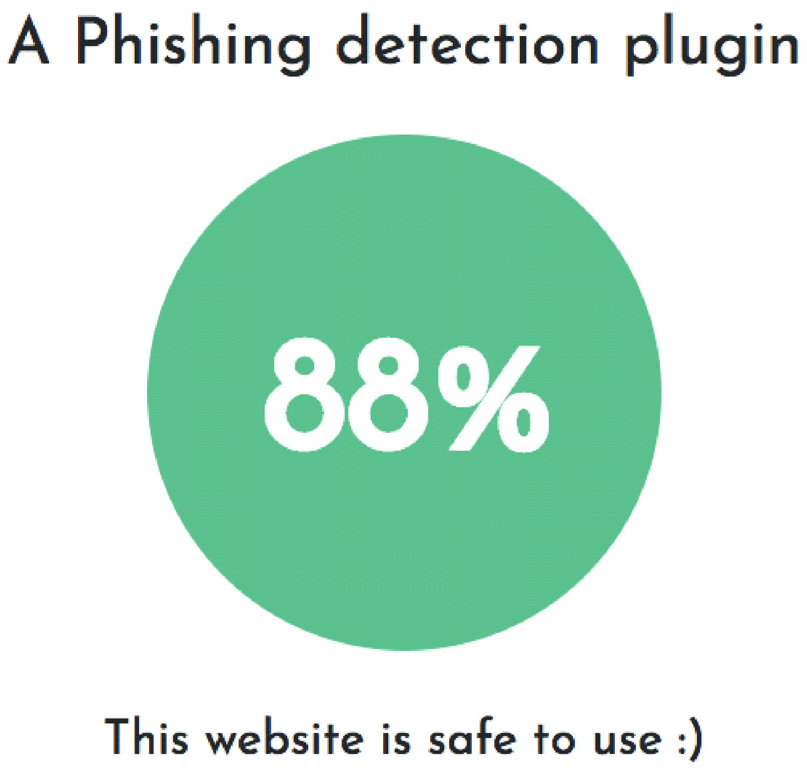


Figure 6.5 Classification Output

- **SAMPLE SCREENSHOTS DURING TESTING**

The output of the plugin while visiting a phishing site collected from PhishTank. In addition to having a low trust ranking, this website indicates phishing with its light red circle.

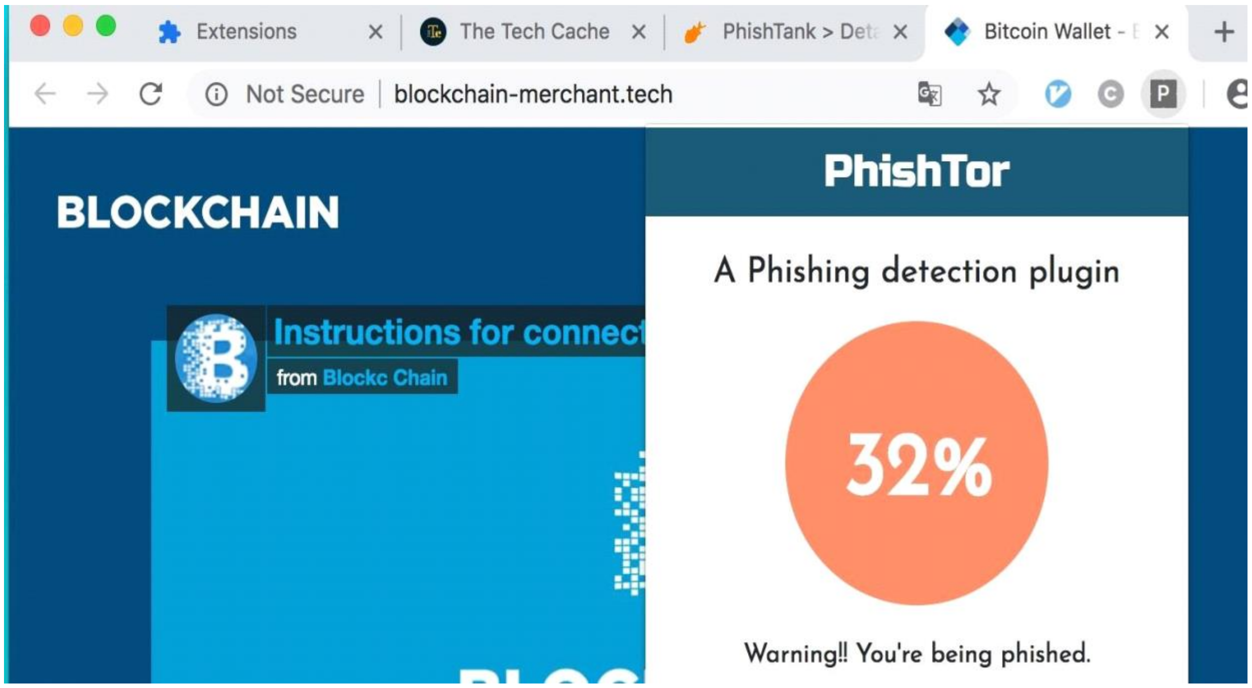


Figure 6.6 Test Output

- **PERFORMANCE EVALUATION**
- **The standard parameters listed below are used to assess the system's overall performance.**
- **Cross Validation score**

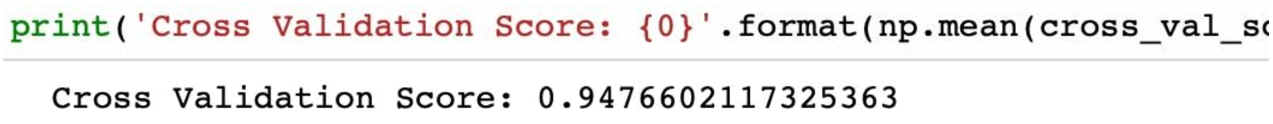
It is incorrect to discover why a prediction failed and then retest it using the same set of data: Reiterating the label of the discovered model will yield a perfect score but prevent it from making any predictions. Not yet located information We refer to this state as overfitting.

An additional portion of the data set can be referred to as a "validation set" in order to address this issue. Training is conducted on the training set, evaluation is conducted on the validation procedure, and if the experiment is deemed successful upon completion, a final evaluation can be conducted on the experiment. However, we decrease the amount of samples available to learn the model by splitting the given data into three groups.

and the outcome will be determined by the particular random selection of the pair group for training and validation.

Cross-validation, or CV for short, is the technique that solves this issue. The existing approach is no longer necessary when preparing the CV, however the test procedure should be kept for the final assessment. The training set is broken into k smaller pieces using the straightforward k-fold CV method (additional approaches are given below, although most use the same language of core ideas). The process that follows is carried out for every k "folds":

The design is verified against the remaining data once the model has been trained using k folds of training data (i.e., used as a test to generate performance measures such as accuracy). The 10x cross validation score is as follows



```
print('Cross Validation Score: {0}'.format(np.mean(cross_val_s

  Cross Validation Score: 0.9476602117325363
```

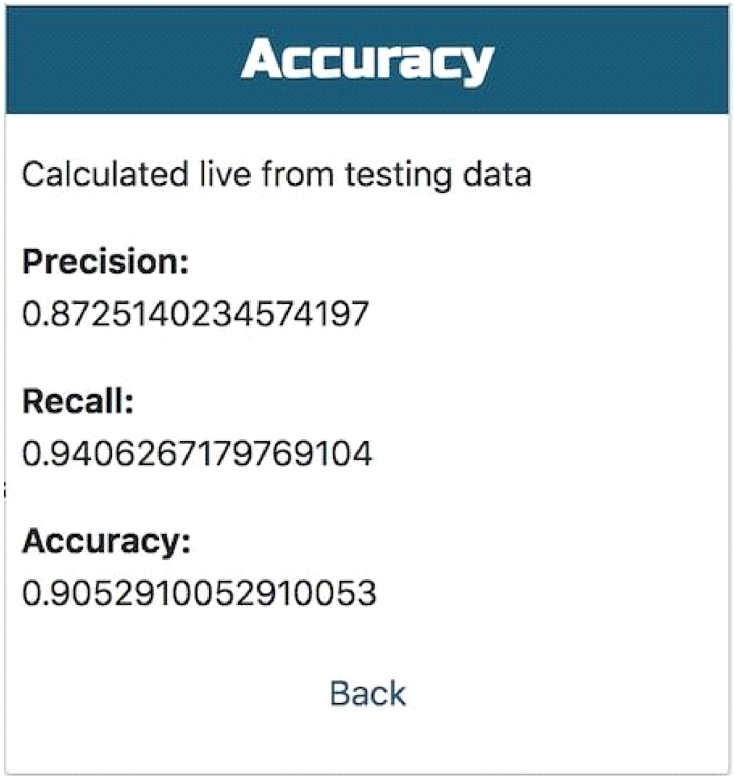
Figure 6.7 Cross Validation Output (calculated with cross_val_score of scikit-learn)

- **F1 score**

  The test's correctness is gauged by the F1 score. The test's accuracy and return are calculated using the score: accuracy is the number of results divided by the total number of positive results the distribution returned, and return is the number of positive results divided by the total number of positive results the distribution returned. samples that are pertinent (all samples that ought to be classified as positive). The F1 score is a trade-off between recall and precision, with a maximum value of 1 (very good recall and precision) and a minimum value of 0. < br>The weight between real and real is the definition of the F1 score. Conversely, when the F1 score hits the optimal value (1) and Zero is the lowest possible score. Precision and return both contribute equally to the F1 score. The following is the formula for the F1 score.:

F1 = 2 * (precision * recall) / (precision + recall)

Javascript is used to manually calculate the phishing classifier's precision, recall, and F1 score using the test data set. The outcomes are displayed in figure 6.8.



Accuracy

Calculated live from testing data

Precision:
0.8725140234574197

Recall:
0.9406267179769104

Accuracy:
0.9052910052910053

Figure 6.8 Performance measure

CHAPTER-7
CONCLUSIONS

- ## SUMMARY

The goal of this phishing website detection system is to quickly identify phishing websites so that consumers can be alerted before falling victim to them. The primary purpose is to obtain JavaScript's random forest distribution. Similar processes frequently do network-based discovery and use web pages that cannot be retrieved from the client. However, this system might offer faster detection and better privacy because it employs features that are exclusive to the customer. Reducing the number of characteristics improves the system's usability even though it could marginally decrease accuracy. Without sacrificing precision, this function specifies the collection of web page functions that are offered to clients.

- Fast searching is further aided by the personal random forest utilized in javascript and the conversion from Python to JavaScript. The temporal complexity is taken into consideration when creating the standard JSON representation and classification of text. Phishing can be identified by the plugin even before the page loads completely. The client's exam resulted in an F1 score of 0.886.

CRITICISMS

- Although the system's accuracy is not as good as that of cutting-edge systems, it is nonetheless very useable and strikes a good compromise between speed and accuracy. Add-ons are not significantly impacted by the Chrome Extension API. Unfortunately, the plugin is unable to stop harmful JavaScript code because these methods are pulled from the script during page load.

- Furthermore, there is a need for improvement as the accuracy decreases from 0.94 to 0.886 while switching from Python to JavaScript.Only JavaScript is supported by the browser;

multiple threads are not supported by Javascript. As a result, splitting cannot be completed more quickly using parallel threads. Even during peak website traffic, results are now counted twice without being cached by the plugin.

FUTURE WORK

17 features that may be added as long as they didn't slow down or compromise privacy were covered in the existing class. This add-on makes it possible to minimize computation for frequently viewed websites. This, however, may make pharming attacks go unnoticed. It is necessary to provide a solution that can cache the results without sacrificing its capacity to spot fraud. WorkerThreads can be used to partition Javascript, which can speed up the partitioning process. As a result, the system is capable of numerous enhancements to increase its efficacy in phishing detection.

REFERENCES

- "UCI Machine Learning Repository: Data Set of Phishing Websites," [Online]. Phishing websites are accessible at https://archive.ics.uci.edu/ml/datasets/.

- PhishBox: An Approach for Phishing Validation and Detection, J.-H. Li and S.-D. Wang, 2017 IEEE 15th International Conference on Secure, Autonomous, and Reliable Computing, DASC/PiCom/DataCom/CyberSciTech, the 15th International Conference on Pervasive Intelligence and Computing, the 3rd International Conference on Big Data Intelligence and Computing, and the Cyber Science and Technology Congress

- "Real time detection of phishing websites," 7th IEEE Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2016, A. A. Ahmed and N. A. Abdullah. R. Aravindhan, R. Shanmugalakshmi, K. Ramya, and S. C., "Cer- tain investigation on web application security: Phishing detection and phishing target discovery," *2016 3rd International Conference on Advanced Computing and Communication Systems* (ICACCS), 2016.

- S. B. K.p., "Using Neural Networks and Firefly for Phishing Detection in Websites," International Journal of Engineering and Computer Science, 2016.

- "A Supervised Machine Learning Technique for Efficient Website Phishing Detection Approaches," International Journal of Advance Engineering and Research Development, vol. 2, no. 05, 2015.

- "Artifical neural network with PSO for Phishing URL detection," 2nd International Conference on Telecommunication and Networks (TEL-NET), 2017; S. Gupta and A. Singhal.

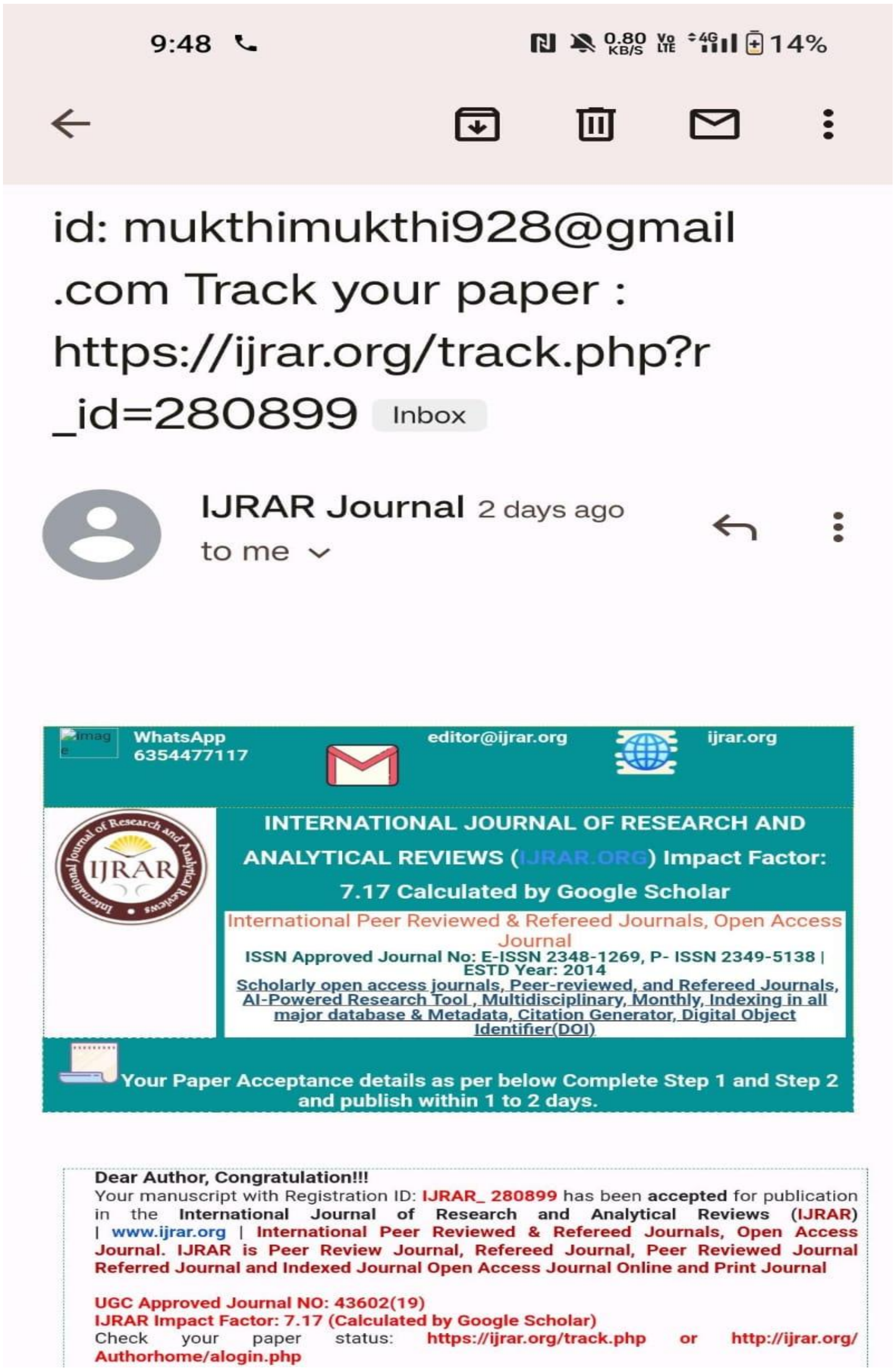
• Phishing Dynamic Evolving Neural Fuzzy Framework for..., Ammar ALmomani, G. B. B, Tat-Chee Wan, Altyeb Altaher, and Selvakumar Manickam, Jan-2013. [Online]. The link https://arxiv.org/pdf/1302.0629 is available.

• "Intelligent phishing website detection using random forest classifier," A. Subasi, E. Molah, F. Almkallawi, and T. J. Chaudhery, 2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA), Nov. 2017.

APPENDIX-A

PSUEDOCODE

https://github.com/SHAMBHVISAMRIDHI110/Phishing-Detection/tree/main



9:48

0.80 KB/S 14%

id: mukthimukthi928@gmail .com Track your paper : https://ijrar.org/track.php?r _id=280899 Inbox

IJRAR Journal 2 days ago
to me

WhatsApp 6354477117

editor@ijrar.org

ijrar.org

INTERNATIONAL JOURNAL OF RESEARCH AND ANALYTICAL REVIEWS (IJRAR.ORG) Impact Factor: 7.17 Calculated by Google Scholar

International Peer Reviewed & Refereed Journals, Open Access Journal

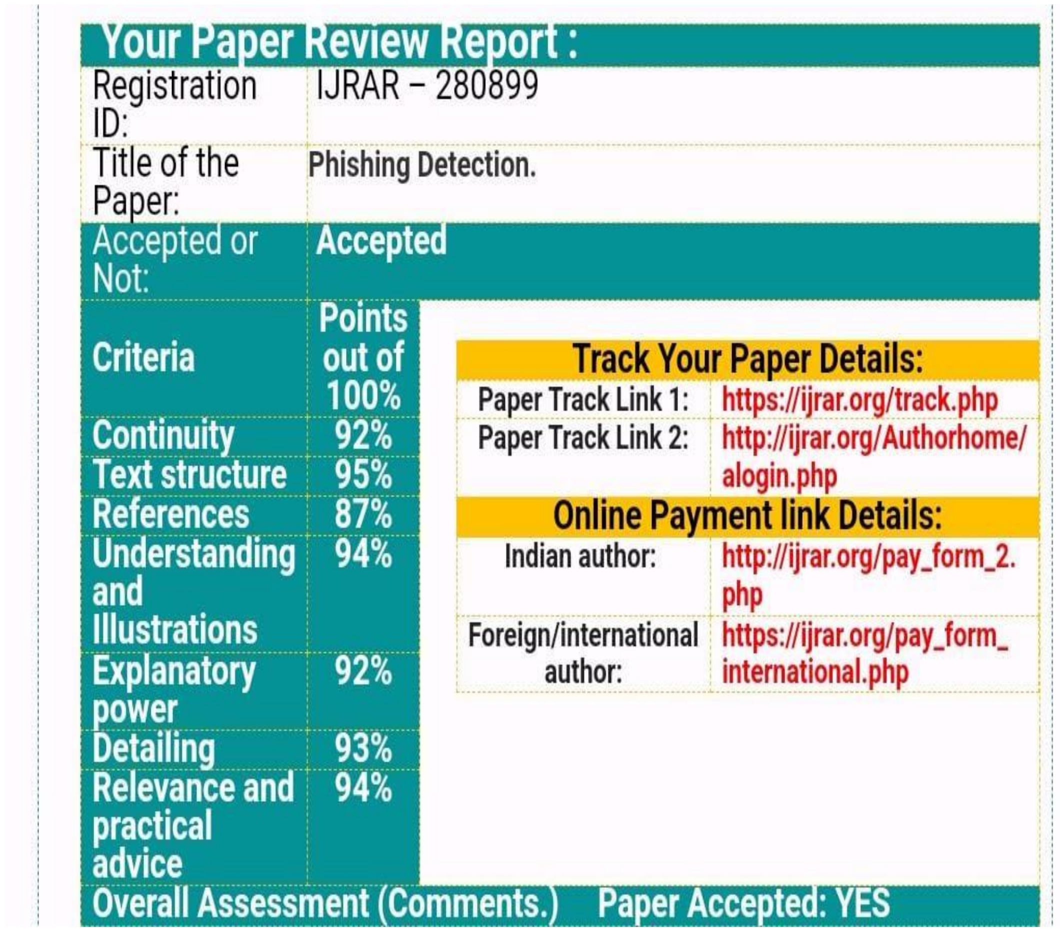
ISSN Approved Journal No: E-ISSN 2348-1269, P- ISSN 2349-5138 | ESTD Year: 2014

Scholarly open access journals, Peer-reviewed, and Refereed Journals, AI-Powered Research Tool , Multidisciplinary, Monthly, Indexing in all major database & Metadata, Citation Generator, Digital Object Identifier(DOI)

Your Paper Acceptance details as per below Complete Step 1 and Step 2 and publish within 1 to 2 days.

Dear Author, Congratulation!!!
Your manuscript with Registration ID: **IJRAR_ 280899** has been **accepted** for publication in the **International Journal of Research and Analytical Reviews (IJRAR)** | **www.ijrar.org** | **International Peer Reviewed & Refereed Journals, Open Access Journal. IJRAR is Peer Review Journal, Refereed Journal, Peer Reviewed Journal Referred Journal and Indexed Journal Open Access Journal Online and Print Journal**

UGC Approved Journal NO: 43602(19)
IJRAR Impact Factor: 7.17 (Calculated by Google Scholar)
Check your paper status: **https://ijrar.org/track.php or http://ijrar.org/Authorhome/alogin.php**



Your Paper Review Report :

Registration ID:	IJRAR – 280899
Title of the Paper:	**Phishing Detection.**
Accepted or Not:	**Accepted**

Criteria	Points out of 100%
Continuity	92%
Text structure	95%
References	87%
Understanding and Illustrations	94%
Explanatory power	92%
Detailing	93%
Relevance and practical advice	94%

Track Your Paper Details:

Paper Track Link 1:	https://ijrar.org/track.php
Paper Track Link 2:	http://ijrar.org/Authorhome/alogin.php

Online Payment link Details:

Indian author:	http://ijrar.org/pay_form_2.php
Foreign/international author:	https://ijrar.org/pay_form_international.php

Overall Assessment (Comments.) **Paper Accepted: YES**

VISHAL SRIVASTAVA

ORIGINALITY REPORT

14%	9%	7%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	wwww.easychair.org Internet Source	4%
2	Senthilkumar Subramanian, Nithya Venkatachalam, Regan Rajendran. "chapter 3 A Novel Phishing Attack Prediction Model With Crowdsouring in Wireless Networks", IGI Global, 2023 Publication	3%
3	fcc08321-8158-469b-b54d-f591e0bd3df4.usrfiles.com Internet Source	2%
4	www.ijert.org Internet Source	2%
5	Submitted to Universiti Sains Islam Malaysia Student Paper	2%
6	Submitted to City University Student Paper	1%
7	Submitted to University of Southampton Student Paper	<1%

8	Submitted to University Of Tasmania Student Paper	<1 %
9	R T F CHEUNG. "MRI in vertebral artery dissection", Journal of Neurology Neurosurgery & Psychiatry, 05/01/1999 Publication	<1 %
10	Submitted to The University of the West of Scotland Student Paper	<1 %
11	www.ncbi.nlm.nih.gov Internet Source	<1 %
12	www.netlearn.ca Internet Source	<1 %
13	vdoc.pub Internet Source	<1 %
14	Joseph S. B. Mitchell. "Finding optimal bipartitions of points and polygons", Lecture Notes in Computer Science, 1991 Publication	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

SDG

Sustainable Development Goal 16 (**SDG 16** or **Global Goal 16**) is one of the 17 Sustainable Development Goals established by the United Nations in 2015, the official wording is: "Promote peaceful and inclusive societies for sustainable development, provide access to justice for all and build effective, accountable and inclusive institutions at all levels".[1] The Goal has 12 targets and 23 indicators.

SDG 16 has ten outcome targets: Reduce violence; protect children from abuse, exploitation, trafficking and violence; promote the rule of law and ensure equal access to justice; combat organized crime and illicit financial and arms flows, substantially reduce corruption and bribery; develop effective, accountable and transparent institutions; ensure responsive, inclusive and representative decision-making; strengthen the participation in global governance; provide universal legal identity; ensure public access to information and protect fundamental freedoms. There are also two *means of implementation targets*[2]: Strengthen national institutions to prevent violence and combat crime and terrorism; promote and enforce non-discriminatory laws and policies.