Sri Lanka Institute of Information Technology

Identification of Phishing Websites using Random Forest Classifier Algorithm

Project Report

SE4060 - Machine Learning

IT15135308
B.Kiruthiga

Contents

List of Figures

List of Tables

List of Abbreviations

URL - Uniform Resource Locator
SFH - Server Form Handler
SSL - Secure Socket Layer
PCA - Principal Component Analysis
SVM - Support Vector Machines

1. Introduction

1.1 Problem Statement

Phishing is a criminal mechanism to collect sensitive details like personal information, credit card details and passwords by pretending like a trustworthy object in the digital world. In the context of websites, the confidential information is obtained by tricking the user to believe that he is on a legitimate webpage. Further, phishing leads to login to other person's account and may result in financial loss.

In modern phishing, a copy of an official well-known website could be a phishing website. This can be identified by the Uniform Resource Locator (URL) in the address bar. The URL would look like the original website. However, if we notice the URL carefully, there are some letters that are missed and some are incorporated to it. The following figure 1 shows the legitimate login page of Facebook and figure 2 shows the phishing webpage of it.

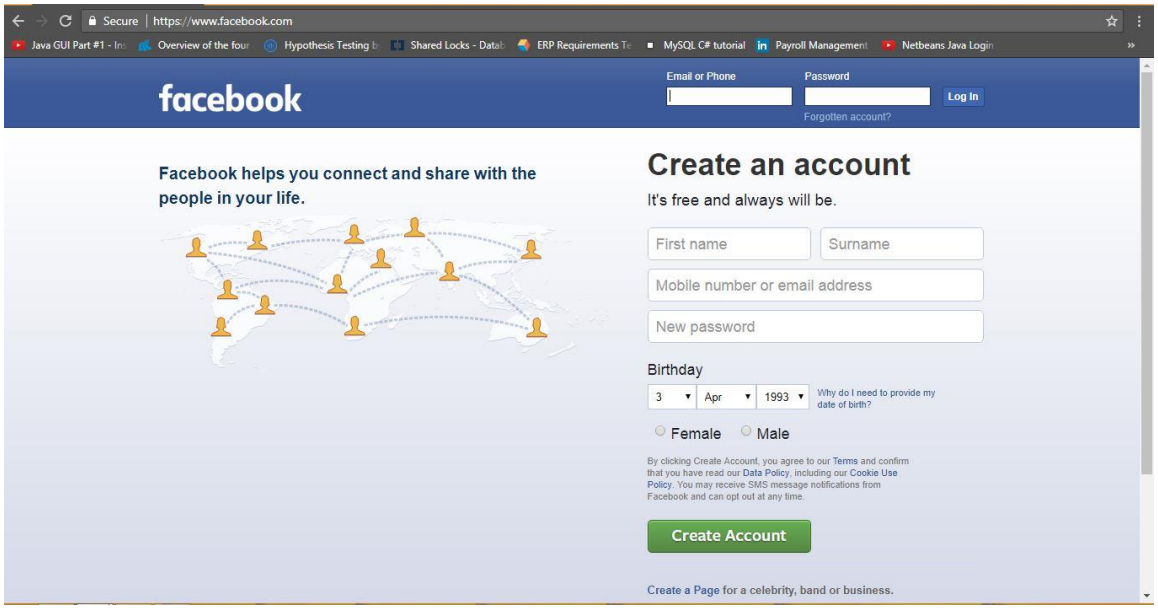

Figure 1 - Legitimate webpage of Facebook

Figure 2 - Phishing webpage of Facebook

1.2 Product Scope

Detecting the phishing websites is essential to safeguard the personal information. Most of the web users unintentionally access the phishing webpages daily perhaps for every hour. Attackers target both on the digital industry and every individual. This leads industry to face high loss in the financial state. The absence of awareness of the users is the main reason to steal the sensitive information. People should be aware of it and security defenders should protect users from accessing the phishing domains.

In this report, the detection process of phishing websites will be discussed in detail through Machine Learning Algorithm called Random Forest Classifier. This algorithm trains a model using the given dataset of features of phishing URLs and identify whether a URL is legitimate, suspicious or phishing. In this way, a software could be developed and incorporated to the browser so that users will be protected from sharing their information to the attackers. This would also have the following key-benefits to the users.

- Safeguard the personal information
- Deny access to phishing domains
- Alert users while they type the phishing URL in the address bar
- Provide with a list of phishing websites

This mechanism would also benefit the corporate world by saving their important credentials and financial information. Investing on this type of software will prevent from bankrupt and promote their business strategies without any obstacles.

2. Methodology

2.1 Data Collection

2.1.1 Dataset

The dataset for this report was taken from UCI Machine Learning Repository [1]. It contains features of all three classes: legitimate, suspicious and phishing. The data for legitimate websites were gathered from Yahoo and using a PHP webscript which is plugged to the browser [1]. Phishtank data repository [2] was used to obtain data for phishing websites. The characteristics of this dataset are listed in the following table.

Source of the dataset	[1]
Number of instances	1353
Number of attributes	10 (including class)
Number of classes	03
Number of missing values	Null
Data type	Multivariate
Attribute type	Integer
Realated tasks	Classification
Number of legitimate websites	548
Number of suspicious websites	103
Number of phishing websites	702

Table 1 - Characteristics of dataset

All the attributes in this dataset holds the following three categorical values

- 1 if the feature is legitimate
- 0 if the feature is suspicious (i.e. either legitimate or phishing)
- -1 if the feature is phishing

2.1.2 Description of Attributes

These are the attributes that are used in the dataset. The description and how the classes are classified to the attributes are as follows [3]

1) Server Form Handler (SFH)

SFH is used to handle actions of forms in server side. If SFH contains an empty string then this is acknowledged to be doubtful as there should be an action taken from the submitted information.

$$\text{IF} \begin{cases} \text{SFH is "about: blank" Or Is Empty} \rightarrow \text{Phishing} \\ \text{SFH Refers To A Different Domain} \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

2) Popup window

It occurs in both legitimate and phishing sites. If the popup window asks personal information then that site is considered as phishing. Some legitimate sites use popup window to alert the user or to show some announcements

$$\text{IF} \begin{cases} \text{Popoup Window Contains Text Fields} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

3) SSL final state

Websites with URL starting of 'https' and have a SSL certificate are mostly considered to be legitimate websites otherwise phishing.

$$\text{IF} \begin{cases} \text{Use https and Issuer Is Trusted and Age of Certificate} \geq \text{1 Years} \rightarrow \text{Legitimate} \\ \text{Using https and Issuer Is Not Trusted} \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

4) Request URL

Request URL finds whether there are any images, audios and videos in the current site that are loaded from another domain. If this is the case, then this site would be known as Phishing site. A legitimate site normally loads all the necessary media from its domain itself.

$$\text{IF} \begin{cases} \%\text{ of Request URL} < 22\% \rightarrow \text{Legitimate} \\ \%\text{of Request URL} \geq 22\% \text{ and } 61\% \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{feature} = \text{Phishing} \end{cases}$$

5) URL of anchor

This attribute is similar to Request URL. However, the <a> tag is examined here. This tag is used to link to another page. If the anchor tag doesn't have any links then it is considered as Phishing.

$$\text{IF} \begin{cases} \%\text{ of URL Of Anchor} < 31\% \rightarrow \text{Legitimate} \\ \%\text{ of URL Of Anchor} \geq 31\% \text{ And} \leq 67\% \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

6) Web traffic

This attribute is defined by the number of pages being visited by the users. There is a web ranking defined by the Alexa database [4]. Legitimate websites are ranked to come under 100,00 otherwise they are known as suspicious. If the websites have no traffic and not being ranked by the Alexa then it considered as Phishing

$$\text{IF} \begin{cases} \text{Website Rank} < 100{,}000 \rightarrow \text{Legitimate} \\ \text{Website Rank} > 100{,}000 \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

7) URL length

There is no defined rule for the URL length to decide whether a site is legitimate. However, researches have proved and showed the higher accuracy for certain URL length

$$\text{IF} \begin{cases} \text{URL length} < 54 \rightarrow \text{feature} = \text{Legitimate} \\ \text{else if URL length} \geq 54 \text{ and } \leq 75 \rightarrow \text{feature} = \text{Suspicious} \\ \text{otherwise} \rightarrow \text{feature} = \text{Phishing} \end{cases}$$

8) Age of domain

Most of the phishing web domains remain for a short period.

$$\text{IF} \begin{cases} \text{Age Of Domain} \geq 6 \text{ months} \rightarrow \text{Legitimate} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

9) Having an IP address

Phishing sites use IP address as a part of the domain name.

$$\text{IF} \begin{cases} \text{If The Domain Part has an IP Address} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

2.2 Data Preprocessing

Initially, the dataset is read from the Excel file. Features and the result columns are separated. The first five instances of the dataset are shown in the following figure.

	SFH	popUp Window	SSL Final State	Request URL	URL of Anchor	Web Traffic	URL Length	Age of Domain	Having IP Address	Result
0	1	-1	1	-1	-1	1	1	1	0	0
1	-1	-1	-1	-1	-1	0	1	1	1	1
2	1	-1	0	0	-1	0	-1	1	0	1
3	1	0	1	-1	-1	0	1	1	0	0
4	-1	-1	1	-1	0	0	-1	1	0	1

Figure 3 - First five instances of the dataset

Standardizing the dataset is important so that the features will be scaled into one-unit scale where the mean is zero and variance is one. This will help in optimal performance of the classifier. Label Encoding is one of the technique to standardize the categorical variables.

Principal Component Analysis (PCA) is used to analyze the patterns of the data and identify the correlation between variables. If there is high correlation between variables, then reducing the dimension of the data would help to solve the problem. PCA could be used in the following situations.

1. **For visualization**

When the data has too many features (more than three), visualizing it in a plot is impossible. Therefore, PCA is used to reduce the n-dimension into 2 or 3-dimension to view the graph.

2. **To speed up the machine learning algorithm**

High dimension of the dataset will slow down the performance of the algorithm. Reducing the dimension would help to sort the problem.

Our dataset has nine features. PCA has chosen eight principal components to retain the 95% of variance. The following figure shows the eight principal components of first five instances of the dataset after implying PCA.

	0	1	2	3	4	5	6	7	Result
0	-0.040007	-0.534025	1.396626	0.657558	0.070759	0.410673	-1.317227	1.134072	0
1	1.126874	-1.694566	-0.602241	1.499749	0.151729	-0.102142	-0.358969	0.431160	1
2	-0.048551	-0.771857	0.049534	-1.083485	0.707807	-0.547327	-1.039926	0.308606	1
3	-0.655722	-0.809362	1.487724	0.779619	0.072811	-0.126211	-0.400537	0.517497	0
4	0.462426	-1.250212	-0.285455	-0.897072	-0.932769	1.017973	0.058857	0.286242	1

Figure 4 - Dataset after implying PCA

Dataset should be divided into two sets; training set and test set. A library called train_test_split is used to separate the dataset. This dataset is divided as 75% for training and 25% for testing.

2.3 Random Forest Classifier

It is a supervised learning algorithm and an ensemble classifier as it uses multiple classifiers to predict the class. As the name implies, there is a forest of number of decision trees(In this dataset, there are 1000 decision trees being used) used to obtain the result. Each decision tree uses randomly selected subsets of training instances to predict a result and high voted predicted value is considered as the outcome.

2.3.1 Selection of the algorithm

Random Forest Classifier reduces overfitting since it obtains result from many trees. It is easy to model categorical variables compared to other learning algorithms. This learning algorithm handles missing values of the dataset. It also has a higher accuracy for predicted data. Understanding the logic is simplified in this algorithm.

2.3.2 Basic concept of Decision Tree

It is a supervised learning algorithm used to solve both classification and regression problems. Decision Tree creates a training model by using decision rules from the training data. Each internal node represents the attribute of the dataset and each leaf node depicts the target class. The following pseudocode explains how the Decision Tree algorithm works.

Pseudocode of Decision Tree algorithm

1) The best choice of attribute is chosen and placed as root node

2) The training set is divided into subsets so that every subset will have the data of identical value for an attribute
3) Step 1 and 2 are repeated until the leaf nodes are created for all the branches

Identifying the best attribute for the root and the other nodes is a challenging task. Some of the popular mechanisms for attribute selection are information gain and Gini index. In our dataset, Gini index is used. It calculates how a randomly selected element is incorrectly detected. Finally, an attribute with lower Gini value is considered. The following figure shows one decision tree of three levels from the forest.

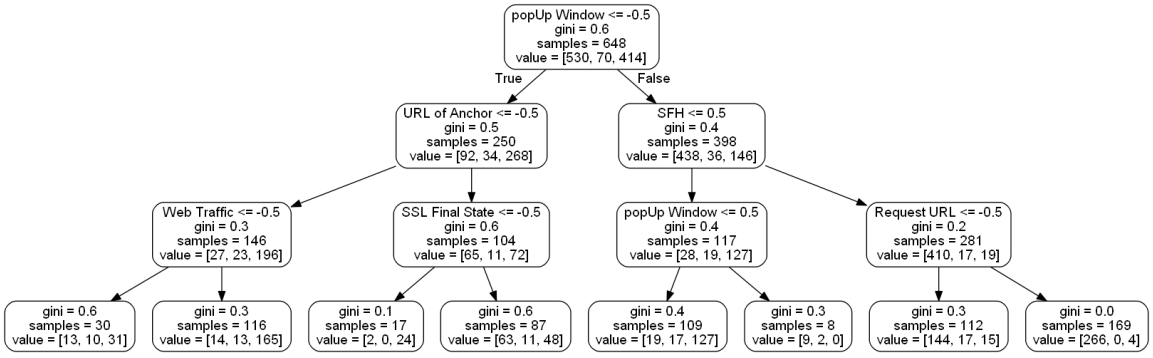

Figure 5 - Decision Tree with three levels

Refer the Appendix A to view the full Decision Tree

2.3.3 Implementation of Random Forest Classifier

The following pseudo codes are used in this algorithm.

- Creation of Random Forest
- Prediction from Random Forest

Creation of Random Forest

1) Selecting ‘x’ number of features randomly out of total ‘p’ features where x<<p
2) Using Gini index (best split approach), choose the root node out of ‘x’ number of features
3) Splitting the root node using the same best split approach to find child nodes
4) Repeating the first three stages until we find the leaf nodes (reaching target values)
5) Building the forest by repeating the first four stages for ‘n’ number of times to have ‘n’ number of trees in the forest

Prediction from Random Forest

1) Using the trained Random Forest Classifier, test features are passed through the rules of each randomly created trees and the predicted value is saved for each tree. (Each tree can predict different outcome for a same test feature)
2) Calculating the votes given by each tree for a test feature

3) Finding the high voted outcome is considered as the final prediction of the algorithm

2.4 Testing

These are the following testing methods being used in this classifier.

1) Finding the accuracy of the predicted data

Using the test data (25% of the dataset), predictions are made from the classifier. The accuracy of test data obtained for this classifier is 90.26%

2) Calculating the cross-validation score for the dataset using K-fold cross validation

In this method, the dataset is divided into K subsets. This will run for K times where one of the subset will be test set and the other K-1 sets are used for training. Therefore, each subset is validated once and trained K-1 times. Hence, it reduces bias as most of the data is being used for training and reduces variance as most of the subsets are validated. In this dataset, we have taken K=10

We have obtained the accuracy of 89.88% for the 10-fold cross validation.

3) Generating a confusion matrix to compare with predicted and true labels

Confusion matrix is used to count how many times the true value and the predicted value are equal for a class. If they are not equal at all, the value will remain as zero. Otherwise, the value gets added.

4) Generating classification report for the predictions

Classification report is used to create a text report containing all the necessary classification metrics. These are the following metrics being used.

- **Precision**
  This gives the score of a class for a negative sample that shouldn't be labelled as positive.
- **Recall**
  This gives the score of a class for a sample that is identified correctly by the classifier.
- **F1-score**
  This gives the accuracy of a class for classifying the samples belong to that class compared to other classes
- **Support**
  This gives the number of samples of the dataset for a class

All the metrics have produced a score above 0.8 to our predicted data with the average of 0.9.

Refer the Appendix B to view the snippets and figures of these test results

3. Evaluation

3.1 Assessment of the Project results

Random Forest Classifier algorithm has provided higher accuracy for this dataset. Even though it has some limitations like quite slow in creating prediction, it is a simple flexible predictive modelling. This dataset was also tried training with Logistic Regression and Support Vector Machines (SVM) and the following table shows the results obtained for the test data.

Classifier	**Success Rate (%)**	**Error Rate (%)**
Random Forest Classifier	90.26	9.74
Logistic Regression	85.20	14.8
SVM	83.88	16.12

Table 2 - Performance of Classifiers

The following improvements can be done to increase the accuracy rate of the Random Forest Classifier.

- Allocating more data for training would help to train the model a lot and could result in high success rate.
- Tuning the parameters of Random Forest Classifier such as depth of a tree, number of trees and number of features used in a split.
- Selecting the most important features of the dataset would increase the accuracy rate.

3.2 Lessons Learned

- Getting hands on experience with Python Machine Learning Algorithms using Scikit libraries was a great lesson
- Even though the classifiers were taken from the library, it was difficult to understand how they work. Learning them was another important lesson

3.3 Future Work

- Training this dataset with more complex Machine Learning algorithms like Artificial Neural Network and calculating the accuracy should be done. This helps to understand the depth of handling data.
- Trying different feature engineering techniques would lead to choose the important features.
- Different methods from various libraries are used to obtain the same result. Analyzing and trying those methods would help to understand the differences with the currently used methods.

4. References

[1] "UCI Machine Learning Repository: Website Phishing Data Set", *Archive.ics.uci.edu*, 2018. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Website+Phishing. [Accessed: 20- Mar- 2018].

[2] "PhishTank | Join the fight against phishing", *Phishtank.com*, 2018. [Online]. Available: https://www.phishtank.com/. [Accessed: 24- Mar- 2018].

[3] R. Mohammad, L. McCluskey and F. Thabtah, "Intelligent rule-based phishing websites classification", IET Information Security, vol. 8, no. 3, pp. 153-160, 2014.

[4] "Keyword Research, Competitor Analysis, & Website Ranking | Alexa", Alexa Internet, 2018. [Online]. Available: https://www.alexa.com/. [Accessed: 02- Apr- 2018].

5. Appendix

Appendix A - Diagrams

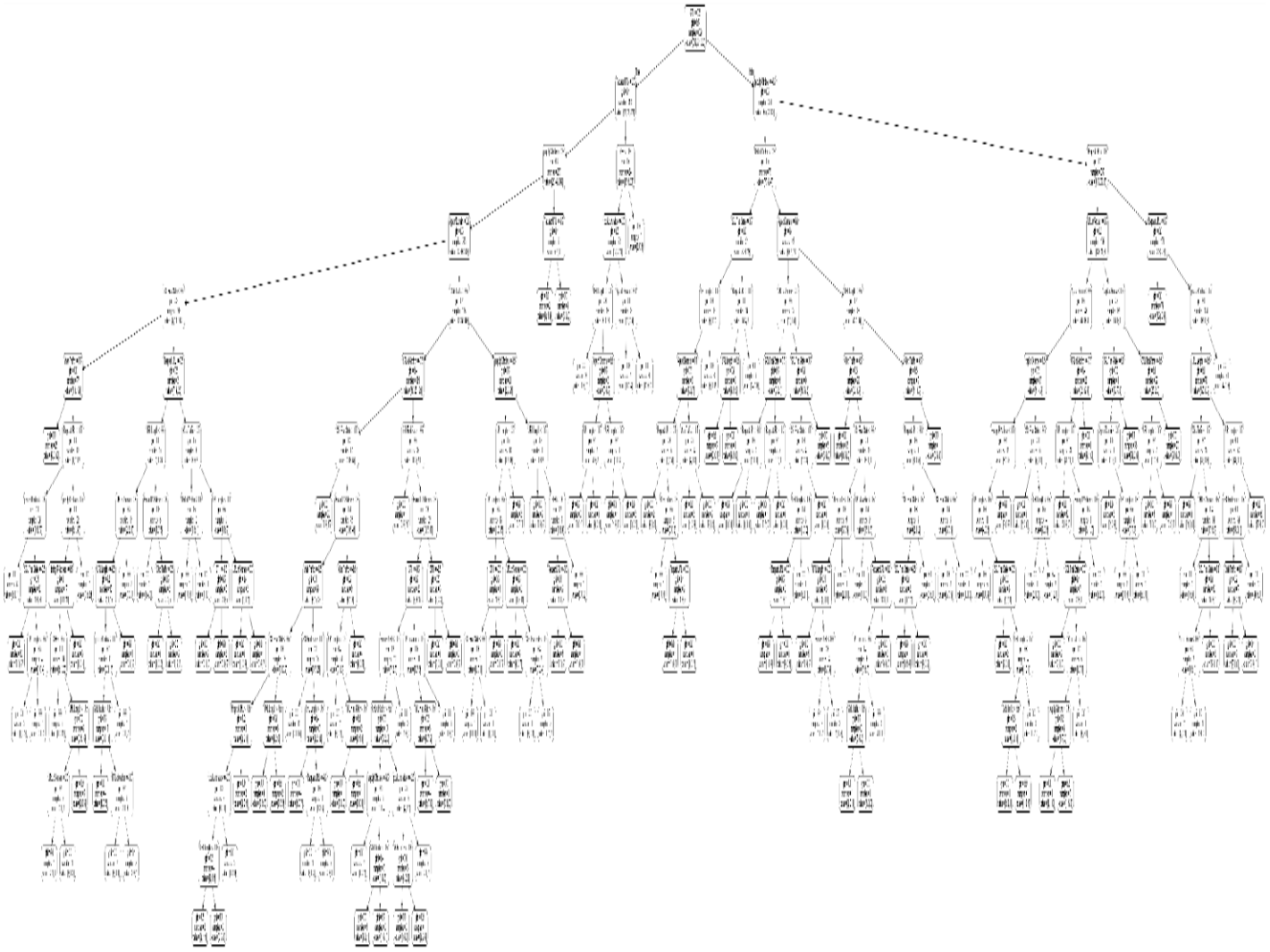

Figure 6 - Decision Tree of the classifier

A clear view of the tree image is attached with this report

Appendix B: Test Results

```
# Finding the accuracy of the predicted data compared to the test values
accuracy = 100.0 * accuracy_score(test_labels, predictions)
print ("The accuracy of your algorithm on testing data is: " + str(accuracy))
```

```
The accuracy of your algorithm on testing data is: 90.2654867256637
```

```
# Calculating the cross validation score of the dataset using the 10 fold cross validation
scores = cross_val_score(RFC, X, Y,cv=10)
print("10-fold cross validation average accuracy: " + str(scores.mean()*100.0) )
```

```
10-fold cross validation average accuracy: 89.88622906886302
```

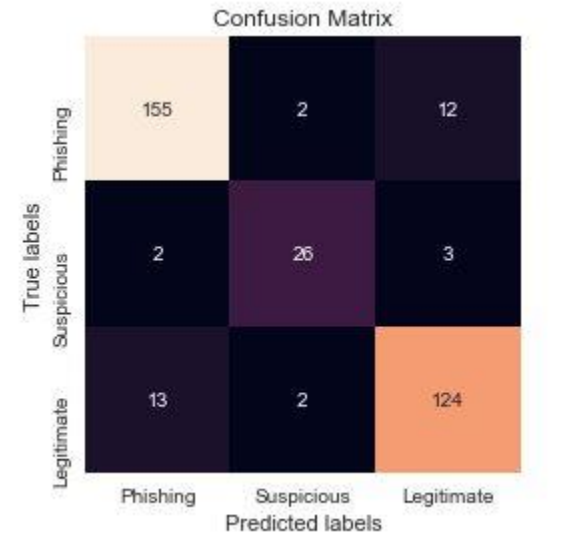

Figure 7 - Confusion Matrix

	precision	recall	f1-score	support
Phishing	0.92	0.91	0.92	171
Suspicious	0.84	0.87	0.85	30
Legitimate	0.89	0.90	0.90	138
avg / total	0.90	0.90	0.90	339

Figure 8 - Classification Report

Appendix C: Code Listings

```
# Importing all necessary libraries to work with the dataset
# Panda library is used to get a dataframe with rows and columns out of the dataset
import pandas as pd
# Seaborn library is used to provide an interface to generate statiscal visualization
import seaborn as sns; sns.set()
# Matplotlib library is used to plot all types 2D graphs in python
import matplotlib.pyplot as plt
# Numpy library is used to handle mathematical and numerical functions
import numpy as np
# We have used PCA to reduce the dimension of the dataset
from sklearn.decomposition import PCA
# Train_test_split package is used to split the training and testing data from the dataset
from sklearn.model_selection import train_test_split
# Importing the classifier that we will be working
from sklearn.ensemble import RandomForestClassifier
# This package is used to obtain the accuracy of the classifier
from sklearn.metrics import accuracy_score
# Cross validation library helps to get the accuracy of the cross validation set
from sklearn.cross_validation import  cross_val_score
# Confusion matrix is used to get the statistics between the true and predicted labels
from sklearn.metrics import confusion_matrix
# This package is used to generate the classification report
from sklearn import metrics
# In order to plot the tree we need to add these two lines to the PATH
import os
os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/bin/'
# We are importing essential packages to visualize the decision trees.
from sklearn.tree import export_graphviz
import pydot
# Reading the Phishing websites data from the excel file and saving them to a variable
'features'
features = pd.read_excel('dataset.xlsx')
# Displaying the first 5 rows of the dataset
features.head(5)
# X variable contains all data without the Result column.Axis 1 refers to the columns
X= features.drop('Result', axis = 1)
# Y variable contains only the Result column of the dataset
Y= features.loc[:,['Result']].values
# Converting Y into 1D array
Y= Y.ravel()
# Saving the column headings(feature names)
feature_list = list(X.columns)
target_list =['Phishing', 'Suspicious', 'Legitimate']
print(feature_list)
```

```
# Choose the minimum number of principal components so that 95% of variance is
maintained
pca = PCA(0.95)
# Fitting the features data and applying transform
principalComponents = pca.fit_transform(X)
# Framing the number of components into a data frame
pcaDf = pd.DataFrame(data = principalComponents)
# Adding the result column
finalDf = pd.concat([pcaDf, features[['Result']]], axis = 1)
# Viewing the first 5 rows of the final dataframe
finalDf.head(5)
# Splitting the dataset into training and testing sets
train_features, test_features, train_labels, test_labels = train_test_split(X, Y, test_size =
0.25, random_state = 42)
print('Training Features Shape:', train_features.shape)
print('Training Labels Shape:', train_labels.shape)
print('Testing Features Shape:', test_features.shape)
print('Testing Labels Shape:', test_labels.shape)
# Using the RandomForestClassifier model with 1000 decision trees
RFC = RandomForestClassifier(n_estimators=1000)
# Training the classifier on training data
RFC.fit(train_features, train_labels);
# Predicting the classifier with the test data
predictions = RFC.predict(test_features)
# Finding the accuracy of the predicted data compared to the test values
accuracy = 100.0 * accuracy_score(test_labels, predictions)
print ("The accuracy of your algorithm on testing data is: " + str(accuracy))
# Calculating the cross validation score of the dataset using the 10 fold cross validation
scores = cross_val_score(RFC, X, Y,cv=10)
print("10-fold cross validation average accuracy: " + str(scores.mean()*100.0) )
# Generating classification report for the predictions
print(metrics.classification_report(predictions, test_labels,target_names=target_list))
# Finding the values for the confusion matrix by comparing the predicted values with the
test values
confusionMatrix = confusion_matrix(test_labels, predictions)
subGraph= plt.subplot()
# Generating a map to visualize the matrix
sns.heatmap(confusionMatrix, square=True, annot=True, fmt='d', cbar=False,
ax=subGraph); #annot=True to annotate cells
# Setting the labels for X and Y axis
subGraph.set_xlabel('Predicted labels');
subGraph.set_ylabel('True labels');
# Setting title
subGraph.set_title('Confusion Matrix');
# Setting the class labels for both axis
subGraph.xaxis.set_ticklabels(['Phishing', 'Suspicious','Legitimate']);
```

```
subGraph.yaxis.set_ticklabels(['Phishing', 'Suspicious','Legitimate']);
# From the forest we are going to take one tree
oneTree = RFC.estimators_[5]
# We are exporting the decision tree to a dot file
export_graphviz(oneTree, out_file = 'oneTree.dot', feature_names = feature_list, rounded
= True, precision = 1)
# With the help of the dot file we are generating the graph
(graph, ) = pydot.graph_from_dot_file('oneTree.dot')
# To view the image, we are converting the dot file to png format
graph.write_png('oneTree.png')
# Limiting the depth of the single tree to 3 levels
RFC_small = RandomForestClassifier(n_estimators=10, max_depth = 3)
RFC_small.fit(train_features, train_labels)
# Extracting the small tree
decisionTree_small = RFC_small.estimators_[5]
# Saving the tree to png format
export_graphviz(decisionTree_small, out_file = 'small_tree.dot', feature_names =
feature_list, rounded = True, precision = 1)
(graph, ) = pydot.graph_from_dot_file('small_tree.dot')
graph.write_png('small_tree.png');
```