Journal of Advances in Mathematical & Computational Sciences,
Vol. 7 No. 2, June 2019

Article Citation Format
Longe, O.B, Okorejior, M.S. & Isong, E.B. (2019): Design Architecture for Training Random Forest Classifiers to Detect Phishing Attacks. Journal of Advances in Mathematical & Computational Sciences Vol. 7, No. 2. Pp 1-8

Article Progress Time Stamps

Article Type: Research Article
Manuscript Received 13th April. 2019
Review Type: Blind
Final Acceptance: 17th June, 2019
Article DOI:
dx.doi.org/10.22624/AIMS/MATHS/V7N2P1

Design Architecture for Training Random Forest Classifiers to Detect Phishing Attacks

Longe, O.B (PhD) & Okorejior, M.S.
Information Systems Programme
American University of Nigeria
Yola, Adamawa State, Nigeria
E-mails: Olumide.longe@aun.edu.ng; mohammed.okorejior@aun.edu.ng

Isong, E.B. (PhD)
Department of Computer Science
Akwa Ibom State University
Ikot-Akpaden, Akwa Ibom State, Nigeria
E-mail: etebongisong@aksu.edu.ng

ABSTRACT

An attempt is made in this paper to develop an anti-phishing system using the Random Forest (RF) classifier/algorithm. The RF classifier is trained on phishing sites dataset using python scikit-learn. A JSON format to represent the random forest classier was devised and the learned classifier is exported to the same. A browser script was then implemented which uses the exported model JSON to classify the website being loaded in the active browser tab. The intention is to initiate a warning system that prompts users in the event of phishing. The architecture for the development is presented in this paper.

Keywords: Random, Classifier, Forest, Alorithm, Phising, Users, Security and JSON

1. INTRODUCTION

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 [1][2]. 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.

One way for an end user to benefit from this is to implement detection in a browser plugin. So that the user can be warned in real time as he browses a phishing site. However, browser extensions have restrictions such as they can be written only in JavaScript and they have limited access to page URLs and resources. 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 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 considering the privacy aspects, we decided to implement this on a chrome browser plugin which can do the classification without an external server [3][4][5].

2. SYSTEM ARCHITECTURE

Random Forest classifier on 17 features of a website is used to classify whether the site is phishing or legitimate. The dataset raff file is loaded using python raff library and 17 features are chosen from the existing 30 features [6]. 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 JSON in cache and downloads it again in case it is not there in cache. The System Architecture is shown in Fig. 1

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

2.1 User Interface (UI) Design

A simple and easy to use User Interface has been designed for the plugin using HTML and CSS. The UI contains a large circle indicating the percentage of the legitimacy of the website in active tab. The circle also changes its color with respect to the classification output (Green for legitimate website and Light Red for phishing). Below the circle, the analysis results containing the extracted features are displayed in the following color code.

Green - Legitimate
Yellow - Suspicious
Red - Phishing

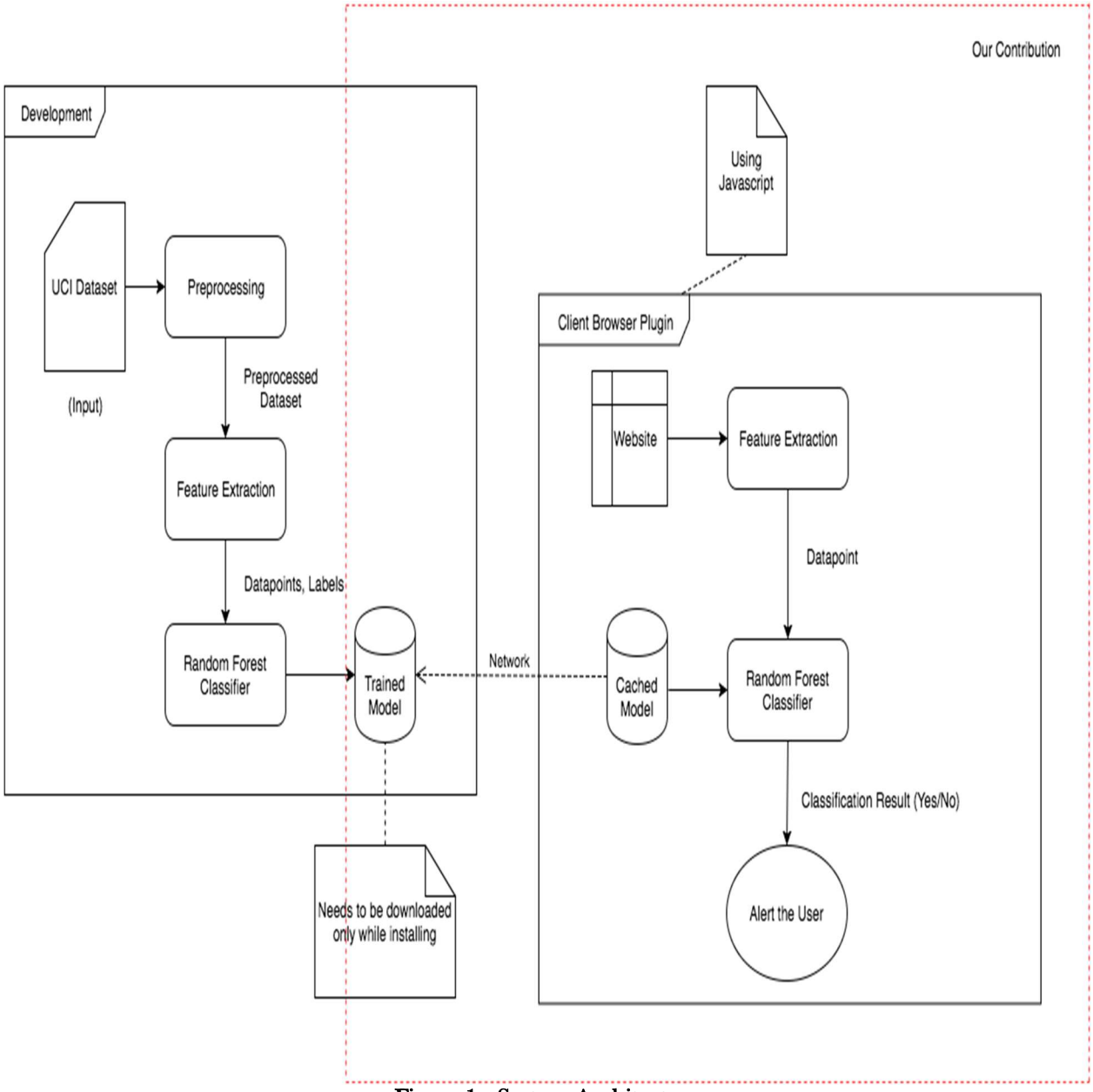

Figure 1: System Architecture

2.2 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.

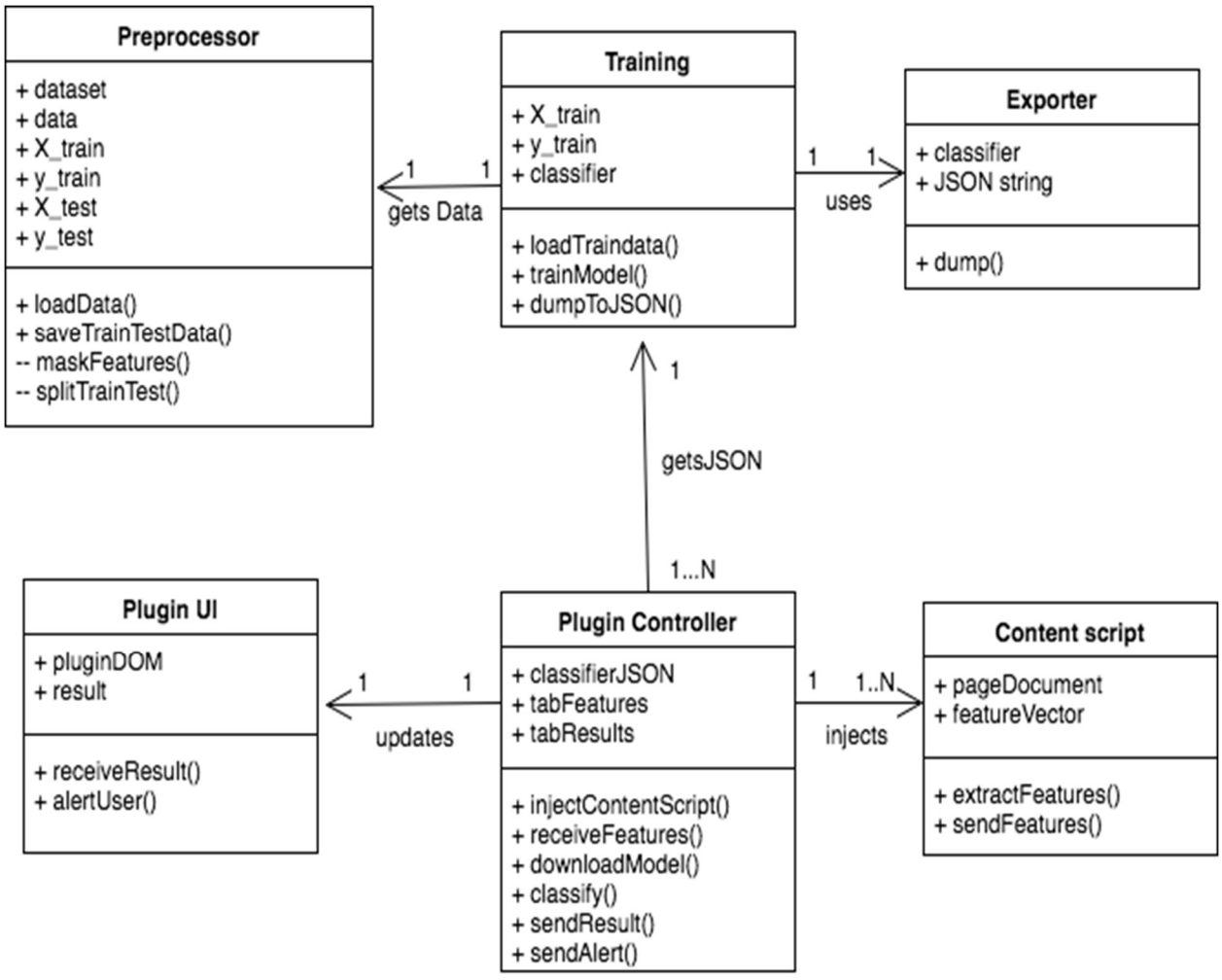

Figure 2: Class Diagram

2.3 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.

Thus a subset of features is chosen in a way that the tradeoff is balanced.

1. IP address
2. Degree of subdomain
3. Anchor tag href of domains
4. URL length
5. HTTPS or not
6. Script & link tag domains
7. URL shortener
8. Favicon domain
9. Empty server form handler
10. '@' in URL
11. TCP Port
12. Use of mailto
13. Redirection with '//'
14. HTTPS in domain name
15. Use of iFrame
16. "-" in domain
17. Cross domain requests

The plugin also displays an alert warning in case of phishing to prevent the user from entering any sensitive information on the website. The test results such as precision, recall and accuracy are displayed in a separate screen. The UI is shown in Fgure 3

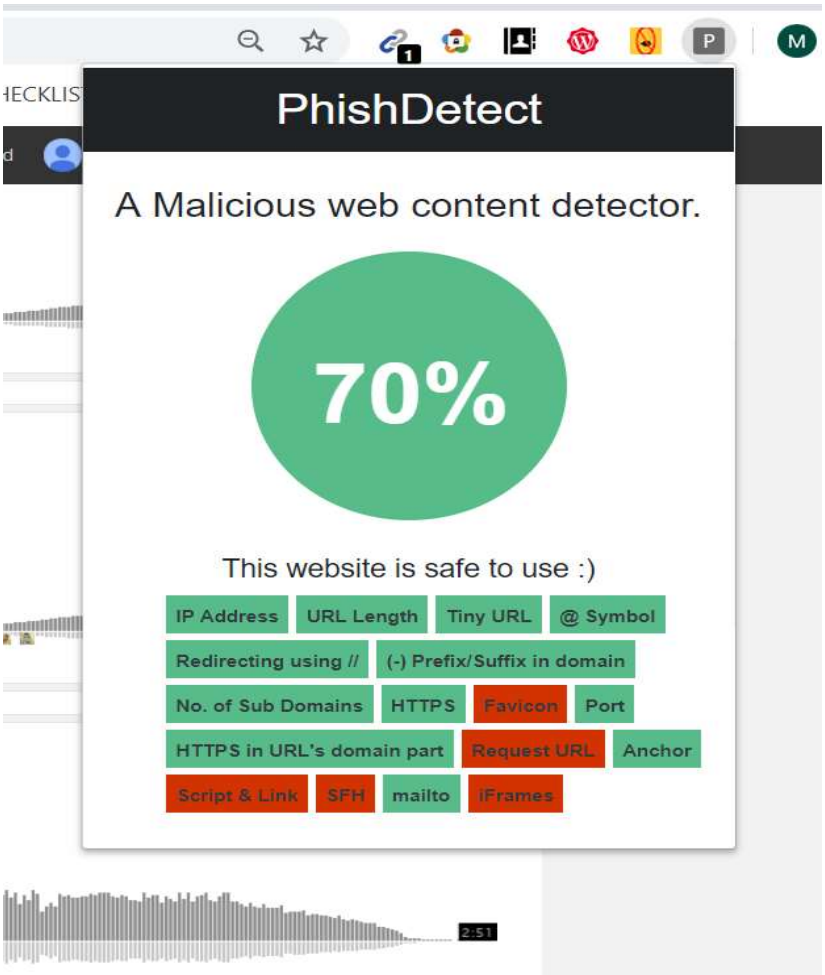

Figure 3: UI Design

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

3. TRAINING

The data from the preprocessing 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 19 an ensemble of 10 decision tree estimators is used. Each decision tree follows CART algorithm and tries to reduce the Gini impurity.

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

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.

3.1 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 de- vised to represent the Random Forest in JSON. The overall JSON structure consists of keys such as number of estimators, number of classes etc. Further it contains an array in Figure 4.4 Random Forest JSON structure 20 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.

3.2 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 0 - Suspicious 1 - Phishing

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

3.3 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 behavior 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.

4. COMPLEXITY ANALYSIS

4.1 Complexity of the Design Architecture

The complexity of the project lies in balancing the tradeoff be- tween 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 format. For example, JSON.
- Reproducing the Random Forest behavior in JavaScript reduced the accuracy by a small margin.
- Many features are not feasible to extract without using an 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.

5. CONCLUDING REMARKS AND FUTURE WORKS

We presented a design architecture for the development of an anti-phishing system using the Random Forest (RF) classifier/algorithm. The RF classifier is trained on phishing sites dataset using python scikit-learn. A JSON format to represent the random forest classier was devised and the learned classifier is exported to the same. Future work will look at scaling the architecture high enough to address the complexity analysis issues emanating from the current effort.

REFERENCES

[1] 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 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.

[3] 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.

[4] 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.

[5] 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.

[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.

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/