International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 06, June-2021

Comparative Analysis of Various Machine Learning Techniques for Detecting Malicious Webpages

Gayaksha Kandolkar
ME Student,
Information Technology Department,
Goa College of Engineering
Goa-India

Soniya Usgaonkar
Assistant Professor,
Information Technology Department,
Goa College of Engineering
Goa-India

***Abstract*— Due to the rapid growth of the internet, websites have become the intruder's main target. Malicious websites, when visited by an unsuspecting victim infect their machine to steal valuable information, redirect them to malicious targets or compromise their system to mount future attack. At times a malicious dynamic HTML code is usually embedded in a normal webpage. Anti-virus software packages commonly use signature-based approaches which might not be able to efficiently identify camouflaged malicious HTML codes. Therefore, using machine learning approach to detect malicious web content is a better alternative. The objective of this project is to train various machine learning classifier models on the dataset created to predict malicious websites. The study is also conducted to measure and compare the performance level of these machine learning classifiers.**

Keywords*— Machine Learning, Malicious webpages, Random Forest, XGBoost

I. INTRODUCTION

Malicious web pages are those, which contain content that can be used by attackers to exploit end users. This includes web pages with phishing URLs, spam URLS, JavaScript malware scripts, Adware etc. A Malicious URL or a malicious web site hosts a variety of spontaneous content within the shape of spam, phishing, or drive-by- exploits in order to dispatch attacks. Attackers on the web put up hyperlinks to malicious content that can harm the victims in one way or another. They may choose customers to download their malware, phish their credentials, steal sessions etc. Nowadays, it is becoming very difficult to detect such vulnerabilities due to the continuous development of new techniques for carrying out such attacks. At times a malicious dynamic HTML code is embedded in a normal webpage. Dynamic HTML gives attackers a new and powerful technique to compromise the security of computer systems. The malicious webpage infects the victim when a user browses it. Naive users using a browser have no idea about the back-end of the page. The users might be tricked into giving away their credentials or downloading malicious data. Furthermore, such DHTML code can disguise itself easily through obfuscation or transformation, which makes the detection even harder.

So, detecting and preventing the user from these attacks are significant task. Malicious attack detection and prevention system plays an immense role against these attacks by protecting the system's critical information. The internet security softwares and firewalls are not enough to provide full protection to the system. Anti-virus software packages commonly use signature-based approaches which might not be able to efficiently identify camouflaged malicious HTML codes. Therefore, using machine learning to detect such web content will be a better alternative.

II. LITERATURE SURVEY

Significant research has been carried out on Malicious Website detection using Machine Learning Techniques. However, these papers have restricted themselves to few attributes for machine learning. In the paper "Detection of Malicious URLs using Machine Learning Techniques" [1] the authors extracted the Lexical Analysis Features in addition to the 3rd party feature, geo ranking and used the Convolutional Neural Networks (CNN) algorithm for feature extraction and classification. They said that malicious URLs could be detected by extracting the lexical features. Their presented work was an early effort in malicious URL detection.

In this paper "Malicious web content detection using machine learning" [2] the authors proposed the development of an extension for Google Chrome which acted as middleware between the users and the malicious websites to ensure safe browsing. They used the UCI Dataset of Phishing Website to train the classifier. Whenever a user entered the URL, the features were extracted and the URL was tested on the trained classifier to obtain the result. They extracted around 22 URL features also called as lexical features and domain-based features called as host-based features. They also compared the following three classification algorithms - K-Nearest Neighbours (kNN), Support Vector Machines (SVM), and Random Forest and concluded that Random Forest algorithm gave better accuracy in comparison with the two others.

Yuan-Tsung Hou et.al [3] proposed a malicious web page detection model based on dynamic HTML and some Java script native functions using the boosted decision tree algorithm. The author used the technique

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 06, June-2021

of machine learning for the classification of web pages. They explored the possible content-based Features (dynamic HTML and some Java script native functions) and compared the following four classification algorithms - Naive Bayes, Decision Tree, Support Vector Machine, and Boosted Decision Tree. Their experiment showed that Boosted Decision Tree gave better test results and so the same was implemented.

In this paper "Using supervised machine learning algorithms to detect suspicious URLs in online social networks" [4] the authors built a supervised machine learning classification model to detect the distribution of malicious content in online social networks (ONSs). Multisource features were used to detect social network posts that contain malicious Uniform Resource Locators. For the data collection stage, the Twitter streaming application programming interface (API) was used and VirusTotal was used for labelling the dataset. To explore the best-performance machine learning algorithms for the classification of spam and non-spam URLs associated with tweets the following algorithms were used: - Naïve Bayes (NB), k-Nearest Neighbors (k-NN), Random Forest (RF), and Logistic Regression (LR). They trained and tested four classifiers using the same set of 36 features and results showed that RF had the best performance. They aimed to find the highest performance model using the smallest number of features and the smallest structural parameters (tree number, max tree depth and maximum leaf size) in order to find the least complex but high performing classifier.

[5] gives the insight about the Dynamic attack detection method in which the JavaScript was embedded to the URL to bypass the detection mechanisms the false positive rate produced by this technique is less than 4.2% in the best case.

Sahingoz et al. [6] proposed a model to detect whether the URL in the email is legitimate or phishing in real time. Features used are Word Vector, NLP based and Hybrid. To find the word vector a word list is created. Each word from the URL is separated from one another using separators, removed the digits and random letter words. Meaningful existing words are added to the list to be analyzed. Words consisting of 2 meaningful words is separated using a word decomposition module. For example, SECURELOGIN is separated to SECURE and LOGIN and are added to the list. 7 classification algorithms were used namely Naive Bayes, Random Forest, KNN (n = 3), Ada boost, K-star, SMO and Decision Tree. Random Forest Algorithm proved to give the best giving 97.98% accuracy.

In the paper "Detection of phishing URLs using machine learning techniques", the author [7] discusses about the rise of phishing websites and give techniques to extract features and implement machine learning algorithms to classify the same. They have extracted features like traffic rank details, lexical features, page rank etc. They have presented a study of different machine learning algorithms. A fixed result showing the best algorithm is not done in the paper, we will give statistical analysis of all the algorithms and even the accuracy of the chosen algorithm to prove the result.

III. METHODOLOGY

The idea is to use URL/Lexical features, Page content features as well as Host-based features.

Lexical features: Lexical features are features obtained from the properties of the URL name or the URL string.

Page-Content features: The content-based features of a webpage can be drawn primarily from its HTML and JavaScript content. The idea is to capture the URLs which bypass the lexical features.

Host-based features: In addition to URL-based, HTML-based and JavaScript features, we extract Host based features which are dependent on the domain of the URL.

In this work, the Host-based features are extracted using the WHOIS information of the domain.

A. *Obtaining Dataset*

The set of malicious URLs are collected from opensource service called Phish Tank[8]. This service provides a set of phishing URLs in multiple formats like csv, json etc. From this dataset, 5000 random phishing URLs are collected to train the ML models. The legitimate URLs are obtained from the open datasets of the University of New Brunswick[9]. This dataset has a collection of benign, spam, phishing, malware & defacement URLs. Out of all these types, the benign URL dataset is considered for this project. 5000 random legitimate URLs are collected from this dataset to train the ML models.

B. *Feature Extraction*

In the context of machine learning, features are used to provide discriminative power in the classification process. Fig. 1 depicts a more detailed insight of the feature extraction step in the proposed system.

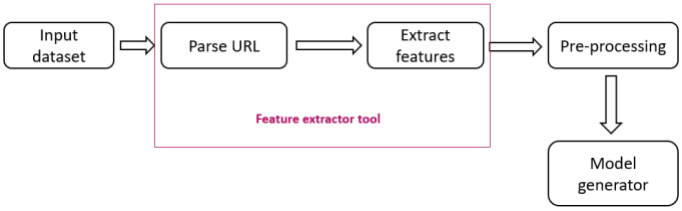

Fig. 1. Overview of feature extraction

When a URL is appended into this model, the first step is to read the URL and parse through it. Once the URL is parsed, next step is to extract relevant features from the URLs. Python in-built libraries are used to parse and extract features some of the features. In our study we have extracted a total of 17 features as shown in Table I. These extracted features are categorized into

- URL based Features
- Domain based Features
- HTML & JavaScript based Features

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 06, June-2021

TABLE I. PROPOSED FEATURE SET

Feature	Feature Source
Domain of URL	URL string
IP Address in URL	URL string
"@" Symbol in URL	URL string
Length of URL	URL string
Depth of URL	URL string
Redirection "//" in URL	URL string
"http/https" in Domain name	URL string
Using URL Shortening Services "TinyURL"	URL string
Prefix or Suffix "-" in Domain	URL string
DNS Record	Domain WHOIS Info
Website Traffic	Domain WHOIS Info
Age of Domain	Domain WHOIS Info
End Period of Domain	Domain WHOIS Info
IFrame Redirection	Web Page Content
Status Bar Customization	Web Page Content
Disabling Right Click	Web Page Content
Website Forwarding	Web Page Content

C. *Model selection*

The following machine learning algorithms were considered to evaluate and compare the best performance. Support Vector Machines classifiers - The SVM algorithm uses a dataset where the input samples are divided into two classes with labels either 0 or 1. The methodology includes finding a line (in two-dimension space) or a plane (in multi- dimension space) also called as a hyperplane which will most efficiently separate the two classes [10].

K-Nearest Neighbours (kNN) - kNN algorithm can be used for both classification as well as regression problems. However, mostly it is used for classification problems. 'k' in the kNN algorithm stands for the number of nearest neighbours we wish to take vote from. When predicting fora new data sample, this algorithm will run a search on the training dataset to find the closest k-samples. The predicted class of those similar samples is then found out and the summary of that is given out as the class label for this new data sample [11].

Naive Bayes - Naive Bayes classifier is one of the commonly used learning algorithms. The Naive Bayes classifier is a probabilistic model based on the Bayes rule. 'Naïve' refers to the assumption of conditional independence among features [12].

Logistic Regression - LR classifier: a probabilistic classifier, typically working on binary classification problems [13].

Decision tree classifier - A Decision tree is a machine learning classifier based on the tree structure. Each node in the tree is associated with a particular feature, and the edges from the node separate the data based on the value of the feature. Each leaf node binds to a class in the classifier model. The training data is key point for the information gain (IG) of the feature selection policy. A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees [14].

Random forest classifier - A Random forest classifier is an ensemble algorithm. Ensemble algorithm is the combination of same or different kind of algorithms. Set of trees make a forest. Here, Random forest is a set of decision trees. The voting of each decision tree is taken and the new output case is added to that class which has highest vote [15].

XGBOOST - (Extreme Gradient Boosted Tree) is an optimized implementation of gradient boosted trees first introduced by [16]. It is mostly employed in classification task where it is used as a classifier for mapping input pattern into a specific class. It is a recent supervised learning algorithm that implements a process known as boosting to improve the performance of gradient boosted trees. All the algorithms above were implemented in this study by using Scikit-learn, which is an open-source machine learning library in Python.

We trained and tested these classifiers using the same set of 17 features described in Table 1. The dataset was randomly divided into a 80% training and 20% testing set. The said classifiers were trained and tested.

IV. EVALUATION CRITERIA

We used the Scikit learn default parameter values for all the algorithms. Table II shows the confusion matrix in which TP (True Positive) is a case where the predicted value matches the actual value. The actual value was positive and the model predicted a positive value.

FP (False Positive) is a case where the predicted value was falsely predicted. The actual value was negative but the model predicted a positive value.

TN (True Negative) is a case where the predicted value matches the actual value. The actual value was negative and the model predicted a negative value.

FN (False negative) is when the predicted value was falsely predicted. The actual value was positive but the model predicted a negative value.

TABLE II. CONFUSION MATRIX

	Predicted Positive Class	Predicted Negative Class
Actual positive class	TP	FP
Actual negative class	FN	TN

There exists several metrics that utilize the confusion matrix.

• Precision is the ratio of true positives to the sum of a true positive and false positive, as shown in (1).

$$\text{Precision} = TP/(TP + FP) \quad (1)$$

• Recall is the ratio of correct true positive classifier decisions to the all true positive examples in the test set as shown in (2).

$$\text{Recall} = TP/(TP + FN) \quad (2)$$

• F-measure (F1) represents the previous metrics precision and recall combined as follows in (3)

$$F1 = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall}) \quad (3)$$

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 06, June-2021

• Accuracy is the measure of the true predictions divided by the total, shown in (4)

Accuracy = (TP + TN)/(TP +TN + FP+ FN) (4)

• Support is the number of actual occurrences of the class in the specified dataset.

VI. EXPERIMENTAL RESULT AND COMPARISON

All the classifiers were implemented using the same performance metric for a fair comparison. Below is the screenshot depicting the results of all the algorithms with the different performance metrics and the same is summarized in Table III.

SVM : classifaction report on test Data:

	precision	recall	f1-score	support
0	0.71	0.98	0.82	976
1	0.97	0.61	0.75	1024
accuracy			0.79	2000
macro avg	0.84	0.80	0.79	2000
weighted avg	0.84	0.79	0.79	2000

SVM : Accuracy on test Data: 79.300 %

KNN : classifaction report on test Data:

	precision	recall	f1-score	support
0	0.77	0.81	0.79	976
1	0.81	0.77	0.79	1024
accuracy			0.79	2000
macro avg	0.79	0.79	0.79	2000
weighted avg	0.79	0.79	0.79	2000

KNN : Accuracy on test Data: 79.100 %

Naive Bayes: classifaction report on test Data:

	precision	recall	f1-score	support
0	0.67	0.90	0.76	976
1	0.86	0.57	0.68	1024
accuracy			0.73	2000
macro avg	0.76	0.73	0.72	2000
weighted avg	0.76	0.73	0.72	2000

Naive Bayes: Accuracy on test Data: 73.050 %

Logistic Regression: classifaction report on test Data:

	precision	recall	f1-score	support
0	0.72	0.94	0.82	976
1	0.92	0.64	0.76	1024
accuracy			0.79	2000
macro avg	0.82	0.79	0.79	2000
weighted avg	0.82	0.79	0.79	2000

Logistic Regression: Accuracy on test Data: 79.100 %

Decion Tree : classifaction report on test Data:

	precision	recall	f1-score	support
0	0.72	0.99	0.84	976
1	0.98	0.64	0.78	1024
accuracy			0.81	2000
macro avg	0.85	0.81	0.81	2000
weighted avg	0.86	0.81	0.81	2000

Decision Tree: Accuracy on test Data: 81.050 %

Random Forest : classifaction report on test Data:

	precision	recall	f1-score	support
0	0.80	0.93	0.86	976
1	0.92	0.78	0.85	1024
accuracy			0.85	2000
macro avg	0.86	0.86	0.85	2000
weighted avg	0.86	0.85	0.85	2000

Random Forest: Accuracy on test Data: 85.400 %

XG Boost : classifaction report on test Data:

	precision	recall	f1-score	support
0	0.81	0.93	0.86	976
1	0.92	0.79	0.85	1024
accuracy			0.86	2000
macro avg	0.86	0.86	0.86	2000
weighted avg	0.86	0.86	0.86	2000

XG Boost: Accuracy on test Data: 85.600 %

Fig. 2. Classification reports for different algorithms

TABLE III. TEST PERFORMANCE OF THE ALGORITHMS

Machine Learning algorithms	Accuracy (%)	Precision	Recall	F1-Score
Support Vector Machine	79.30	0.97	0.61	0.75
K-Nearest Neighbour	79.10	0.81	0.77	0.79
Naïve Bayes	73.05	0.86	0.57	0.68
Logistic Regression	79.10	0.92	0.64	0.76
Decision Tree	81.05	0.98	0.64	0.78
Random Forest	85.40	0.92	0.78	0.85
XGBoost	85.60	0.92	0.79	0.85

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 06, June-2021

The result can be represented in graphical form for better analysis and understanding.

Fig 4 shows that XGBoost Classifier returned the best accuracy among all other the algorithms followed by Random Forest, whereas Naïve Bayes obtained lowest accuracy rate.

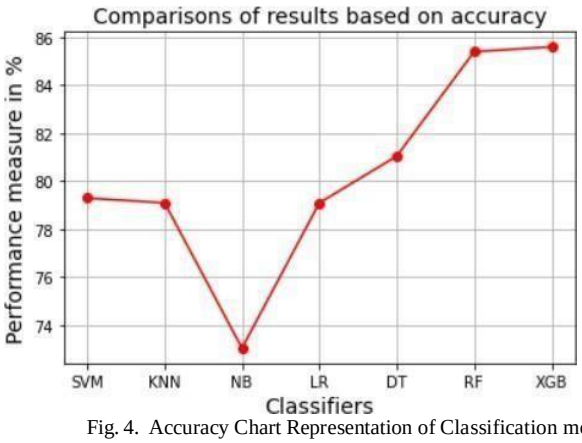

Fig. 4. Accuracy Chart Representation of Classification models

VI. CONCLUSION

The detection of malicious URLs is a binary classification problem and various machine learning classifier models are trained on the dataset created to predict malicious websites. In this paper, we aimed to find the highest performance model. Out of the seven different classifiers that are evaluated XGBoost Classifier model gave the best accuracy of 85.60% followed by Random Forest Classifier.

VII. REFERENCES

[1] Immadisetti Naga Venkata Durga Naveen, Manamohana K, Rohit Verma, “Detection of Malicious URLs using Machine Learning Techniques”, International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-4S2 March, 2019.

[2] A. Desai, J. Jatakia, R. Naik and N. Raul, "Malicious web content detection using machine leaning," 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, 2017.

[3] Hou, Y.T., Chang, Y., Chen, T., Laih, C.S., Chen, C.M.: Malicious web content detection by machine learning. Expert Systems with Applications, International Journal 2010.

[4] Mohammed Al-Janabi, Ed de Quincey, and Peter Andras. 2017. Using supervised machine learning algorithms to detect suspicious URLs in online social networks. In Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017.

[5] Hung Le, Quang Pham, Doyen Sahoo, Steven C.H Ho, “URL Net: Learning a URL Representation with Deep Learning for Malicious URL Detection”, arXiv:1802.03162v2 Mar 2018.

[6] Sahingoz, Ozgur & Buber, Ebubekir & Demir, Onder & Diri, Banu. (2019). Machine learning based phishing detection from URLs. Expert Systems with Applications.

[7] James, Joby, L. Sandhya, and Ciza Thomas, ”Detection of phishing URLs using machine learning techniques,” Control Communication and Computing (ICCC), 2013 International Conference on. IEEE, 2013.

[8] https://www.phishtank.com/developer_info.php

[9] https://www.unb.ca/cic/datasets/url-2016.html

[10] J. Brownlee, “Support Vector Machines for Machine Learning - Machine Learning Mastery”, Machine Learning Mastery, 2017. [Online] https://machinelearningmastery.com/support-vectormachines-for-machine-learning/

[11] J. Brownlee, “Tutorial To Implement k- Nearest Neighbors in Python From Scratch - Machine Learning Mastery”, Machine Learning Mastery, 2017[Online] https://machinelearningmastery.com/k-nearest- neighbors-for machine-learning/

[12] https://machinelearningmastery.com/naive-bayes-for-machine-learning/

[13] https://machinelearningmastery.com/naive-bayes-for-machine-learning/

[14] https://www.javatpoint.com/machine- learning-decision-tree-classification-algorithm

[15] “Random Forests Algorithm”, Datasciencecentral.com, 2017[Online] https://www.datasciencecentral.com/profiles/blogs/random-forests-algorithm

[16] T. Chen and C. Guestrin. XGBOOST: A scalable tree boosting system. In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016, pp. 785-794.