Machine Learning-Based Phishing URL Detection

Shambhavi Shandilya
Information Technology Department
ABV-IIITM Gwalior
Gwalior, India
imt_2019089@iiitm.ac.in

Sunil Kumar
Machine Learning Department
ABV-IIITM Gwalior
Gwalior, India
snk@iiitm.ac.in

***Abstract*—Phishing is a form of cyber-attack where the actor pretends to be a legitimate authority and it utilizes that trust to extract sensitive credentials from users. Phishing is carried out by various methods. Algorithm Phishing, Email Phishing, URL Phishing, Spear Phishing etc. are to name a few. The core of Phishing attack is to persuade user to click on a malicious URL, which seems legitimate and then steal credentials from the user. This project explores various machine learning algorithms and evaluates their performances. The primary focus is to accurately classify legitimate and malicious links using Machine Learning.**

***Index Terms*—Phishing, Machine Learning, Data Security, Social Engineering**

I. Introduction

Because it is so simple to develop a phoney website that closely resembles a legitimate website, phishing has recently become a top worry for security specialists. Although experts can spot fraudulent websites, not all users can, and as a result, some users fall prey to phishing scams. The attacker's primary goal is to obtain bank account credentials. The ”blacklist” method, which is the standard technique for detecting phishing websites, involves adding blacklisted URLs and Internet Protocol (IP) addresses to the antivirus database. Attackers modify the URL to appear authentic by obfuscation and many other straightforward ways, such as fast-flux, in which proxies are automatically constructed to host the website, algorithmic production of new URLs, etc., to dodge blacklists. This method's primary flaw is its inability to identify zero-hour phishing attacks. Many security experts are now focusing on machine learning techniques to overcome the limitations of blacklist and heuristics-based methods. Machine learning technology is made up of numerous algorithms that use historical data to forecast or make decisions about future data.

II. Literature Review

A. Classical Methods

This section describes some classical methods to detect phishing URLs. These methods work on a static set of knowledge and cannot dynamically learn from the results. The paper titled ”Machine Learning-Based Phishing Attack Detection” [1] describes these methods.

Blacklist Filter The paper ”Filtering spam with behavioral blacklisting” [2] explains in depth about this mechanism. It imparts the usage of a list containing unwanted IP addresses, websites, etc. which must be blocked from being accessed by the client's system. This technique has often been classified as static that cannot deal with the alarmingly quick adaptations by attackers. However, a timely update of the blacklisted resources, as proposed in the paper ”A phishing sites blacklist generator” [3], improved the algorithm's accuracy multi-fold.

Whitelist Filter On contrary to the blacklist filter, the whitelist filter method maintains a list of the allowed websites, URLs etc. that can be accessed by the client system. Similar to the blacklist filter, the whitelist resources must be updated frequently to reduce the false positives. The paper ”A usability test of whitelist and blacklist-based anti-phishing application” [4] studied the effectiveness of both Blacklist and Whitelist filters and found them quite similar.

Pattern Matching This technique works on the basis of certain blacklisted and whitelisted sequences of characters. If the requested URL matches any of these patterns, the results are generated accordingly. It uses a great deal of list-matching technique. The paper ”Preventing from phishing attack by implementing url pattern matching technique in web” [5] explains the details of this algorithm.

B. Rule based Detection

Another more effective method to detect phishing URLs was a rule based detection. The paper ”Rule-Based Phishing Attack Detection” [5] devises a set of 15 rules to identify if a link is malicious or legitimate. A few of them are mentioned for phishing URLs: ”If a webpage's URL is not present in all search enginesíndexes”, ”If a webpage's URL is IP based (hex-based, octal, or decimal-based)”, ”If a webpage contains password input field AND the webpage has more external than internal links”, etc.
These rules were subsequently utilised as features in learning algorithms for decision trees and logistic regression.

C. Machine Learning Algorithms

Looking at the inability of the classical methods to adapt to new dataset, several machine learning algorithms were then used in the detection of Phishing Attacks. I have researched on papers ” Phishing Attacks Detection using Machine Learning Approach” [6] and ”Machine Learning-Based Phishing Attack

Detection" [2] to understand the different steps and algorithms best suited for this task.

Feature Extraction Several papers focus on the feature extraction process. Since the URLs contain multiple components and meta data, it is important to reduce dataset dimensions for better performance of the algorithm. This can be removed by identifying co-related and irrelevant features. The following algorithms were used to achieve feature extraction: Principal Component Analysis, Intonation Groups and Parallel Coordinates.

Phishing Detection The most important part of the algorithm is to classify the URLs as legitimate or malicious using the Phishing detection algorithm. Most papers converged towards the following algorithms for this task: Decision Tree, Random Forest Algorithm, Logistic Regression and K-Nearest Neighbour Algorithm.
The Random Forest algorithm provided most accurate results. It performed better than a single decision tree and also overcame overfitting problem because of the randomness in the dataset and the features.

III. Methodology

In this section, we discuss the details of the proposed data-driven phishing website detection system. For the classification, we will be using the algorithms: K-Nearest Neighbours, Support Vector Machines, Decision Trees and Random Forest.

A. Dataset

The dataset has been obtained from the online repository of Mendeley. The complete dataset has 48 features, categorized as Lexical, Host-based and co-related features for the URL. It has 10000 values, having half legitimate and the other half malicious URLs. The class label 0 indicates a phishing website and 1 a legitimate website.

B. Data cleaning and formatting

Data cleansing, also known as data cleaning, is the process of identifying and correcting (or removing) inaccurate or corrupt records from a record set, table, or database. It involves determining which parts of the data are incomplete, incorrect, inaccurate, or irrelevant, and then replacing, changing, or deleting the soiled or coarse data. The data formatting was a two-step process in this phishing detection system:

Data formatting The initial process is to visualize the data parameters and identify the important columns. A total number of 27 lexical features are described like NumDots, SubdomainLevel, PathLevel, and so on. Another 15 features are included in the host-based features set. And the remaining 6 features are correlation features.

Training and Testing data split After data pre-processing, we will divide the dataset into two parts for training and testing purposes respectively. 80% of the data will be for training the model and the rest 20% will be for testing the model's metrics.

C. Model Training

This project employed the usage of KNN, SVM, decision tree and random forest in our system. Here is a brief on their working as classifiers in the system:

K-Nearest Neighbours We calculated the distance using the Euclidean method from equation (1),

$$d\left(p,q\right) = \sqrt{\sum_{i=1}^{n} \left(q_i - p_i\right)^2} \tag{1}$$

Where, $d\left(p,q\right)$ is the distance between points p and q in N-dimensions.

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. KNN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. Our dataset has 48 features and a Class label where 0 indicates a phishing website, and 1 indicates a legitimate website. When given an unknown sample, KNN will first measure the distance of the unknown sample with its neighbors by using Euclidean distance. The number of neighbors that it will check will be the value of K that can be chosen by setting the value of "n_neighbors." The distances will be measured by taking in the features of the samples that are in the dataset. The majority class of the neighbors that are the closest will be then assigned to the unknown sample.

We have used K=1 for the classification because it resulted in maximum accuracy as described in Figure 1.

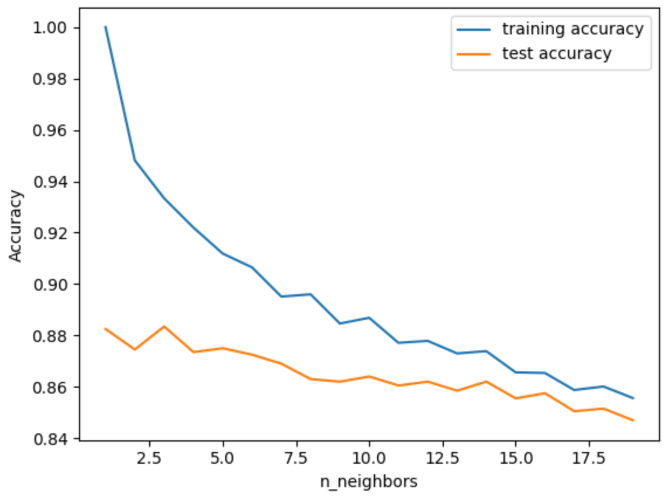

Fig. 1. Accuracy of KNN Classifier

Decision Tree We used Gini importance to calculate a node's importance for each decision tree. This was based under the assumption that the tree is binary, and so each node has at most two children. Gini Index can be calculated from the equation (2),

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2 \tag{2}$$

Where, p_i is the probability of an object being classified to a particular class.
While building the decision tree, we would prefer choosing the attribute/feature with the least Gini index as the root node.

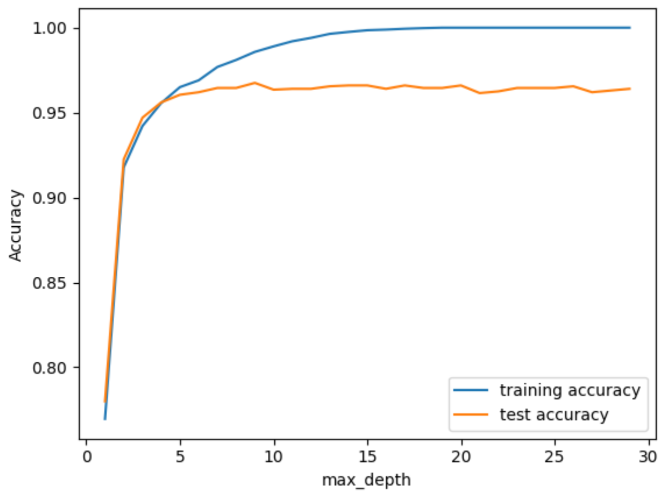

Fig. 2. Accuracy of Decision Tree

Random Forest A random forest classifier consists of a large number of decision trees that work as an ensemble. At first, it will create a bootstrap dataset of size "N" that will randomly take samples from our dataset. A random forest can then use these bootstrap samples to create a tree. For example, if our training data was [a, b, c, d, e, f], we might give one of our trees the following list [a, a, b, c, f, f]. It should be noticed that both samples are of the same size, and "a" and "f" are repeated in the bootstrap dataset because we sample with replacement. After taking in the samples from the bootstrap dataset, it begins to build trees by first choosing a root node. Random forest differs from decision trees because it uses a method called Feature Randomness. This means that when it comes to choosing a root node for a random tree forest will only allow the trees to choose a root node from a subset of features. The Gini impurity is measured among these subsets of features, and the lowest score will be used as the root node, and the different subsequent nodes are chosen in the same way. After creating the trees, the random forest classifier is ready to make predictions. It will take an unknown sample from our test dataset and run the sample among all of the trees. All of the individual trees give a class prediction, and the class that has the most votes will be the class of the unknown sample. One of the main reasons random forest classifier does well with large datasets is because it maintains diversity between models by using bootstrap aggregation and feature randomness.

IV. Results

The implemented model works on three different classifiers. The details of their performances are described in Table I.

V. Conclusion

Our work analyses different machine learning techniques when implemented over a dataset of features regarding web-

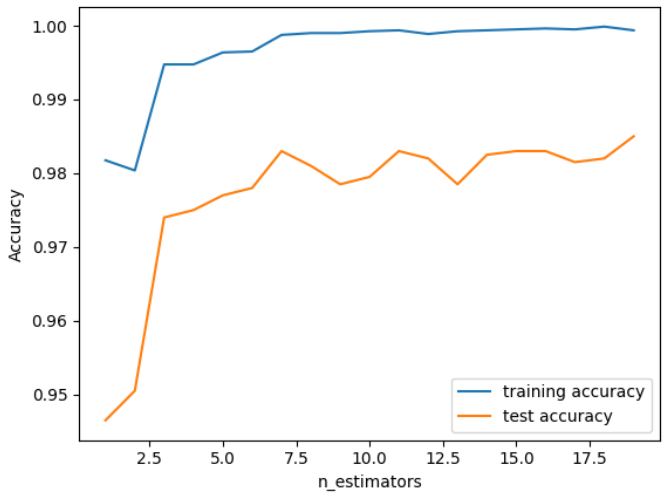

Fig. 3. Accuracy of Random Forest Classifier

TABLE I
Comparison of different classifiers

ML Model	Accuracy	f1_score	Recall	Precision
Random Forest	0.982	0.982	0.998	1.0
Decision Tree	0.962	0.961	1.000	1.0
K-Nearest Neighbors	0.882	0.883	1.000	1.0

sites and their corresponding details that may prove useful to detect a possible phishing website. This report aims to be useful to its readers to provide a conclusive analysis of these methods and to verify our observations regarding the random forest classifier's optimal performance. F1 score for the random forest is 0.99, which indicate that both false positive and false negative rate are in the satisfactory level. The graphs and details have been added to the document with aim to help others carry out further experimentation to conclude this work.

VI. Future Work

This project was fairly basic comparison among various classification methods. In the future, we might work on the deployment of deep learning techniques like multi-layer perception and artificial neural networks to improve the performance of the detection system.

References

[1] Hossain, Sohrab Sarma, Dhiman Chakma, Rana. (2020). Machine Learning-Based Phishing Attack Detection. International Journal of Advanced Computer Science and Applications. 11. 378-388. 10.14569/IJACSA.2020.0110945.

[2] Ramachandran, Anirudh Feamster, Nick Vempala, Santosh. (2007). Filtering spam with behavioral blacklisting. 342-351. 10.1145/1315245.1315288.

[3] M. Sharifi and S. H. Siadati, "A phishing sites blacklist generator," 2008 IEEE/ACS International Conference on Computer Systems and Applications, 2008, pp. 840-843, doi: 10.1109/AICCSA.2008.4493625.

[4] Li, Linfeng Helenius, Marko Berki, Eleni. (2012). A usability test of whitelist and blacklist-based anti-phishing application. 195-202. 10.1145/2393132.2393170.

[5] Basnet, Ram Sung, Andrew Liu, Qingzhong. (2012). Rule-Based Phishing Attack Detection.

[6] M. N. Alam, D. Sarma, F. F. Lima, I. Saha, R. -E. -. Ulfath and S. Hossain, ”Phishing Attacks Detection using Machine Learning Approach,” 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), 2020, pp. 1173-1179, doi: 10.1109/ICSSIT48917.2020.9214225.