

# Phishing Detection

## A PROJECT REPORT

*Submitted by,*

SHAMBHVI SAMRIDHI - 20201CSE0495

MUKTHI -20201CSE0480 MANYTHA RAJ- 20201CSE0472

PRANAV BHATT -20201CSE048 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**



**PRESIDENCY UNIVERSITY BENGALURU JANUARY 2024**

*PRESIDENCY UNIVERSITY*

**SCHOOL OF COMPUTER SCIENCE ENGINEERING &  
INFORMATION SCIENCE**

**CERTIFICATE**

This is to certify that the Project report “**Phishing Detection**” being submitted by SHAMBHVI SAMRIDHI - 20201CSE0495  
MUKTHI -20201CSE0480 MANYTHA RAJ- 20201CSE0472  
PRANAV BHATT -20201CSE048 VISHAL SRIVASTAVA -  
20201CSE0496

in partial fulfilment 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. SREELATHA P.K**

Asst. Professor

School of CSE&IS Presidency University

**Dr. Pallavi R**

Prof. & HoD

School of CSE&IS Presidency University

<b>Dr. C. KALAIARASAN</b> Associate Dean School of CSE&IS Presidency University	<b>Dr. SHAKKEERA L</b> Associate Dean School of CSE&IS Presidency University	<b>Dr. SAMEERUDDIN KHAN</b> Dean School of CSE&IS Presidency University
--	---	--

# **SCHOOL OF COMPUTER SCIENCE ENGINEERING & INFORMATION SCIENCE**

## **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 P.K**

Professor

Presidency University **School of Computer Science Engineering & Information Science, Presidency University, Bengaluru.**

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

## **ACKNOWLEDGEMENT**

First of all, we indebted to the for giving me an opportunity to excel in our efforts to complete this project on time.

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

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

We would like to convey our gratitude and thanks to the University Project-II Coordinators **Dr. Sanjeev P Kaulgud, Dr. Mrutyunjaya MS** and also the department Project Coordinators.

We are greatly indebted to our guide **Ms. Sreelatha P.K, Professor**, School of Computer Science Engineering & Information Science, Presidency University for her inspirational guidance, valuable suggestions and providing us a chance to express our technical capabilities in every respect for the completion of the project work.

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

SHAMBHVI SAMRIDHI MUKTHI K MANYTHA RAJ PRANAV BHATT VISHAL  
SRIVASTAVA

## **ABSTRACT**

This document proposes a phishing<sup>1</sup> 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 classifier<sup>2</sup>, 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<sup>3</sup> using python scikit-learn and then the learned model parameters need to be exported into a portable format for using in javascript.

<sup>1</sup> Phishing is mimicking a creditable company's website to take private information of a user.

<sup>2</sup> <https://ieeexplore.ieee.org/abstract/document/8252051/>

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

## TABLE OF CONTENTS

Figure 2.1 Approaches to phishing detection.....	6
Figure 3.1 Use case diagram of the system.....	12
Figure 3.2 System Sequence diagram.....	13
Figure 4.1 System Architecture.....	15
Figure 4.2 UI Design.....	17
Figure 4.3 Class Diagram .....	17
Figure 4.4 Random Forest JSON structure .....	19
Figure 5.1 Code Overview .....	23
Figure 6.1 Preprocessing output .....	26
Figure 6.2 Training output .....	27
Figure 6.3 Model JSON .....	27
Figure 6.4 Webpage features.....	28
Figure 6.5 Classification Output .....	28
Figure 6.6 Test Output.....	29
Figure 6.7 Cross Validation Output.....	30

Figure 6.8 Performance measure .....	31
--------------------------------------	----

Table 1.1 SWOT analysis.....	3
Table 4.1 Webpage Features.....	18
Table 4.2 Time Complexity of various modules.....	21

URL	Uniform Resource Locator
API	Application Programming Interface
HTTP	Hyper Text Transfer Protocol
SSL	Secured Socket Layer
DNS	Domain Name System
GDPR	General Data Privacy Regulation
JSON	JavaScript Object Notation
DOM	Document Object Model
UI	User Interface
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheet



# **CHAPTER 1**

## **INTRODUCTI**

### **ON**

- **PROBLEM DOMAIN**

Phishing is a fraudulent attempt to obtain sensitive information such as usernames, passwords and credit card details (and money), often for malicious purposes. It is usually done through email spoofing or instant messaging and often allows users to access personal information on a fake website that looks and feels like a legitimate

website. The only difference is the targeted website URL. Requests from chat rooms, merchants, banks and online payment systems are often used to lure victims. Phishing emails may contain links to websites that distribute malware. Identifying phishing websites often involves looking at a list of malicious websites. Since most phishing sites are short, the list cannot be tracked everywhere, including new phishing sites. Therefore, machine learning can better solve the problem of detecting phishing websites. Based on the comparison of different learning machines, it seems that the random forest classifier performs better. The only way end users can benefit from this is by searching in the browser add-on. So users can receive instant alerts when checking phishing websites. However, browser extensions have some limitations, such as being written only in javascript and having limited access to page URLs and resources. Current plugins pass the URL to the server, so deployment can be done on the server and results are returned to the plugin. This approach may raise questions about user privacy, discovery may be delayed due to network outages, and the plugin may not notify users in a timely manner. Since this is a major security issue and keeping privacy in mind, we decided to use this Chrome browser extension that allows sharing without third-party food.

- **PROBLEM DESCRIPTION**

Creating a browser add-on that, once installed, will warn users when they visit a phishing website. The plugin must not call other web services that may share the user's browsing data for this purpose. Detection must be made immediately to warn users before they access important information on the phishing website.

- **SCOPE**

According to Wikipedia, 76% of organizations experienced phishing attacks in 2017. Almost half of security professionals surveyed reported an increase in attacks compared to 2016. In the first half of 2017, businesses and residents in Qatar were subjected to more than 93,570 phishing incidents. within three months. As the number of internet users increases, the need for security solutions against attacks such as phishing also increases. Therefore, this extension is a useful tool for Chrome users.

- **CONTRIBUTION**

This is the first web browser phishing to be used in a browser plug-in without the need for an external service. This takes existing efforts in phishing detection and uses them in a way that benefits the end user. This involves porting an existing python classifier (random forest) to javascript. The plugin downloads the learning model in one go and will be able to deploy the website instantly. This includes generating such patterns (random forest) in javascript as the browser plugin only supports javascript. Therefore, this project will help retain information better and detect phishing quickly.

## • **SWOT ANALYSIS**

<b>STRENGTHS</b>	<b>WEAKNESSES</b>
<ul style="list-style-type: none"> <li>• Enables user privacy.</li> <li>• Rapid detection of phishing.</li> <li>• Can detect new phishing sites too.</li> <li>• Can interrupt the user incase of phishing.</li> </ul>	<ul style="list-style-type: none"> <li>• Javascript limits functionality.</li> <li>• Cannot use features that needs a external service such as SSL, DNS, page ranks.</li> <li>• No library support.</li> </ul>
<b>OPPORTUNITIES</b>	<b>THREATS</b>
<ul style="list-style-type: none"> <li>• Everyone conscious of privacy and security can use this plugin.</li> <li>• Non technical people who do business transactions are vulnerable to phishing and they are potential end users for this.</li> </ul>	<ul style="list-style-type: none"> <li>• Server side classification plugins may perform better than this and users without privacy concerns may opt of those.</li> <li>• Chrome Plugin API will be continuously changed.</li> </ul>

**Table 1.1** SWOT analysis

## • **PESTLE ANALYSIS**

### • **Political**

The project is rarely controlled by politics. One condition is that the government takes measures to prevent phishing, in which case the plugin will lose its ability.

### • **Economical**

The plugin is made as a public document and opens the Chrome plugin API. Therefore it is not controlled by economic factors.

- **Social**

The social factor that will have effect on this plugin is awareness of the user. Phishing detection systems aims to aid an user in finding a phishing sites. At least the users need to be aware about phishing to in- stall this plugin.

- **Technological**

Technological advancements or improved techniques for phishing detection can make this plugin become outdated and are a major threat to this plugin.

- **Legal**

Legal policies those in concern of user privacy such as GDPR will enhance the potential of this plugin as user will be moving to more pri- vacy based products.

- **Environmental**

This plugin has no environmental factors affecting it.

- **ORGANISATION OF THESIS**

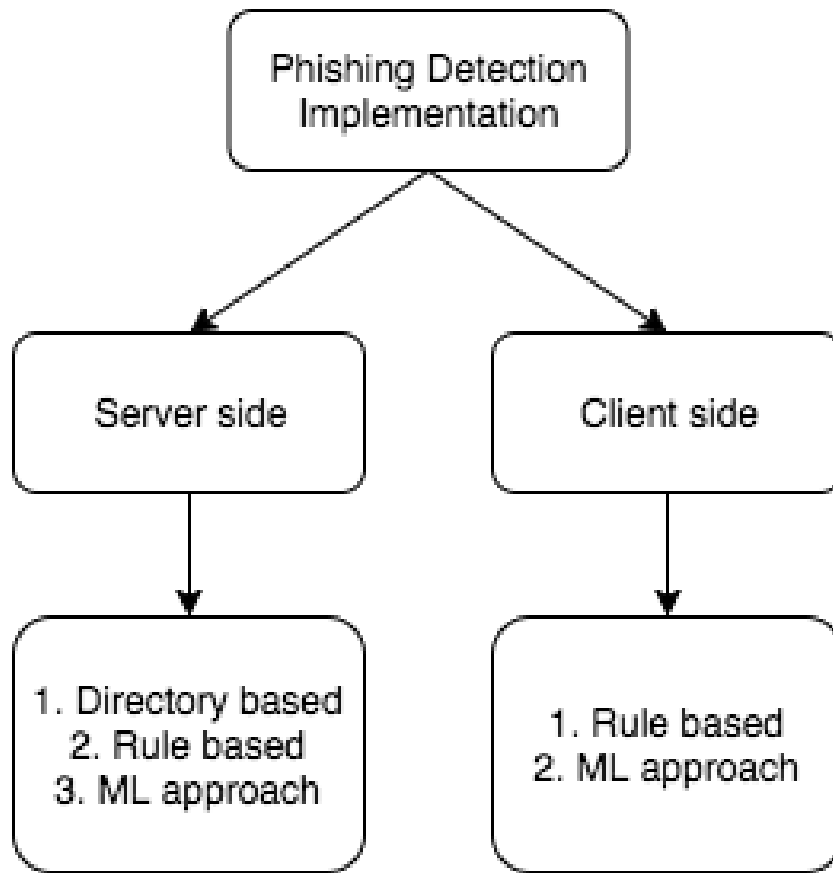
Chapter 2 discusses current phishing techniques in more detail. Section 3 presents the system requirements analysis. It explains functionality and non-functionality, constraints and assumptions along with various UML diagrams during operation. Section 4 describes the overall architecture, design, and complexity of each module. Chapter 5 provides details on the implementation of each module. Section 6 shows the results of the application. Chapter 7 introduces the entire text and summarizes its criticisms.

## **CHAPTER 2**

### **RELATED**

### **WORKS**

This chapter gives a survey of the possible approaches to phishing website detection. This survey helps to identify various existing approaches and to find the drawbacks in them. The difficulty in most of the approaches is that they are not implemented in real time so that an end user will benefit from it.



**Figure 2.1** Approaches to phishing detection

- **DIRECTORY BASED APPROACHES**

One of the most popular is PhishTank. According to Phish-Tank4, this is a public clearinghouse where data and information related to Internet phishing are collected. Additionally, PhishTank provides an open API for developers and researchers to integrate anti-phishing data into their applications for free. Therefore, PhishTank is a list of all phishing sites discovered and reported by people on the Internet so that developers can use its API to identify phishing sites.

Google has an API called Google Safe Browsing API which again follows a grid-based approach and also provides an open API similar to PhishTank.

This suggestion obviously doesn't work because new phishing websites are constantly being created and this list cannot be maintained indefinitely. When URLs are sent to the PhishTank API, this also causes user browsing behavior.

- **RULE BASED APPROACHES**

An existing Chrome extension called PhishDetector<sup>5</sup> uses a rules-based approach to detect phishing without the need for an external service. Although rule-based methods promote user-friendliness, they are less accurate than machine learning methods. Similar work by Shreeram.V to investigate phishing attacks Research using Genetic Algorithm <sup>6</sup> algorithm.

Using rules created by genetic algorithms.

<sup>4</sup> <http://phishtank.com/>

<sup>5</sup> <https://chrome.google.com/webstore/detail/phishdetector-true-phishi/kgelcldbalfgmgelepbbld-foogmjdgmj>

<sup>6</sup> <https://ieeexplore.ieee.org/document/5670593/>

PhishNet is a predictive blacklisting method. TLDs use rules that match domain names, IP addresses, HTTP header fields, and more.

SpoofGuard<sup>7</sup> is a Chrome extension from Stanford that uses similar rules, taking into account DNS, URLs, images, and links.

Phishwish: A Stateless Filter Using a Few Rules Authors: Debra L. Cook, Vijay K. Gurbani, Michael Daniluk have developed a phishing filter that has the advantages of already existing filters: no training required and no security measures. whitebase or blacklists to reference. They use only 11 rules to determine the authenticity of the website.

- **ML BASED APPROACHES**

Intelligent Phishing Detection Using Random Forest Classifiers (IEEE-2017), authors: Abdulhamit Subaşı, Esraa Molah, Fatin Almkallawi and Touseef J. Chaudhery discuss phishing detection using random forest classifiers. Random forest showed the best performance among classification methods, with an accuracy rate as high as 97.36%.

PhishBox: Phishing Detection and Analysis Methods (IEEE-2017), Jhen-Hao Li, Sheng-De Wang discuss a common model for phishing detection. As a result, the negative rate for phishing searches dropped by an average of 43.7%.

Instant Detection of Phishing Websites (IEEE-2016), Abdulghani Ali Ahmed, Nurul Amirah Abdullah talks about a feature based solely on URLs on websites.

They may come with a detection engine that can identify different types of phishing attacks at low cost.

Nuttapong Sanngrersinlapachai uses domain homepage similarity features in machine learning based phishing detection,

<sup>7</sup> <https://crypto.stanford.edu/SpoofGuard/>

Arnon Rungsawang presents a study on using a concept feature to detect web phishing problem. They applied additional domain top-page similarity feature to a machine learning based phishing detection system. The evaluation result in terms of f-measure was up to 0.9250, with 7.50% of error rate.

Netcraft<sup>8</sup> is one popular phishing detection plugin for chrome that uses server side prediction.

- **DRAWBACKS**

Based on the above mentioned related works, It can be seen that the plugins either use rule based approach or server side ML based approach. Rule based approach doesn't seem to perform well compared to ML based approaches and on the other side ML based approaches need libraries support and so they are not implemented in client side plugin. All the existing plugins send the target URL to an external web server for classification. This project aims to implement the same in browser plugin removing the need of external web service and improving user privacy.



## **CHAPTER 3**

### **REQUIREMENTS ANALYSIS**

- **FUNCTIONAL REQUIREMENTS**
- The plugin will warn users when they visit a phishing website.
- Plugins must comply with the following rules:
- Plugins must be fast enough to prevent users from sending sensitive information to phishing websites.
- Plugins should not use external services or APIs that could reveal user browsing patterns.
- The plugin must be able to detect new phishing websites.
- Plugins need to have a strategy to update themselves against emerging phishing techniques.
- **NON FUNCTIONAL REQUIREMENTS**
- **User Interface**

There must be a simple and easy to use user interface where the user should be able to quickly identify the phishing website. The input should be automatically taken from the webpage in the current tab and the output should be clearly identifiable. Further the user should be interrupted on the event of phishing.

- **Hardware**

No special hardware interface is required for the successful implementation of the system.

- **Software**

- Python for training the model
- Chrome browser

- **Performance**

The plugin should be always available and should make fast detection with low false negatives.

- **CONSTRAINTS AND ASSUMPTIONS**

- **Constraints**

- Some technologies include SSL, page rank, etc. uses. Without an external API, this information cannot be retrieved from the plugin client. Therefore, these features cannot be used for prediction.
- Considering the processing power of the user's machine and the loading time of the site, heavy methods cannot be used.
- Use only Javascript to create Chrome extensions. Machine learning libraries have less javascript support compared to python and R.

- **Assumptions**

- The plugin is provided with the needed permissions in the chrome environment.

- The user has a basic knowledge about phishing and extensions.

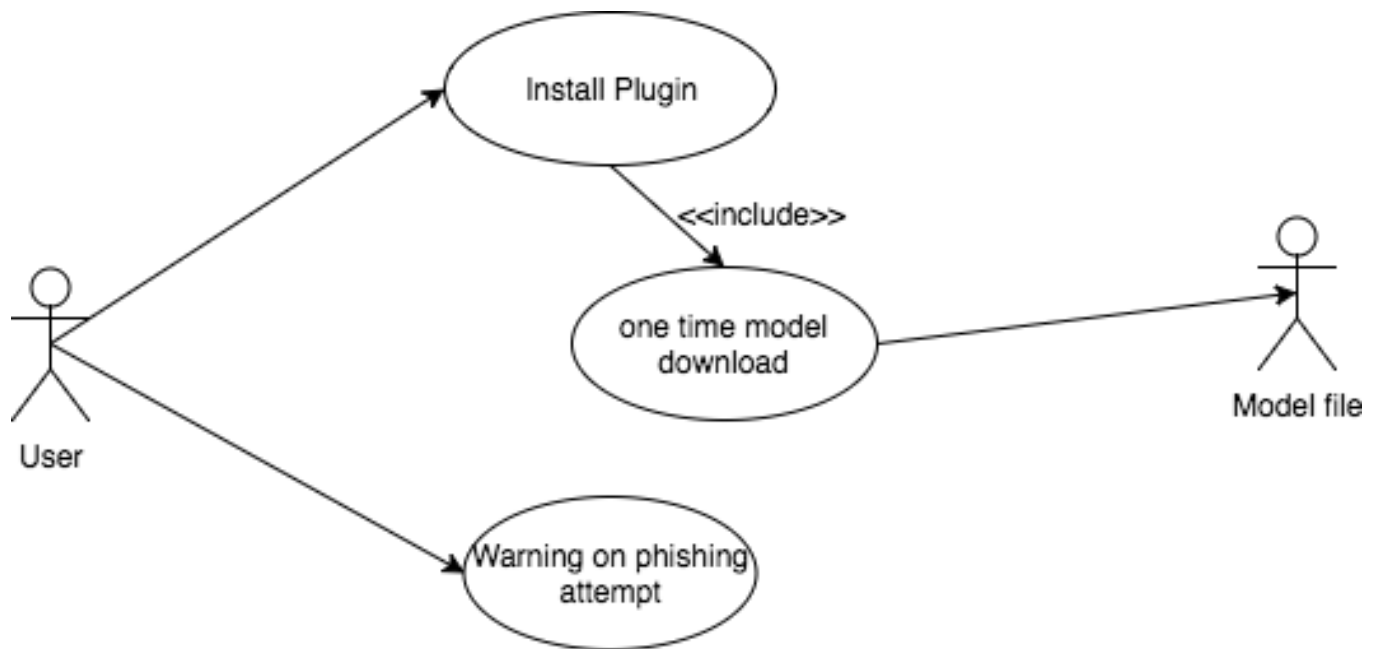
- **SYSTEM MODELS**

- **Use Case Diagram**

The overall use case diagram of the entire system is shown in figure 3.1. The user can install the plugin and then can continue his normal browsing behaviour. This plugin will automatically check the browsing pages for phishing and warns the user of the same.

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

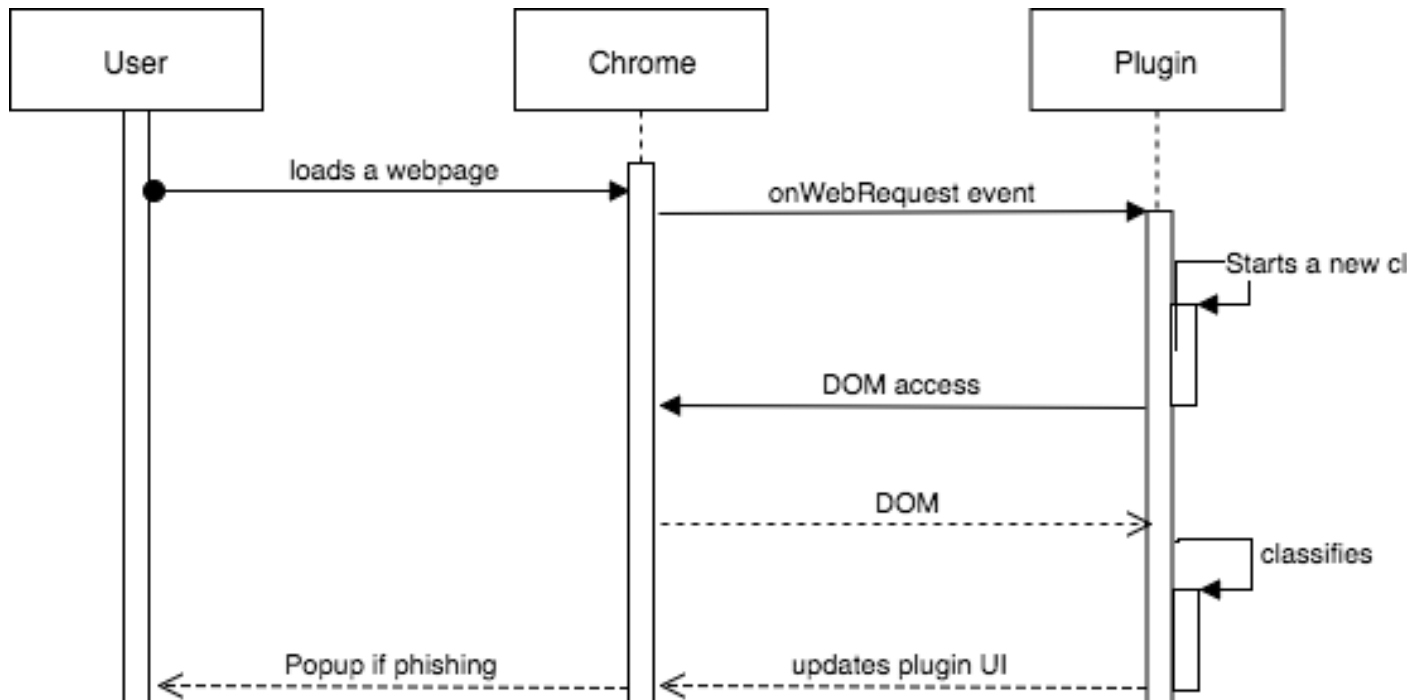
**Post condition:** The user is warned incase it's a phishing website.



**Figure 3.1** Use case diagram of the system

- **Sequence diagram**

The sequence of interactions between the user and the plugin are shown in the figure 3.2



**Figure 3.2** System Sequence diagram

# CHAPTER 4

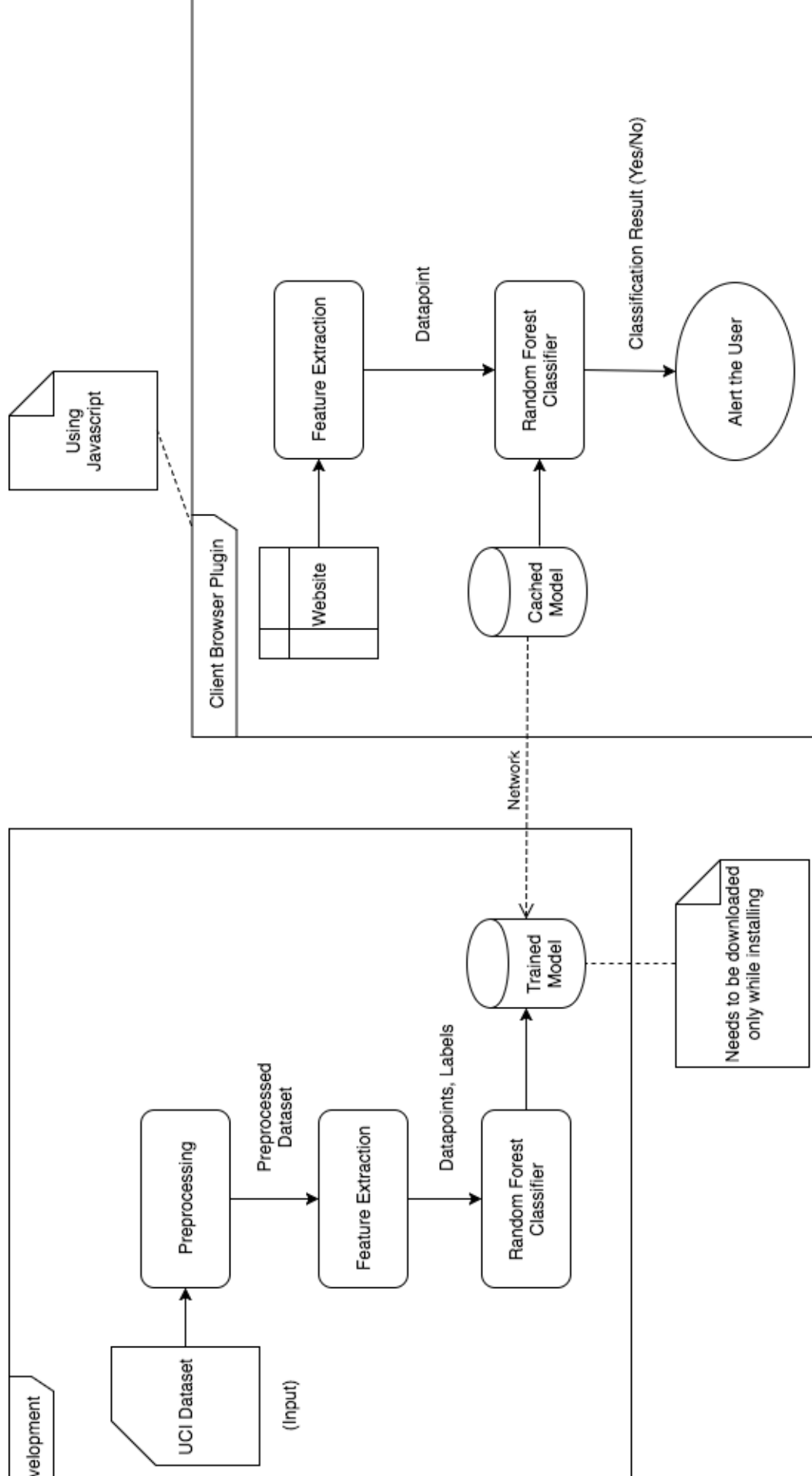
## SYSTEM

## DESIGN

- **SYSTEM ARCHITECTURE**

The block diagram of the entire system is shown in the figure 4.1. A Random Forest classifier is trained on phishing sites dataset using python scikit-learn. A JSON format to represent the random forest classifier has been devised and the learned classifier is exported to the same. A browser script has been implemented which uses the exported model JSON to classify the website being loaded in the active browser tab.

The system aims at warning the user in the event of phishing. Random Forest classifier on 17 features of a website is used 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. Features are selected on the basis that they can be downloaded completely offline from the client, without relying on web services or third parties. Data with selected features are classified for training and testing purposes. Then the random forest is trained based on the data shown and exported to the above JSON format. The JSON archive is hosted in the URL. Client-side Chrome extension is used to create a script every time the page loads and start extracting and accessing the functions selected above. Once the property is coded, the plugin checks the exported JSON format in the cache and re-downloads it if it is not in the cache.



## **Figure 4.1 System Architecture**

With the encoded feature vector and model JSON, the script can run the classification. Then a warning is displayed to the user, incase the website is classified as phishing. The entire system is designed lightweight so that the detection will be rapid.

- **UI DESIGN**

- A simple and easy-to-use user interface was created for the plugin using HTML and CSS. The user interface has a large circle showing the percentage of space in the Active tab. The color of the circle also varies by classification (green for legitimate sites, red for phishing). Analysis results, including the values displayed at the bottom of the circle, are displayed in the following color codes.
- Green - Legal Yellow - Suspicious Red - Phishing
- The plugin also shows an alert when phishing occurs to prevent phishing. Test results such as precision, recall, and accuracy are displayed on separate screens. The user interface is shown in Figure 4.2

- **CLASS DIAGRAM**

The class diagram of the entire Machine Translation system is shown in figure 4.3. This diagram depicts the functions of various modules in the system clearly. It also shows the interaction between the modules of the system thereby providing a clear idea for implementation.

## PhishTor

A Phishing detection plugin



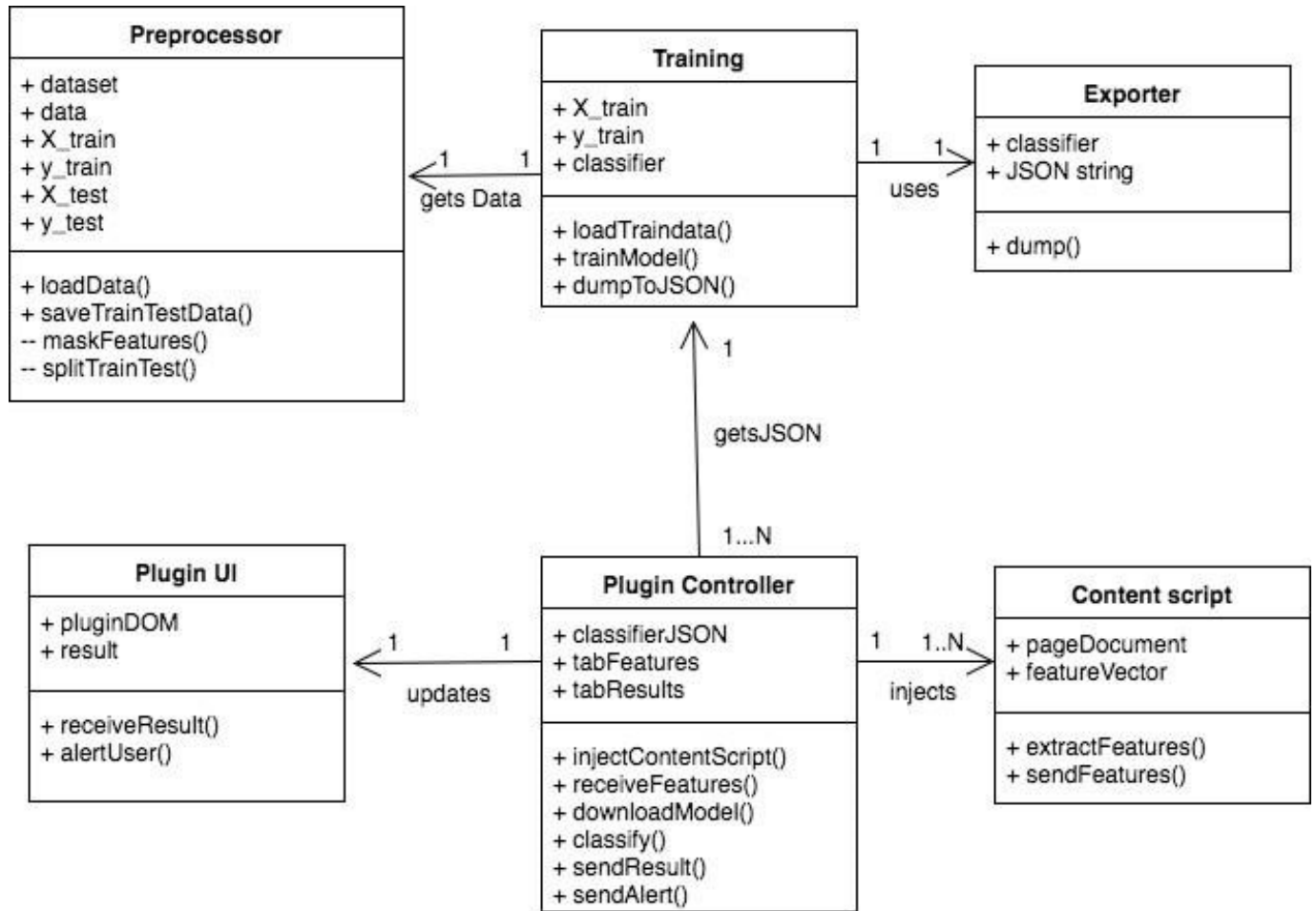
Warning!! You're being phished.

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**

The dataset is downloaded from UCI repository and loaded into a numpy array. The dataset consists of 30 features, which needs to be reduced so that they can be extracted on the browser. Each feature is experimented on the browser so that it will be feasible to extract it without using any external web service or third party. Based on the experiments,

17 features have been chosen out of 30 without much loss in the accuracy on the test data. More number of features increases the accuracy and on the other hand, reduces the ability to detect rapidly considering the feature extraction time. Therefore, feature subsets are selected in a balanced 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

Then the dataset is split into training and testing set with 30% for testing. Both the training and testing data are saved to disk.

- **Training**

Training data from the preprocessing module is loaded from disk. Use the scikit-learn library to train a random forest distribution on data. Random Forest is an ensemble learning technique and therefore uses a set of 10 decision tree predictions. All decision trees follow the CART algorithm and try to minimize Gini impurities.

$$\text{Gini}(E) = 1 - \sum_{j=1}^c p_j^2$$

Cross-validation scores are also included as data points. F1 scores are calculated based on test data. Then use the extension model to export the training model to JSON.

## Exporting Model

Every machine learning algorithm learns its values during training. In a random forest, each decision tree is an independent study, each decision tree learns the starting point and leaf of the learning category result. Therefore, a format must be created to represent the random forest in JSON.

Each JSON structure contains the estimator's number, class number, etc. Contains keys such as. There is also 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

which each value is an estimator represented in JSON. Each decision tree is encoded as a JSON tree with nested objects containing threshold for that node and left and right node objects recursively.

- **Plugin Feature Extraction**

The above mentioned 17 features needs to be extracted and encoded for each webpage in realtime while the page is being loaded. A content script is used so that it can access the DOM of the webpage. The content script is automatically injected into each page while it loads. The content script is responsible to collect the features and then send them to the plugin. The main objective of this work is not to use any external web service and the features needs to be independent of network latency and the extraction should be rapid.

All these are made sure while developing techniques for extraction of features.

Once a feature is extracted it is encoded into values  $\{-1, 0, 1\}$  based on the following notation.

- 1 - Legitimate
- - Suspicious
- - Phishing

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

- **Classification**

Feature vectors obtained from labels were classified according to random forest. Download the forest random parameter JSON and cache it on disk. The script attempts to load JSON from disk and if the cache is missing the JSON is re-downloaded. Created a JavaScript library to simulate random forest character nodes using JSON by comparing feature vectors with thresholds. The result of binary classification is determined by the values of the page and warns the user if the web page is classified 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	$O(E * v * n \log(n))$
3	Exporting model	$O(E * n \log(n))$
4	Plugin feature extraction	$O(v)$
5	Classification	$O(E * n \log(n))$

**Table 4.2** Time Complexity of various modules

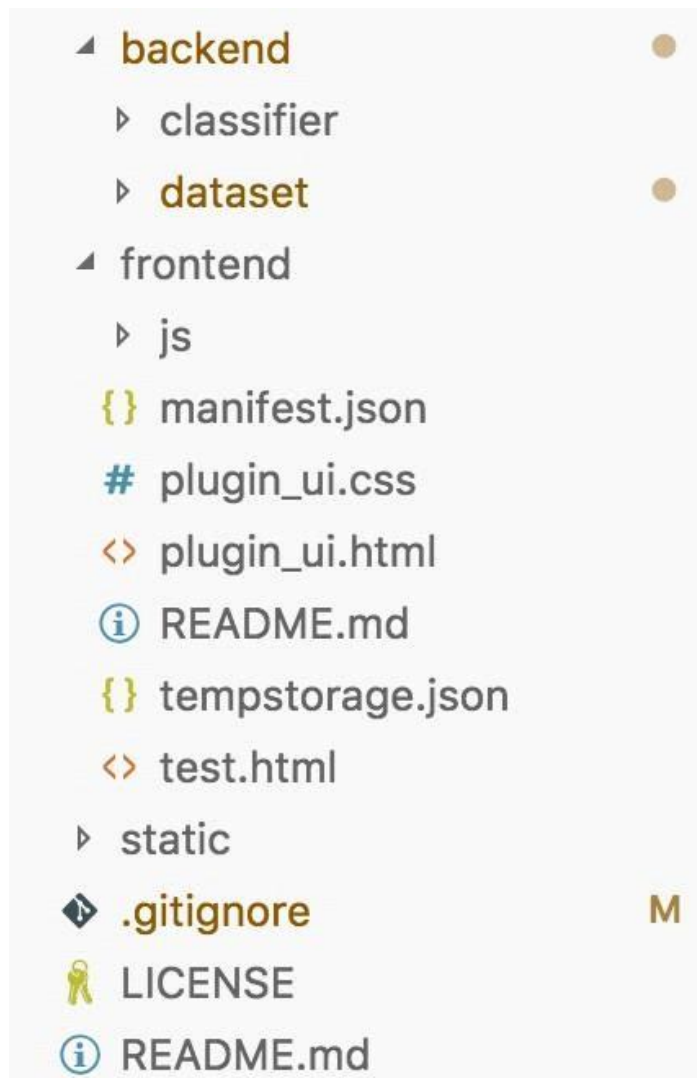
- ‘n’ denotes number of data points.
  - ‘E’ denotes number of ensembles (decision trees).
  - ‘v’ denotes number of features.
- **Complexity of the project**
    - The challenge of this project is to gauge the balance between accuracy and rapid detection. Selecting a subset of features can increase accuracy without decreasing accuracy.
    - Port Scikit-learn python components to javascript via format. For example JSON.
    - Reproducing random forest behavior in Javascript slightly reduces accuracy.
    - Many things cannot be extracted without using external services. Use of external services will again affect call time.
    - It is important to maintain a high level of security as the system must detect phishing before the user sends sensitive information.

## CHAPTER 5

### SYSTEM

### DEVELOPMENT

The system is generally divided into backend and plugins. The backend contains predefined data and training modules. The front end of the plugin contains javascript files for content scripts and background scripts (like random forest scripts). The plugin also includes HTML and CSS files for the user interface. A general outline showing the organization of these different structures is shown in Figure 5.1.



**Figure 5.1** Code Overview

- **PROTOTYPE ACROSS THE MODULES**

The input and output to each module of the system is described in this section.

- **Preprocessing:** This module takes the downloaded dataset in arff format and creates four new files listed as training features, training class labels, testing features, testing class labels.
- **Training:** This module takes the four output files from pre-

processor and gives a trained Random Forest object along with the cross validation score on the training set.

- **Exporting model:** This module takes the learned Random Forest classifier object and the recursively generates its JSON representation which is written to file in disk.
- **Plugin Feature Extraction:** This module takes a webpage as input and generates a feature vector with 17 encoded features.
- **Classification:** This model retrieves the image vectors from the Feature Extraction module, retrieves the JSON format from the Export Model module, and provides a Boolean output message indicating whether the page is legitimate or phishing.

- **EXPORTING ALGORITHM**

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

**TREE\_TO\_JSON(NODE):**

- $tree\_json \leftarrow \{ \}$
- **if** (node has threshold) **then**
- $tree\_json["type"] \leftarrow "split"$
- $tree\_json["threshold"] \leftarrow node.threshold$
- $tree\_json["left"] \leftarrow TREE\_TO\_JSON(node.left)$
- $tree\_json["right"] \leftarrow TREE\_TO\_JSON(node.right)$
- **else**
- $tree\_json["type"] \leftarrow "leaf"$
- $tree\_json["values"] \leftarrow node.values$
- **return**  $tree\_json$

### **RANDOM\_FOREST\_TO\_JSON(RF):**

- $\text{forest\_json} \leftarrow \{\}$
- $\text{forest\_json}['n\_features'] \leftarrow \text{rf.n\_features\_}$
- $\text{forest\_json}['n\_classes'] \leftarrow \text{rf.n\_classes\_}$
- $\text{forest\_json}['classes'] \leftarrow \text{rf.classes\_}$
- $\text{forest\_json}['n\_outputs'] \leftarrow \text{rf.n\_outputs\_}$
- $\text{forest\_json}['n\_estimators'] \leftarrow \text{rf.n\_estimators}$
- $\text{forest\_json}['estimators'] \leftarrow []$
- $e \leftarrow \text{rf.estimators}$
- **for** ( $i \leftarrow 0$  **to**  $\text{rf.n\_estimators}$ )
- $\text{forest\_json}['estimators'][i] \leftarrow \text{TREE\_TO\_JSON}(e[i])$
- **return**  $\text{forest\_json}$

### **• DEPLOYMENT DETAILS**

The backend requires Python 3 and the Classifier JSON and Test set are served over HTTP using Github. The plugin is distributed as single file and requires Chrome browser to run. The plugin (frontend) is packed into a crx file for distribution.



## **CHAPTER 6**

### **RESULTS AND DISCUSSION**

- **DATASET FOR TESTING**

The test system consists of data points separated from the data set in a ratio of 70:30. The plugin also tries to use websites listed on phishTank. New phishing websites are also added to PhishTank as soon as they are discovered. It is worth noting that the plugin can also detect new phishing websites. The results of this evaluation and the overall process are summarized below.

- **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 ➤ 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_registration_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 splitting
X:(11055, 17), y:(11055,)
After splitting
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 ➤ 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.



---

▼ 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 output of the classification is shown right in the Plugin UI. Green circle indicates legitimate site and Light red indicates phishing.

## A Phishing detection plugin

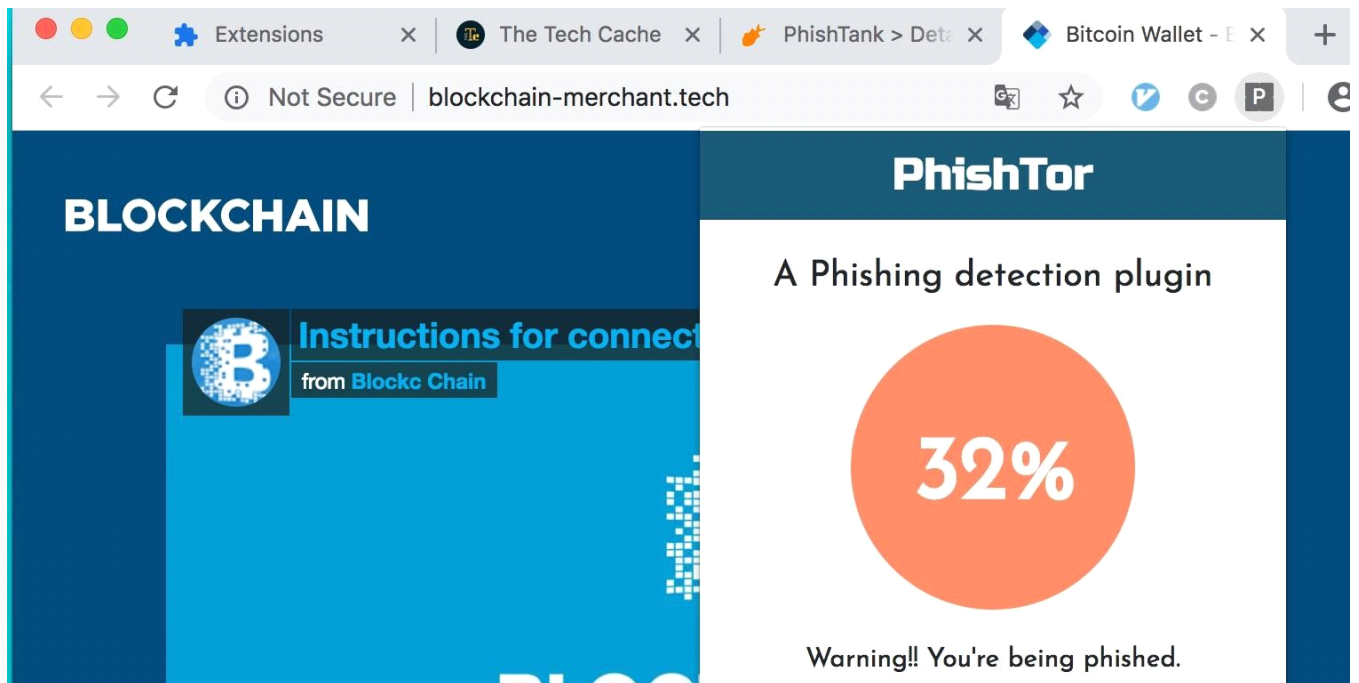


This website is safe to use :)

**Figure 6.5** Classification Output

- **SAMPLE SCREENSHOTS DURING TESTING**

The output of the plugin while visiting a phishing site taken from PhishTank. This site has a low trust value and also the light red circle indicates phishing.



**Figure 6.6** Test Output

- **PERFORMANCE EVALUATION**

The performance of the entire system is evaluated using the standard parameters described below.

- **Cross Validation score**

Learning about the failure of a prediction and testing it on the same data is a mistake: A model that repeats the label of the model it just found will score perfectly but fail to predict any value. Information not yet found. This situation is called overfitting.

To solve this problem, another part of the data set can be called a "validation set": training is done on the training set, >Then evaluation is done on the validation process and if the experiment appears successful when it is done, the final evaluation can be done on the experiment. However, by dividing the available data into three groups, we reduce the number of samples available to learn the model, and the results will depend on the specific random selection of the (training, validation) pair group.



The solution to this problem is a process called cross-validation (CV for short). The test procedure should still be retained for final evaluation, but the current procedure is no longer required when preparing the CV. In the simple method known as k-fold CV, the training set is divided into k smaller pieces (other methods are described below, but most follow the same language of basic principles). For each k "folds" the following procedure is followed:

The model is trained using k folds as training data; the design is validated against the rest of the data (i.e. used as a test to calculate performance metrics such as accuracy).

The 10x cross validation score is as follows

```
print('Cross Validation Score: {0}'.format(np.mean(cross_val_score  
Cross Validation Score: 0.9476602117325363
```

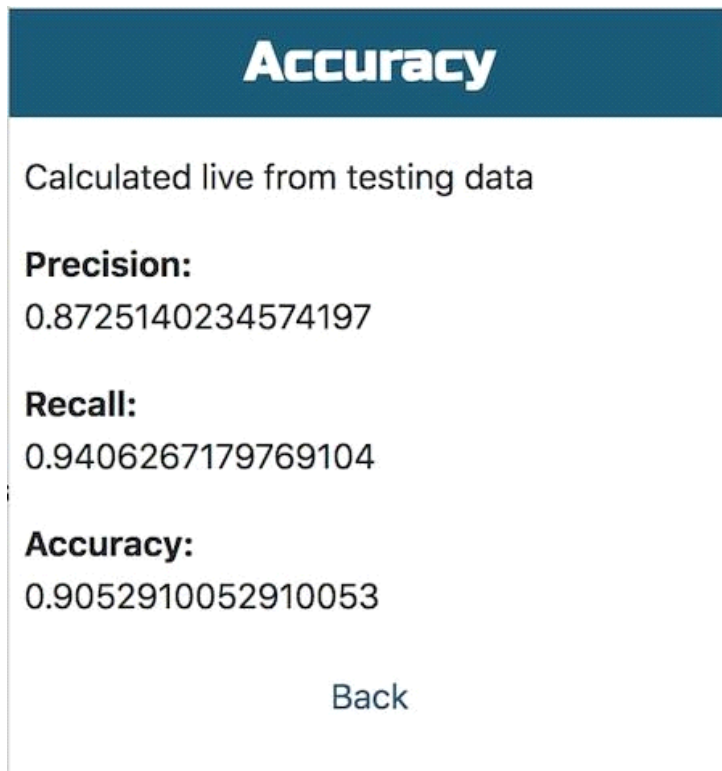
**Figure 6.7** Cross Validation Output  
(calculated with cross\_val\_score of scikit-learn)

- **F1 score**

The F1 score is a measure of the accuracy of the test. It calculates the score to determine the test's accuracy and return: accuracy is the number of results divided by the number of all positive results returned by the distribution, and return is the number of positive results divided by the number of each positive result returned by the distribution. relevant samples (all samples that should be identified as positive). The F1 score is a compromise between precision and recall; where the F1 score reaches its best value of 1 (excellent precision and recall) and its worst value of 0. < br>F1 score can be defined as the weight between real and real. Vice versa, where the F1 score reaches the best value (1) and the worst score is 0. The relative contribution of precision and return to F1 score is equal. The formula for the F1 score is as follows:

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

The precision, recall and F1 score of the phishing classifier is calculated manually using javascript on the test data set. The results are shown in the figure 6.8



**Figure 6.8** Performance measure



# CHAPTER 7

## CONCLUSIO

### NS

- **SUMMARY**

This is a phishing website detection system that aims to provide rapid detection to users so that users are warned before they are phished. The main use is to access random forest distribution for JavaScript. Similar operations often use web pages that cannot be extracted from the client and perform network-based discovery. On the other hand, this system uses features that can only be obtained from the customer and thus can provide rapid detection and better privacy. Although using fewer features may slightly reduce accuracy, it increases the usability of the system. This function defines the set of web page functions that are available to customers without compromising accuracy.

Python to javascript migration and personal random forest used in javascript further aids fast searching as the standard JSON representation and classification of text is created keeping in mind the time complexity. The plugin can detect phishing even before the page is fully loaded. The calculated F1 score of the client's test was 0.886.

- **CRITICISMS**

The system is less accurate than state-of-the-art systems, but it is more usable and the balance between accuracy and search speed works well. The Chrome Extension API has limited impact on add-ons. Since these functions are extracted from the script when the page loads, the plugin cannot prevent malicious JavaScript code.

Additionally, when transferring from python to javascript, the accuracy drops from 0.94 to 0.886, which needs to be worked on. Javascript does not support multiple threads, the browser only supports javascript. Therefore, splitting cannot be done

faster with parallel threads. Currently results are not cached in the plugin and are counted twice even when the site is visited most frequently.

- **FUTURE WORK**

The current class learned about 17 features that could be added as long as they didn't slow down or cause loss of privacy. This extension allows frequently visited sites to reduce calculation. However, this can cause pharming attacks to go undetected. A solution must be designed to cache the results without losing the ability to detect fraud. Partitioning in Javascript can be done using WorkerThreads which can improve partitioning time. Therefore, the system can perform many improvements and improvements to be more effective in detecting phishing.

## **REFERENCES**

- A. Subasi, E. Molah, F. Almkallawi, and T. J. Chaudhery, “Intelli- gent phishing website detection using random forest classifier,” 2017 *International Conference on Electrical and Computing Tech- nologies and Applications (ICECTA)*, Nov. 2017.
- “UCI Machine Learning Repository: Phishing Websites Data Set,” [Online]. Available: [https://archive.ics.uci.edu/ml/datasets/phishing websites](https://archive.ics.uci.edu/ml/datasets/phishing%20websites).
- J.-H. Li and S.-D. Wang, “PhishBox: An Approach for Phishing Validation and Detection,” 2017 *IEEE 15th Intl Conf on Depend- able, Autonomic and Secure Computing, 15th Intl Conf on Perva- sive Intelligence and Computing, 3rd Intl Conf on Big Data Intelli- gence and Computing and Cyber Science and Technology Con- gress(DASC/PiCom/DataCom/CyberSciTech)*, 2017.
- A. A. Ahmed and N. A. Abdullah, “Real time detection of phishing websites,” 2016 *IEEE 7th Annual Information Technology, Elec- tronics and Mobile Communication Conference (IEMCON)*, 2016.
- 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, “Phishing Detection in Websites Using Neural Networks and Firefly,” *International Journal Of Engineering And Computer Science*, 2016.
- “An Efficient Approaches For Website Phishing Detection Using Supervised Machine Learning Technique,” *International Journal of Advance Engineering and Research Development*, vol. 2, no. 05, 2015.
- S. Gupta and A. Singhal, “Phishing URL detection by using artificial neural network with PSO,” 2017 *2nd International Conference on Telecommunication and Networks (TEL-NET)*, 2017.
- Ammar ALmomani, G. B. B, Tat-Chee Wan, Altyeb Altaher, and Selvakumar Manickam, “Phishing Dynamic Evolving Neural Fuzzy Framework for ...,” Jan-2013. [Online]. Available: <https://arxiv.org/pdf/1302.0629>.