

A MAJOR PROJECT REPORT ON

**An improvement on Detection of Phishing Websites using Web  
Extension**

A dissertation submitted in partial fulfilment of the  
Requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY**

in

**INFORMATION TECHNOLOGY**

*Submitted by*

**Chanda Koushik (17B81A1228)**

**Theratipally Balaji (17B81A1212)**

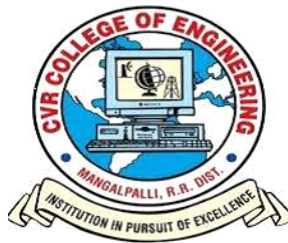
**D.Sankeerthana Reddy (17B81A1259)**

*Under the esteemed guidance of*

**Mrs. Bhagya Sri. G**

Assistant Professor, IT Department

CVR College of Engineering



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**CVR COLLEGE OF ENGINEERING**

ACCREDITED BY NBA, AICTE & Affiliated to JNTU-H  
Vastunagar, Mangalpally (V), Ibrahimpatnam (M), R.R. District, PIN-501 510  
2020-21



Cherabuddi Education Society's  
**CVR COLLEGE OF ENGINEERING**

(An Autonomous Institution)

**ACCREDITED BY NATIONAL BOARD OF ACCREDITATION, AICTE**

(Approved by AICTE & Govt. of Telangana and Affiliated to JNT University)

**Vastunagar, Mangalpalli (V), Ibrahimpatan (M), R.R. District, PIN - 501 510**

Web : <http://cvr.ac.in>, email : [info@cvr.ac.in](mailto:info@cvr.ac.in)

Ph : 08414 - 252222, 252369, Office Telefax : 252396, Principal : 252396 (O)

---

**DEPARTMENT OF INFORMATION TECHNOLOGY**  
**CERTIFICATE**

This is to certify that the Major Project Report entitled “**An improvement on detection of phishing websites using web extension**” is a bonafide record of work carried out by T.Balaji(17B81A1212),Ch.Koushik(17B81A1228),D.Sankeerthana(17B81A1259) under my supervision in partial fulfilment of the requirement for the degree of Bachelor of Technology in Information Technology, CVR College of Engineering.

**Project Guide**

**Mrs. Bhagya Sri**

Asst.Professor

Information Technology.

**Head of Department**

**Dr. Bipin Bihari Jayasingh**

Information Technology.

**Project Coordinator**

**Dr. J. Sengathir**

Associate Professor.

Information Technology.

## ACKNOWLEDGEMENT

The satisfaction of completing this project would be incomplete without mentioning our thanks to all the people who have supported us and constant guidance and encouragement have been instrumental in its completion.

We offer our sincere gratitude to **Mrs. Bhagya Sri**, Asst. Professor, Department of Information Technology, CVR College of Engineering for his immense support, timely co-operation and valuable advice throughout the course of our project work.

We would like to thank the Head of Department Professor **Dr. Bipin Bihari Jayasingh**, for meticulous care and cooperation throughout the project work.

We are thankful to **Dr. J. Sengathir**, Project Coordinator, Department of Information Technology, CVR College of Engineering for his supportive guidelines and for having provided the necessary help for carrying forward this project without any obstacles and hindrances.

We also thank the **Project Review Committee Members** for their valuable suggestions.

## Declaration

We hereby declare that the project report entitled “**An improvement on detection of phishing websites using web extension**” submitted by us to CVR College of Engineering, in partial fulfilment of the requirement for the award of the degree of B Tech, in INFORMATION TECHNOLOGY is a record of bonafide project work carried out by us under the guidance of **Mrs. Bhagya Sri. G.**

We further declare that the work reported in this project has not been submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

---

Signature of the Student  
(Ch.Koushik)  
(17B81A1228)

---

Signature of the Student  
(Therati pally. Balaji)  
(17B81A1212)

---

Signature of the Student  
(D.Sai Sankeerthana Reddy)  
(17B81A1259)

## CONTENTS

ABSTRACT .....	8
INTRODUCTION.....	10
1.1 PROBLEM DOMAIN .....	10
1.2 PROBLEM DESCRIPTION.....	10
1.3 SCOPE.....	10
1.4 CONTRIBUTION .....	11
1.5 SWOT ANALYSIS .....	11
1.6 PESTLE ANALYSIS .....	11
1.7 ORGANISATION OF THESIS .....	12
LITERATURE SURVEY .....	13
RELATED WORKS.....	16
3.1 DIRECTORY BASED APPROACHES .....	17
3.2 RULE BASED APPROACHES .....	17
3.3 ML BASED APPROACHES .....	17
3.4 DRAWBACKS.....	18
REQUIREMENTS ANALYSIS .....	19
4.1 FUNCTIONAL REQUIREMENTS .....	19
4.2 NON FUNCTIONAL REQUIREMENTS .....	19
4.3 CONSTRAINTS AND ASSUMPTIONS .....	20
4.4 SYSTEM MODELS .....	21
SYSTEM DESIGN.....	23
5.1 SYSTEM ARCHITECTURE .....	23
5.2 UI DESIGN .....	24
5.3 CLASS DIAGRAM.....	25
5.4 MODULE DESIGN .....	26
5.5 COMPLEXITY ANALYSIS .....	28
SYSTEM DEVELOPMENT .....	30
6.1 PROTOTYPE ACROSS THE MODULES .....	30
6.2 EXPORTING ALGORITHM.....	31
6.3 DEPLOYMENT DETAILS.....	32
RESULTS AND DISCUSSION .....	33
7.1 DATASET FOR TESTING.....	33

7.2 OUTPUT OBTAINED IN VARIOUS STAGES .....	33
7.3 SAMPLE SCREENSHOTS DURING TESTING.....	38
7.4 PERFORMANCE EVALUATION .....	40
CONCLUSION.....	42
8.1 SUMMARY .....	42
8.2 CRITICISMS.....	42
8.3 FUTURE WORK.....	43
REFERENCES .....	44
Appendix A .....	46
Appendix B .....	47
Appendix C .....	48

## LIST OF FIGURES

Figure 3.1	Approaches to phishing detection	16
Figure 4.4.1	Use case diagram of the system	21
Figure 4.4.2	System Sequence diagram	22
Figure 5.1	System Architecture	24
Figure 5.3	Class Diagram	25
Figure 6.2	Random Forest JSON structure	31
Figure 7.1	Preprocessing output	33
Figure 7.2	Training output	34
Figure 7.3	Model JSON	35
Figure 7.4	Webpage features	36
Figure 7.5	Classification Output	37
Figure 7.6	Test Output	38
Figure 7.7	Test Output	39
Figure 7.8	Test Output	39
Figure 7.9	Performance measure	41
Figure 8.1	Demo of Python usage in IDE	51

## LIST OF TABLES

Table 1.5	SWOT analysis	12
Table 5.5	Time Complexity of various modules	29



## **ABSTRACT**

Phishing is a type of cyber threat practice where, the intruder acts as a trustworthy entity and attempts to steal sensitive information such as Login Credentials, Credit and Debit card details by making the user feel the websites look and feel exactly as the original ones. It is generally carried out in various ways like E-Mail Spoofing, Instant Messaging or redirecting to the fake websites which user cannot detect the differences between the original and malicious ones. The core idea of this project is to detect those phishing websites by analysing the characteristics of URL (Uniform Resource Locator) using Machine Learning Algorithms.

We analyse various features of the URL like presence of '@' symbol, presence of Redirection (//) Symbol, Length of URL, Subdomains present in the URL which are relevant to the system and help in performing prediction on a new URL. All these extracted features are then trained in to a best suited machine learning algorithm. These features play a major role in classifying a URL whether a safe one or malicious.

The data we have used here is in high-level language which is first converted into machine-level language first and then these features are used as parameters to train the model. We have designed a web-app where the user can upload a file of URLs that he/she wants to predict and then obtain a result from it. With the assistance of this project, one can easily stop entering malicious websites and be safe from potential cyber threats.

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 PROBLEM DOMAIN**

Phishing is the fraudulent attempt to obtain sensitive information such as usernames, passwords, and credit card details (and money), often for malicious reason. It is typically carried out by email spoofing or instant messaging, and it often directs users to enter personal information at a fake website, the look and feel of which are identical to the legitimate site, the only difference being the URL of the website in concern. Communications purporting to be from social web sites, auction sites, banks, online payment processors are often used to lure victims. Phishing emails may contain links to websites that distribute malware. Detecting phishing websites often include lookup in a directory of malicious sites. Since most of the phishing websites are short lived, the directory cannot always keep track of all, including new phishing web- sites. So the problem of detecting phishing websites can be solved in a better way by machine learning techniques. Based on a comparison of different ML techniques, the random forest classifier seems to perform better.

Existing plugins send the URL to a server, so that the classification can be done in the server and the result is returned to the plugin. With this approach, user privacy is questioned and also the detection may be delayed due to network latency and the plugin may fail to warn the user in right time. As it is an important security problem and also considering the privacy aspects, we decided to implement this on a chrome browser plugin which can do the classification without an external server.

### **1.2 PROBLEM DESCRIPTION**

To develop a browser plugin which once installed, should warn the user on the event of he/she visiting a phishing website. The plugin should not contact any external web service for this which may leak the user's browsing data. The detection should be instant so that the user will be warned before entering any sensitive information on the phishing website.

### **1.3 SCOPE**

According to Wikipedia, In 2017, 76% of organisations experienced phishing attacks. In a survey professionals said that the rate of attacks increased from 2016 and the first quarter of the year 2017 nearly 100000 phishing events were experienced by people in Qatar. With increasing number of internet users, there is a prominent need for security solutions against attacks such as phishing. Hence this plugin would be a good contribution for the chrome users.

## **1.4 CONTRIBUTION**

This is the first implementation of phishing website detection in browser plugin without use of an external web service. This makes use of existing works done on phishing detection and implements them in a manner that it will benefit end users. This involves porting the existing python classifier (random forest) to JavaScript. The plugin with an one-time download of the learned model, will be able to classify websites in real time. This involves developing such a model (random forest) in javascript, as browser plugin supports only javascript. Thus this project contributes to better privacy and rapid detection of phishing.

## **1.5 SWOT ANALYSIS**

### **STRENGTHS**

- Enables user privacy.
- Rapid detection of phishing.
- Can detect new phishing sites too.
- Can interrupt the user in case of phishing.

### **WEAKNESS**

- Javascript limits functionality.
- Cannot use features that needs an external service such as SSL, DNS, page ranks.
- No library support.

### **OPPORTUNITIES**

Everyone conscious of privacy and security can use this plugin.

### **THREATS**

Non-technical people who do business transactions are vulnerable to phishing and they are potential end users for this

## **1.6 PESTLE ANALYSIS**

### **1.6.1 POLITICAL**

This project is rarely controlled by political factors. One such scenario is that the government may

take any measure to prevent phishing and in such cases the plugin may lose its potential.

### **1.6.2 ECONOMICAL**

The plugin is completely based on public dataset and open chrome plugin API. Thus it has no Economic factors controlling it.

### **1.6.3 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 install this plugin.

### **1.6.4 TECHNOLOGICAL**

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

### **1.6.5 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 privacy based products.

### **1.6.6 ENVIRONMENTAL**

This plugin has no environmental factors affecting it.

## **1.7 ORGANISATION OF THESIS**

Chapter 2 discusses the existing approaches to phishing detection in greater detail. Chapter 3 gives the requirements analysis of the system. It explains the functional and non-functional requirements, constraints and assumptions made in the implementation of the system and the various UML diagrams. Chapter 4 explains the overall system architecture and the design of various modules along with their complexity. Chapter 5 gives the implementation details of each module. Chapter 6 elaborates on the results of the implementation. Chapter 7 concludes the thesis and gives an overview of its criticisms.

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 A COMPARATIVE ANALYSIS AND AWARENESS SURVEY OF PHISHING DETECTION TOOLS**

Nowadays various tools are available to detect phishing websites in which most of them are browser extensions that generate a warning to user showing that it is a phishing website. The authors of this paper did a performance analysis for different phishing detection tools they also conducted a survey to raise awareness about different phishing websites and also the ways in which our personal information is misused and other details like bank credentials can be misused.

They tested different phishing detection tools in which they made a note of different values like true positive rate, true negative rate, false positive rate, false negative rate for each tool and they also calculated accuracy for each tool on how each tool is performing on several phishing URLs. Furthermore, drawbacks of some tools are discussed in this paper are Nucraft toolbar only checks for the URL's mentioned in the blacklist, Spoof guard toolbar calculates a threshold value where in some cases if the threshold value is not reached then it indicates that determination of this site was unsuccessful. Phisdetector toolbar is mostly recommended for the banking sites.

They also conducted an awareness survey which involved MTech students they were given with the systems in which tools were preinstalled and there were some questions. At the end they made a table format considering different working conditions of each tool.

## 2.2 DETECTING PHISHING WEBSITES USING DATA MINING

In this paper a brief about phishing and various ways of phishing and the consequences faced by the phishing. Furthermore, the authors mentioned the various types of techniques used for detecting phishing web sites like visual similarity approach, content based approach, data mining, etc and it also explains about their drawbacks.so, to overcome the drawback from heuristic design approach which uses only the common features (HTML,DOM) of websites so data mining approach is used which also refers large datasets.

In this approach a cloud-based model is used for phishing websites detection. The model will be based on classification algorithm (random forest) and will be trained using training dataset. The dataset is taken from UCI repository consist of 11055 records out of which 4,898 are phishing and 6,157 are legitimate. This model will be deployed in the cloud, which will directly communicate with the web extension.so, when the user enters a URL it is sent to the cloud and detects whether it's a phishing website or not then the chrome extension displays a message based on the result from the cloud. The extracted features through the URL will act as a test data for the model trained.

The URL features used for detecting phishing websites are address bar based featured (IP address, @ symbol, favicon, etc), abnormal features (request URL, abnormal URL), html and JavaScript based features (Iframe redirection, website forwarding), domain-based features. And they used various training algorithms like J48, SVM, neural network, random forest from which random forest is used with accuracy 97.2592. A machine learning classifier with very high accuracy is more likely to produce either false positives or true negatives.

## **2.3 DETECTING PHISHING WEBSITES USING MACHINE LEARNING**

The authors of this paper firstly explained the meaning of phishing and the major problems we face because of phishing, then they have written about the conventional approaches such as blacklist-whitelist approach, heuristic approach and content analysis and also the drawbacks associated with these approaches. So, to overcome the problems from the conventional approaches application of machine learning is proposed.

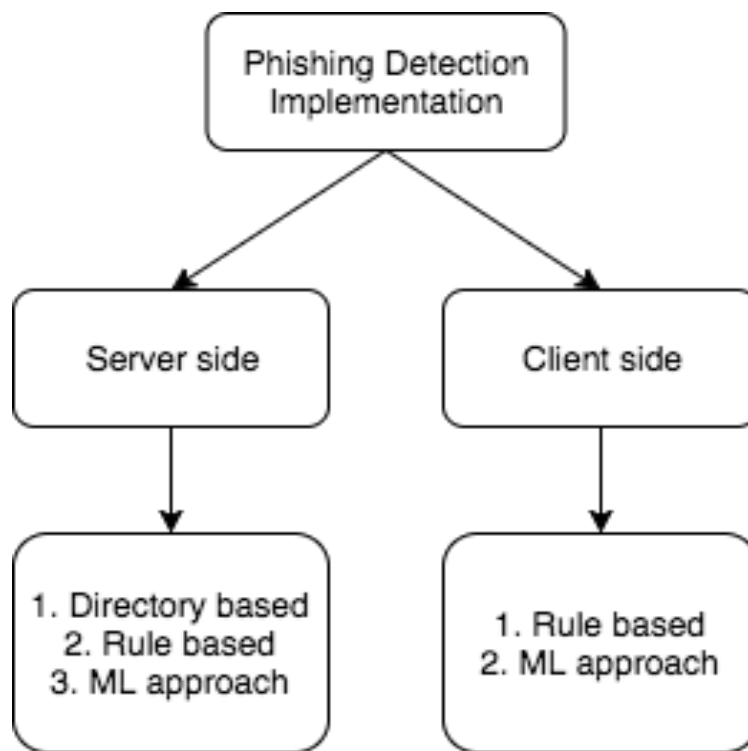
In machine learning approach they used a classifier to identify the phishing websites after t with a chosen dataset. The dataset is a custom build one consisting of 16000 datapoints with 31 attributes(secure layer, URL length, web traffic, domain details etc.), these are collected from various online sources and 10 membered surveys. They classified the features into three categories namely features from URL, page content and page rank.

The data set is further subjected to a feature selection algorithm for the elimination of intercorrelated features before feeding the data into the classifier for training, and this gave the output of 29 features those can be used for classification. Furthermore, the data was split into two sets i.e., 80% for training and the remaining for testing the classifier. Their data set shown the maximum accuracy when the selected features given into random forest classifier, and the most important feature here is web traffic.

## CHAPTER 3

### 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 3.1**  
Approaches to phishing detection



### 3.1 DIRECTORY BASED APPROACHES

Most popular one of this kind is Phish Tank. According to Phish-Tank , it is a collaborative clearing house for data and information about <sup>4</sup> phishing on the Internet. Also, PhishTank provides an open API for developers and researchers to integrate anti-phishing data into their applications at no charge. Thus PhishTank is a directory of all phishing websites that are found and reported by people across the web so that developers can use their API for detecting phishing websites. Google has a API called Google Safe Browsing API which also follows directory based approach and also provides open API similar to PhishTank. This kind of approach clearly can't be effective as new phishing web sites are continuously developed and the directory can't be kept up to date always. This also leaks users browsing behaviour as the URLs are sent to the PhishTank API.

### 3.2 RULE BASED APPROACHES

An existing chrome plugin named PhishDetector uses a rule <sup>5</sup> based approach so that it can detect phishing without external web service. Although rule based approaches support easier implementation on client side, they can't be accurate compared to Machine Learning based approaches. Similar work by Shreeram.V on detection of phishing attacks using genetic algorithm uses a rule that is generated by a genetic <sup>6</sup> algorithm for detection. PhishNet is one such Predictive blacklisting approach. It used rules that can match with TLD, directory structure, IP address, HTTP header response and some other. SpoofGuard by Stanford is a chrome plugin which used similar <sup>7</sup> rule based approach by considering DNS, URL, images and links. Phishwish: A Stateless Phishing Filter Using Minimal Rules by Debra L. Cook, Vijay K. Gurbani, Michael Daniluk worked on a phishing filter that offers advantages over existing filters: It does not need any training and does not consult centralized white or black lists. They used only 11 rules to determine the veracity of an website.

### 3.3 ML BASED APPROACHES

Intelligent phishing website detection using random forest classifier (IEEE-2017) by Abdulhamit Subasi, Esraa Molah, Fatin Almkallawi and Touseef J. Chaudhery discusses the use the random forest classifier for phishing detection. Random Forest has performed the best among the classification methods by achieving the highest accuracy 97.36%. PhishBox: An Approach for Phishing Validation and Detection (IEEE-2017) by Jhen-Hao Li, and Sheng-De Wang discusses ensemble models for phishing detection. As a result, The false-positive rate of phishing detection is dropped by 43.7% in average. Real time detection of phishing websites (IEEE-2016) by Abdulghani Ali Ahmed, and Nurul Amirah Abdullah discusses an approach based on features from only the URL of the website. They were able to come up with a detection mechanism that is capable of detecting various types of phishing attacks maintaining a low rate of false alarms. Using Domain Top-page Similarity Feature in Machine Learning- Based Web Phishing Detection by Nuttapon

Sanglerdsinlapachai, 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. Evaluation result was up to 0.9250 in terms of f-measure, with 7.5% of error rate. Netcraft is one popular phishing detection plugin for chrome that <sup>8</sup> uses server side prediction.

### **3.4 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 4**

### **REQUIREMENTS ANALYSIS**

#### **4.1 FUNCTIONAL REQUIREMENTS**

The plugin warns the user when he/she visits a phishing website.

The plugin should adhere to the following requirements:

- The plugin should be fast enough to prevent the user from submitting any sensitive information to the phishing website.
- The plugin should not use any external web service or API which can expose user's browsing pattern.
- The plugin should not use any external web service or API which can expose user's browsing pattern.
- The plugin should be able to detect newly created phishing websites.
- The plugin should have a mechanism of updating itself to emerging phishing techniques.

#### **4.2 NON FUNCTIONAL REQUIREMENTS**

##### **4.2.1 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.

##### **4.2.2 HARDWARE**

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

##### **4.2.3 SOFTWARE**

- Python for training the model
- Chrome browser

##### **4.2.4 PERFORMANCE**

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

## **4.3 CONSTRAINTS AND ASSUMPTIONS**

### **4.3.1 CONSTRAINTS**

- Certain techniques use features such as SSL, page rank etc. Such information cannot be obtained from client side plugin without external API. Thus those features can't be used for prediction.
- Heavy techniques can't be used considering the processing power of client machines and the page load time of the website.
- Only Javascript can be used to develop chrome plugins. Machine learning libraries support for javascript is far less compared to python and R.

### **4.3.2 ASSUMPTIONS**

- The plugin is provided with the needed permissions in the chrome environment.
- The user has a basic knowledge about phishing and extensions.

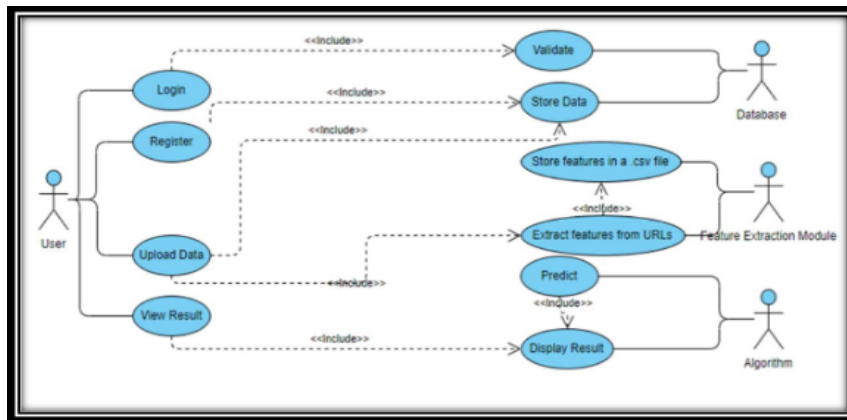
## 4.4 SYSTEM MODELS

### 4.4.1 USE CASE DIAGRAM

The overall use case diagram of the entire system is shown in figure 4.1. 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.

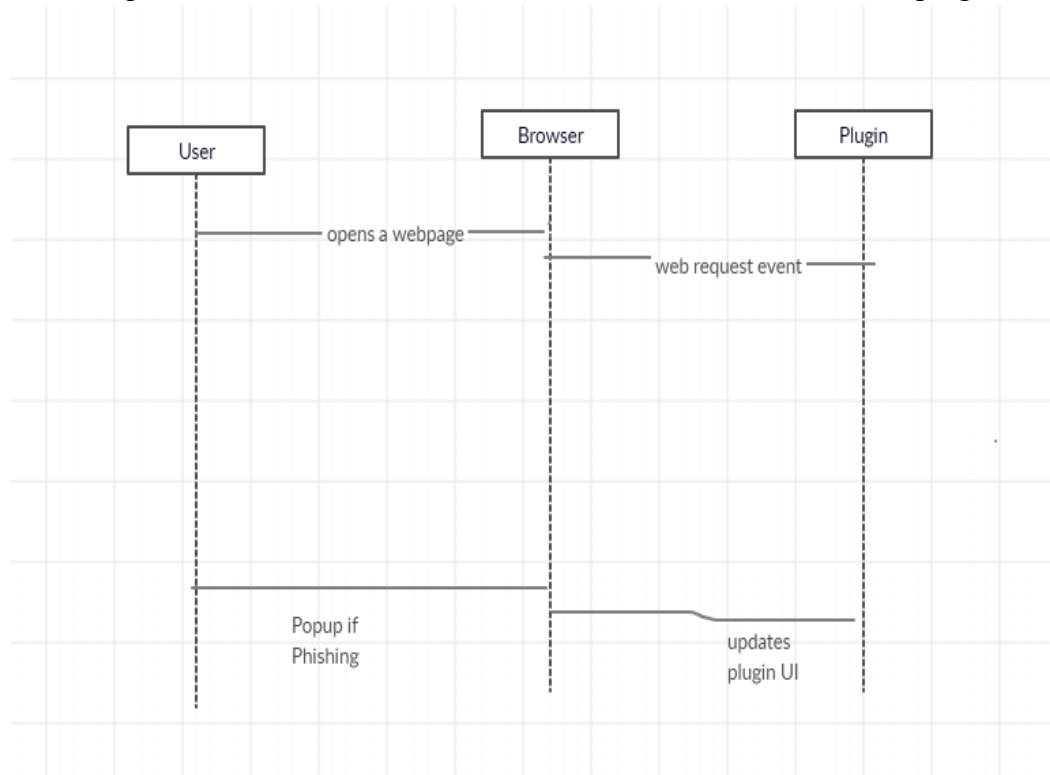
**Post condition:** The user is warned incase it's a phishing website.



**Figure 4.4.1** Use case diagram of the system

## 4.4.2 SEQUENCE DIAGRAM

The sequence of interactions between the user and the plugin are shown



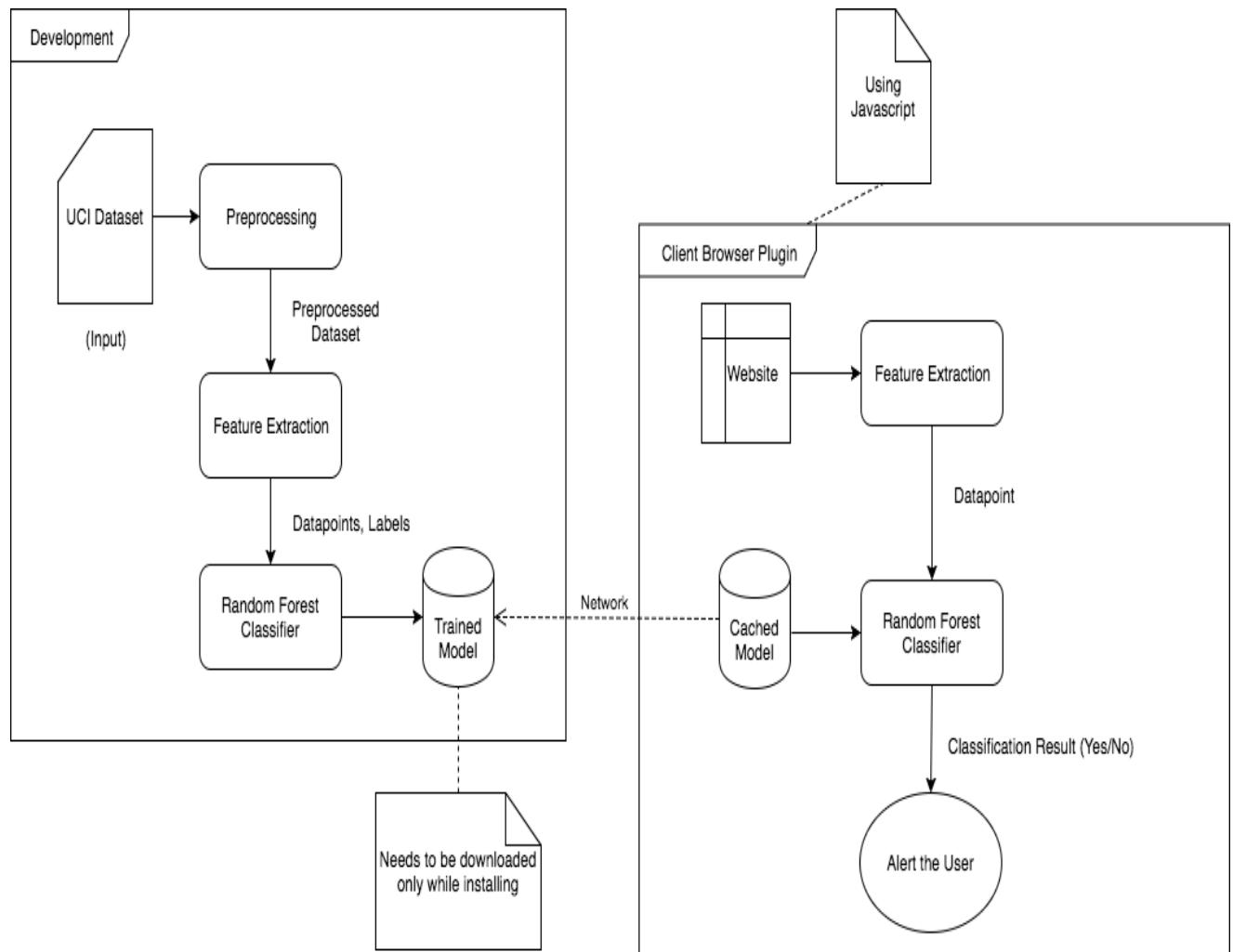
**Figure 4.2** System Sequence diagram

## CHAPTER 5

### SYSTEM DESIGN

#### 5.1 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 basis that they can be extracted completely offline on the client side without being dependent on a web service or third party. The dataset with chosen features are then separated for training and testing. Then the Random Forest is trained on the training data and exported to the above mentioned JSON format. The JSON file is hosted on a URL. The client side chrome plugin is made to execute a script on each page load and it starts to extract and encode the above selected features. Once the features are encoded, the plugin then checks for the exported model.



**Figure 5.1** 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.

## 5.2 UI DESIGN

A simple and easy to use User Interface has been designed for the plugin using HTML and CSS. The UI indicates Phishing if it is a Phishing website and safe if it is not. It displays both the results with different colours.(Green for legitimate website and Light Red for phishing).Below the result, the analysis results containing the extracted features are displayed in the following code.

-1 - Safe

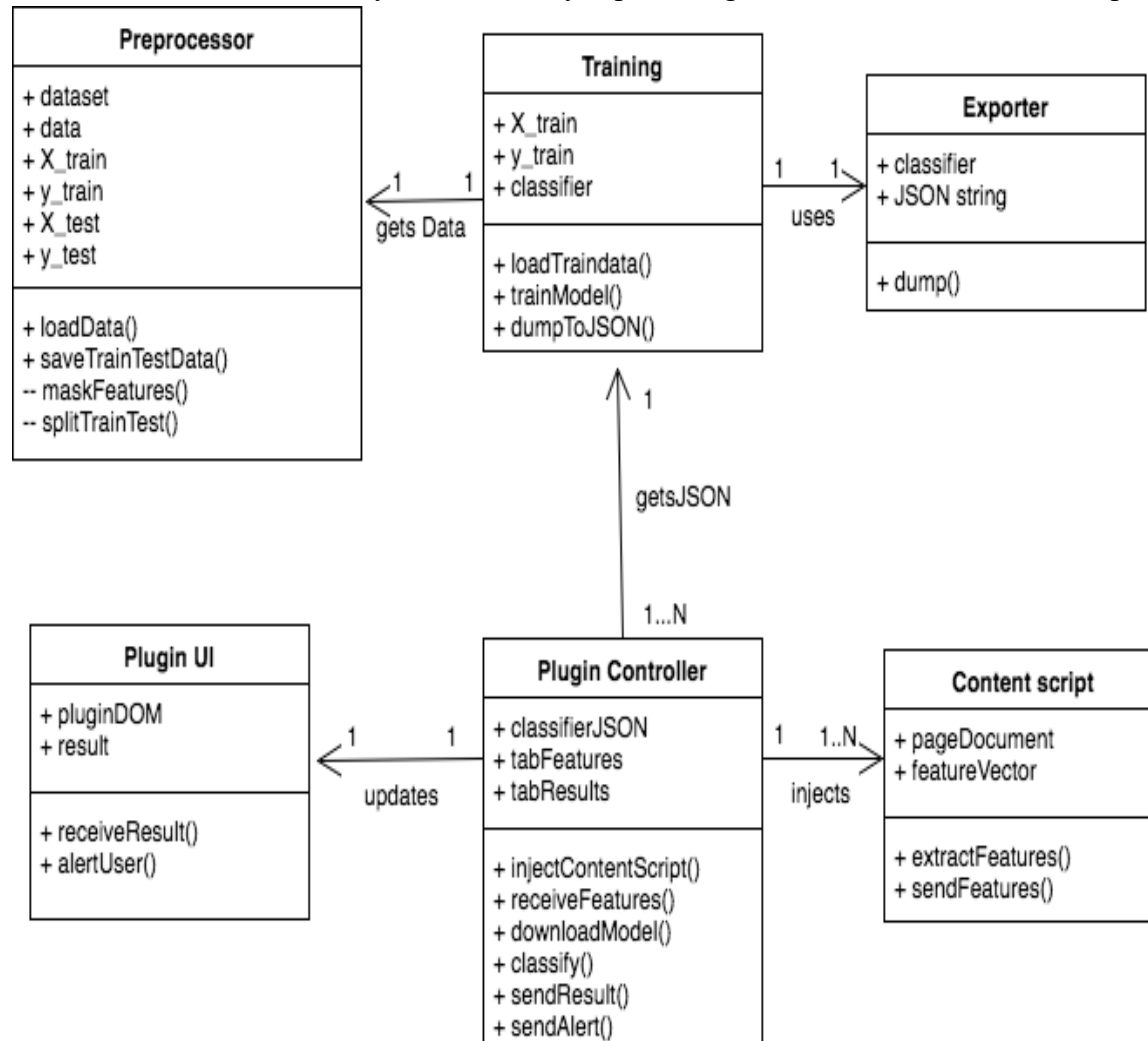
1 - Phishing



The test results such as precision, recall and accuracy are displayed in a separate screen. The UI is shown in figure 5.2

### 5.3 CLASS DIAGRAM

The class diagram of the entire Machine Translation system is shown in figure 5.3. 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.



**Figure 5.3** Class Diagram

## 5.4 MODULE DESIGN

### 5.4.1 PRE-PROCESSING

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. Thus a subset of features is chosen in a way that the trade-off is balanced.

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 5.1** Webpage Features

### 5.4.2 TRAINING

The training data from the pre-processing module is loaded from the disk. A random forest classifier is trained on the data using scikit-learn library. Random Forest is an ensemble learning technique and thus an ensemble of 10 decision tree estimators is used. Each decision tree follows CART algorithm and tries to reduce the gini impurity.

$$c_{\sum j=1} Gini(E)=1 -$$

$p_j$  <sup>2</sup>The cross validation score is also calculated on the training data.

The F1 score is calculated on the testing data. Then the trained model is exported to JSON using the next module.

### 5.4.3 EXPORTING MODEL

Every machine learning algorithm learns its parameter values during the training phase. In Random Forest, each decision tree is an independent learner and each decision tree learns node threshold

values and the leaf nodes learn class probabilities. Thus a format needs to be devised to represent the Random Forest in JSON. The overall JSON structure consists of keys such as number of estimators, number of classes and etc. Further it contains an array in 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.

#### **5.4.4 PLUGIN FEATURE EXTRACTION**

The above mentioned 17 features needs to be extracted and encoded for each webpage in real-time 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

1 - Phishing

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

#### **5.4.5 CLASSIFICATION**

The feature vector obtained from the content script is ran through the Random Forest for classification. The Random Forest parameters JSON is downloaded and cached in disk. The script tries to load the JSON from disk and incase of cache miss, the JSON is downloaded again. A javascript library has been developed to mimic the Random Forest behaviour using the JSON by comparing feature vector against the threshold of the nodes. The output of the binary classification is based on the leaf node values and the user is warned if the webpage is classified as phishing.

## 5.5 COMPLEXITY ANALYSIS

### 5.5.1 TIME COMPLEXITY

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

Table 5.2

S.No	Module Complexity
1	Pre-processing $O(n)$
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 5.2** Time Complexity of various modules

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

## 5.5.2 COMPLEXITY OF THE PROJECT

- The complexity of the project lies in balancing the trade-off between accuracy and rapid detection. Choosing a subset of features that will make the detection fast and at the same time without much drop in accuracy.
- Porting of scikit-learn python object to javascript compatible for- mat. For example, JSON.
- Reproducing the Random Forest behaviour in javascript reduced the accuracy by a small margin.
- Many features are not feasible to extract without using a external web service. Use of an external web service will again affect the detection time.
- Maintaining rapid detection is important as the system should detect the phishing before the user submit any sensitive information.

## CHAPTER 6

### SYSTEM DEVELOPMENT

The system is overall split into backend and plugin. The backend consists of dataset preprocessing and training modules. The frontend which is the plugin consists of javascript files for content script and background script including the Random Forest script. The plugin also consists of HTML and CSS files for the user interface.

#### 6.1 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 the 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 in- put and generates a feature vector with 17 encoded features.
- **Classification:** This module takes the feature vector from feature extraction module and the JSON format from the Exporting mod- el module and then gives a boolean output which denotes whether the webpage is legitimate or phishing.

## 6.2 EXPORTING ALGORITHM

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

### 6.2.1 TREE\_TO\_JSON(NODE):

6.2.1 Converting Decision Trees to JSON as an extension is coded in javascript.

1.  $\text{tree\_json} \leftarrow \{\}$
2. **if** (node has threshold) **then**
3.  $\text{tree\_json}["\text{type}"] \leftarrow \text{"split"}$
4.  $\text{tree\_json}["\text{threshold}"] \leftarrow \text{node.threshold}$
5.  $\text{tree\_json}["\text{left}"] \leftarrow \text{TREE\_TO\_JSON}(\text{node.left})$
6.  $\text{tree\_json}["\text{right}"] \leftarrow \text{TREE\_TO\_JSON}(\text{node.right})$
7. **else**
8.  $\text{tree\_json}["\text{type}"] \leftarrow \text{"leaf"}$
9.  $\text{tree\_json}["\text{values}"] \leftarrow \text{node.values}$
10. **return**  $\text{tree\_json}$

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

6.2.2 Random Forest to JSON as an extension is coded in javascript.

1.  $\text{forest\_json} \leftarrow \{\}$
2.  $\text{forest\_json}['\text{n\_features}'] \leftarrow \text{rf.n\_features\_}$
3.  $\text{forest\_json}['\text{n\_classes}'] \leftarrow \text{rf.n\_classes\_}$
4.  $\text{forest\_json}['\text{classes}'] \leftarrow \text{rf.classes\_}$
5.  $\text{forest\_json}['\text{n\_outputs}'] \leftarrow \text{rf.n\_outputs\_}$
6.  $\text{forest\_json}['\text{n\_estimators}'] \leftarrow \text{rf.n\_estimators}$
7.  $\text{forest\_json}['\text{estimators}'] \leftarrow []$

```
8.  $e \leftarrow \text{rf.estimators}$   
9. for ( $i \leftarrow 0$  to  $\text{rf.n\_estimators}$ )  
10.  $\text{forest\_json}[\text{'estimators'}][i] \leftarrow \text{TREE\_TO\_JSON}(e[i])$   
11. return  $\text{forest\_json}$ 
```

## 6.3 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 7

### RESULTS AND DISCUSSION

#### 7.1 DATASET FOR TESTING

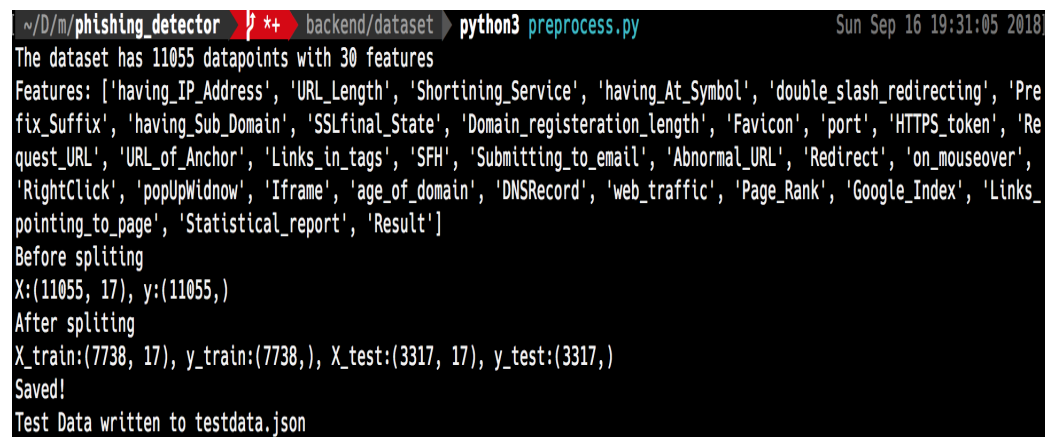
The test set consists of data points separated from the dataset by ratio 70:30. Also the plugin is tested with websites that are listed in PhishTank. New phishing sites are also added to PhishTank as soon as they are found. It should be noted that the plugin is able to detect new phishing sites too. The results of this module testing as well as the testing of the entire system are summarised below.

#### 7.2 OUTPUT OBTAINED IN VARIOUS STAGES

This section shows the results obtained during module testing.

##### 7.2.1 PREPROCESSING

The output of the preprocessing module is shown in figure 7.1.

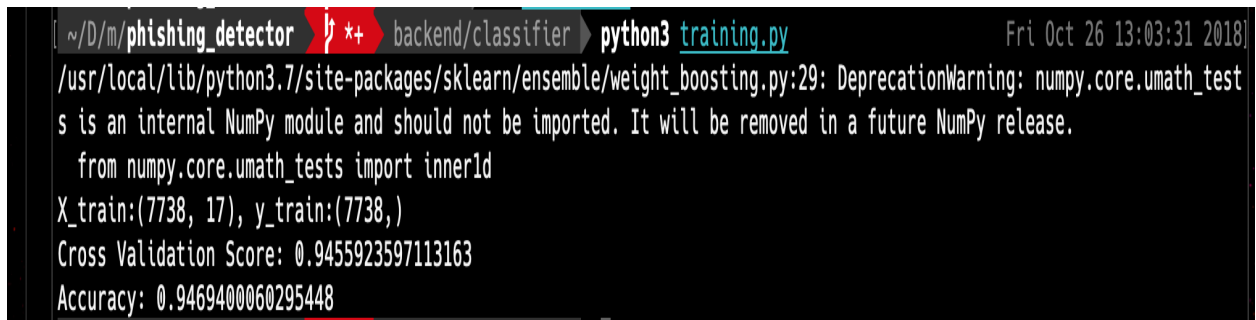


```
~/D/m/phishing_detector ➤ backend/dataset ➤ python3 preprocess.py Sun Sep 16 19:31:05 2018
The dataset has 11055 datapoints with 30 features
Features: ['having_IP_Address', 'URL_Length', 'Shortining_Service', 'having_At_Symbol', 'double_slash_redirecting', 'Pre
fix_Suffix', 'having_Sub_Domain', 'SSLfinal_State', 'Domain_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 7.1** Preprocessing output

## 7.2.2 TRAINING

The output the training module is shown in figure 7.2.

A terminal window with a dark background. The top bar shows the path ~/D/m/phishing\_detector, a red icon with a plus sign, the subdirectory backend/classifier, and the command python3 training.py. The timestamp is Fri Oct 26 13:03:31 2018. The output text is as follows:

```
/usr/local/lib/python3.7/site-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarning: numpy.core.umath_tests 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: 0.9455923597113163  
Accuracy: 0.9469400060295448
```

**Figure 7.2** Training output

### 7.2.3 EXPORTING MODEL

The output the export module is shown in figure 7.3. It outputs a JSON file representing the Random Forest.


```
{
  "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 7.3** Model JSON

## 7.2.4 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 discussed above.

---

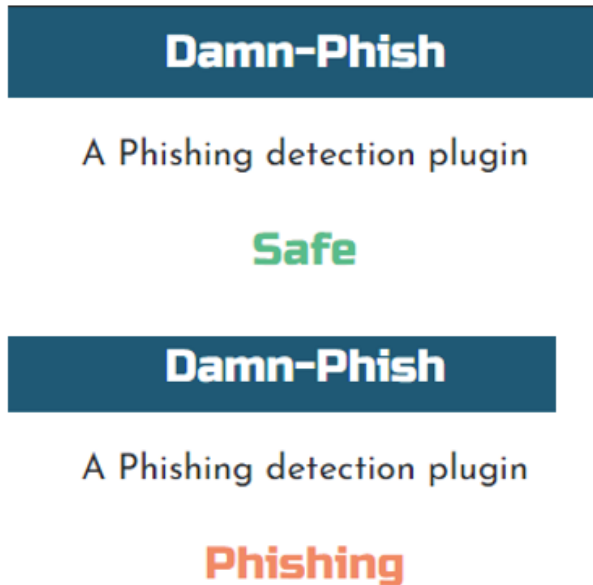
▼ Object 

```
(-) 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 7.4** Webpage features

### 7.2.5 CLASSIFICATION

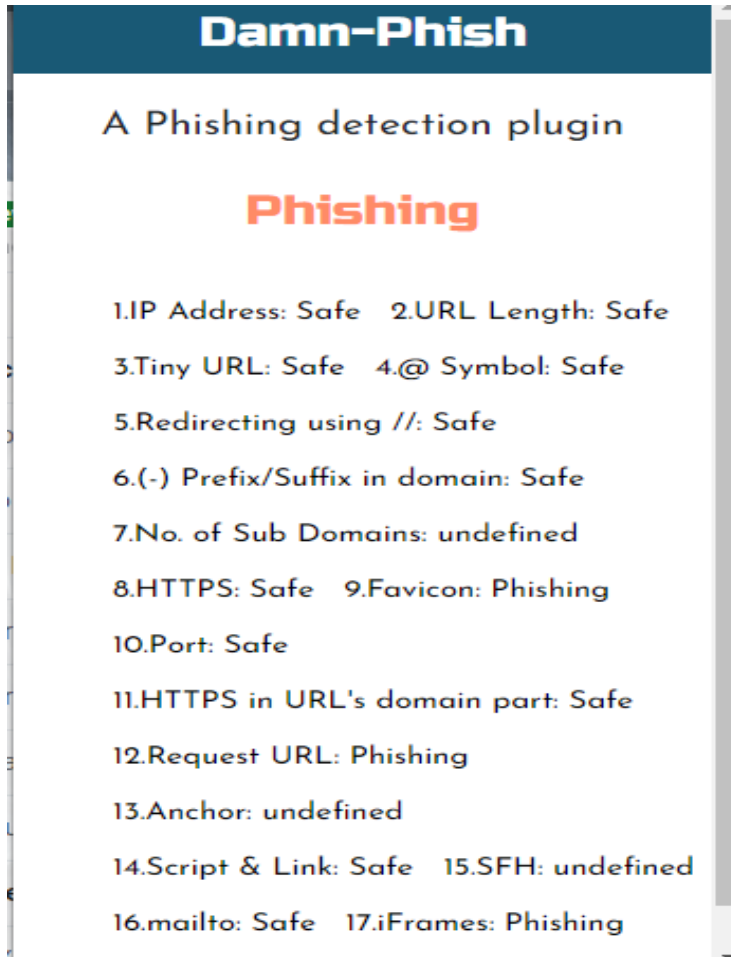
The output of the classification is shown right in the Plugin UI. If it is a safe website then it displays safe with light green colour and if phishing it displays phishing with light red colour.



**Figure 7.5** Classification Output

### 7.3 SAMPLE SCREENSHOTS DURING TESTING

The output of the plugin while visiting a phishing site taken from Damn-Phish. This site has a low trust value it displays phishing with a light red color.



**Figure 7.6** Test Output

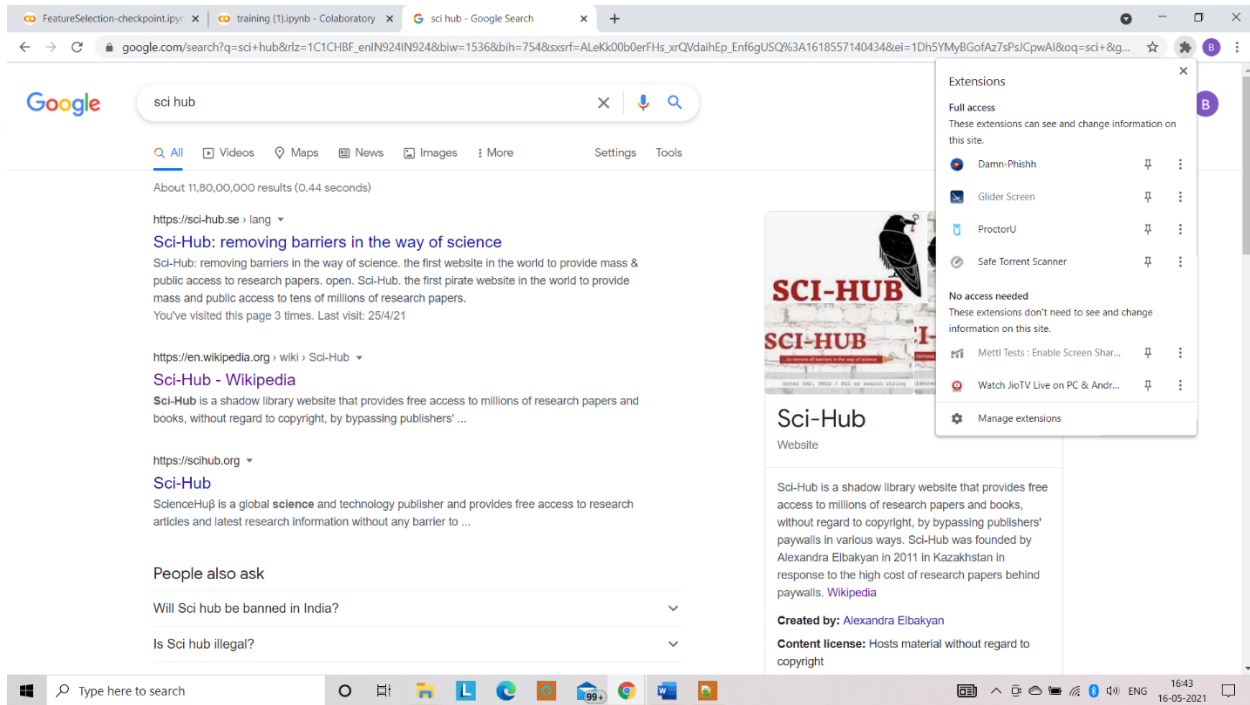


Figure 7.7 Test Output

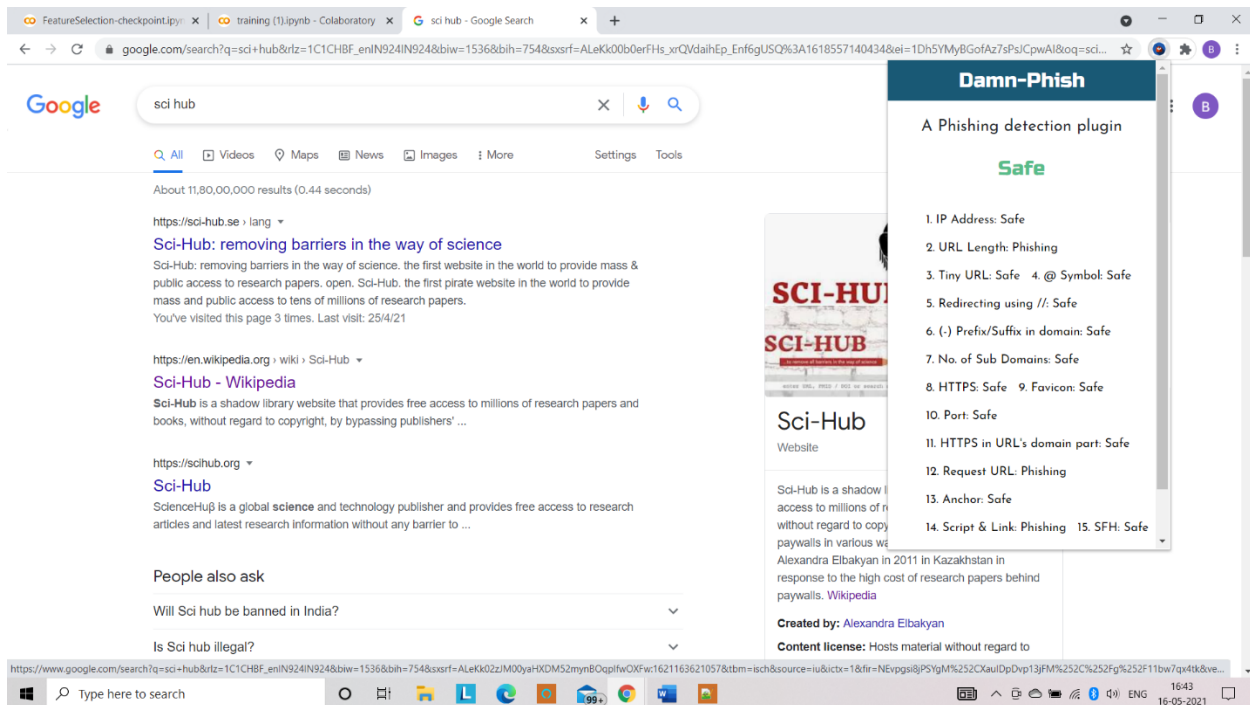


Figure 7.8 Test Output

## 7.4 PERFORMANCE EVALUATION

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

### 7.4.1 CROSS VALIDATION SCORE

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples which are seen would have perfect score but would fail to predict anything useful on unseen data. This situation is called **overfitting**. To solve such problem, another part of the dataset can be taken out as a “**validation set**”: training is done on the training set, after which evaluation is done on the validation set, and when the experiment seems to be successful, final evaluation can be done on the test set. By partitioning the data into three sets, we reduce the samples which can be used for learning the model, and the results can depend on a particular random choice for the pair of sets.

A solution for this problem is cross-validation. Test set should be kept for final evaluation validation set can be ignored. In the basic approach, called k-fold CV, the training set is split into k smaller sets (other approaches are described below, but generally follow the same principles). The following procedure is followed for each of the k “folds”: A model is trained using k of the folds as training data; the resulting model is validated on the left out part of the data.

### 7.4.2 F1 SCORE

F1 score is a measure of a classifier accuracy. It considers both the precision and the recall of the classifier prediction to compute the metric: precision is the number of correct positive results divided by the number of all positive results returned by the classifier, and recall is the number of correct positive results divided by the number of all relevant samples. The F1 score is the harmonic mean of the precision and recall.

F1 score gives its best value at 1 and worst at 0.

The formula for the F1 score is:

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

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 7.8



## Model Metrics

Model: **Random Forest**

**Precision:**

0.9419354838709677

**Recall:**

0.954248366013072

**F1 score:**

0.948051948051948

[Back](#)

**Figure 7.9** Performance measure

## CHAPTER 8

### CONCLUSION

#### 8.1 SUMMARY

This is a phishing website detection system that focuses on client side implementation with rapid detection so that the users will be warned before getting phished. The main implementation is porting of Random Forest classifier to javascript. Similar works often use webpage features that are not feasible to extract on the client side and this results in the detection being dependent on the network. On the other side, this system uses only features that are possible to extract on the client side and thus it is able to provide rapid detection and better privacy. Although using lesser features results in mild drop in accuracy, it increases the usability of the system. This work has identified a subset of webpage feature that can be implemented on the client side without much effect in accuracy. The port from python to javascript and own implementation Random Forest in javascript further helped in rapid detection as the JSON representation of the model and the classification script is designed with time complexity in mind. The plugin is able to detect the phishing even before the page loads completely. The F1 score calculated on the test set on the client side is **0.94**.

#### 8.2 CRITICISMS

The system has a lower accuracy than the state-of-the art but it is more usable and the trade-off between accuracy and rapid detection is handled well enough. The chrome extension API restrictions has a small effect on the plugin. Since the features are extracted in content script which is injected on page load, this plugin can't prevent a malicious javascript code from executing. Further the accuracy reduces from 0.94 to 0.886 while porting from python to javascript and this needs to be investigated. Javascript doesn't support multithreading and browser execute only javascript. Thus the classification can't be made faster by using parallel threads. Currently the results are not cached on the plugin and it's computed repeatedly even for frequently visited sites.

### **8.3 FUTURE WORK**

The classifier is currently trained on 17 features which can be increased provided that, they don't make the detection slower or result in loss of privacy. The extension can be made to cache results of frequently visited sites and hence reducing computation. But this may result in pharming attack being undetected. A solution needs to be devised for caching of results without losing the ability to detect pharming. The classification in javascript can be done using Worker Threads which may result in better classification time. Thus a lot of improvements and enhancements are possible this system offers a more usable solution in the field of phishing detection.

## REFERENCES

- [1] A. Subasi, E. Molah, F. Almkallawi, and T. J. Chaudhery, "Intelligent phishing website detection using random forest classifier," 2017 *International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, Nov. 2017.
- [2] "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+websites).
- [3] J.-H. Li and S.-D. Wang, "PhishBox: An Approach for Phishing Validation and Detection," 2017 *IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, 2017.
- [4] A. A. Ahmed and N. A. Abdullah, "Real time detection of phishing websites," 2016 *IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2016.
- [5] R. Aravindhana, R. Shanmugalakshmi, K. Ramya, and S. C., "Certain investigation on web application security: Phishing detection and phishing target discovery," 2016 *3rd International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2016.
- [6] S. B. K.P, "Phishing Detection in Websites Using Neural Networks and Firefly," *International Journal Of Engineering And Computer Science*, 2016.
- [7] "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.
- [8] 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.
- [9] 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>.

## **Appendix A**

### **LIST OF ABBREVIATIONS**

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

## Appendix B - INSTALLATIONS

### Python - IDLE

IDLE (Integrated Development and Learning Environment) is an integrated development environment (IDE) for Python. The Python installer for Windows contains the IDLE module by default. IDLE can be used to execute a single statement just like Python Shell and also to create, modify, and execute Python scripts. IDLE provides a fully-featured text editor to create Python script that includes features like syntax highlighting, autocompletion, and smart indent. It also has a debugger with stepping and breakpoints features.

- Download python from <https://www.python.org/downloads/>
- Select Version of **Python** to **Install**.
- **Download Python** Executable Installer.
- Run Executable Installer.
- Verify **Python** Was Installed On Windows.
- Verify Pip Was Installed.
- Add **Python** Path to Environment Variables (Optional)
- **Install** virtual NV (Optional)

### Jupyter Lab

Visit <https://jupyter.org/try> and choose 'try **JupyterLab**' option. The launcher tab shows currently available kernels and consoles. You can start a new **notebook** based/terminal based on any of them. The left column is also having tabs for file browser, running kernels and tabs and settings view.

### Chrome

- If prompted, click Run or Save.
- If you chose Save, double-click the download to start **installing**.
- Start **Chrome**: Windows 7: A **Chrome** window opens once everything is done. Windows 8 & 8.1: A welcome dialog appears. Click Next to select your default browser.

## Appendix C

### Software Usage Process

Python is a general purpose and high level programming language. You can use Python for developing desktop GUI applications, websites and web applications. Also, Python, as a high level programming language, allows you to focus on core functionality of the application by taking care of common programming tasks. The simple syntax rules of the programming language further makes it easier for you to keep the code base readable and application maintainable.

### 7 Reasons Why One Must Consider Writing Software Applications in Python

#### 1) Readable and Maintainable Code

While writing a software application, you must focus on the quality of its source code to simplify maintenance and updates. The syntax rules of Python allow you to express concepts without writing additional code. At the same time, Python, unlike other programming languages, emphasizes on code readability, and allows you to use English keywords instead of punctuations. Hence, you can use Python to build custom applications without writing additional code. The readable and clean code base will help you to maintain and update the software without putting extra time and effort.

#### 2) Multiple Programming Paradigms

Like other modern programming languages, Python also supports several programming paradigm. It supports object oriented and structured programming fully. Also, its language features support various concepts in functional and aspect-oriented programming. At the same time, Python also features a dynamic type system and automatic memory management. The programming paradigms and language features help you to use Python for developing large and complex software applications.

#### 3) Compatible with Major Platforms and Systems

At present, Python is supporting many operating systems. You can even use Python interpreters to run the code on specific platforms and tools. Also, Python is an interpreted programming language. It allows you to you to run the same code on multiple platforms without recompilation. Hence, you are not required to recompile the code after making any alteration. You can run the modified application code without recompiling and check the impact of changes made to the code immediately. The feature makes it easier for you to make changes to the code without increasing development time.



#### 4) **Robust Standard Library**

Its large and robust standard library makes Python score over other programming languages. The standard library allows you to choose from a wide range of modules according to your precise needs. Each module further enables you to add functionality to the Python application without writing additional code. For instance, while writing a web application in Python, you can use specific modules to implement web services, perform string operations, manage operating system interface or work with internet protocols. You can even gather information about various modules by browsing through the Python Standard Library documentation.

#### 5) **Many Open Source Frameworks and Tools**

As an open source programming language, Python helps you to curtail software development cost significantly. You can even use several open source Python frameworks, libraries, and development tools to curtail development time without increasing development cost. You even have option to choose from a wide range of open source Python frameworks and development tools according to your precise needs. For instance, you can simplify and speedup web application development by using robust Python web frameworks like Django, Flask, Pyramid, Bottle and CherryPy. Likewise, you can accelerate desktop GUI application development using **Python GUI frameworks** and toolkits like PyQt, PyJs, PyGUI, Kivy, PyGTK and WxPython.

#### 6) **Simplify Complex Software Development**

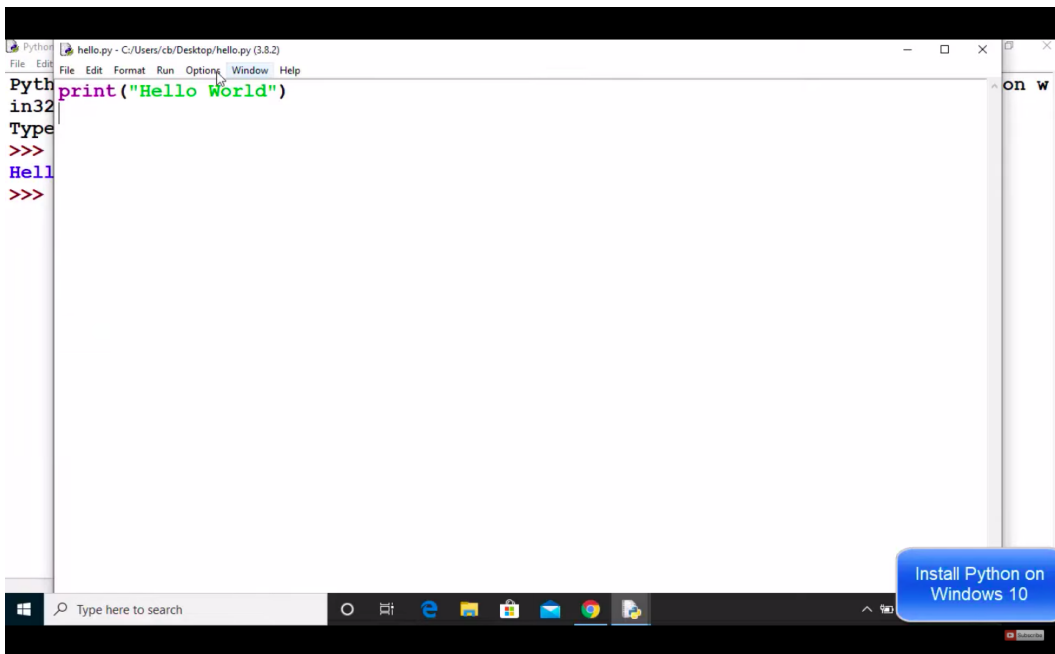
Python is a general purpose programming language. Hence, you can use the programming language for developing both desktop and web applications. Also, you can use Python for developing complex scientific and numeric applications. Python is designed with features to facilitate data analysis and visualization. You can take advantage of the data analysis features of Python to create custom big data solutions without putting extra time and effort. At the same time, the data visualization libraries and APIs provided by Python help you to visualize and present data in a more appealing and effective way. Many **Python developers** even use Python to accomplish artificial intelligence (AI) and natural language processing tasks.

#### 7) **Adopt Test Driven Development**

You can use Python to create prototype of the software application rapidly. Also, you can build the software application directly from the prototype simply by refactoring the Python code. Python even makes it easier for you to perform coding and testing simultaneously by adopting test driven development (TDD) approach. You can easily write the required tests before writing code and use the

tests to assess the application code continuously. The tests can also be used for checking if the application meets predefined requirements based on its source code.

However, Python, like other programming languages, has its own shortcomings. It lacks some of the built-in features provided by other modern programming language. Hence, you have to use Python libraries, modules, and frameworks to accelerate custom software development. Also, several studies have shown that Python is slower than several widely used programming languages including Java and C++. You have to speed up the Python application by making changes to the application code or using custom runtime. But you can always use Python to speed up software development and simplify software maintenance.



The image shows a screenshot of a Windows desktop with a Python 3.8.2 Shell window open. The window title is "Python 3.8.2 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area displays the following content:

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>>
===== RESTART: C:/Users/cb/Desktop/hello.py =====
Hello World
>>> |
```

The Windows taskbar is visible at the bottom, showing the search bar with the text "Type here to search" and several application icons. A blue notification bubble in the bottom right corner of the taskbar says "Install Python on Windows 10".

## 8.1 Demo of Python usage in IDE