A Machine-Learning Based Approach for Detecting Phishing URLs

Mahmoud Atari, Amjed Al-Mousa
Computer Engineering Department
Princess Sumayya University for Technology
Amman, Jordan
E-mail: mah20180344@std.psut.edu.jo, a.almousa@psut.edu.jo

2022 International Conference on Intelligent Data Science Technologies and Applications (IDSTA) | 978-1-6654-9960-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/IDSTA55301.2022.9923050

***Abstract*— This research's focus is to utilize different machine learning classification models to predict whether a given URL is a legitimate or a phishing URL. A legitimate URL directs users to a benign authentic webpage and typically serves the user's request. In contrast, a phishing URL directs users to a fraudulent website, usually impersonating another entity, luring visitors to believe otherwise, and eventually allowing the attacker to perform limitless post-exploitation attacks. Given the little-to-no internet safety awareness of average individuals, this paper aims to take an adaptive approach to detect phishing URLs on the client-side, which can significantly protect users from falling victims to cyber-attacks such as stealing important personal credentials. The proposed approach is to build a machine-learning powered tool that can help individuals stay safe and assist security researchers in identifying patterns and relations that correlate to these attacks, which will help maintain high-security standards for everyday internet users. Finally, the proposed model yielded a 97% detection accuracy using the XGBoost classifier and the random forest classifier.**

Keywords— Classification, Cyber Security, legitimate, Machine learning, phishing, URL.

I. INTRODUCTION

Social engineering attacks remain one of the most robust and dangerous threats to an organization's overall cyber security stand, alongside the online security of ordinary users. Phishing constitutes a major branch of social engineering attacks; such attacks target vulnerabilities present in the human factor of information systems. Phishing attacks hold no stable definition in literature. However, to narrow the scope of this research, mitigating phishing URLs will be specifically discussed. Phishing URLs are commonly used when an attacker attempts to masquerade as a trusted entity and sends a well-crafted message containing a fraudulent phishing link that usually simulates a trusted 3rd-party page in an attempt to lure a victim into clicking it. Once clicked, the attacker has several payloads he can plant, such as stealing credentials, installing malware, and disclosing personal information. According to the FBI [1], phishing was the most common type of cybercrime in 2020. Moreover, in 2020 alone, 75% of organizations around the globe experienced a phishing attack [2].

Phishing attacks are growing at an alarming rate. Phishing instances almost doubled from 2019 (114,702) to 2020 (241,324) [1]. These rates pose a great question, what is causing this incredible growth? First, during the COVID-19 pandemic, society was shifting towards using virtual tools to complete stuff normally done in physical presence, which translates to a greater pool of novice users that attackers could leverage. Second, the technological advancement in social engineering tools made it easier and cheaper for anyone to perform a phishing attack [3]. Furthermore, studies done by F5 security operations center [4] suggest that a 15% increase in phishing incidents can be seen annually.

This research focuses on building an adaptive supervised machine learning model that can act as a tool to detect phishing links before the user falls into clicking them. The tool is meant to act as a Host-Based Intrusion Detection System (HIDS) against phishing URLs.

The model was trained and tested on the 'Web page phishing detection' dataset collected by Abdelhakim Hannousse and Salima Yahiouche and published on Mendeley Data [5]. The dataset was constructed in May 2020, consisting of 11,430 URLs alongside 87 extracted features, and the outcome, namely, the status, which indicates whether the URL is either phishing or legitimate. The dataset is perfectly balanced, 50% phishing and 50% legitimate.

This research's main contribution can be summarized as follows:

- Studying the dataset at hand through exploratory data analysis techniques, which include statistical correlations of parameters and data visualization. This allows to discover patterns, and relationships that are not readily clear in raw data.
- Studying the performance of independent machine-learning models through accuracy, recall, and precision on unseen data.

The motivation behind using a machine-learning approach to detect phishing:

- Rule-based detection is limited in scope and eventually proves obsolete with the exponential emergence of attacking strategies. On the other hand, machine-learning powered detection adapts better to changing circumstances.
- Training and educating non-technical users are usually faced with resistance, so a machine-learning powered tool that is self-supervised can bridge this untouched gap in knowledge.

The remainder of this paper is structured as follows: Section II reviews the literature on previous work done on the topic. Section III consists of the experimental setup done before building the models, which includes studying the data in-depth and preprocessing it for more accurate results. Section IV will present the machine learning algorithms used to build different models. Section V discusses the results and findings. Finally, a conclusion is drawn in Section VI.

II. Related Work

Phishing has long been a hurdle in the way of security analysts and blue teamers; it has always made its way and exploited the vulnerabilities in human nature. Awareness programs and formal training have always been known as one of the best strategies to mitigate phishing attacks. However, with the incredible growth in mass phishing e-mails, several tools and ideas have been proposed to mitigate the problem.

The other direct method traditionally used by organizations and enterprises is blacklisting and whitelisting, whereby a URL is searched against a preset database to check its authenticity [6]. This method presents a major limitation, as it cannot detect zero-day attacks in real-time. Research shows that it takes an average of two hours for 47% to 83% of phishing websites to be blacklisted. This marks a significant delay as 63% of phishing campaigns end within the first 2 hours [7].

Machine Learning has been increasingly used to address several classification and detection problems. For example, it has been used to successfully detect diabetes and heart disease with high accuracy [8, 9]. It was also deployed to help students predict the possibility of being admitted to a particular school [10].

Jeeva and Rajsingh [11] proposed subjecting features that discriminate a phishing URL to associative rule-mining and found that transport layer security and the unavailability of a top-level domain are sensitive indicators of a phishing URL. Senturk and his colleagues [12] proposed a machine learning-based approach using the Weka classifier and reached an 89% true positive rate on unseen test data. Sankhe and his distinguished students [13] also used a machine-learning approach to detect phishing URLs and achieved a 95% accuracy using the RandomForest Classifier.

This work is a continuation of Shafana's and Fanoon's work [14], where they used an older version of the same dataset used in this research and achieved an accuracy of 97.6% using the PART algorithm. This research is meant to improve their work by studying the new version of the dataset and studying the performance of a broader range of machine learning algorithms.

III. Experimental Setup

The main objective of this paper is to build a classification model to predict whether a given URL is a phishing URL or a legitimate one using the dataset mentioned earlier. The dataset will be split into two parts; one used to train the model and the other to test the accuracy of the model on new unseen data. It is worth noting that the source code used in this work is available on GitHub [15].

A. Dataset Attribute Information

The dataset used consists of 11430 entries; each entry represents a given URL. Every URL is described with 87 features that come from different classes, precisely, 56 features from the syntax and structure of the URL, 24 features from content within the corresponding webpage, and finally, seven features from querying external resources such as whois which is a service that stores the registered users or assignees of Internet resources. Moreover, the input features are only numerical or dichotomous (True/False) attributes, 54 attributes are numerical, and 33 are dichotomous.

B. Data Analysis

Two different metrics were used to measure the effect of input variables on the label being predicted. First, the Point Biserial correlation was used to calculate the correlation between numerical attributes and the target. Then, the Chi-Square statistic was used to measure the independence between dichotomous attributes and the target.

1) Point Biserial Correlation

The Point Biserial correlation is a correlation coefficient used to measure the strength between a continuous variable, specifically the input features, and a dichotomous variable, the target. The correlation is presented as a number between -1 and 1, where -1 and 1 indicate a perfect correlation, and 0 indicates no correlation. The correlation coefficient was calculated for all numerical attributes, but only coefficients with a magnitude larger than 0.25 will be presented in Table 1.

Table 1: Point Biserial Correlation with target

Attribute	Point Biserial Correlation
ratio_digits_url	0.357505
phish_hints	0.334417
nb_qm	0.291925
nb_hyperlinks	-0.339540
domain_age	-0.377562
nb_www	-0.442184
page_rank	-0.505181

It can be seen that several features significantly contribute to the target variable, most notably, the page rank, which is a value assigned to a webpage as a measure of its importance or popularity. Search engines use this value as a way to help to sort search results. From the correlations calculated, the lower the page rank of a given URL, the higher chance it is a phishing link. Furthermore, the higher the ratio of digits in a URL, the higher the probability it is a phishing link.

2) Chi-Square Statistic

The Chi-Square statistic can be used to test the relationship between two categorical variables. It will be used to test independence between the dichotomous attributes and the target variable. The null hypothesis of the Chi-Square test is that there exists no relationship between the two variables; they are independent. The p-value of the Chi-Square test is of interest here. The p-value is a probability describing how likely the data would have occurred by random chance (i.e., that the null hypothesis is true). Figure 1 shows the variables with a p-value greater than 0.05, which is not statistically significant and indicates strong evidence for the null hypothesis. Meaning that there is no relationship between the variables and the target variable.

The feature 'path_extension', which indicates whether the object requested within the given URL has a .txt file extension or something else does not play any role in the classification of that given URL.

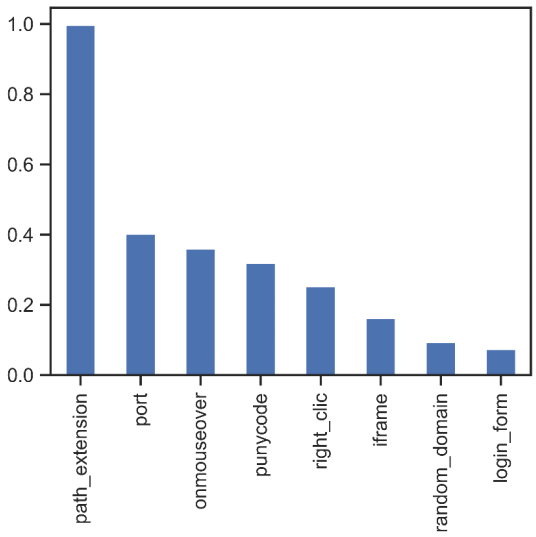

Figure 1: Chi-Square P-Values > 0.05

C. Data Visualization

This section displays plots that can help in studying the features and their contribution to predicting the target variable deeper. Figure 2 shows the distribution of the page_rank feature against the target variable, and Figure 3 is a histogram that shows the distribution of the average values of high cardinality features across the target variable.

Figure 2 stresses the solid contribution of the 'page_rank' feature to the classification of the target. The phishing URL distribution presented is right-skewed. On the other hand, the distribution of legitimate URLs forms a normal distribution as the 'page_rank' increases.

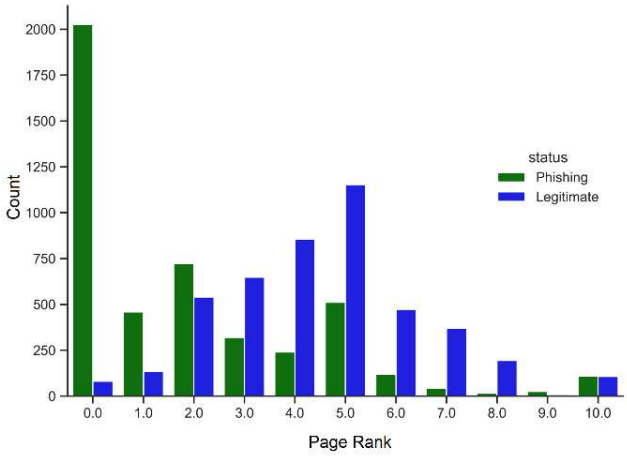

Figure 2: Page Rank Distribution

From Figure 3, it can be seen that the greater the URL length, the higher the probability it is a phishing URL. Another distinctive relation can be spotted here is the number of hyperlinks. The more the hyperlinks in the corresponding webpage imply a legitimate URL.

D. Data Preprocessing

At this stage, the dataset was split into a training set and a test set, where the training set consisted of 80% of the dataset and was used to train models and test them. The test set consisted of the remaining 20% and was used to validate the model on unseen data to see if it generalizes well.

Real-time collected data is likely to contain errors, missing values, and outliers. Building a successful machine-learning model greatly depends on the quality of data supplied to train the model. Therefore, preprocessing of data to deal with such problems yields considerably greater accuracy than the trained model.

In the dataset at hand, two features, namely, "domain_registration_length" and "domain_age" had negative values, which is unsound. Instead of deleting the URLs with incorrect data, the problem was solved by replacing the negative values with the median value of the corresponding column. To speed up the training, all columns with zero variance (i.e., all 0's) were dropped as those do not affect the target. Note that all preprocessing till this stage was done before studying correlations and plotting different graphs.

Finally, the features need to be scaled before feeding the data to machine learning algorithms. Feature scaling is an essential part of preparing the data to make sure all values contribute almost equally to the model regardless of their magnitude. In the URLs dataset, numerical features had different scales, so a standard scaler that scales the data such that it has a mean of 0 and a variance of 1.

Data preprocessing was done on the train set and the test set but using the train set features (i.e., the median of domain_age was calculated from the train set).

IV. Machine Learning Algorithms

A. RandomForest Classifier

A random forest classifier is one of the most widely used models in machine learning. It is an ensemble method classifier that fits many decision trees on different subsets of the training set and then averages their predictive probabilities to improve accuracy and regulate over-fitting [16].

B. K-Nearest Neighbors Classifier

K-Nearest Neighbors classifier is an instance-based classifier. As the name suggests, this classifier builds its decisions based on a similarity index rather than a parameterized model. It uses the distance between points to determine the similarity between instances, then it lists the top K neighbors labels and takes their mode to make a decision [17].

C. Linear Support Vector Machine (SVC)

The linear support vector classifier is commonly used when the data is said to be linearly separable. This algorithm tries to 'fit' the data to return the 'best' hyperplane that divides the instances. Consequently, when fed with new data, it can predict the label.

D. Logistic Regression

The logistic regression classifier work is similar to linear regression. Where it computes a weighted sum of the input variables, however, it does not output the results directly. Instead, it calculates the logistic of the result to estimate the probability of belonging to a particular class, and then it picks its decision based on maximum probability.

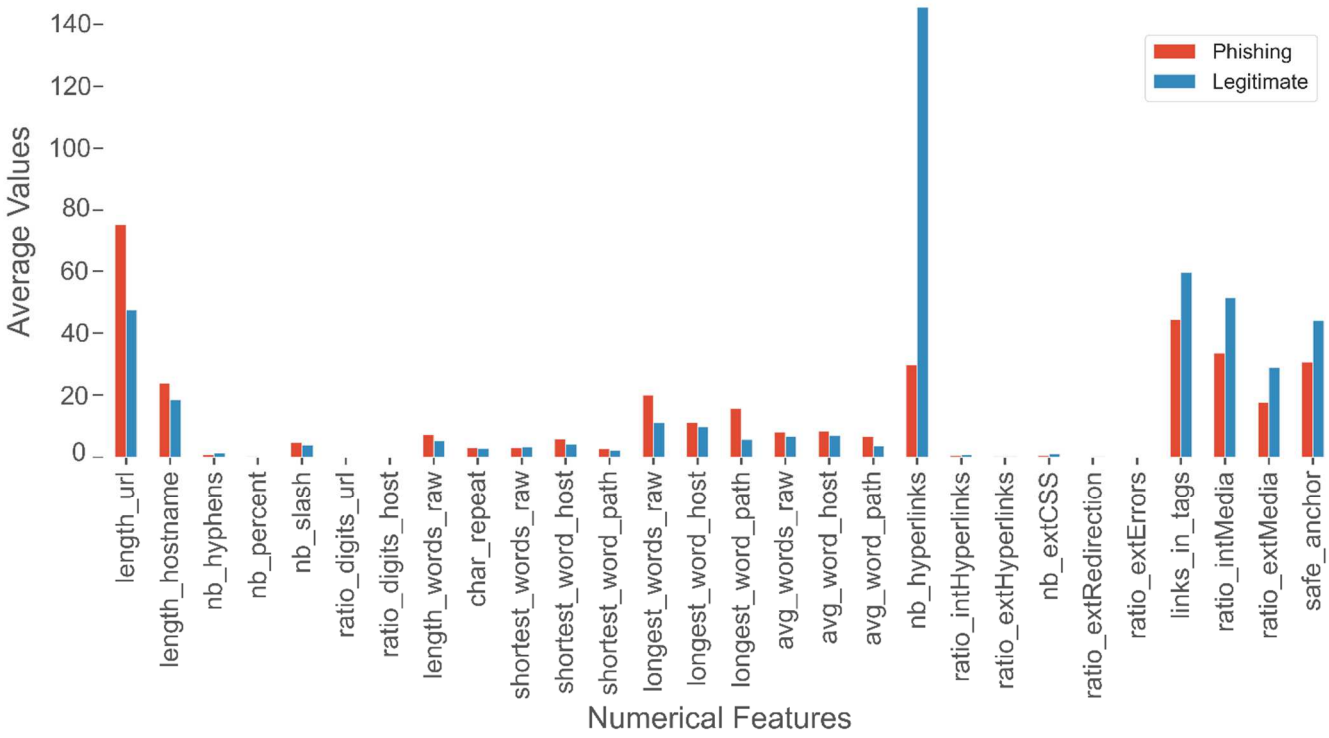

Figure 3: Distribution of Average Values across Target

E. *Extreme Gradient Boosting Classifier (XGBoost)*

XGBoost is a relatively new algorithm dominating the applied machine-learning scene recently. This algorithm uses gradient boosting to calculate cost functions; it focuses on optimizing run time through parallelization and distributed computing.

F. *SVC with Polynomial Kernel*

The polynomial kernel method works similarly to the linear svc but opens room for a non-linear model when a linear model cannot fit the data. To do this, the polynomial kernel adds new features that are combinations of the features already present; by introducing new features, the data might become separable, translating to a higher accuracy model.

G. *Voting Classifier*

A voting classifier makes decisions on a voting basis across different classifiers, where it predicts classes based on the predicted labels from each classifier. In this report, hard voting was used where the outcome is predicted through the mode of different classifiers' predictions. The models used to build this classifier are XGBoost, SVC with the polynomial kernel, and Random forest.

V. RESULTS

The models were tested on the unseen test set to see how they generalize on new data. Since the dataset has evenly separated labels, accuracy can be comfortably used as an evaluation metric to compare different classifiers. A normalized confusion matrix for each model will be shown to present a visual representation of the model's result.

Starting with the RandomForest classifier, the model's parameters were fine-tuned using GridSearchCV. The model's accuracy with the default parameters on the unseen test set was 97%. When fine-tuning the parameters, the accuracy stayed at 97%; however, there was a 1% increase in the recall rate of phishing websites. Figure 4 shows the normalized confusion matrix of the fine-tuned model.

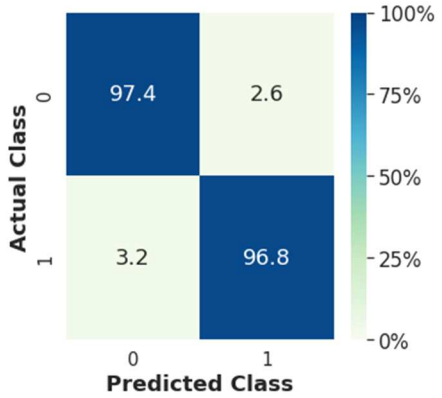

Figure 4: Normalized confusion matrix of the RandomForest classifier

The second model was trained using the K-Nearest Neighbors classifier. When trained on the default hyper-parameters, it scored an accuracy of 94%, but when hyper-parameters were tuned using GridSearchCV, it yielded an accuracy of 96%. Figure 5 shows the normalized confusion matrix of the tuned model.

The third model trained was using the Linear Support Vector Classifier. The accuracy obtained from this model was also 97%. Figure 6 shows its normalized confusion matrix.

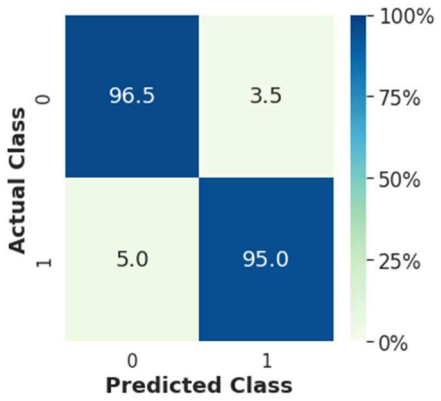

Figure 5: Normalized confusion matrix of K-Nearest classifier

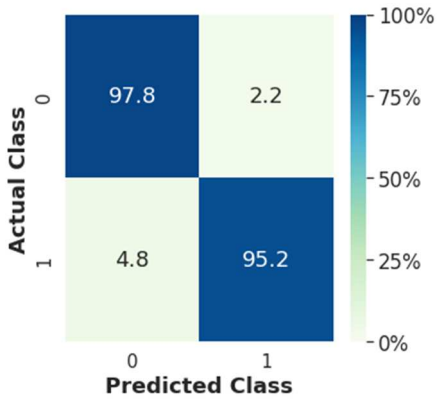

Figure 6: Normalized confusion matrix of Linear SVC

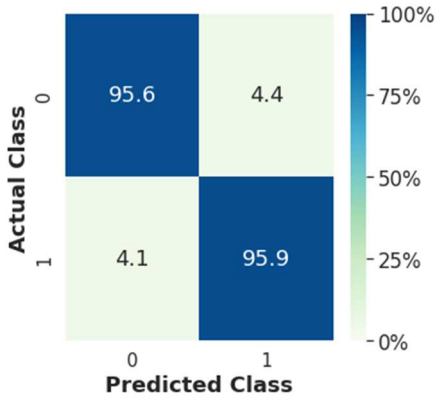

Figure 7: Normalized confusion matrix of Logistic Regression classifier

The next model trained was the Logistic Regression model. The model scored an accuracy of 96%. Figure 7 shows its normalized confusion matrix.

XGBoost Classifier was used to train the next model. This model scored an accuracy of 97% on the unseen test set with default parameters. When fine-tuning it using GridSearchCV, the accuracy remained the same. However, there was a trade-off, recall for phishing websites went down from 97% to 96%, and recall for legitimate websites went up from 97% to 98%. This is a trade-off between the user's convenience and security level that can be decided based on context. Figure 8 shows the normalized confusion matrix for the model with the default parameters.

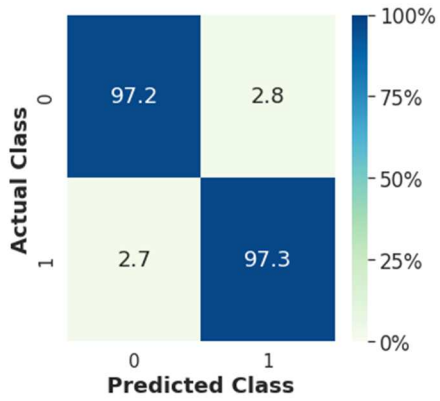

Figure 8: Normalized confusion matrix for XGBoost

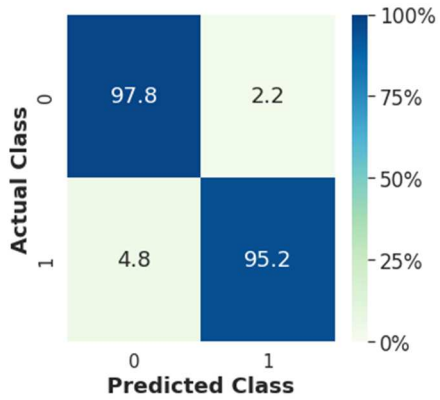

Figure 9: Normalized confusion matrix for SVC with polynomial kernel

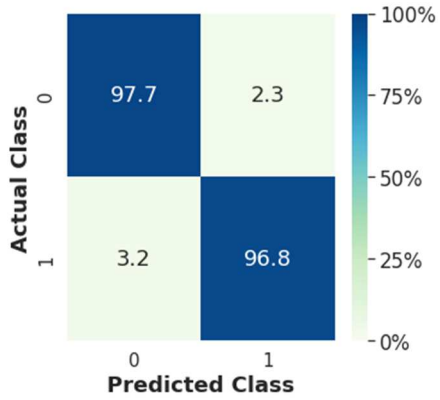

Figure 10: Normalized confusion matric for Voting Classifier

The penultimate model trained was the SVC with a polynomial kernel. To determine the polynomial's degree, GridSearchCV was used to tune the model, and it turned out that the degree is 3 and C is 3. The model scored an accuracy of 97%. Figure 9 shows the normalized confusion matrix.

Finally, the last model was built using the hard-voting classifier. The voting classifier at hand consisted of the XGBoost classifier, RandomForest classifier, and SVC with the polynomial kernel. The model also scored an accuracy of 97%. Figure 10 shows its normalized confusion matrix.

A ROC (receiver operating characteristics) curve was plotted for all classifiers to compare further and analyze the different models. Also, the area under the graph was calculated. Figure 11 shows the comparison.

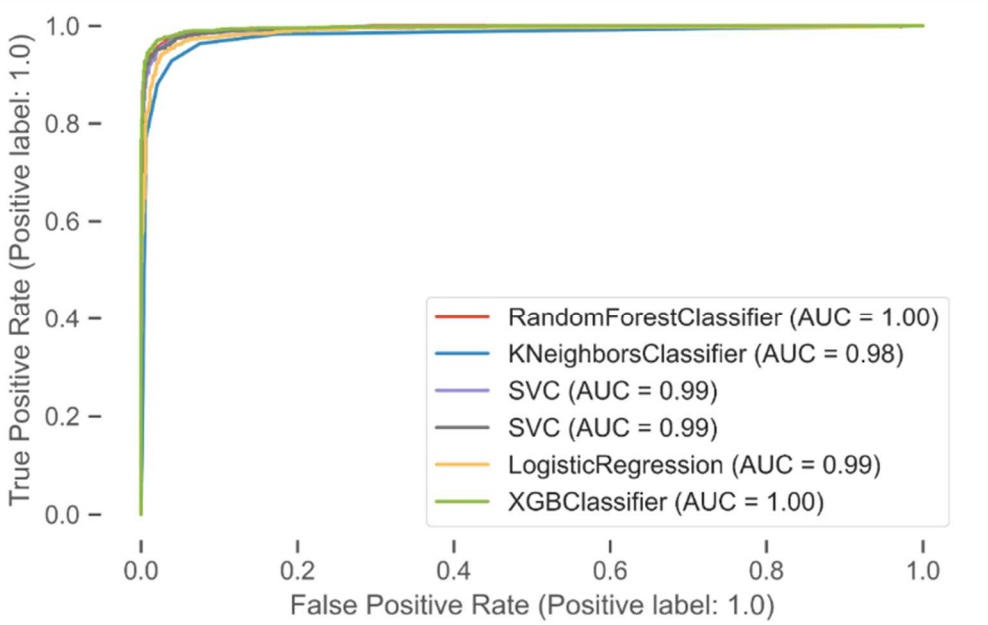

Figure 11: ROC Curve for different classifiers

Now, to consider the practicality of this model and to ensure the reliability of the data used for training, a minimum threshold for the upvotes can be imposed on a link before its classification is changed. Also, it is still possible that problems can arise, including phishing campaigns that use links from a distribution not covered in the training dataset. In addition, sophisticated attacks can learn the internals of the trained models and hence build their URLs to escape detection.

VI. Conclusion

To sum it up, the main goal of this paper was to build a supervised classification model that can act as a tool to detect whether a given URL is phishing or not. The model intends to assist security analysts in their jobs to protect the overall cybersecurity of organizations and to protect average internet users from fraudulent phishing links in a dynamic and adaptable way.

After tweaking several hyper-parameters of the trained models demonstrate a trade-off between user convenience and high-security standards, this trade-off introduces adaptability to the model where it can be tweaked to suit the application. For instance, the model should be tweaked on a highly critical host to increase recall on the cost of reducing precision, that is, to minimize the chances of a phishing URL being classified as benign.

Moving forward, this model can be further improved in terms of accuracy in two ways; first, the models can be trained on larger datasets that cover a more comprehensive range of the internet URLs, and the second model can be trained using deep neural networks which might yield a slightly better classification accuracy.

On the other spectrum, to further raise the usability of the model, an in-browser extension can be built on top of it to allow users to detect attempted attacks in real-time (before occurring). Moreover, the model can be coupled with another model to detect URLs in e-mails and quickly notify a potential victim.

This paper presented multiple techniques that were used to train various models. The highest accuracy achieved was 97%.

References

[1] M. Rosenthal, "Must-Know Phishing Statistics: Updated 2021," 16 September 2021. [Online]. Available: https://www.tessian.com/blog/phishing-statistics-2020/. [Accessed 28 December 2021].

[2] C. Jones, "50 Phishing Stats You Should Know In 2022," 22 December 2021. [Online]. Available: https://expertinsights.com/insights/50-phishing-stats-you-should-know/. [Accessed 28 December 2021].

[3] M. Samarati, "Phishing attacks: 6 reasons why we keep taking the bait," 4 September 2020. [Online]. Available: https://www.itgovernance.co.uk/blog/6-reasons-phishing-is-so-popular-and-so-successful. [Accessed 26 December 2021].

[4] D. Warburton, "Phishing Attacks Soar 220% During COVID-19 Peak as Cybercriminal Opportunism Intensifies," 2021. [Online]. Available: https://www.f5.com/company/news/features/phishing-attacks-soar-220--during-covid-19-peak-as-cybercriminal. [Accessed 22 December 2021].

[5] A. Hannousse and S. Yahiouche, "Web Page Phishing Detection," Mendley Data, London, 2021.

[6] Y. Guan, F. Zou, Y. Yao, W. Wang and T. Zhu, "Web Phishing Detection Using a Deep Learning Framework," *Wireless Communications and Mobile Computing,* 2018.

[7] M. Khonji, Y. Iraqi and A. Jones, "Phishing Detection: A Literature Survey," *IEEE COMMUNICATIONS SURVEYS & TUTORIALS,* vol. 15, no. 4, pp. 2091 - 2121, FOURTH QUARTER 2013.

[8] R. Atallah and A. A. Al-Mousa, "Heart Disease Detection Using Machine Learning Majority Voting Ensemble Method," in *2nd International Conference on new Trends in Computing Sciences (ICTCS)*, Amman, 2019.

[9] N. Abdulhadi and A. A. Al-Mousa, "Diabetes Detection Using Machine Learning Classification Methods," in *2021 International Conference on Information Technology (ICIT)*, Amman, 2021.

[10] Z. Bitar and A. A. Al-Mousa, "Prediction of Graduate Admission using Multiple Supervised Machine Learning Models," in *IEEE SoutheastCon*, Raleigh, 2020.

[11] S. Jeeva and R. E. , "Intelligent phishing url detection using association rule mining," *Human-centric Computing and Information Sciences,* vol. 6, no. 12, 2016.

[12] S. Sentürk, E. Yerli and I. Sogukpinar, "Email Phishing Detection and Prevention by Using Data Mining Techniques," in *(UBMK'17) 2nd International Conference on Computer Science and Engineering*, 2017.

[13] S. Parekh, D. Parikh, S. Kotak and S. Sankhe, "A new method for Detection of Phishing Websites: URL," in *2nd International Conference on Inventive Communication and Computational Technologies*, 2018.

[14] A. R. Shafana and A. R. Fanoon, "Predictive Data Mining for Phishing Websites: A Rule," *Journal of Information Systems & Information Technology,* vol. 5, no. 2, 2020.

[15] M. Atari, "Source code for the ML Based Approach for Detecting Phishing URLs," [Online]. Available: https://github.com/Mahmoudatari/ml_phishing.git. [Accessed 6 8 2022].

[16] SciKit Learn, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[17] O. Harrison, "Machine Learning Basics with the K-Nearest Neighbors Algorithm," 10 September 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761. [Accessed 29 December 2021].