=

**Malicious URL detection using supervised machine learning techniques**

Arvind Narayan Srinivasan,Gundeep Singh,Vigneshwaran Balasubramani,Rudra Pathak

***Abstract-* With the rapid advancement of the internet and social media, cyber attackers prey on website and try to embed trojans and viruses without the awareness of the user and take advantage of naive users. Mostly, the website are reached through emails, hyper-links, social media advertisements or web searches, however different the type of attack can be but the web page is reached through the Uniform Resource Locator(URL). There are many antivirus software available in the market that gives alert to the user based on the website they are currently surfing; but the classification of malicious and benign URL is established on the basis of some repository that requires predetermined knowledge of the website. So if any website that is malicious and is generated newly, will not be available in the repository of the antivirus software and users data would become liable to be stolen because maintaining up to date blacklisting is difficult. Therefore, there is a need for a more learning based approach to solve the issue. This project focuses on certain machine learning techniques such as Logistic Regression, KNN and Random Forest Classifier to classify the URLs based on the URL string and extracting useful features from it. After cleaning the data, we test our dataset with different machine learning techniques and conclude which would be the best model for a given set of data.**

1. INTRODUCTION

With the advent of the World Wide Web, the number of users active on the internet has increased drastically. This has lead the users of the internet prone to many attacks such as malwares, phishing, trojans and many other viruses[1].

URL stands for Uniform Resource Locator which serves as an address for web based documents in the world wide web. A URL consists of mainly two components : the protocol and resource name. The protocol would mostly be *http* or *https.* The resource name consists of the domain address and the path.

A malicious URL are generally web pages that are embedded with some harmful code that may intend to do some unwanted task which the user might not have given access to although the website would display some other information to deceive the user’s attraction. A malicious URL hosts a variety of unwanted content such as phishing, trojans, uninformed downloads etc. Most of the time these malware codes are embedded in the javascript which once downloaded releases a package in the users machine to steal data. Some social engineering attacks steal sensitive and private information by pretending they require only basic information from the user’s account[2]

Detecting and stopping malicious URL have been on research for over a decade and many solutions have been devised. One well known approach is blacklisting by different antivirus software[3]. It contains a repository of list of blacklisted site; with a query of new URL by the user, the URL is searched if it is in the blacklisted data and it is classified as malicious or not based on the output. However maintaining an exhaustive list of blacklisted sites is impossible as thousands of new web pages pop up everyday. Being a simpler model, this system has major flaw of identifying malicious web pages that are not available in the data. This can happen more often as attackers tend to use Domain Generation Algorithms(DGA) that generates different URLs with different domains, so even if a URL is been blacklisted; the hacker can surpass this system by creating another web page with the same malicious content. The attackers can also use “Cloaking” to render content differently for different users which allows them to visually manipulate the users and get them to click the links to start the attack.

In order to overcome this issue, researchers have suggested a more comprehensive approach that is learning the patterns of malicious URLs. For this we need to extract features from the URLs of the training data. The features to be extracted should be informative and should describe the URL. Simply just using the string of the URL may not give best result as compared to blacklisting. These features should be then processed to be fed into the machine learning algorithm. The only known and feasible data when we enter a website is the URL, so our goal is to extract features from the URL(texts, IPs,length etc) and construct a machine learning algorithm that falls into binary classification of the URL- “Malicious” and “Benign”. After successfully extracting the features, the next goal is to identify a suitable model,more accurately a classification algorithm such as Logistic Regression, Support Vector Machine, Naive bayes etc. Comparing with the blacklisting approach, this approach generalizes the concept of classification and examines the pattern with the known data rather than just checking with existing repositories or checking the contents of the webpage.

1. RELATED WORKS

There are several works taken up by machine learning enthusiasts all over the world to come up with an optimal solution for this problem. Here are some top ideas which were accurate in predicting the malicious URLs.

Let us take a look at a Phishing attack detection proposal [4], the author compares different machine learning techniques for phishing detection such as logistic regression, CART, BART, SVM and Random Forest. The author has used 1171 raw phishing emails and 1718 legitimate emails and also has extracted 43 features from those. The author has categorized the phishing solutions into - detective solutions, preventive solutions, corrective solutions. Then the author uses each of the machine learning techniques and performs binary classification. In the results, RF outperforms all the classifier with an error rate of 7.72 % followed by CART, LR,BART, SVM, and NNet respectively. However, RF has a high false positive rate of 8.29 %.

For Malicious website detection [5], A K Singh et al have analyzed different machine learning approaches. The author has presented with 25 possible attributes or features that can be used to classify malicious URLs. The attributes have been chosen based on classification accuracy and computational requirements. Some of the major features to be noted are the location of the website, domain name, DNS and WHOIS information, Title tag values etc. Then the author has used these features in Naive Bayes and C4.5 algorithm to test the efficiency of the attributes. In the result, the author performs ten fold classification and found the feature *flash components* has the highest accuracy.

To differentiate malicious URLs [6], the author have used different machine learning techniques. The author have used 2.4 million URLs with 3.2 million features. Out of the 3.2 million features, 64 features are non binary that contain both numerical and ordinal values. After performing training and testing, the author concludes that Random Forest and Multi layer perceptron shows high precision and recall. Also, the author concludes that numerical features are more effective than the categorical features.

Justin Ma et al specifies the disadvantages of blacklisting and proposes a lightweight solution of using the URL to detect malicious websites [7]. The author have taken from 20000 to 30000 URLs and aims an accuracy of 95-99 %. The author first specifies about the lexical feature - the host name and the path. A binary feature is created for each of the host name called the bag of words. The next feature is the hostname property. The author justifies this attribute by stating that malicious URLs may not have been hosted with a reputed hosting service. The host name feature has been sub categorized into IP address properties, WHOIS properties and Domain name properties. Then the author has used SVM, Naive Bayes and Logistic Regression to train the model. The author performs L1 regularized logistic regression to yield low error rate.The SVM and LR classifiers have at least half of the error of Naive Bayes.

In an entropy-based approach [8], the author states that the use of URL for malicious website detection is a faster type of classification. The URL is chunked into meaningful components that are sequential and orthographic. In the feature inventory, the author mentions the used features such as the URL component and the length, distinct tokens such as years and sequential n-grams. Maximum entropy(ME) based learning is applied and all the results are at a 95% confidence level. Then leave-one out cross validation is applied that showed a significant increase in accuracy from 78% to 92%.

1. APPROACH

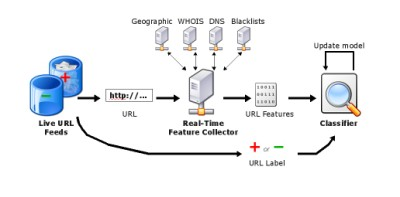
In this section, we will explain the different approaches used to predict the malicious URLs which includes nature of dataset, feature identification or extraction, various classification techniques used etc.

For our dataset, we treat URL authenticity as a classification problem with the labels ‘good’ and ‘bad’ which represents the respective benign or malicious URL. This learning based approach is used for our supervised learning techniques which employs cross validation to split the world dataset (400,000 entries) as training and test data which can then be used in suitable learning techniques to measure the accuracy of prediction.

Significantly, we identify URLs based on their word length, lexical features and host or port number based characteristics which we later define them as features. However we exclude the two potential characteristics which could also stand as a potential candidates to be used as features for training the model, they are: the content of the URL page, and the URL’s context (e.g., the page or email in which the URL is embedded) [7]. The reason for the exclusion of these features can be backed by a variety of reasons as follows. First, by avoiding to download the page content we secure the user’s system from any harmful virus that can be embedded along with the content. Second, downloading the page content and then using it for classification can cause a memory overhead and increase the prediction time. Third, when we use the URL features for training the model it makes the model generic for all contexts (Web pages, Email, Social media etc). Finally, the reliability for obtaining a malicious page content for training and test data can be practically difficult because the malicious websites have the ability of “cloaking” their contents differently to different users [10]. Provided we have identified the potential features we can see that the nature of the problem is by itself a classification problem with only two classes which are “Malicious” or “Benign”, so we have picked the training models to be logistic regression, Random forests for supervised classification learning and K-nearest neighbors model for supervised learning.

1. ARCHITECTURE

The basic architecture of the malicious URL prediction system will look like ***Fig1***

******

***Fig1:*** *Architecture of Malicious URL detection technique [7]*

The architecture diagram depicts our future work where the URLs will be taken from a live repository of URLs which are being identified as malicious by browsers or when users mark it as spam and the like.We then pass the URL to a Real-time feature collector which queries DNS, WHOIS, blacklist URL lists and geographical server information and identifies IP related and lexical features from the URL. The features are then fed to the classifier which is a machine learning model of our choice which will provide predictions based on real time data thus avoiding the chance of the dataset being outdated, this is because as the data set is real time so are the predictions, which enables the model to stay updated no matter the number of URLs added every day.

The architecture we have used for our training model is a similar one, except that instead of the Live URL feeds we are feeding a static dataset of 400,000 URLs consisting of both ‘Malicious’ and ‘Benign’ classes and also feeding the features to select directly to the classifier.

The data set we obtained will be cleaned and a feature extraction was performed to identify the potential features which can provide a better prediction after which cross validation was performed to train the different models which we use in the classifier like logistic regression, K nearest neighbours technique and Random forests. Let us look in to each in detail in the following topics.

1. DATA PREPROCESSING

Data as we know is the essence of Machine Learning and the accuracy or the performance of a Machine Learning Model is determined solely by the quality of the data provided while training the model.

However, data available in the real world is raw, inconsistent and sometimes incomplete, making it unfit for model training. Considering the fact that high quality data leads to better models and predictions, data preprocessing has become the most fundamental part of any machine learning or data science approach.

For our project we selected a dataset[10] of as much as 400,000 URLs. The dataset contains approximately 325,000 URLs which are safe to access and are thus labelled as *good* while the other 75,000 of the URLs are identified as malicious and are hence labelled as *bad.*

While the dataset was being selected, the approach was to find a consistent, diverse, simple dataset with features minimum in number but most meaningful in nature for the model to learn efficiently and make as much accurate predictions as possible.

The dataset as a whole has just one **feature**/**predictor**(the full URL of the website) and one **label**(*good/bad*). A sample of dataset is shown in the picture below(*fig2.*) A decision was made to select this dataset as it was huge in number when compared to other available datasets and simple as the whole URL was taken as a single feature. It was decided to process the data before feeding it to the model rather than by just increasing the dimensionality of the dataset by adding multiple features. Cross validation was performed to split the dataset into appropriate training and test data to be fed to the learning model.

*TABLE 1*:*Sample Data*

|  |  |
| --- | --- |
| **url** | **label** |
| nkckbkalcabkkndc.website / | bad |
| apple-search.info | bad |
| 23.227.196.215/ | bad |
| answers.yahoo.com/question/index?qid=20071223204515AAgZF8B | good |
| apanews.si.edu/2010/11/ | good |
| decorpad.com/photo.htm?photoId=6771 | good |

While feeding the data to any model, the data was processed in desired format using ***sklearn*** *library of Python.*

The identification as we know of a URL is done by identifying some patterns/combination of words, this makes the individual word/token as the key element to be considered as a feature and a suitable candidate to aid in the prediction.

The URLs provided in the dataset are inconsistent and contain some of the noise (unwanted features which can interfere with the prediction) that needs to be filtered.

For the purpose of cleaning the provided data, we use the Tfidfvectorizer method of sklearn library (sklearn.feature\_extraction.text.TfidfVectorizer)[12] that is used to convert a collection of raw documents into a matrix of Tfidf features.

The method Tfidfvectorizer has quite a handful of parameters but we focus mainly on the *tokenizer* parameter of TfidfVectorizer. This parameter based on the value provided makes a decision on how to split the (string/byte data). For our model we decided to make a seperate function by ourselves to set up the criteria for fetching tokens of a URL.

In this method for tokenizing the URLs, we have implemented such a functionality so that the URLs can be split based on the occurrence of redundant variables like slash(/), hyphen(-) and most frequently occurring words like .com/.net that do not help in identifying whether the URLs are *good* or *bad.*

Also, since ours is a classification problem with two classes (good/bad), it was necessary to encode these values to make it machine understandable.To fulfill these requirements we use the ***LabelEncoder***method of *sklearn.preprocessing* library and we follow the *one hot encoding approach.* The label of the dataset is now classified as 0/1 with 0 represented as *good* and 1 as *bad.*

VI. LOGISTIC REGRESSION

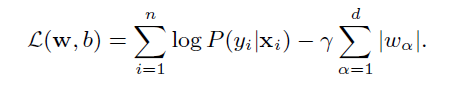
Linear regression works well for continuous data but fails for discrete response data. The linear regression tries to fit a straight line instead of calculating the probabilities of occurring of events. Also, if we extrapolate the line generated if at all we fit linear model for classification models, the probability can sometimes decrease below 0 or give results more than 1.

A better approach is Logistic regression: it tries to squeeze the output between 0 and 1 that is plotted as a sigmoid function .

where Ø(x) is the same as we use in linear regression i.e

= β 0 + β 1 x 1( i ) + … + β p xp ( i )

After tokenizing each of the URL, the input is the vector that is Vectorized by the TFID Vectorizer. Firstly, the number of features are high since each of the words in the URL can be considered as a token thereby one possible feature. Also features are so sparse and is available in only certain URLs. For example, say some random combination of words such as “svadf” can be found in only one URL at most. Also some feature can be irrelevant. Logistic regression is suitable for large, irrelevant features and also sparse data. Say a vector Xi = {x1,x2…,xn} and Yi represents the binary classifier if the website is malicious or benign. We use maximum likelihood estimation, to regularize the logistic regression with the following function:



The first component of the right-hand side specifies the log-likelihood that ensures the labels correct in the dataset. We use l1-norm regularization to reduce large magnitude values that is the second term which is used for sparse solutions. The γ is the regularization parameter is calculated by cross validation.

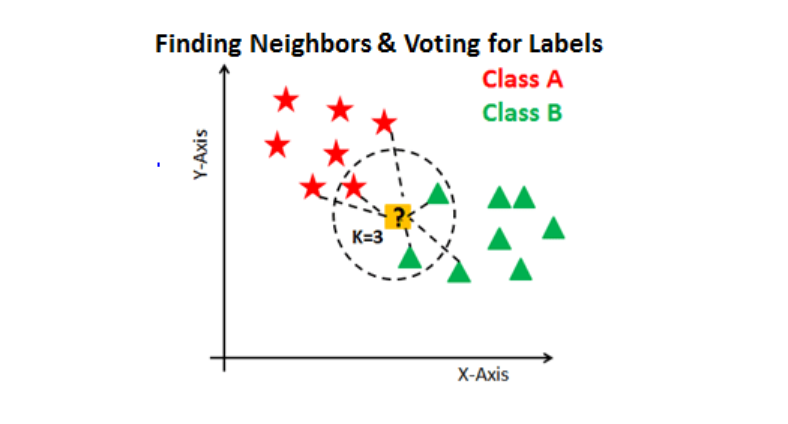
The next step is to find the threshold value of the probability. The threshold value is calculated through the greedy approach as the incoming features are different for each dataset. Then the model is trained based on the extracted features as discussed above and the accuracy was obtained as 96.16%.

VII K-NEAREST NEIGHBORS

*K-nearest neighbor (also called k-nn)* algorithm falls under the *supervised learning* technique and is a non-parametric method used both for classification as well as regression.

For a classification problem the input taken is the k nearest neighbors in the feature space and in the output the object is classified based upon the class most common among its k nearest neighbors. To avoid any conflicts the value of k is always selected to be an odd number and hence the concept of majority vote helps in determining the class of the outcome[11].

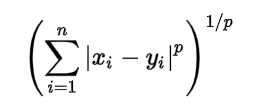
In this approach all the k nearest neighbors are assigned some weight and the prediction for a sample is made by finding the difference in the weights of the sample and k nearest neighbors. In vector space, one way of achieving this is to compute the euclidean distance of each neighbor and the sample and an average can be taken of the closest samples to make a fair prediction.



***Fig 2:*** *Demonstration of K-nn and majority voting*

The above *Fig 2* demonstrates how the similar classes exist all together and for k = 3 how the distance is calculated and a decision is made using the average of the closely occuring neighbors (concept of majority voting).

Since it is advisable to have a small value of k (the neighbors of neighbors to be selected), for our project we have kept it equal to the default value of 5 and for computing the distance among the sample and neighboring data points we use *minkowski distance:*

**

we keep the value of p = 2(default) and hence the above formula behaves like *Euclidean distance.*

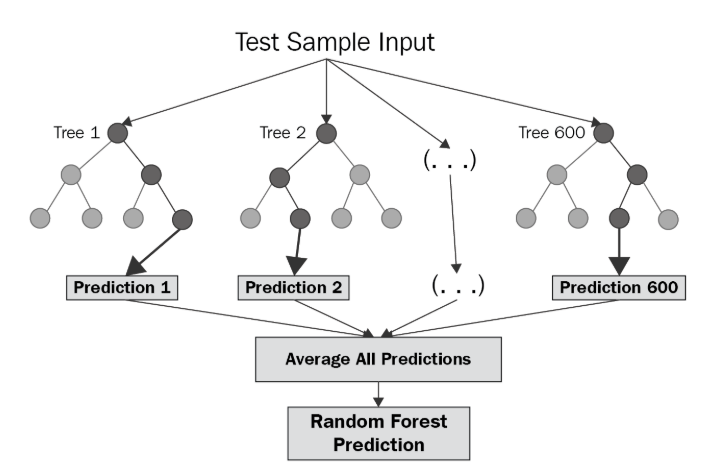
*K-NN model* in the project is trained using cross validation technique[12] with 80% of the data reserved for training split while 20% is kept for the test split. The features for KNN model are kept the same as we used for logistic regression.

The model on completing learning gives an accuracy of as high as **97.9%** on test split however we do not consider it to be the best approach as the model takes a lot of time while training and also the model is not a practical approach in situations where predictions need to be made rapidly.

VIII. RANDOM FORESTS

We chose Random forests as one of our classification methods because of its ability to classify datasets with the use of an ensemble of decision trees which makes it one of the best among the real-world classification problems.

Random forests uses a meta estimator that fits a sub-0 samples of dataset on numerous decision tree classifiers which improves accuracy in prediction and controls the problem of overfitting. Random forest boosts the performance of the model using two steps. In the first step the algorithm selections several samples from the world data (the whole dataset) called bootstrap samples each of size which are roughly two-third of the entire training dataset, after which the cases are selected in random by a replacement technique which selects random cases with replacement from original data and the observations in the original data that did not occur in the bootstrap sample. This kind of random selection is called out-of-bag (OOB) observation. The second step is to train a classification tree from the observations taken from first step. The tree is trained using each sample of observation but only with a selected variables which is used in the tree partition. The final output will be a mean of all predictions from the individual trees as shown in *Fig3.*



***Fig 3:*** *Random forest prediction flow*

Here is the detailed algorithm working of random forests which involves techniques like bagging and construction of decision trees. Consider the following parameters:

X = x1, x2, …., xn - Input attributes

Y = y1, y2, ….., yn - Class (Benign or Malicious)

After reading the input the Random forest model follows the below steps to generate a prediction

1. Split each URL data according to their features (Lexical, word count, IP etc)

2. Calculate term frequency of each URL

3. A feature matrix is created for all URLs with inverse document frequency

4. Now the actual classification occurs, consider the following

s = 1,...,S be the samples

The sampling is repeated with replacement for n training examples

5. The samples are trained on a regression tree on ,

6. Let be the prediction for unseen samples which can be found by averaging all the prediction from individual decision trees on based on the below formula

()

For our dataset the accuracy obtained was 0.82 which approximated to 82%.

IX. EXPERIMENTAL RESULTS

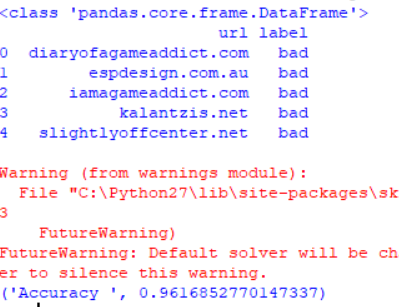
After training and testing each of the model, the computation time taken to train each of the model is tabulated below:

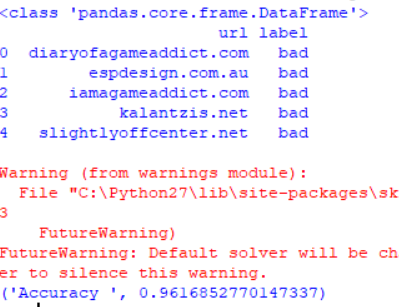
*TABLE 2*:*Computation time*

|  |  |  |
| --- | --- | --- |
| **Logistic Regression** | **KNN** | **Random Forest** |
| 1 minute | 50 mins | 1 hour 10 mins |

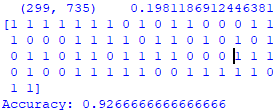
From the computation time, we can infer that logistic regression outperforms the other models in terms of time complexity and response time.

The accuracies of each of the model are listed below:



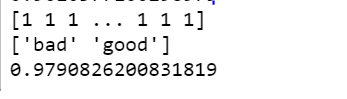


***Fig 4:*** *Logistic Regression Accuracy*



***Fig 5:*** *Random forest Accuracy*

From *Fig5* the accuracy of Random forest is less when compared to that of Logistic regression and the time taken for it’s response is no match for Logistic regression.

**

***Fig 6:*** *K-Nearest Neighbor Accuracy*

From *Fig6* though the accuracy of K-NN is more than logistic regression while considering the response time the logistic regression out performs K-NN, as it is a responsive time intensive model which involves a huge security risk if the model takes longer time for the prediction.

X. FUTURE WORK

In the current implementation, Logistic Regression model is predicting the URLs at an efficiency of 96.16% under 1 minute but the scope of the project is still limited to static URL prediction and we wish to increase the scope of the project in certain possible aspects.

Our future work will be to implement the Architecture as shown in *Fig1,* Where we will replace the static URL dataset by a live URL feed which will have a real time repository of updated list of malicious URLs from the user reports and the current blacklisted websites. Also the feature selector will also be dynamic and real time. This whole architecture will be wrapped with an interactive UI and it will be modularized as an extension or a browser add-on to warn users about the malicious URLs they are about to visit or to integrate the model to an Antivirus software which will prevent the user from accessing the URL or preventing the browser in downloading the page contents.

X. CONCLUSION

This paper describes an approach for categorizing the URLs into malicious and benign with the help of machine learning classification techniques. We can infer that this approach is much faster and better than blacklisting where new URLs cannot be classified. Further, we use different classifier to justify our statement through Logistic regression, KNN and Random Forest.

From the results obtained, we can infer that the ratio of computation time and accuracy for Logistic regression were better followed by KNN and Random Forest. Also, we can state that Logistic regression can be used for real time prediction as it is optimum in scenarios where predictions are to be made rapidly and accurately.

REFERENCES

[1] S. P. Ramesh and Murugan, “Survey on Web based Application Attacks & Improves Security,” *Asian Journal of Research in Social Sciences and Humanities*, vol. 6, no. 6, p. 382, 2016.

[2] Dharmaraj Rajaram Patil and JB Patil. 2015. Survey on Malicious Web Pages Detection Techniques. International Journal of u-and e-Service, Science and Technology (2015).

[3]M. Akiyama, T. Yagi, and T. Hariu, “Improved Blacklisting: Inspecting the Structural Neighborhood of Malicious URLs,” *IT Professional*, vol. 15, no. 4, pp. 50–56, 2013.

[4]S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, “A comparison of machine learning techniques for phishing detection,” *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit on - eCrime 07*, 2007.

[5] A. K. Singh and N. Goyal, “A Comparison of Machine Learning Attributes for Detecting Malicious Websites,” *2019 11th International Conference on Communication Systems & Networks (COMSNETS)*, 2019.

[6] F. Vanhoenshoven, G. Napoles, R. Falcon, K. Vanhoof, and M. Koppen, “Detecting malicious URLs using machine learning techniques,” 2016 IEEE Symposium Series on Computational Intelligence (SSCI), 2016.

[7] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Beyond blacklists,” *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 09*, 2009.

[8] M.-Y. Kan and H. O. N. Thi, “Fast webpage classification using URL features,” *Proceedings of the 14th ACM international conference on Information and knowledge management - CIKM 05*, 2005.

[9] Y. Niu, Y.-M. Wang, H. Chen, M. Ma, and F. Hsu. A Quantitative Study of Forum Spamming Using Context-based Analysis. In Proceedings of the Symposium on Network and Distributed System Security (NDSS), San Diego, CA, Mar. 2007.

[10]faizann24,“faizann24/Using-machine-learning-to-detect-malicious-URLs,” GitHub. [Online]. Available: https://github.com/faizann24/Using-machine-learning-to-detect-malicious-URLs/blob/master/data/data.csv. [Accessed: 10-Dec-2019].

[11]“K-nearest neighbors algorithm,” Wikipedia, 17-Dec-2019.[Online].Available:https://en.wikipedia.org/wiki/K-nearest\_neighbors\_algorithm.[Accessed:20-Dec-2019].

[12]“sklearn.feature\_extraction.text.TfidfVectorizer,” scikit.[Online].Available:https://scikit-learn.org/stable/modules/generated/sklearn.feature\_extraction.text.TfidfVectorizer.html. [Accessed: 20-Dec-2019].