Ensemble Machine Learning Model for Phishing Intrusion Detection and Classification from URLs

R. Pravali[1], Sk. Raha[1], Y. Rachana[1], Dr. D. B. K. Kamesh[2]

[1]UG Student, [2]Professor, [1,2]Department of Computer Science and Engineering

[1,2]Malla Reddy Engineering College for Women (UGC-Autonomous), Maisammaguda, Secunderabad, Telangana, India

ABSTRACT

Phishing sounds like fishing (which means to cash fish) is a term used for an attempt to commit financial fraud on the internet. An e-mail scam is carried out on individuals or corporate organizations in an attempt to defraud them by falsely obtaining their sensitive details such as usernames, passwords, credit card information, and account numbers. For example, an email may be sent to an individual and appears with a link to click, such as “click me” showing that the recipient has won a certain amount of money, and thereafter requesting him to provide account information for verification. Unfortunately, the credentials are actually transmitted to a phisher who may exploit the person's account when the receiver sends the account details for validation. This research’s focus is to utilize different machine learning classification models to predict whether a given URL is 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.

Keywords: Phishing intrusion, URLs, cyber-attack, machine learning, XGBoost algorithm.

1. INTRODUCTION

1.1 Overview

Phishing is a fraudulent technique that uses social and technological tricks to steal customer identification and financial credentials. Social media systems use spoofed e-mails from legitimate companies and agencies to enable users to use fake websites to divulge financial details like usernames and passwords [1]. Hackers install malicious software on computers to steal credentials, often using systems to intercept username and passwords of consumers’ online accounts. Phishers use multiple methods, including email, Uniform Resource Locators (URL), instant messages, forum postings, telephone calls, and text messages to steal user information. The structure of phishing content is similar to the original content and trick users to access the content in order to obtain their sensitive data. The primary objective of phishing is to gain certain personal information for financial gain or use of identity theft. Phishing attacks are causing severe economic damage around the world. Moreover, most phishing attacks target financial/payment institutions and webmail, according to the Anti-Phishing Working Group (APWG) latest Phishing pattern studies [1]. In order to receive confidential data, criminals develop unauthorized replicas of a real website and email, typically from a financial institution or other organization dealing with financial data. This e-mail is rendered using a legitimate company's logos and slogans. The design and structure of HTML allow copying of images or an entire website. Also, it is one of the factors for the rapid growth of Internet as a communication medium, and enables the misuse of brands, trademarks, and other company identifiers that customers rely on as authentication mechanisms. To trap users, Phisher sends "spooled" mails to as many people as possible. When these e-mails are opened, the customers tend to be diverted from the legitimate entity to a spoofed website.

There is a significant chance of exploitation of user information. For these reasons, phishing in modern society is highly urgent, challenging, and overly critical. There have been several recent studies against phishing based on the characteristics of a domain, such as website URLs, website content, incorporating both the website URLs and content, the source code of the website and the screenshot of the website. However, there is a lack of useful anti-phishing tools to detect malicious URL in an organization to protect its users. In the event of malicious code being implanted on the website, hackers may steal user information and install malware, which poses a serious risk to cybersecurity and user privacy.

1.2 Problem Statement

Phishing assault is a most straightforward approach to get delicate data from honest clients. Point of the phishers is to obtain basic data like username, secret key, and ledger subtleties. Network safety people are currently searching for dependable and consistent location methods for phishing sites recognition. To overcome the drawbacks of blacklist and heuristics-based method, many security researchers now focused on machine learning techniques. Machine learning technology consists of many algorithms which requires past data to decide or prediction on future data. Using this technique, algorithm will analyze various blacklisted and legitimate URLs and their features to accurately detect the phishing websites including zero- hour phishing websites.

2. LITERATURE SURVEY

Phishing attacks are categorized according to Phisher's mechanism for trapping alleged users. Several forms of these attacks are keyloggers, DNS toxicity, Etc., [2]. The initiation processes in social engineering include online blogs, short message services (SMS), social media platforms that use web 2.0 services, such as Facebook and Twitter, file-sharing services for peers, Voice over IP (VoIP) systems where the attackers use caller spoofing IDs [3, 4]. Each form of phishing has a little difference in how the process is carried out in order to defraud the unsuspecting consumer. E-mail phishing attacks occur when an attacker sends an e-mail with a link to potential users to direct them to phishing websites.

Phishing websites are challenging to an organization and individual due to its similarities with the legitimate websites [5]. There are multiple forms of phishing attacks. Technical subterfuge refers to the attacks include Keylogging, DNS poisoning, and Malwares. In these attacks, attacker intends to gain the access through a tool/technique. On the one hand, users believe the network and on the other hand, the network is compromised by the attackers. Social engineering attacks include Spear phishing, Whaling, SMS, Vishing, and mobile applications. In these attacks, attackers focus on the group of people or an organization and trick them to use the phishing URL [6, 7]. Apart from these attacks, many new attacks are emerging exponentially as the technology evolves constantly. Phishing detection schemes which detect phishing on the server side are better than phishing prevention strategies and user training systems. These systems can be used either via a web browser on the client or through specific host-site software [8, 9].

3. EXISTING SYSTEM

3.1 Logistic Regression

Logistic regression predicts the probability of an outcome that can only have two values (i.e., a dichotomy). The prediction is based on the use of one or several predictors (numerical and categorical). A linear regression is not appropriate for predicting the value of a binary variable for two reasons:

- A linear regression will predict values outside the acceptable range (e.g., predicting probabilities
- outside the range 0 to 1)
- Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.

On the other hand, a logistic regression produces a logistic curve, which is limited to values between 0 and 1. Logistic regression is similar to a linear regression, but the curve is constructed using the natural logarithm of the "odds" of the target variable, rather than the probability. Moreover, the predictors do not have to be normally distributed or have equal variance in each group.

In the logistic regression the constant (b0) moves the curve left and right and the slope (b1) defines the steepness of the curve. By simple transformation, the logistic regression equation can be written in terms of an odds ratio.

$$\frac{p}{1-p} = \exp\left(b_0 + b_1 x\right)$$

Finally, taking the natural log of both sides, we can write the equation in terms of log-odds (logit) which is a linear function of the predictors. The coefficient (*b_1*) is the amount the logit (log-odds) changes with a one unit change in *x*.

$$ln\left(\frac{p}{1-p}\right) = b_0 + b_1 x$$

As mentioned before, logistic regression can handle any number of numerical and/or categorical variables.

$$p = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p)}}$$

There are several analogies between linear regression and logistic regression. Just as ordinary least square regression is the method used to estimate coefficients for the best fit line in linear regression, logistic regression uses maximum likelihood estimation (MLE) to obtain the model coefficients that relate predictors to the target. After this initial function is estimated, the process is repeated until LL (Log Likelihood) does not change significantly.

$$\boldsymbol{\beta}^1 = \boldsymbol{\beta}^0 + [\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}]^{-1}.\boldsymbol{X}^T(\boldsymbol{y}-\boldsymbol{\mu})$$

$\boldsymbol{\beta}$ is a vector of the logistic regression coefficients.

$\boldsymbol{W}$ is a square matrix of order N with elements $n_i\pi_i(1-\pi_i)$ on the diagonal and zeros everywhere else.

$\boldsymbol{\mu}$ is a vector of length N with elements $\mu_i = n_i\pi_i$.

A pseudo R^2 value is also available to indicate the adequacy of the regression model. Likelihood ratio test is a test of the significance of the difference between the likelihood ratio for the baseline model minus the likelihood ratio for a reduced model. This difference is called "model chi-square “. Wald test is used to test the statistical significance of each coefficient (*b*) in the model (i.e., predictors contribution).

Pseudo R2

There are several measures intended to mimic the R2 analysis to evaluate the goodness-of-fit of logistic models, but they cannot be interpreted as one would interpret an R2 and different pseudo R2 can arrive at very different values. Here we discuss three pseudo R2measures.

Likelihood Ratio Test

The likelihood ratio test provides the means for comparing the likelihood of the data under one model (e.g., full model) against the likelihood of the data under another, more restricted model (e.g., intercept model).

$$LL = \sum_{i=1}^{n} y_i ln(p_i) + (1-y_i)ln(1-p_i)$$

where *'p'* is the logistic model predicted probability. The next step is to calculate the difference between these two log-likelihoods.

$$2(LL_1 - LL_2)$$

The difference between two likelihoods is multiplied by a factor of 2 in order to be assessed for statistical significance using standard significance levels (Chi^2 test). The degrees of freedom for the test will equal the difference in the number of parameters being estimated under the models (e.g., full and intercept).

3.2 Drawbacks of existing system

- If the number of observations is lesser than the number of features, Logistic Regression should not be used, otherwise, it may lead to overfitting.
- It can only be used to predict discrete functions. Hence, the dependent variable of Logistic Regression is bound to the discrete number set.
- Logistic Regression requires average or no multicollinearity between independent variables.

4. PROPOSED SYSTEM

This section describes the proposed ensemble machine learning model for the detection of phishing intrusions from URL dataset. Fig. 1 demonstrate the proposed architecture of phishing intrusion detection using ensemble machine learning model, where the system training phase includes data uploading, preprocessing, building, and training the machine learning model such as logistic regression, and XGBoost, and performance evaluation. Here the performance evaluation is done using the calculation of confusion matrix, and training accuracy. The second phase i.e., prediction involves the test URL data uploading, preprocessing, TF-IDF vectorizer, applying XGBoost model and final prediction of given URL.

4.1 Data Preprocessing in Machine learning

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put it in a formatted way. So, for this, we use data pre-processing tasks.

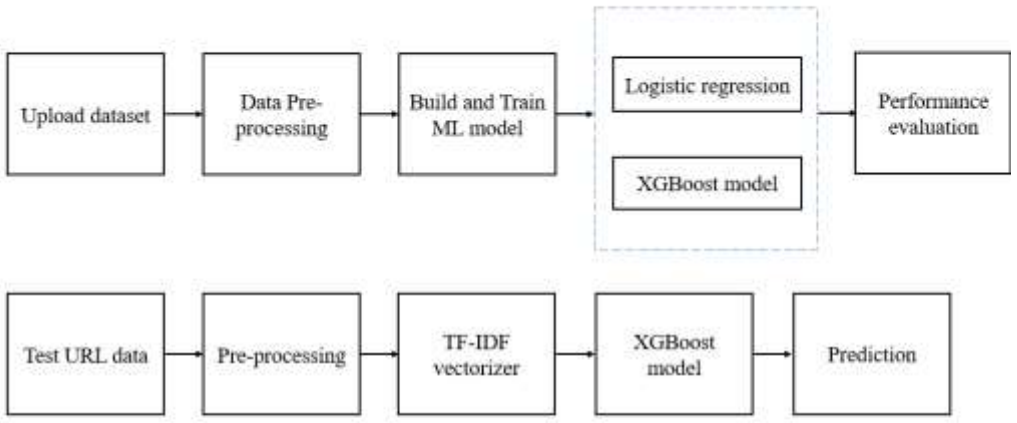


Fig. 1: Proposed architecture for phishing intrusion detection from URLs using ensemble learning model.

4.2 TF-IDF Feature extraction

TF-IDF which stands for Term Frequency – Inverse Document Frequency. It is one of the most important techniques used for information retrieval to represent how important a specific word or phrase is to a given document. Let’s take an example, we have a string or Bag of Words (BOW) and we have to extract information from it, then we can use this approach.

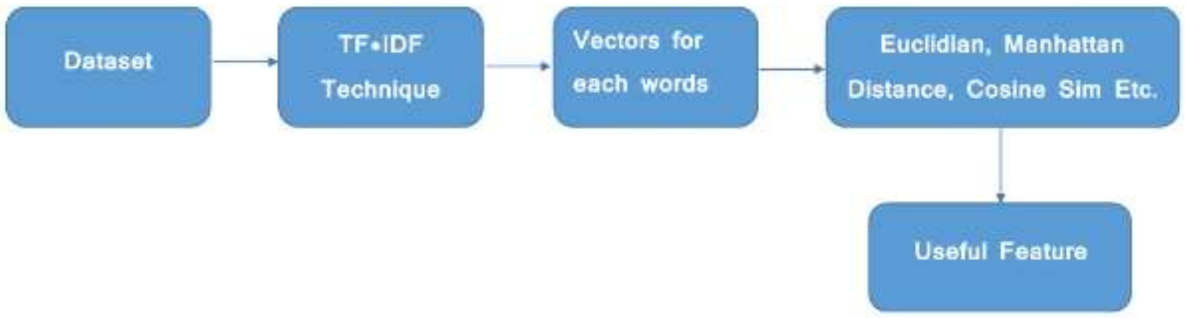


Fig. 2: TF-IDF block diagram.

The tf-idf value increases in proportion to the number of times a word appears in the document but is often offset by the frequency of the word in the corpus, which helps to adjust with respect to the fact that some words appear more frequently in general. TF-IDF use two statistical methods, first is Term

Frequency and the other is Inverse Document Frequency. Term frequency refers to the total number of times a given term t appears in the document doc against (per) the total number of all words in the document and The inverse document frequency measure of how much information the word provides. It measures the weight of a given word in the entire document. IDF show how common or rare a given word is across all documents. TF-IDF can be computed as tf * idf

TF-IDF do not convert directly raw data into useful features. Firstly, it converts raw strings or dataset into vectors and each word has its own vector. Then we'll use a particular technique for retrieving the feature like Cosine Similarity which works on vectors, etc.

Terminology

t — term (word)

d — document (set of words)

N — count of corpus

corpus — the total document set

Step 1: Term Frequency (TF): Suppose we have a set of English text documents and wish to rank which document is most relevant to the query, "Data Science is awesome!" A simple way to start out is by eliminating documents that do not contain all three words "Data" is", "Science", and "awesome", but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document; the number of times a term occurs in a document is called its term frequency. The weight of a term that occurs in a document is simply proportional to the term frequency.

$$tf(t,d) = count\ of\ t\ in\ d\ /\ number\ of\ words\ in\ d$$

Step 2: Document Frequency: This measures the importance of document in whole set of corpora, this is very similar to TF. The only difference is that TF is frequency counter for a term t in document d, whereas DF is the count of occurrences of term t in the document set N. In other words, DF is the number of documents in which the word is present. We consider one occurrence if the term consists in the document at least once, we do not need to know the number of times the term is present.

$$df(t) = occurrence\ of\ t\ in\ documents$$

Step 3: Inverse Document Frequency (IDF): While computing TF, all terms are considered equally important. However, it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus, we need to weigh down the frequent terms while scale up the rare ones, by computing IDF, an inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely. **The** IDF is the inverse of the document frequency which measures the informativeness of term t. When we calculate IDF, it will be very low for the most occurring words such as stop words (because stop words such as "is" is present in almost all of the documents, and N/df will give a very low value to that word). This finally gives what we want, a relative weightage.

$$idf(t) = N/df$$

Now there are few other problems with the IDF, in case of a large corpus, say 100,000,000 , the IDF value explodes , to avoid the effect we take the log of idf . During the query time, when a word which is not in vocab occurs, the df will be 0. As we cannot divide by 0, we smoothen the value by adding 1 to the denominator.

$$idf(t) = log(N/(df + 1))$$

The TF-IDF now is at the right measure to evaluate how important a word is to a document in a collection or corpus. Here are many different variations of TF-IDF but for now let us concentrate on this basic version.

$$tf - idf(t,d) = tf(t,d) * log(N/(df + 1))$$

4.3 XGBoost Algorithm

XgBoost stands for Extreme Gradient Boosting, which was proposed by the researchers at the University of Washington. It is a library written in C++ which optimizes the training for Gradient Boosting. Before understanding the XGBoost, we first need to understand the trees especially the decision tree.

4.3.1 Decision Tree

A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. A tree can be “learned” by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions.

4.3.2 Bagging

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it. Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples(or data) from the original training dataset, where N is the size of the original training set. The training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out. Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance though.

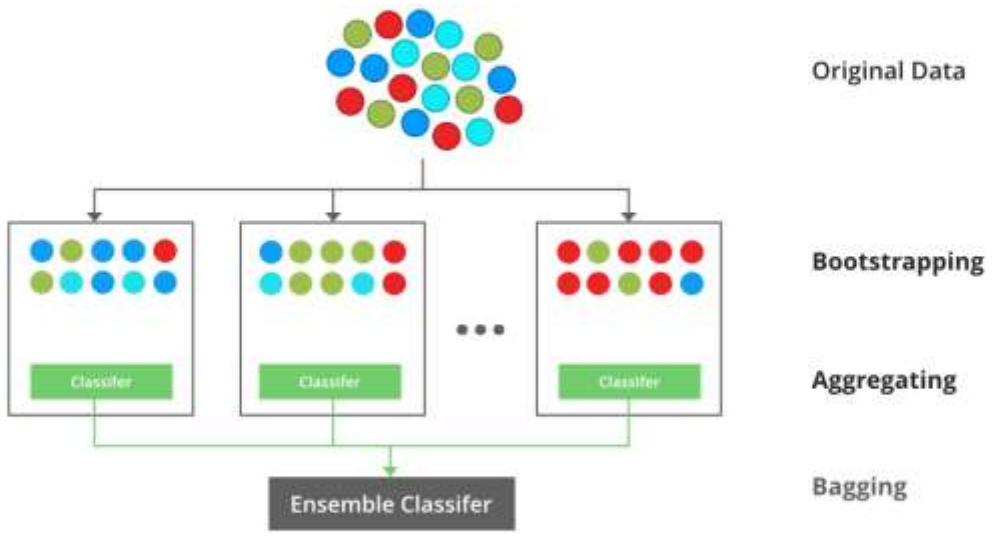


Fig. 3: Architecture of bagging classifier.

4.3.3 Random Forest

Every decision tree has high variance, but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence the output doesn't depend on one decision tree but multiple decision trees. In the case of a classification problem, the final output is taken by using the majority voting classifier. In the case of a regression problem, the final output is the mean of all the outputs. This part is Aggregation.

The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees. Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.

4.3.4 Boosting

Boosting is an ensemble modelling, technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.

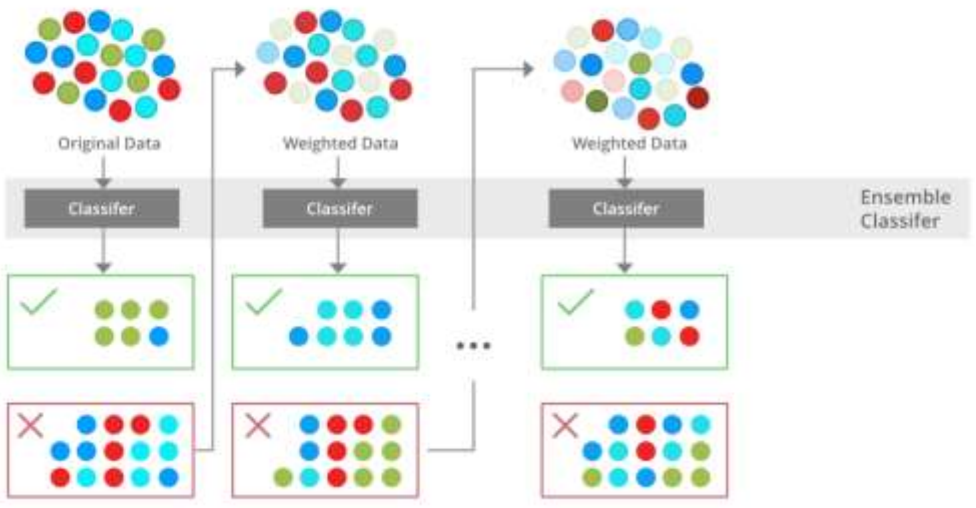


Fig. 4: Architecture of boosting.

4.3.5 Gradient Boosting

Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor's error. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels. There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees).

4.3.6 XGBoost

XGBoost is an implementation of Gradient Boosted decision trees. XGBoost models majorly dominate in many Kaggle Competitions. In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

4.4 Advantages of proposed system

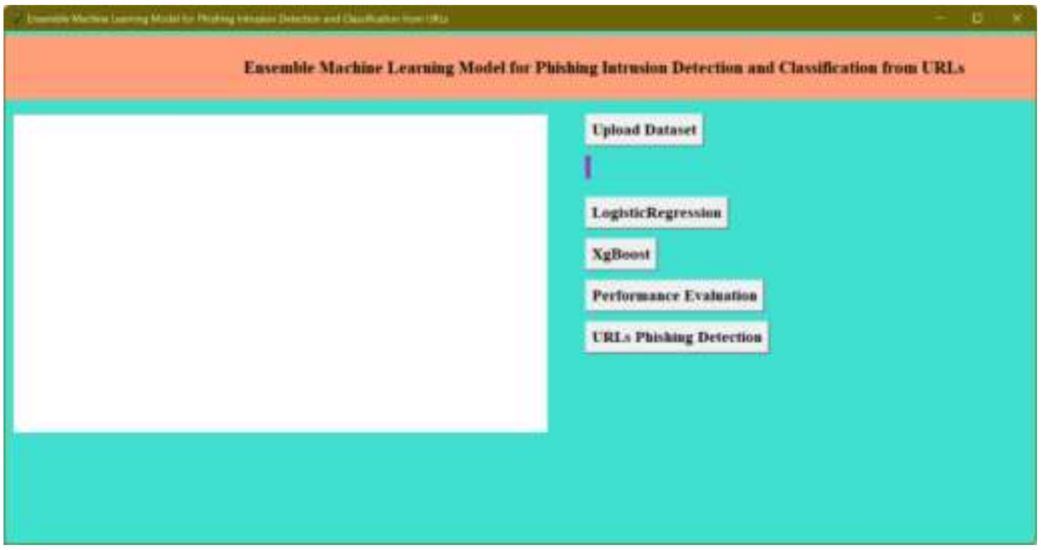
- Simple and easy to implement.
- Training phase is faster due to lazy learning.
- Suitable for multi class problems.
- Works better for continuously changing data due to instance-based learning.

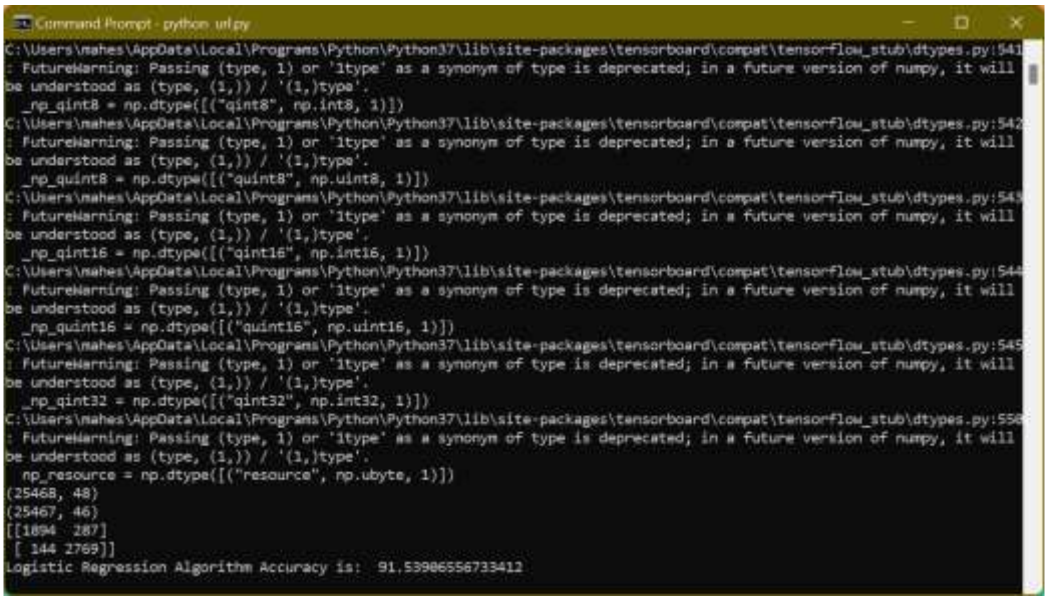
5. RESULTS AND DISCUSSION

1. Loading Data
   - The dataset is collected from open-source platform.
   - A collection of website URLs for 11000+ websites. Each sample has 30 website parameters and a class label identifying it as a phishing website or not (1 or -1).
   - The overview of this dataset is it has 11054 samples with 32 features.
2. EDA: In this step, a few data frame methods are used to look into the data and its features.

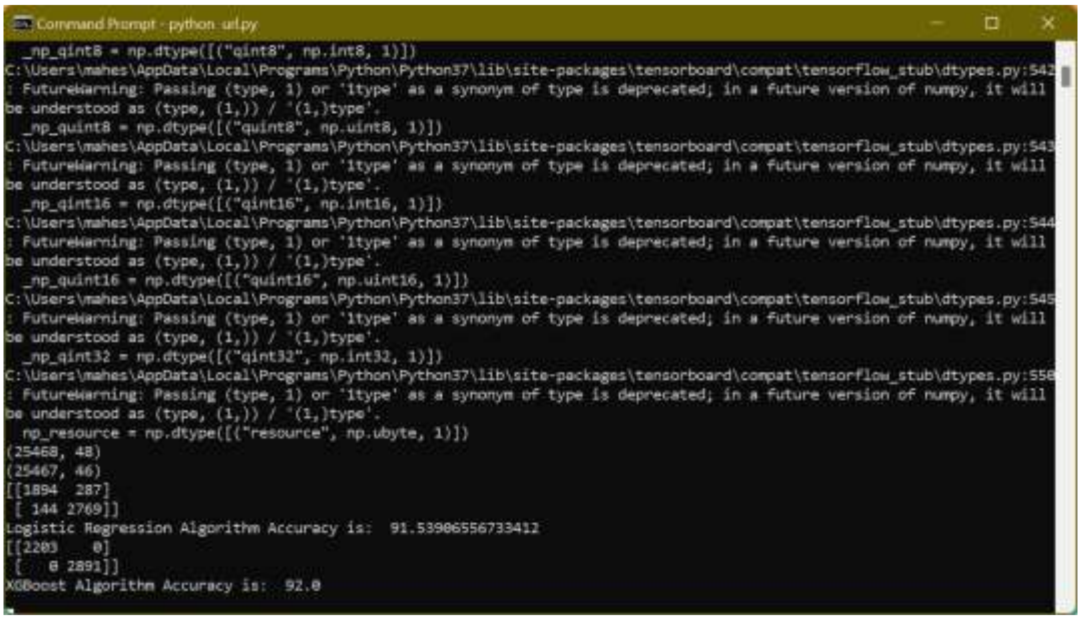
3. Data Splitting: The data is split into train & test sets, 80-20 split.
4. Building and Training ML Model

This project describes the concept of phishing intrusion detection and classification using NLP-based Machine learning algorithms such as logistic regression and XGBoost classification. In addition, it will also detect phishing intrusion from test URL.

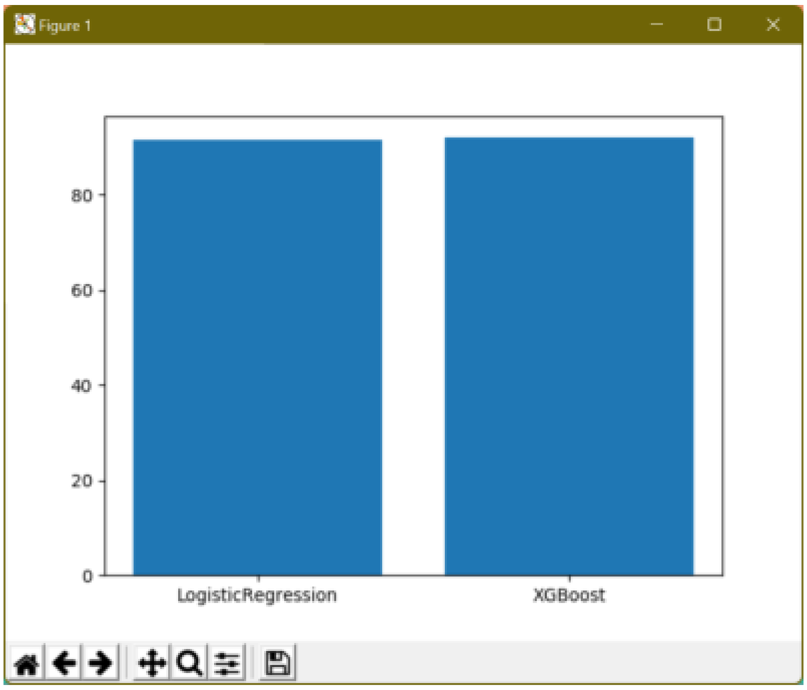

Ensemble Machine Learning Model for Phishing Intrusion Detection and Classification from URLs
Upload Dataset
LogisticRegression
XgBoost
Performance Evaluation
URLs Phishing Detection

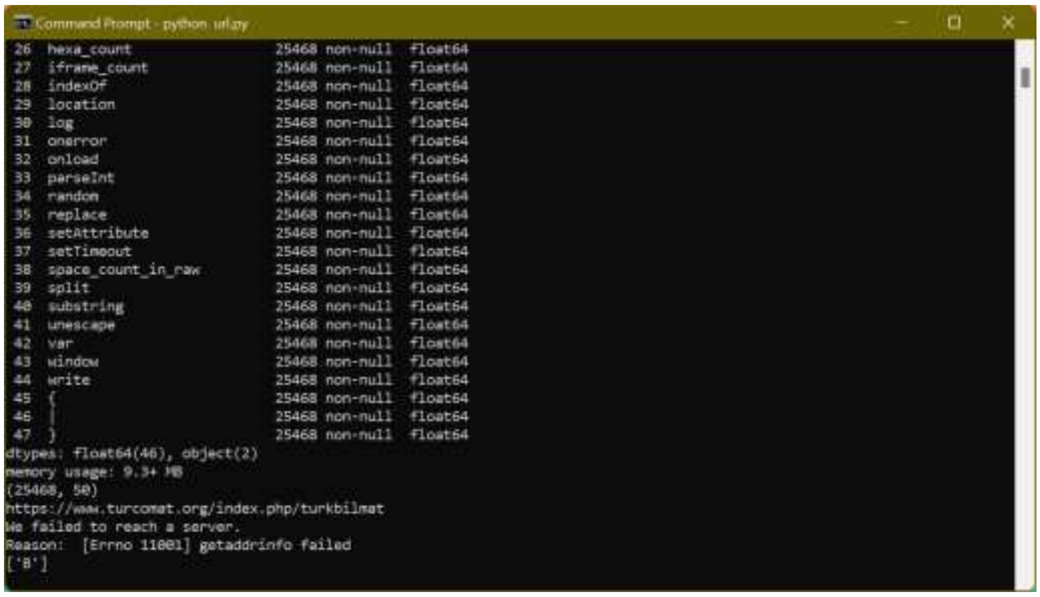


```
C:\Users\mahes\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:541
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\mahes\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:542
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\mahes\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:543
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\mahes\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:544
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\mahes\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:545
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\mahes\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:550
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
(25468, 48)
(25467, 46)
[[1894  287]
 [ 144 2769]]
Logistic Regression Algorithm Accuracy is:  91.53986556733412
```

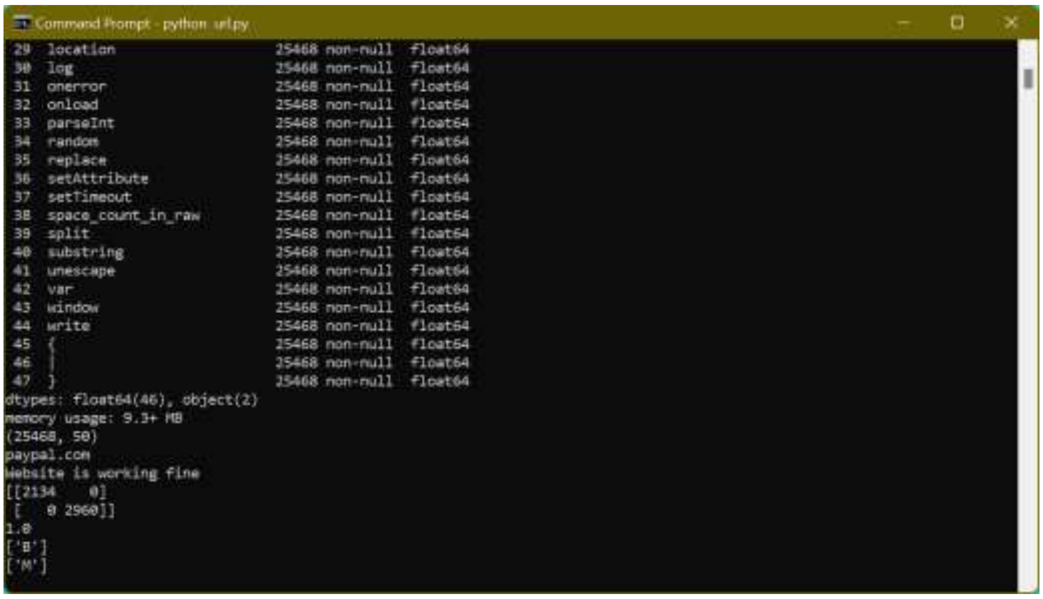


```
Command Prompt - python url.py
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\mahes\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:542
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\mahes\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:543
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\mahes\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:544
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\mahes\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:545
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\mahes\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:550
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
(25468, 48)
(25467, 46)
[[1894  287]
 [ 144 2769]]
Logistic Regression Algorithm Accuracy is:  91.53906556733412
[[2203    0]
 [   0 2891]]
XGBoost Algorithm Accuracy is:  92.0
```

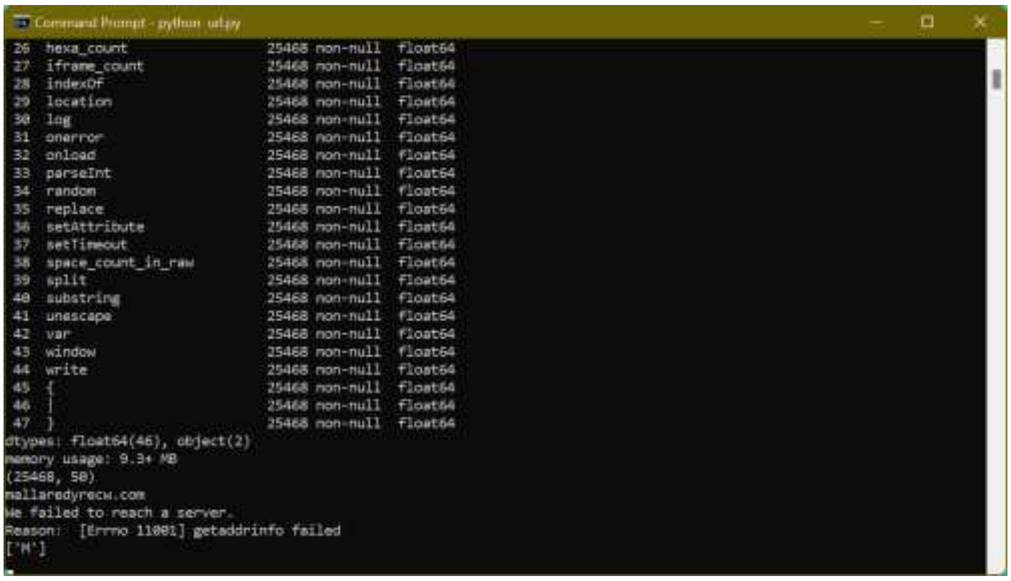




```
 26  hexa_count                 25468 non-null  float64
 27  iframe_count               25468 non-null  float64
 28  indexOf                    25468 non-null  float64
 29  location                   25468 non-null  float64
 30  log                        25468 non-null  float64
 31  onerror                    25468 non-null  float64
 32  onload                     25468 non-null  float64
 33  parseInt                   25468 non-null  float64
 34  random                     25468 non-null  float64
 35  replace                    25468 non-null  float64
 36  setAttribute               25468 non-null  float64
 37  setTimeout                 25468 non-null  float64
 38  space_count_in_raw         25468 non-null  float64
 39  split                      25468 non-null  float64
 40  substring                  25468 non-null  float64
 41  unescape                   25468 non-null  float64
 42  var                        25468 non-null  float64
 43  window                     25468 non-null  float64
 44  write                      25468 non-null  float64
 45  {                          25468 non-null  float64
 46  |                          25468 non-null  float64
 47  }                          25468 non-null  float64
dtypes: float64(46), object(2)
memory usage: 9.3+ MB
(25468, 50)
https://www.turcomat.org/index.php/turkbilmat
We failed to reach a server.
Reason:  [Errno 11001] getaddrinfo failed
['B']
```



```
 29  location                   25468 non-null  float64
 30  log                        25468 non-null  float64
 31  onerror                    25468 non-null  float64
 32  onload                     25468 non-null  float64
 33  parseInt                   25468 non-null  float64
 34  random                     25468 non-null  float64
 35  replace                    25468 non-null  float64
 36  setAttribute               25468 non-null  float64
 37  setTimeout                 25468 non-null  float64
 38  space_count_in_raw         25468 non-null  float64
 39  split                      25468 non-null  float64
 40  substring                  25468 non-null  float64
 41  unescape                   25468 non-null  float64
 42  var                        25468 non-null  float64
 43  window                     25468 non-null  float64
 44  write                      25468 non-null  float64
 45  {                          25468 non-null  float64
 46  |                          25468 non-null  float64
 47  }                          25468 non-null  float64
dtypes: float64(46), object(2)
memory usage: 9.3+ MB
(25468, 50)
paypal.com
Website is working fine
[[2134    0]
 [   0 2960]]
1.0
['B']
['M']
```



```
 26  hexa_count                 25468 non-null  float64
 27  iframe_count               25468 non-null  float64
 28  indexOf                    25468 non-null  float64
 29  location                   25468 non-null  float64
 30  log                        25468 non-null  float64
 31  onerror                    25468 non-null  float64
 32  onload                     25468 non-null  float64
 33  parseInt                   25468 non-null  float64
 34  random                     25468 non-null  float64
 35  replace                    25468 non-null  float64
 36  setAttribute               25468 non-null  float64
 37  setTimeout                 25468 non-null  float64
 38  space_count_in_raw         25468 non-null  float64
 39  split                      25468 non-null  float64
 40  substring                  25468 non-null  float64
 41  unescape                   25468 non-null  float64
 42  var                        25468 non-null  float64
 43  window                     25468 non-null  float64
 44  write                      25468 non-null  float64
 45  {                          25468 non-null  float64
 46  |                          25468 non-null  float64
 47  }                          25468 non-null  float64
dtypes: float64(46), object(2)
memory usage: 9.3+ MB
(25468, 50)
mallaredyrecw.com
We failed to reach a server.
Reason:  [Errno 11001] getaddrinfo failed
['M']
```

6. CONCLUSION AND FUTURE SCOPE

Machine learning (ML) based phishing intrusion detection was proposed in this paper. The investigation utilizes many strategies to identify phishing intrusion detection. Standard datasets of phishing intrusion detection from kaggle.com were used as input for the ML algorithms. The machine learning algorithms called logistic regression and XGBoost algorithm are implemented to analyze and select datasets for classification and detection. The proposed XGBoost obtained enhanced accuracy as compared to logistic regression. In addition, the confusion matrix was also computed to evaluate the performance of two supervised algorithms. Finally, phishing intrusion prediction was also performed with trained ML model.

Future scope

In our future work, fishing attacks will be predicted from the logged dataset of attacks by using a convolution neural network (CNN). It will be added as a tool for intrusion detection system, and we plan to implement these solutions and develop a robust and generalized intrusion detection model.

REFERENCES

[1] Anti-Phishing Working Group (APWG), https://docs.apwg.org//reports/apwg_trends_report_q4_2019.pdf

[2] Jain A.K., Gupta B.B. "PHISH-SAFE: URL Features-Based Phishing Detection System Using Machine Learning", Cyber Security. Advances in Intelligent Systems and Computing, vol. 729, 2018,

[3] Purbay M., Kumar D, "Split Behavior of Supervised Machine Learning Algorithms for Phishing URL Detection", Lecture Notes in Electrical Engineering, vol. 683, 2021,

[4] Gandotra E., Gupta D, "An Efficient Approach for Phishing Detection using Machine Learning", Algorithms for Intelligent Systems, Springer, Singapore, 2021, https://doi.org/10.1007/978-981-15-8711-5_12.

[5] Hung Le, Quang Pham, Doyen Sahoo, and Steven C.H. Hoi, "URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection", Conference'17, Washington, DC, USA, arXiv:1802.03162, July 2017.

[6] Hong J., Kim T., Liu J., Park N., Kim SW, "Phishing URL Detection with Lexical Features and Blacklisted Domains", Autonomous Secure Cyber Systems. Springer, https://doi.org/10.1007/978-3-030-33432-1_12.

[7] J. Kumar, A. Santhanavijayan, B. Janet, B. Rajendran and B. S. Bindhumadhava, "Phishing Website Classification and Detection Using Machine Learning," 2020 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2020, pp. 1–6, 10.1109/ICCCI48352.2020.9104161.

[8] Hassan Y.A. and Abdelfettah B, "Using case- based reasoning for phishing detection", Procedia Computer Science, vol. 109, 2017, pp. 281–288.

[9] Rao RS, Pais AR. Jail-Phish: An improved search engine-based phishing detection system. Computers & Security. 2019 Jun 1; 83:246–67.