Exploring Classical Machine Learning for Botnet Detection

Aditya Raj Singh
MT22136
aditya22136@iiitd.ac.in

Ritik Agrawal
MT22142
ritik22142@iiitd.ac.in

Sumedha Chugh
PhD21123
sumedhac@iiitd.ac.in

Swaib Ilias Mazumder
MT22078
swaib22078@iiitd.ac.in

Abstract

The process of gaining control of user computers through different types of attacks and using these for malicious activities is known as phishing, and the compromised system is called Botnet. In this project, we have developed an end-to-end system using which, a person can identify if the link is of an attacker or not, just by entering the URL on the website. To pre-process the data tokenization, stemming and vectorization have been utilized so that it can be used by algorithms. We have used four classical machine learning algorithms, one bagging based and one boosting-based algorithm. We have also ensembled a Quantum Machine learning algorithm with LGBM.

1 Introduction:

Client computers that have been infected by malware and are controlled by a remote server are known as botnets. Since the safety and data of client computer has been compromised, these can be used for malicious acts, so it's very important to detect botnets. Usually, attackers get a hold of client computers by getting them to click on URLs or emails known as phishing attacks [1]. There are various types of phishing attacks such as Deceptive Phishing, Spear phishing, Whaling, Pharming etc. In Deceptive phishing assailant gains confidential information of the user and then uses this to gain access of computer or launch financial scams. Spear phishing is a targeted phishing in which attackers gain user information by research or social media. Whaling is a type of spear phishing, but the target is someone at big position like company CEO etc.[2]
Though User education and Security Technology play an important role in protection against these attacks, it is equally important to detect these. A review of related work done to detect these attacks is discussed in section 2. In this project we detect these phishing websites using machine learning models like Logistic Regression, Decision Trees, Random Forest, Variational Quantum Classifier, etc., and develop a website where users can enter the suspected link to know if the link is malicious or not. Details of these models are discussed in the Pre-requisites section. Details of the dataset and EDA can be found in section 4. Section 5 shows experimental results and future scope.

2 Related Work

Various methods in Machine Learning and Deep Learning domain have been used to detect phishing URLs. In this section, we discuss Machine Learning i.e. Supervised and Semi-supervised Learning based approaches that have been utilized for tracking suspicious and legitimate URLs in the English language.

Pan et al. proposed a technique to detect phising website from features such as body of URL, title, keyword etc. They extracted ten features from given URLs and trained the modified dataset on SVM classifier. This technique fails when the website is hosted on domains that are jeopardized. CANTINA [3] uses TF-IDF algorithm for feature extraction and uses the features that give high TF-IDF scores. It also adds new features to the dataset and trains it on a linear model. This technique performs really well on text-based sites, but takes high time to run. Miyamoto et al. [4] applied various Machine learning based algos for training along with linear model as in CANTINA. The lowest error was obtained using AdaBoost. Xiang et al proposed using Bayesian Networks to detect these malicious links and compared their work to baselines. Bayesian networks gave the best results among all. He et al. [5] extracted twelve features using Anomaly and PILFER and fed these to SVM. This method faces high time complecity problem. Gowtham et al. [6] use a private dataset of logins to get higher number of true positives. This technique successfully reduces false positives but has a high computation cost. Chiew et al. [7] proposed a technique where they extracted logo of website and retrived the identity using google image search. This technique performs poorly for websites with no logo. PhishAri by Aggarwal et al. [8] use network-based features of tweets and train them on Random Forest classifier.

For extracting features from URLs tokenization, stemming, vectorization, etc., are applied to the dataset, which leads to sparsity in the dataset. Rao et al. propose multiple Random Forest based approaches like oblique RF, orthogonal RF, and PCA-RF that help solve the overfitting problem and works well with sparse and missing data. They also propose novel features that account for malicious links' characteristics and lead to better detection. Authors compared their work with various baseline models like CATINA and found that PCA-RF i.e., Principal Component Analysis Random Forest gave the best results with higher accuracy. Paper [9] uses Decision Tree, Neural Network and Logistic Regression based methods to safeguard users from these malicious attacks. One of the major problems faced by various ML algorithms in phishing detection is drift i.e. computers hardly understand the semantics. Another issue is that security tools lie mainly on the user side; hence this problem arises from the client's side. So nowadays, attackers choose different methods to exploit. Since these methods are previously unseen by machine learning algorithms, detecting them becomes difficult. The general approach is to increase the number of features and datasets continuously. Basit et al.[10] showed in their work that a lot of these techniques have a high false positive rate. A novel neural networks-based approach by Feng at al.[11] solves these problems and when compared with seven classical ML algorithms, their technique gave the best results. Yang et al.[12] use multi-dimensional features to resolve these challenges. Although the method works well in terms of accuracy and False Positive Rates, it can't be applied to websites which require captcha while loading[13].

[14]shows that ensemble methods perform better than classical methods. Various ensemble methods that have been used in literature for phishing detection are AdaBoost, Random Forest, Support Vector Machine with PCA and LDA.

Yuancheng Li et al. use Transductive Support Vector Machine (TSVM) as it also considers information stored in the large quantity of unsampled data, and as a result, it shows better accuracy than SVM. Sharif Amit Kamran et al. use game theory-based approaches and train their GAN architecture in a semi-supervised manner to distinguish between good and malicious links.[16]

3 Pre-requisites:

3.1 Logistic Regression

Logistic regression method is used to predict the class of categorical variables i.e. the output can only be True or False, Yes or No, 0 or 1. But rather than providing an exact value between 0 and 1, it provides probabilistic values that are in the range between 0 and 1. Logistic regression is used to solve classification problems. Instead of constructing a regression line, we fit a "S" shaped softmax function, which predicts two maximum values (0 or 1).

$$Pr(Y_i = 1|X_i) = \frac{exp(\beta_0+\beta_1 X_i+\beta_2 X_2+\beta_3 X_3+\beta_4 X_4)}{1+exp(\beta_0+\beta_1 X_i+\beta_2 X_2+\beta_3 X_3+\beta_4 X_4)}$$

The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight.

3.2 Multinomial Naive Bayes

Multinomial Naive Bayes is a probabilistic learning method commonly used in Natural Language Processing (NLP). The algorithm guesses the tag of a text, such as an email or a newspaper story, using the Bayes theorem. It computes the likelihood of each tag for a given sample and outputs the tag with the highest likelihood.Naive Bayes is a powerful algorithm that is used for text data analysis and with problems with multiple classes.

$$P(A|B) = P(B/A) * P(A)/P(B)$$

3.3 Decision Tree

This classification algorithm works on the principle of assigning an importance score to a feature and then further splitting the tree on its basis. Some methods used for calculating these importance scores are Information gain, Gini index, etc. Tree splitting can be univariate or multivariate. In univariate tree splitting, an internal node is split according to the value of a single feature or attribute. In multivariate tree splitting, it is split according to the value of multiple attributes. Entropy tells about the randomness in the data. More homogeneous the dataset lesser the entropy, and vice versa. Mathematically it van be written as

$$H(Y) = \sum_{i=1}^{n} p_i log2(p_i)$$

where i is the number of classes:
Information gain is defined as Entropy before splitting - Entropy after splitting. i.e. Information gain = H(Y)-H(Y|X)
Another criterion for selecting important features is Gini gain. Gini gain is gini index of child node subtracted from gini index of parent node. Where gini index is defined as:

$$GiniIndex = 1 - \sum_{i=1}^{n} p_i^2$$

The most popular algorithm for creating decision trees is ID3. This algo first selects most important feature based on Information gain or gini index and splits the data gain for node where information gain is maximum. This node is further split to produce subset of data. This process is repeated until all features are covered.
Although this algorithm works very well, it can't be backtracked and suffers from problem of overfitting.

3.4 Random Forest

As seen in section 3.3 Decision Tree tends to overfit and has a high bias; an ensemble method called Random Forest is used to avoid this. In this, multiple deep decision trees are trained on different subsets of features of the same training set with the goal of reducing variance. Then ensemble of the decisions made by these trees is taken as the final decision. It boosts the classifier's performance but leads to a loss of interpretation ability, an advantage offered by decision trees.

3.5 Support Vector Machine

SVMs are used when points are linearly separable as well as non-linearly separable. The goal for SVM remains to maximize the minimum geometric margin. The geometric margin is given by γ. The goal is to

$$max_{w,b} : GM \geq \gamma$$

As we want no points in the margin area, those points will be considered as misclassified too. Error for SVMs is sum of Classification error and margin error i.e. Error = Classification Error + Margin Error. Minimizing this error gives us the classification decision for SVM.
To separate the points that are not linearly separable kernel trick is used by SVM. Kernels map data points to higher dimensional space. Kernel trick is that if we transform data to higher dimensions, they will become linearly separable in that dimension. Various types of kernel are linear kernel, polynomial kernel, RBF kernel etc.

3.6 XG Boost

Another method to improve accuracy and performance of Decision trees is by using boosting. It stores data in compressed sparse column format, divides this data into blocks such that these blocks run on different machines for parallel compute. .The weight of mis-classified points is fed to the next tree and ensemble of result of all of these trees gives final classification result. This makes it faster and more accurate to work with. Mathematically it can be given as :

$$y_i = \sum_{i=1}^{n} f_k(x_i)$$

where i is the number of trees and f is space of F. Its objective function is:

$$(obj(\theta)) = \sum_{i=1}^{n} l(y_i, y_i) + sum_{K=1}^{k} \omega(f_k)$$

3.7 LGBM

LGBM(Light Gradient Boosting Machine) is a decision tree based algorithm. It uses GOSS i.e. Gradient Based One Side Sampling scheme. Undertrained features, i.e., features with a more significant value of gradient, will have more information gain than other features, and features with smaller information gain will be randomly dropped to recompute accuracy. Also in this algorithm decision tree nodes are selected in a breadth-first fashion by assigning weights to all features.

3.8 Variational Quantum Classifier

The use of quantum-enhanced algorithms with machine learning models is known as Quantum Machine Learning. We use a stacking algorithm with VQC as the base classifier and LGBM as the meta classifier. The first training phase takes place by training the base classifier. This trained classifier is used on training and test set. Output labels of these sets are appended with the original data, which is fed to the meta classifier to get final prediction values.

We used VQC (Variational Quantam Classifier) as the base classifier and LGBM(Light Gradient Boosting Machine) as meta classifier. VQC algorithm consists of four steps. For VQC we first prepare the quantum dataset by loading the data into a feature map. The next step is to evaluate gradients and update the parameters to get the cost function finally.
1. Feature vector of x is mapped to the quantum state vector in Hilbert state.

$$\bar{x} \text{ -> } |\Phi(\bar{x}) >< \Phi(\bar{x})|$$

2. A variational Circuit like Entangling CNOT or RY gate is added to the feature map which trains on optimization loop until correct convergence.
3. Output f of this circuit is assigned a value for classification. $(f : 0, 1^n - > 0, 1)$
4. In last step output is updated using classical optimization methods like SPSA, COBYLAT, SLSQP etc.
For coding these Quantum methods Microsoft's open-source kit Quantum Development Kit (QDK) is widely utilized [17,18]

3.9 Tokenization

Tokenization is usually the first step while processing text data. In this text data is split into smaller units like words so that it is easier to process. There are various type of tokenization techniques such as word tokenization, character tokenization etc. As the name suggests word tokenization splits sentences at word level, character tokenization splits sentences at character level.

3.10 Stemming

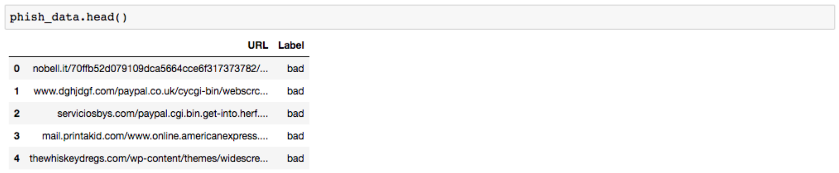
To reduce the size of the data corpus, remove redundant information and get context of data stem of words is extracted in NLP-based tasks e.g. eats, eating, and eaten all have base word eat so everyplace dataset observes these stemming process converts these to eat. Different types of stemmers widely used in NLP applications are: Porter stemmer, Lancaster stemmer,Regexp stemmer etc.

3.11 Vectorization

Since Machine Learning Algorithms work on numbers, it is important to map text data to numbers before feeding it to the algorithm. Vectorization performs the task of converting words to numbers. Some examples of vectorization techniques used are Word2Vec, FeedForward Neural Net Language Model, Continuous Bag-of-Words Model, Continuous Skip-gram Model.

4 Dataset Description

Our dataset consists of 5,00,000 data points and 2 features. Figure 1 shows the screenshot of the first five cells of our dataset. The two features are URL and Label.

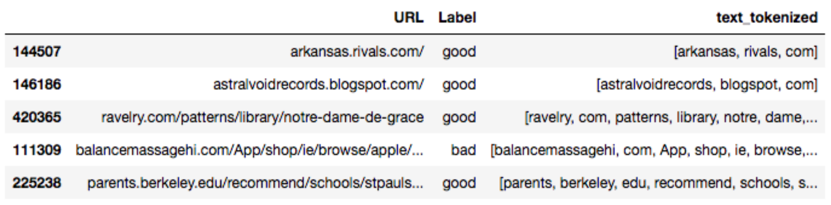


```
phish_data.head()
```

	URL	Label
0	nobell.it/70ffb52d079109dca5664cce6f317373782/...	bad
1	www.dghjdgf.com/paypal.co.uk/cycgi-bin/webscrc...	bad
2	serviciosbys.com/paypal.cgi.bin.get-into.herf....	bad
3	mail.printakid.com/www.online.americanexpress....	bad
4	thewhiskeydregs.com/wp-content/themes/widescre...	bad

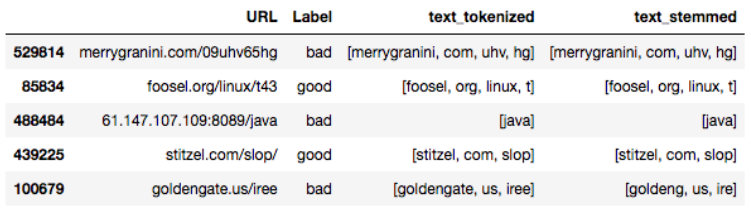
Figure 1: Dataset

For tokenization we used RegExp tokenizer.Figure 2 shows data after tokenization.



	URL	Label	text_tokenized
144507	arkansas.rivals.com/	good	[arkansas, rivals, com]
146186	astralvoidrecords.blogspot.com/	good	[astralvoidrecords, blogspot, com]
420365	ravelry.com/patterns/library/notre-dame-de-grace	good	[ravelry, com, patterns, library, notre, dame,...
111309	balancemassagehi.com/App/shop/ie/browse/apple/...	bad	[balancemassagehi, com, App, shop, ie, browse,...
225238	parents.berkeley.edu/recommend/schools/stpauls...	good	[parents, berkeley, edu, recommend, schools, s...

Figure 2: Dataset after Tokenization



	URL	Label	text_tokenized	text_stemmed
529814	merrygranini.com/09uhv65hg	bad	[merrygranini, com, uhv, hg]	[merrygranini, com, uhv, hg]
85834	foosel.org/linux/t43	good	[foosel, org, linux, t]	[foosel, org, linux, t]
488484	61.147.107.109:8089/java	bad	[java]	[java]
439225	stitzel.com/slop/	good	[stitzel, com, slop]	[stitzel, com, slop]
100679	goldengate.us/iree	bad	[goldengate, us, iree]	[goldeng, us, ire]

Figure 3: Dataset after Stemming Vectorization

For the process of stemming, we used snowball stemmer and for vectorization we used CountVectorizer

The ratio of good to bad links can be seen in figure 4

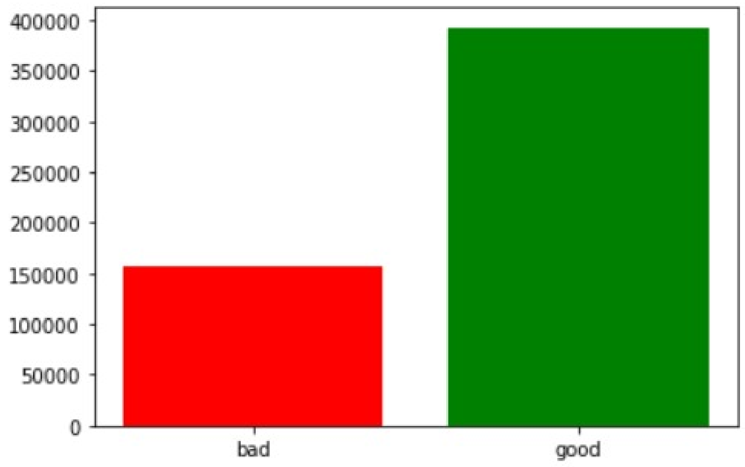


Figure 4: Ratio of good to bad websites

Figures 5 to 12 show Data analysis done to understand data in detail.

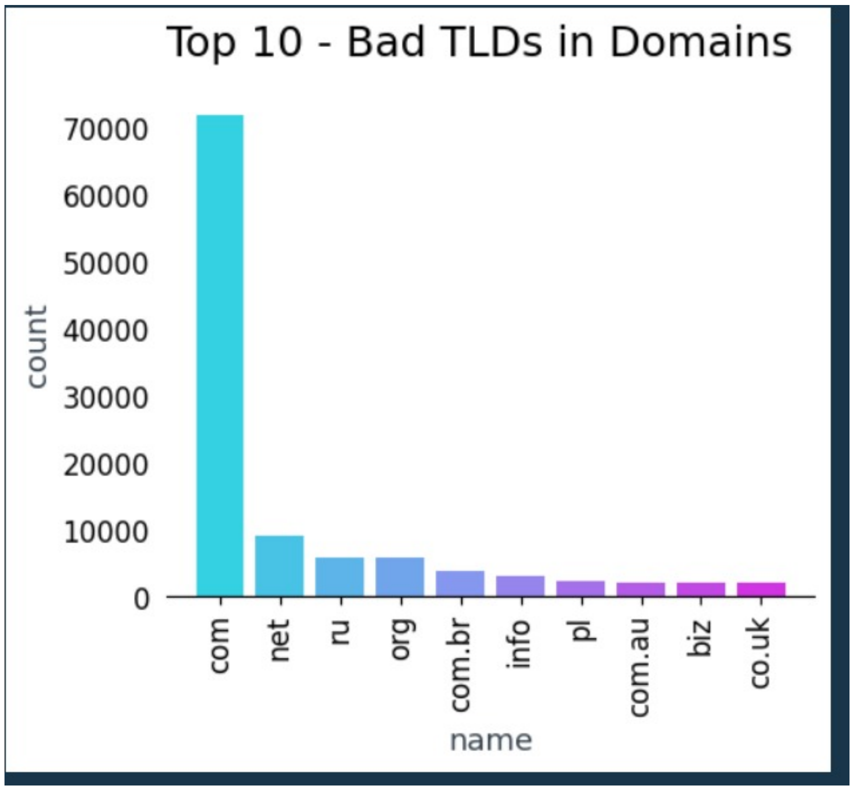


Figure 5: Top 10 Bad TLDs in Domain

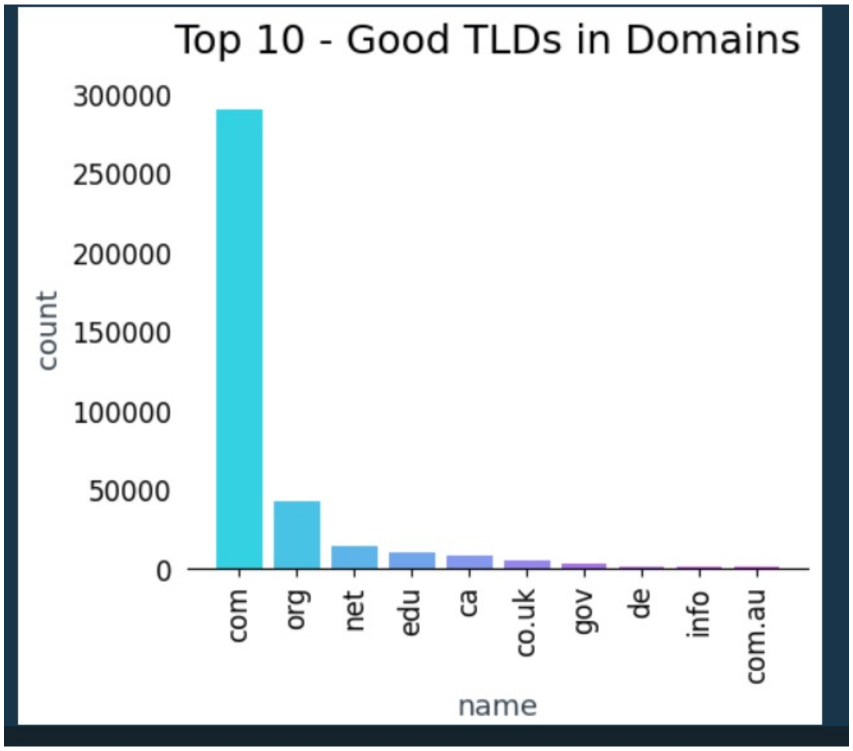


Figure 6: Top 10 Bad TLDs in Domain

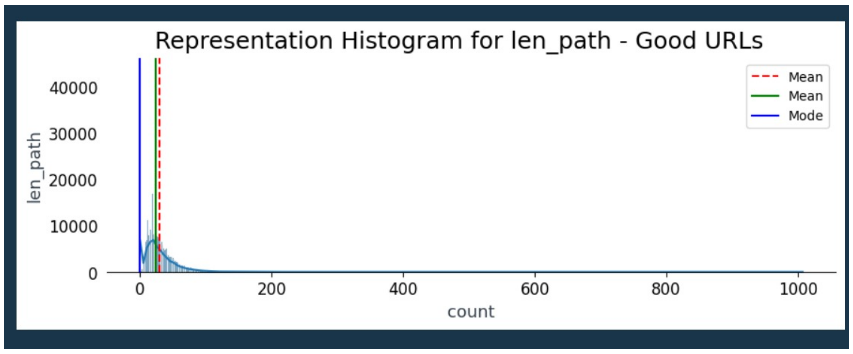


Figure 7: Histogram for path length of good URLs

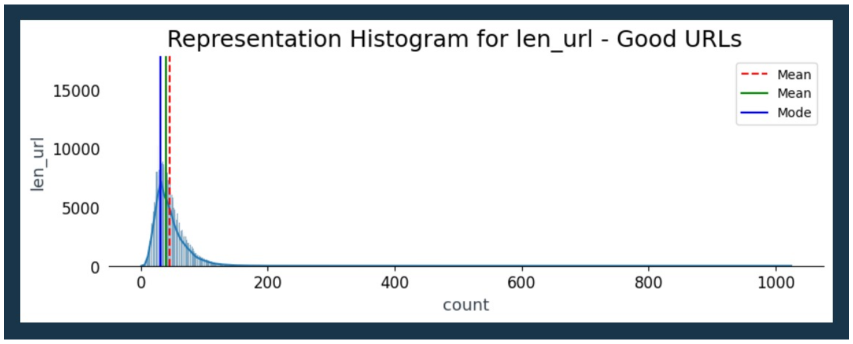


Figure 8: Histogram for path length of good URLs

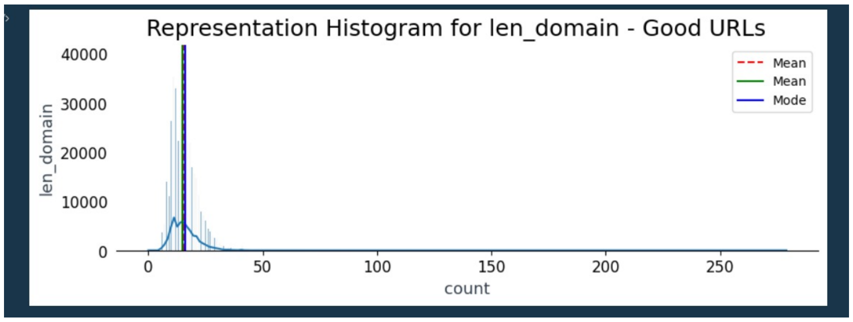


Figure 9: Histogram for domain length of good URLs

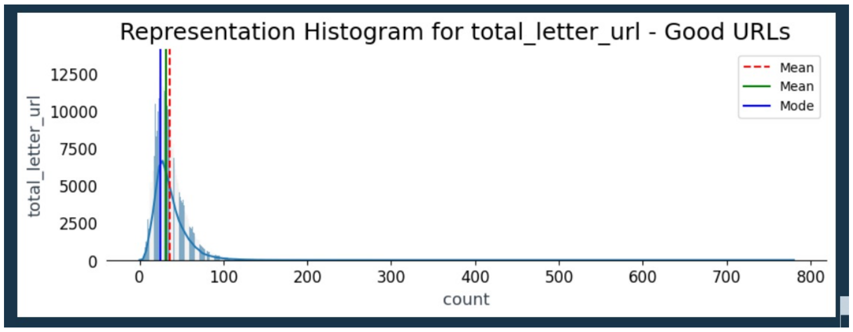


Figure 10: Histogram for domain length of bad URLs

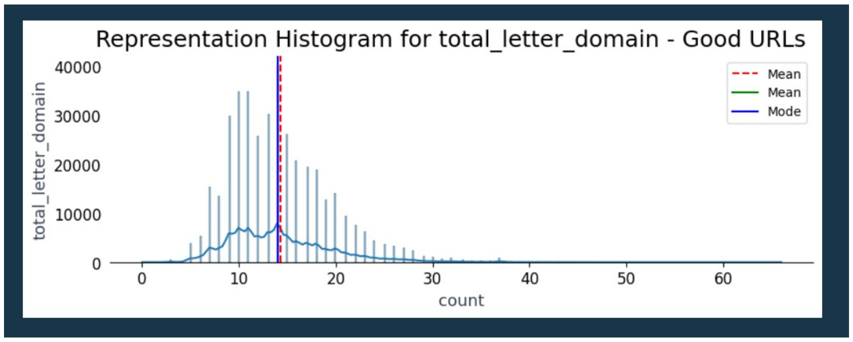


Figure 11: Histogram for the total number of letters in good URLs

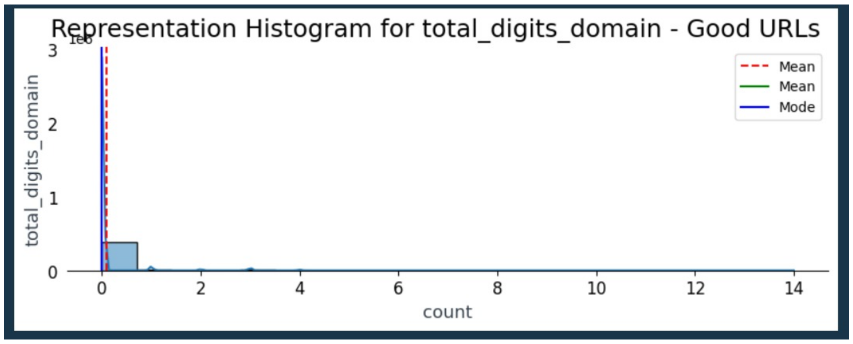
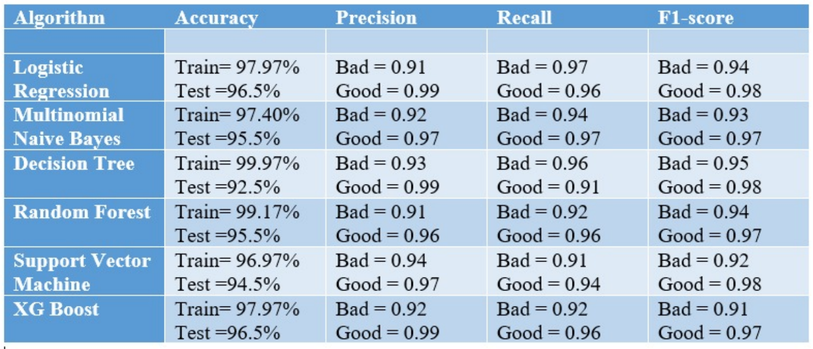


Figure 12: Histogram for the total number of letters in bad URLs

5 Experimental Results and Future Scope

After pre-processing the data we applied various Machine Learning Algorithms on it and calculated the accuracy, precision, recall and F1 score for these. Results for the same can be found below.



Algorithm	Accuracy	Precision	Recall	F1-score

Logistic Regression	Train= 97.97%
	Test =96.5%
		Bad = 0.91
		Good = 0.99
			Bad = 0.97
			Good = 0.96
				Bad = 0.94
				Good = 0.98

Multinomial Naive Bayes	Train= 97.40%
	Test =95.5%
		Bad = 0.92
		Good = 0.97
			Bad = 0.94
			Good = 0.97
				Bad = 0.93
				Good = 0.97

Decision Tree	Train= 99.97%
	Test =92.5%
		Bad = 0.93
		Good = 0.99
			Bad = 0.96
			Good = 0.91
				Bad = 0.95
				Good = 0.98

Random Forest	Train= 99.17%
	Test =95.5%
		Bad = 0.91
		Good = 0.96
			Bad = 0.92
			Good = 0.96
				Bad = 0.94
				Good = 0.97

Support Vector Machine	Train= 96.97%
	Test =94.5%
		Bad = 0.94
		Good = 0.97
			Bad = 0.91
			Good = 0.94
				Bad = 0.92
				Good = 0.98

XG Boost	Train= 97.97%
	Test =96.5%
		Bad = 0.92
		Good = 0.99
			Bad = 0.92
			Good = 0.96
				Bad = 0.91
				Good = 0.97

Figure 13: Experimental Results

Though the ensemble of the Quantum and Classical ML model performs as well as other models, in terms of time consumption it lags behind. In future work, we wish to optimize our algorithm to take lesser time.

References

[1] S. Haq & Y. Singh (2018) Botnet Detection using Machine Learning. In 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC), pp. 240-245. doi: 10.1109/PDGC.2018.8745912.

[2] Bhavsar, Vaishnavi,Kadlak, Aditya & Sharma, Shabnam, D. (2018)Study on Phishing Attacks nternational Journal of Computer Applications, pp. 27-29. doi: 10.5120/ijca2018918286.

[3] Zhang Y, Hong JI & Cranor LF (2007) Cantina: a content-based approach to detecting phishing web sites. In Proceedings of the 16th international conference on World Wide Web, ACM, pp 639–648. doi.org/10.1145/1242572.1242659

[4] Miyamoto D, Hazeyama H & Kadobayashi Y (2008) An evaluation of machine learning-based methods for detection of phishing sites. In: International conference on neural information processing, Springer, pp. 539–546. doi.org/10.1007/978-3- 642-02490-$0_6$6

[5] He M, Horng SJ, Fan P, Khan MK, Run RS, Lai JL, Chen RJ, & Sutanto A (2011) An efficient phishing webpage detector. Expert systems with applications 38(10):12,018–12,027

[6] Gowtham R & Krishnamurthi I (2014) A comprehensive architecture for detecting phishing webpages In International conference on neural information processing pp. 23–37

[7] Chiew KL, Chang EH, & Tiong WK (2015) Utilisation of website logo for phishing detection. Comput Secur 54:16–2

[8] Aggarwal A, Rajadesingan A, & Kumaraguru P (2012) Phishari: automatic realtime phishing detection on twitter. In: eCrime Researchers Summit (eCrime), 2012, IEEE, pp 1–12

[9] Y. Zhang, J. I. Hong, & L. F. Cranor, "Cantina: a content-based approach to detecting phishing web sites," in Proceedings of the 16th international conference on World Wide Web, 2007, pp. 639-648.

[10] A. Basit, M. Zafar, X. Liu, A. R. Javed, Z. Jalil, & K. Kifayat, "A comprehensive survey of AI-enabled phishing attacks detection techniques," Telecommunication Systems, pp. 1-16, 2020.

[11]F. Feng, Q. Zhou, Z. Shen, X. Yang, L. Han, & J. Wang, "The application of a novel neural network in the detection of phishing websites," Journal of Ambient Intelligence and Humanized Computing, pp. 1-15, 2018.

[12]L. Yang, J. Zhang, X. Wang, Z. Li, Z. Li, & Y. He, "An improved ELM-based and data preprocessing integrated approach for phishing detection considering comprehensive features," Expert Systems with Applications, vol. 165, p. 113863.

[13] R. S. Rao & A. R. Pais, "Detection of phishing websites using an efficient feature-based machine learning framework," Neural Computing and Applications, vol. 31, no. 8, pp. 3851-3873, 2019

[14] Ren Y, Zhang L & Suganthan PN (2016) Ensemble classification and regression-recent developments, applications and future directions In IEEE Comput Intell Mag 11(1):41–53

[15] Yuancheng Li, Rui Xiao, Jingang Feng & Liujun Zhao (2013) A semi-supervised learning approach for detection of phishing webpages In Elsevier Optik, Volume 124, Issue 23, Pp. 6027-6033, https://doi.org/10.1016/j.ijleo.2013.04.078.

[16] Kamran, S. A., Sengupta, S., & Tavakkoli, A. (2021). Semi-supervised Conditional GAN for Simultaneous Generation and Detection of Phishing URLs: A Game theoretic Perspective. doi.org/10.48550/arXiv.2108.01852

[17] [online] Syed Farhan Ahmed, (2020) Variational Quantum Classifier, https://born-2learn.github.io/posts/2020/12/variational-quantum-classifier/

[18] Ray, A., Guddanti, S. S., Ajith, V., & Vinayagamurthy, D. (2022). Classical ensemble of Quantum-classical ML algorithms for Phishing detection in Ethereum transaction networks. arXiv. https://doi.org/10.48550/arXiv.2211.00004