Multimedia Tools and Applications (2023) 82:29431–29456
https://doi.org/10.1007/s11042-023-14731-4

Intelligent phishing website detection using machine learning

Ashish Kumar Jha[1] · Raja Muthalagu[1] · Pranav M. Pawar[1]

Received: 3 January 2022 / Revised: 24 March 2022 / Accepted: 4 February 2023 /
Published online: 24 February 2023

Abstract

The need for cyber security is growing every day as the amount of data available online continues to rise exponentially. The cyber security has become a field of prime importance in the recent years and will continue to be so. Hackers and malpractitioners are growing day by day and are using varied methods and techniques to extract information of prime importance from the users. "Phishing" is one of the most common yet unique security concern. It is unique in the way that instead of targeting the system vulnerabilities, it is a social engineering attack targeting human vulnerabilities. Users give up their personal and sensitive data viz. passwords, card details, bank details etc. by falling to scam emails or websites. The target of this research is to create a tool which will help to detect and differentiate a phishing website from a safe website, thus preventing users into opening risky URLs and keeping their personal data safe. Linear Regression and MultinomialNB are used as the prime methods for the classification apart from other techniques viz. Random Forest, Artificial Neural Network and Support Vector Machine. Most common machine learning algorithms require intensive training of data, causing the process to become slow in order to be executed in real time. The aim of the research is to create a model that can work in real time. The designed pipelined model using Logistic regression, achieved an accuracy of around 98%.

Keywords Logistic regression · MultinomialNB · Phishing websites · Machine learning · Classification

✉ Pranav M. Pawar
pranav@dubai.bits–pilani.ac.in

Ashish Kumar Jha
f20180173@dubai.bits-pilani.ac.in

Raja Muthalagu
raja.m@dubai.bits-pilani.ac.in

[1] Department of Computer Science, Birla Institute of Technology and Science Pilani, Dubai Campus, Dubai, UAE

1 Introduction

With the advancement in the science and technology, there is hardly any field that has remained outside this boundary of development. One of the most rapidly growing fields has been the internet. The internet is a hub of all the information and data in the world. The previous decade has brought around a drastic change in the way data is handled. All the information available is now spread over varying different channels. This complexity of data handling leaves most users unaware of the management of their personal and sensitive data. To battle this confusion regarding the safeguarding of a user's data, the European community came up with the General Data Protection Regulation (GDPR). These regulations make it mandatory for institutions and organizations to abide by the rules [5]. The main idea behind these GDPR rules being the anonymity of the data and the confidentiality of the data. Domenico Desiato et al. provided the basis for solving the above-mentioned problems and provided with the methodology for the processing of the GDPR complaint data [8]. With so much information, integrity and reliability on the internet, the onus to safeguard and secure a user's best interest is the need of the hour. Cyber security can be defined as the practice of securing or defending mobile devices, networks, computers, servers and other electronic systems and their corresponding data from attacks with malicious intent. One such attack is Phishing. Phishing is a social engineering attack in which a user gives away his/her personal information such as bank details, login credentials or card details etc. unknowingly to a hacker who uses this information for fulfilling his/her purpose. Various organizations spend huge amount of money every year to safeguard themselves and their information and data from the cybersecurity attacks by hackers. According to the statistics collected by various organizations including the Anti-Phishing Working Group and Kaspersky Lab, there has been a rapid growth in the number of phishing attacks in the recent years. According to the FBI, the most common cybercrime for the previous year, i.e., 2020–2021, was Phishing. The number of incidents nearly doubled from 2019, where the number of cases reported were 114,702 up to a whopping 241,324 in the year [18]. According to "SonicWALL Cyber Threat Report 2019" [20], the most common method of sending phishing links was using the Microsoft office mails and PDF's, as these are trusted worldwide. The numbers clearly show the magnitude of the problem and the need for an effective and immediate solution.

Data from the Symantec's Internet Security Threat Report [7] shows that the most used lines in the BEC attacks or the business email compromise attacks are urgent, request, important followed by payment and finally attention. The phishing websites have increased rapidly in the recent years. According to the data provided by "Google Safe Browsing" [22], since the start of 2016, phishing websites have crossed malware website to be the leading type of unsafe website on the internet. The graph below (Fig. 1) shows the comparison between the number of malware sites and the phishing websites in the recent years.

According to the IBM Cost a of data breach report [10], phishing websites have cost the firms a lot of money on every record that was damaged or stolen. The cost in 2019 stood at $150, which went up to $780 million when around 5.2 million records were stolen during the recent Marriot breach [18]. Not only does the recovery cost a tremendous amount of money, the after-breach effects lead to a fall in the stock prices for the company as well. According to "Data Breach Investigations Report (DBIR), 2021" from Verizon [24], there is an average drop of 5% in the stock prices of the company following a breach.

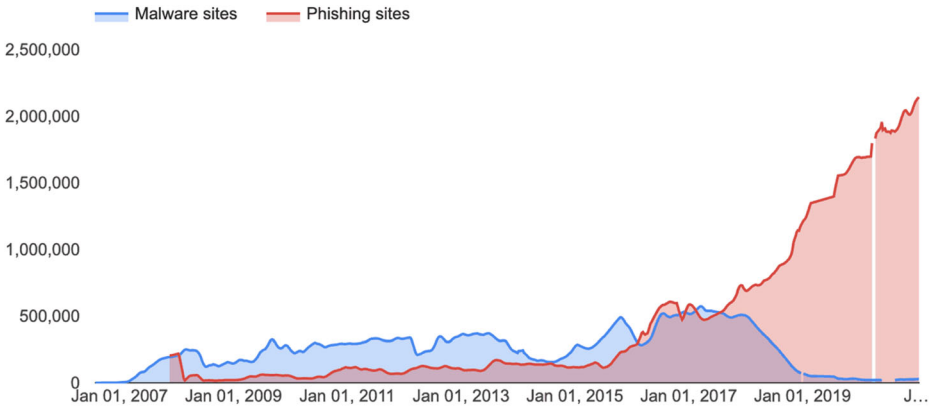


Fig. 1 Unsafe Websites Trend in recent years [22]

Phishing attacks came into picture in the early 90s, in the company AOL (American online), a company for providing online web services and web portal. Phishers created fake accounts using fake identities and fake credit cards on the site. This way, they exploited all the features without paying any money to the company and this led the company into changing its online security features and strengthening it to prevent such attacks. Attackers started asking the users for their passwords in a social engineering attack using either emails or message services. [11]

In the current scenario, the popularity of social media is increasing day by day, creating opportunities for hackers. Fake account creation to lure people into traps is a popular method being utilized in the recent times, affecting the real life, business activities and even politics. Loredana Caruccio et al. in their work on Fake account identification have discussed the impact of social media dependencies on the life of people, and provided a novel method to identify meaningful patterns to characterize and differentiate the behavior of a human to that of a bot. [6]

Machine learning can be referred to as the development of computers in a way that they are able to learn and adapt without following any explicit commands and instructions. These relate the patterns in data using mathematical models to make choices and decisions. Machine Learning can broadly be classified into two main types, viz., Supervised Learning and Unsupervised Learning. As the name suggests, supervised learning involves the algorithm to have the prior information and knowledge about a given specimen with proper label, and then it is provided with training and testing data. In unsupervised learning, labeling is not done before hand and the algorithm must find the best labels on its own.

This paper takes the approach of supervised learning, with the help of various known websites having prior classifications. Archives for malicious website that have been maintained over the years can be found on the internet which was the training data for the model. Various machine learning techniques can be used to solve the problem. However, a choice has to be made between the speed of the classifier and the accuracy of the classifier. If there is an increase in the total accuracy, the classifier will take a longer time identifying the URLs, which beats the purpose of creating a real time detection system. On the other hand, an increase in the classification speed leads to a decline in the accuracy aspect of the model. There has been

significant work in this area over the past few years and successful methods and algorithms have been studied to do the same. This model was trained using the technique of Logistic Regression, which not only is very effective in terms of accuracy but also very fast as it includes few and faster calculations. The contributions of the paper are as follows,

- Create a machine learning model using the logistic regression classifier.
- Train the model to identify and differentiate a malicious website/ URL from a safe URL.
- Create a webapp that runs on a server for identifying phishing websites based on the model trained. The webapp runs in real time with high computation speed to undergo classification while a user is surfing the internet
- Achieving a relatively higher accuracy for detection while maintaining the high computation speed

This paper has been organized further into five sections. Section 2 discusses the literature review, which gives comparative review of state-of-the-art work. Section 3 research methodology for designing intelligent phishing website detection, which includes dataset used, data preprocessing and feature extraction, web application implementation for intelligent phishing, and complexity analysis. Section 4 provides the experimentation details and results. Section 5 concludes the work with future work.

2 Literature review

To solve the above-mentioned problems, various methods have been used by different researchers and security experts. Throughout the years of research, monitoring the way phishing websites work, has provided an edge in identifying how to differentiate a normal site from a malicious one. There are different approaches from the problems defined in the previous section.

Starting off with older techniques, this section will move towards the currently used methods. Older techniques used for the identification was to compare the site URL with an existing list of sites, called the blacklist. This list contained names of all malicious sites. This list was updated frequently to contain the new sites identified as phishing. This approach was disregarded as the best approach because of the delay in updating the list. Any form of a Zero-day attack could not be prevented as the list would not have the site as soon as it is launched onto the web, unless the list was updated, and necessary changes were made [4].

Another approach was using the heuristics pattern of a given URL. The pattern was matched to the signature databases. However, using techniques like obfuscation, novel attacks could not be prevented, especially the sites that were formed recently, or the Zero-Day attack as mentioned earlier [16].

Researchers in the current time even use Amazon Alexa to predict the site URL nature. This prediction is based on the site traffic and the page rank provided by Alexa. The only downside to this method is its reliability. Page traffic cannot be used as a sure determinator for the nature of a URL. However, it can be a good quality measurement value. The value given by the Alexa page rank is used along with the other methods as described below [27].

Therefore, in order to remove all the drawbacks and shortcomings of the previous listed methods, newer techniques are being used. Overall, the different approaches to identify the URL as safe or malicious can be classified in the manner as shown in the following subsection.

These approaches are based on the factors that are used for identifying the safety and credibility of a given website URL.

2.1 Content-based approaches

The first method that is discussed in this section is the "Content-based Approach". As the name clearly suggests, in this method, the classification of a website as phishing, or non-phishing is based on the contents of the site in question. The content analysis is done based on the principle of "Term Frequency/ Inverse Document Frequency" or more commonly known as the TF-IDF algorithm for finding out phishing websites based on the page content [16, 27]. Zhang. et al. [27] used similar content-based approach in their research work, calling the novel approach CANTINA and it can be said with fair confidence based on their research that this technique gives high accuracy in finding out phishing websites (around 97%). To reduce the number of false positives, heuristic changes were applied. The false positives with pure TF-IDF were 6%, which then reduced to a mere 1% with the proper heuristics. Taking the help of heuristics, the reduction in the number of false positives is even more, however, this led the accuracy to go down to around 90%. Apart from this, even though the tool built, was effective against identifying almost all malicious websites, it had usability issues, which led users to fall victim to the malicious sites.

The advanced version of this technique, which included a desktop application for the same, was used and implemented by "Rao and Ali" [17] and they managed to reduce the false positive to 0.035% with a high accuracy of 96.57%. They used the technique based on URL using novel heuristics. The product which is called as "Phish Shield" uses null footer links along with links having the maximum frequency domains, copyrights and using the whitelists for the detection process. This process, however, had a limitation. The response time had scope to be improved based on the relatively newer methodologies such as genetic algorithms and neural networks [17].

2.2 URL-based approaches

Another approach in this domain is to detect phishing websites using the information gathered from the given site's URL. This method was suggested by Nguyen et al. [11] in their research paper. URL or Uniform Resource Locator is the information present on the address bar of a website. Initial stage is to collect the various components and then compute the metric for all the components in question. Using page rank obtained from Alexa, the next step includes picking up known information from a given data set. This can help distinguish several features in a website that can point towards the conclusion, whether the website is a phishing website or not. The accuracy for this approach is high, with around 97% of websites being detected using this technique. But as the authors Nguyen et al. mentioned in their work, there is a scope of classifying weight parameters for each of the used heuristics, and add new properties, viz., properties related to Domain Name, Geography and the WHOIS properties [28].

"Jeeva and Rajsingh" [28] in their work used the "Apriori" algorithm for creating features to be extracted from the URL that will most affect the training. Rules are generated using the association mining rule, which are then used for the prediction model. The algorithm was performed on the hidden data, which in turn would produce the accuracy for the association rules. This helped them increase the number of URL features for their work of phishing website detection. The set size for the features is however, taken as 36, which increases the

overall time complexity of the solution. Similar accuracy could have been achieved by eliminating features of less importance as done in this paper.

2.3 Machine learning approaches

Sanglerdsinlapachai and Rungsawang [19] in their work enhanced the heuristics of the 'CANTINA'. They also upgraded the blocking efficiency of the algorithm with the addition of six different machine learning techniques. This also allowed them to boost their detection rate by up to 15% in terms of the f-measure parameter. There was a boost of 20% in the error rate as well.

Xiang, Hong, Rose and Cranor in their work on the 'CANTINA+' demonstrated a layered solution. This work was an improvement from that of Zhang [27]. CANTINA+ had more features than CANTINA because of the application of machine learning techniques. The false positive rate fell down to only 0.4% and the true positive was as high as 92%.

Mamun et al. [13] used the machine learning approach for the classification of the websites using categorization. This categorization was based on the attack type of the malicious URL. Using 'K-nearest-neighbors', 'Random Forest' and other tree-based classifiers, they tried to classify the URLs in the following types, viz., malware, spam, phishing, and defacement. Out of the above classification algorithms, 'Random Forest' had the best performance followed by the other two that performed nearly the same.

Marchal et al. [14] in his work called the "Phish-Storm" used a system of lexical analyzers for phishing website detection. A dataset of 96,018 URLs containing both phishing and non-phishing URLs was used to train over a given set of 12 features. These features included the Alexa rank, the number of associated words, the domain status and the data that was based and acquired from that present in the URL. The accuracy achieved by this model was 94.91%. The false rate was low as well, on 1.44%. The system performed well on testing dataset as well, with an accuracy of 92.22% in detecting legitimate URL. The accuracy for phishing website detection was at 83.97%.

Hieu Nguyen and Thai Nguyen [15] did extensive evaluation on various types of classifiers that are popular among researchers in this field. They tried to evaluate the performance of these classifiers on the basis of the classification accuracy percentage in order for making a proper comparison. The various classification models that they showed in their work are as follows, J48 Dt with an accuracy of 98.5%, Random Forest with an accuracy of 98.8%, Support Vector Machine had an accuracy of 86.1%, Naïve Bayes had an accuracy of 96.9% and the Neural Network model had an accuracy of 98.4%. From the results obtained above, it is clear that for the given data set and training model, Random Forest classifier had the best results in terms of the accuracy classifier.

Random forest algorithm for classifying was evaluated again by Weedon et al. [25]. They used a lexical data set for the purpose of comparison. Comparisons were made with other popular classifiers such as J48, logistic regression and Naïve Bayes. Random Forest achieved an accuracy of 86.9% during the testing phase and also performed better than the others in terms of the number of false negatives algorithms.

Mustafa Aydin and Nazife Baykal [2] in their work, tried two popular algorithms for tracking phishing websites, viz., the Naïve Bayes classifier and the Sequential Minimal Optimization [SMO]. The SMO algorithm yielded better results than the NB classifier over a set of 8538 URLs. They used the WEKA, which is a tool for data mining and classification.

The NB classifier gave an accuracy of 88.17% with 0.093% false positives. However, the SMO had high accuracy of 94.67% with false positive rates at 0.059% only.

Amani Alswailem et al. [1] used supervised machine learning algorithm using the random forest technique to obtain accurate results. The final accuracy was 98.8% using a combination of 26 different features extracted from a given website. Random forest classifier was used to test a combination of a total of 36 features in a website and after the experiment, it was found out that not all features contribute in similar fashion to a given website classification into phishing or non-phishing. After the experiment, they sidelined 26 features that gave the highest accuracy when considered. The experiment also classified different features based on their importance in finding out if a given website is malicious or not. Alexa page rank was found to be the most important feature followed by secureLayer, numGet and then numberOfOuterSRCinScript.

Tang et al. worked on conducting a survey of the various available machine learning algorithms for the phishing website detection. They discussed the life cycle of phishing and the various related steps to provide anti-phishing techniques. They ran a quantitative comparison for various popular machine learning algorithms such as random forest, priority-based algorithms, Deep neural networks, Recurrent Neural networks among the others. They concluded by saying that most machine learning algorithms can produce accuracy scores of 95% or more. [21]

Peng Yang et. el. Did work on the Deep learning aspect for the detection of phishing website. They take the multidimensional features into consideration and perform a quick classification based on the URL. The next stage includes a combination of features from the URL, the code and text features on the webpage and then performing the classification. They used convolutional neural networks along with LSTM (long short-term memory) to achieve high accuracy scores of 98.99%. [26]

Bac et. el. Showed the flaws of regarding the understanding of the machine learning approach for identifying phishing website. In their work called the PWDGAN, they create a system to deceive a phishing website detector using the generative adversarial network (GANs). Their work showed how the popular machine learning and deep learning algorithms fail at identifying adversarial URLs and thus generating phishing URLs that are impossible to detect. [3]

Like [3], Nimisha Dey et. el. Developed Phishing emails and websites to evaluate the detection effectiveness of machine learning algorithms. Their work concluded the effectiveness of Multinomial naïve bayes for phishing email detection and that of the decision tree-based classifiers for website detection. They even achieved an accuracy of 98.06% and 95.41% respectively. [9]

The above-mentioned information has been summarized in tabular form shown below in tables (Tables 1 and 2). There are two separate tables, one showing the details of the literatures reviewed for the content-based approach and the URL approach. The 2nd table contains the details of the machine learning approach for the similar problem.

The primary problem with the state-of-the-art solutions is the fact that in order to enhance the accuracy of the classifier, the overall execution speed has increased, thereby not making it a real time feasible solution. The solution that this paper aims to achieve is a fast detection tool that can be integrated with the web browser like an extension to provide real life confidence measures. Therefore, the execution speed is a prime deciding factor in the trade off analysis. The current paper takes into consideration the benefits and learnings of all the papers and the

Table 1 Summary of content-based and URL approaches

Ref. No.	Working Mechanism	Algorithm	Type of Solution	Performance Parameters	Implementation Environment
[4]	Using a known blacklist of phishing websites and cascading style sheets to identify phishing websites to prevent zero-day attacks (2009)	TF-IDF and Bayesian Filter	Content-based	Accuracy: 87.9% False Positive: 12%	Implemented as a toolbar on web browsers
[16]	Case base reasoning phishing detection system (2013)	CBR-PDS	URL-based	Accuracy: 98.07% False Positive: 2%	Integrated Web based system
[27]	Novel approach called CANTINA which used the frequency of the different terms on the site (2007)	TF-IDF Information retrieval	Content-based	Accuracy: 95% False Positive:10%	Microsoft Internet Explorer Extension
[17]	Creation of a metric using all the features in a given URL and utilizing the help of heuristics (2015)	Multi-Layer Heuristic Model	URL-based	Accuracy: 96.5% False Positive:0.035%	Implemented a tool: PhishShield
[28]	Rules are used for classification which are generated using Association Mining Rule (2017)	Apriori Algorithm	URL-based	Accuracy: 98.8% False Positive: 0.4%	As a desktop application

decades of research in this area. Based on all the learning, juxtaposed with the objectives of this paper, the current approach is justified.

3 Intelligent phishing website detection

The section gives brief overview of proposed intelligent phishing website detection mechanism. First, it discusses about the dataset and why was it particularly chosen to train the model. Later, it discusses the distribution of the dataset and the labelling. From there, it moves down to the feature extraction and the mathematical reasons for the deciding the number of features used in the paper. It also gives details about the feature extraction process performed in proposed work and it also provide the insight about the implementation of the web application for intelligent phishing website detection.

3.1 Dataset

To train the model for the machine to identify between a phishing website and a non-malicious website, there is a need to have a strong dataset. For better training, there is a need for the dataset to have both types of site URLs. Dataset has been obtained from "PhishTank" [23]. PhishTank has a collection of various URLs, both legitimate and malicious, which can be used for training machine learning algorithm. The number of URLs will be based on the accuracy achieved after doing the first few runs of the algorithm; in case the system needs more data to be trained. While training the data, there is a need to make sure that the system is not being over fitted by excessive training. PhishTank was chosen as the source of dataset, as the model

Table 2 Summary for machine learning approaches

Ref. No.	Working Mechanism	ML Algorithm	Data Set	Accuracy Measure
[27]	Created CANTINA+ which uses two filters, 1st uses hashing to detect phishing and the 2nd is a login form filter. (2007)	Bayesian Network	8118 Phishing URLs and 4883 Legitimate URLs	True Positive: 87% False Positive: 1%
[19]	Trained and tested the dataset using various ML techniques to find out the best alternative. A total of 30 features extracted from the URL was used. (2010)	Random Forest	100,000 Legitimate URLs from Alexa and 10,460 Phishing URLs from PhishTank	Accuracy: 97.39%
[13]	Created a detector using Lexical Analysis and comparison of various ML techniques for obtaining best results. (2016)	K-nearest Neighbors, J48 and Random Forest	114,000 URLs categorized as Spam, Malware, Phishing and Defacement	Accuracy: 99% in Single Identifier and 93 to 99% in detecting attacks containing various classifiers
[14]	Created a tool called Phish Storm which uses Lexical Analysis and URL ratings to identify phishing websites in real time. (2014)	Random Forest	96,018 URLs containing both, phishing and legitimate URLs	Accuracy: 94.91 False Positive:1.44%
[15]	Worked on all popular classification algorithms to conclude the supremacy of one classifier over other. (2016)	Random Forest	4500 URLs from PhishTank	Accuracy: 98.8%
[25]	Identifying the malevolent site and then blocking the diverting site by identifying the iframe symbols. (2017)	Random Forest	Accumulated dataset from PhishTank	Accuracy: 86.9%
[2]	Feature extraction performed using data mining to train models for detection. A feature matrix containing 133 features is generated in this process. (2015)	Naïve Bayes and Sequential Minimal Optimization (SMO)	8538 URLs	Accuracy: 95.03% False Positive Rate: 0.046 Precision: 0.954
[1]	Generating a higher performance classifier using various features extracted from a website and finding out the features that affect the classification the most. (2019)	Random Forest	16,000 Legitimate URLs and 12,000 Phishing websites collected from PhishTank	Accuracy: 98.8%
[21]	Conducting a survey using the various machine learning algorithms to find out the effectiveness of those against zero-day phishing attacks (2021)	Random Forest, RNN, CNN, Deep Neural Network	Various sources, viz., Alexa, PhishTank, OpnePhish etc.	Accuracy: >95%
[26]	Quick classification based on the URL is then proceeded by combining and merging features of the webpage texts and code. (2019)	Convolutional neural network with LSTM	Crawled around 900,000 URLs using dmoztools.net	Accuracy: 98.99%

Table 2 (continued)

Ref. No.	Working Mechanism	ML Algorithm	Data Set	Accuracy Measure
[3]	Generated a tool to deceive machine learning based algorithms in identifying phishing websites to highlight the problems with the current approach (2021)	GANs	_	_
[9]	Created emails and websites with phishing capabilities to test the effectiveness of machine learning algorithms (2021)	MultinomialNB and Decision Tree	11,055 URLs (6157 Phishing sites. 4898 legitimate sites) from Kaggle	Accuracy: 98.06% (Emails) 95.41% (Websites)

required a large number of input values and the collection provided by PhishTank contains a large number of URLs.

The dataset contained 507,195 unique URLs, and these were labelled as good or bad. Good being the site URL that belonged to a genuine and safe site, where as the label bad was given to phishing websites. Out of the total data used, 72% of the data is labelled as "good" and the remaining 28% is labelled as "bad". The entire dataset was loaded into a '.csv' file using the Python Pandas library, and then randomly distributed in 70:30 train to test ratio. Data is stored in comma-separated values (csv) format. The file further contains two columns:

i. Unique URLs
ii. Prediction of the given URL (Figs. 2 and 3)

3.2 Data pre-processing and feature extraction

As discussed before, the features can be extracted in various ways, namely, the URL, page content and the Alexa page rank. Different characteristics can be collected from these three

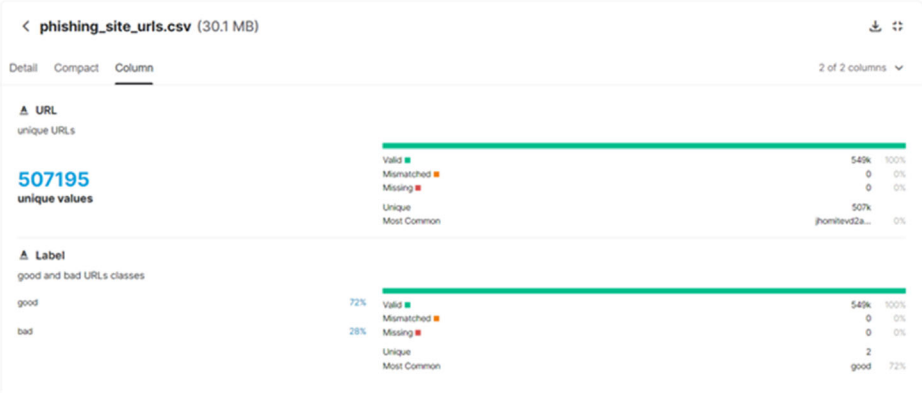
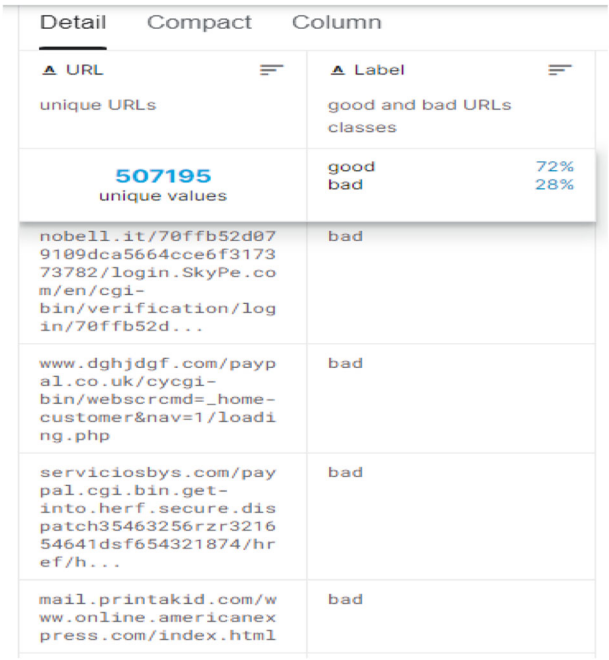


Fig. 2 Dataset distribution [23]

Fig. 3 A Sample of URLs from dataset [23]



properties. A features table has to be created that shows the various properties that can be extracted from the properties of URL, page content and Alexa page rank [15].

These features are then processed in order to obtain the most effective features out of them all. The objective is to get the best features out of all the possible features. As the research started with an aim to make this program fast, so as to make sure it runs in real time. Reduction in the number of the total features is key to increasing the programming speed in real time.

The process of feature extraction was initiated by obtaining the URL from the chosen dataset, upon which further analysis was done. The analysis part of the extraction process can be divided into 5 different processes done at once, as shown in the diagram below (Fig. 4). The first block contains all the URLs present in the considered dataset. It is then sent to the second block, i.e., the data processing block. The data processing block is the soul of the entire process. It performs five different tasks as shown (Fig. 4). The alpha Numeric Character Analysis checks the URL for the special characters or the alpha numeric character and based on the occurrence count predicts the site as being phishing or non-phishing. Phishing websites differ from normal sites on the basis of the total count and position of the alpha numeric character in their URL. The keyword analysis means analyzing the keywords that are present in the search phrases that bring the users or visitors through the search engine. So, the keyword analysis lets the algorithm find out what keywords are users searching to visit the website. Phishing websites have a large number of bogus keywords and there is no such content as promised by the keyword to lure users into the site. Security analysis of a URL refers to the process of analyzing the malicious intent of the website. This process contains checks on the reputation score of the URL and also the IP information associated with the website. Other methods to do the same include URL expander, sandboxing and proxy checking the URL for observing the URL behavior upon visit. The site SSL certificates and its certification authority

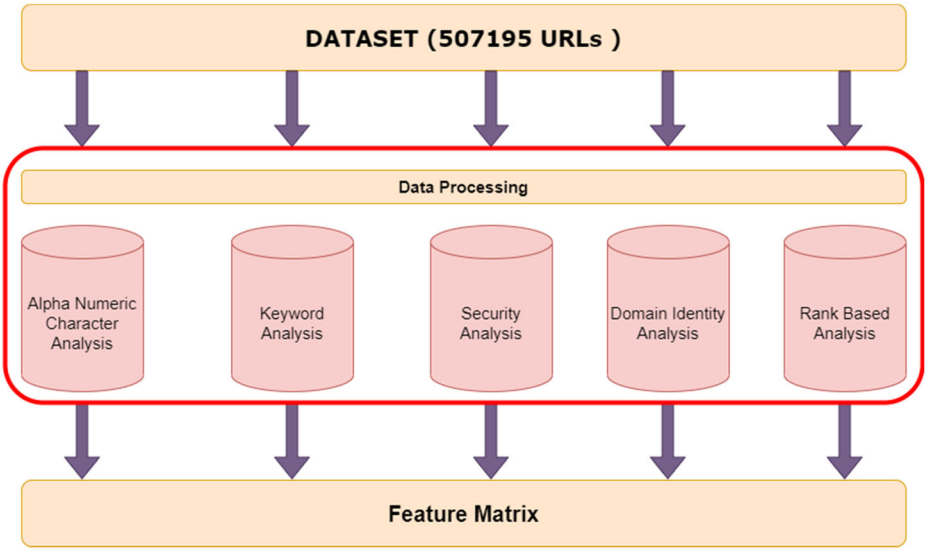


Fig. 4 Data processing

are checked as well for encryption and security purposes. Domain identity analysis is similar to the security analysis and works via IP addresses, URL of the anchor, DNS Details and checking for Strange URLs. A page rank is a term used to measure the importance of a given website in terms of its occurrence upon search. Pages that are visited frequently have a high page rank compared to dormant or unused websites. This helps us stop zero-day attacks as newly formed websites do not have a high page rank. After all the analysis, the end result is a Feature matrix containing 133 features to start with. This has to be reduced in order to obtain the necessary features required for the data training purposes.

Feature extraction is performed by running a python script that extracted all the features of the URL based off Tables 3, 4, 5 and 6. Table 3 shows the features that can be extracted using the available blacklist online. These features can then be compared to find a presence in the blacklist which can confirm the URL in question to be associated with a phishing website. Table 3 lists the features that are considered to match a given website with the pre-existing blacklisted websites. The blacklisted websites are a part of a larger ever-growing list that contains the information about websites, Ip addresses and domains that are used for malicious purposes. These lists are then used for comparison to confirm if the website that a user is visiting is a knows malicious website or is running on a known malicious domain or IP address.

Table 4 mentions about the features that can be extracted given the host of a given website. An example of this can be the check on the presence of the given domain in the RBL (Real-time Blackhole List). Another important feature that can be extracted from the knowledge of

Table 3 Features to be compared using the Blacklist [25]

BLACKLIST		
Presence of the URL in Blacklists	Presence of the IP Address in the Blacklists	Presence of the domain in the Blacklists

Table 4 Features Extracted from Website Host [25]

HOST			
Presence of the domain in RBL (Real-time Blackhole List)	Search time (response) domain (lookup)	Domain has SPF?	Geographical location of IP
AS Number (or ASN)	PTR of IP	Time (in days) of domain activation	Time (in days) of domain expiration
Number of resolved IPs	Number of resolved name servers (NameServers - NS)	Number of MX Servers	Time-to-live (TTL) value associated with hostname

host is the geographical location of a given IP which can help identify the suspicious activities running on a given website.

Table 5 shows the other important feature that need to be observed in order to classify a given website as Phishing or not. For example, Phishing websites have a large number of redirects as it lures the user using fake scenarios into URLs that can compromise the client data. Many attacks happen by creating a fake page which looks like the real page with a different URL and the users log onto their accounts thereby sharing their login credentials with the hackers.

Table 6 shows the various features that can be extracted from a site URL. Most of these features play a very vital role in the classification of site as phishing or non-phishing.

Most of the features discussed above are crucial in the process of identification, however, some features are more important than others in their effect on the classification (Table 7).

Required steps have to be followed on the acquired features which is depicted using a flow chart below (Fig. 5). Initial step is to arrange the features in order to make groupings for training, these features are then sorted in order of their importance in the URL based on the literature reviews conducted for the research. Starting off with an empty feature array and then obtaining suboptimal and then optimal dataset containing features that are of maximum importance in the process. The flowchart (Fig. 5) shown below helps the program to start from a total feature matrix of size 133 down to the features that are the most essential in determining whether a given URL is a phishing URL or not. These features are listed at Table 7 shown above.

The flowchart shown in Fig. 5 starts by taking an empty array of features, which is then populated using the features of prime importance in determining if a given website is malicious or not. All the features are then taken one by one and tried with different combinations and then trained. This gives a measure of the effectiveness of the chosen features in the feature array. This process is repeated till a final feature array is obtained with the most important features to train the final machine learning model.

As its obvious, the number of features is huge, and if an attempt is made to take each feature into account while training the model, the best results might not be obtained. Also, all these

Table 5 miscellaneousFeatures [25]

OTHERS		
Valid TLS / SSL Certificate	Check if URL is indexed on Google	Uses URL shortener service
Number of redirects	Check if domain is indexed on Google	

Table 6 Lexical Features Extracted from the site URL [25]

LEXICAL			
Count (.) in URL	Count (−) in URL	Count (_) in URL	Count (/) in URL
Count (?) in URL	Count (=) in URL	Count (@) in URL	Count (&) in URL
Count (!) in URL	Count () in URL	Count (~) in URL	Count (,) in URL
Count (+) in URL	Count (*) in URL	Count (#) in URL	Count ($) in URL
Count (%) in URL	URL Length	TLD amount in URL	Count (.) in Domain
Count (−) in Domain	Count (_) in Domain	Count (/) in Domain	Count (?) in Domain
Count (=) in Domain	Count (@) in Domain	Count (&) in Domain	Count (!) in Domain
Count () in Domain	Count (~) in Domain	Count (,) in Domain	Count (+) in Domain
Count (*) in Domain	Count (#) in Domain	Count ($) in Domain	Count (%) in Domain
Domain Length	Count the number of vowels in Domain	URL domain in IP address format	Domain contains the key words "server" or "client"
Count (.) in Directory	Count (−) in Directory	Count (_) in Directory	Count (/) in Directory
Count (?) in Directory	Count (=) in Directory	Count (@) in Directory	Count (&) in Directory
Count (!) in Directory	Count () in Directory	Count (~) in Directory	Count (,) in Directory
Count (+) in Directory	Count (*) in Directory	Count (#) in Directory	Count ($) in Directory
Count (%) in Directory	Directory Length	Count (.) in file	Count (−) in file
Count (_) in file	Count (/) in file	Count (?) in file	Count (=) in file
Count (@) in file	Count (&) in file	Count (!) in file	Count () in file
Count (~) in file	Count (,) in file	Count (+) in file	Count (*) in file
Count (#) in file	Count ($) in file	Count (%) in file	File length
Count (.) in parameters	Count (−) in parameters	Count (_) in parameters	Count (/) in parameters
Count (?) in parameters	Count (=) in parameters	Count (@) in parameters	Count (&) in parameters
Count (!) in parameters	Count () in parameters	Count (~) in parameters	Count (,) in parameters
Count (+) in parameters	Count (*) in parameters	Count (#) in parameters	Count ($) in parameters
Count (%) in parameters	Length of parameters	TLD presence in arguments	Number of parameters
Email present at URL	File extension		

futures may or may not affect the authenticity of a page. Some of the features may be of more importance than the others and it will be best to consider the features that are more important than considering all features. So, there is a need to take all these 36 features (Table 7) into account and then try to figure out the best feature group that can help in identify the authenticity of a given website in the most effective way. If all the possibilities were viewed, in which the features can be grouped, 2 or more at a time, the value given by the expression can be defined as:

$$\sum_{f=1}^{36} = \frac{a!}{f!(a-f)!} \tag{1}$$

Here, 'f' represents the number of features that have been taken into consideration at a given point of time. It starts from 1 and goes till 36, taking every feature in a unique way and then together as a group. 'a' represents the number of features, which is 36 in this case.

Table 7 Considered Features [15]

Features Based On		
URL	Length of URL	Length of hostname of URL
	Length of the path of URL	Number of dot (.) in the path
	Number of dot (.) in hostname	Number of slashes (/) in URL
	Number of hyphen (−) in hostname	Number of special char acters (:;%&?+)
	Number of at (@) in the URL	Number of digit in host name
	Number of underscore (in hostname)	Number of underscore () in path
	Number of certain key word in URL	Number of hexadecimal with %
	Transport layer security	IP address
	Presence of www	Post redirect
	Unicode in URL	Hexadecimal characters
Page Content	Number of forms	Number of forms with action 'GET'
	Number of forms with action 'POST'	Number of script
	Number of outer src script	Number of <Iframe>
	Number of <Applet>	Number of <Embed>
	Number of <F rame>	Number of link
	Number of non-link	Number of submit
	Number of input email	Number of input pass word
	Number of button	
Rank	Alexa rank	Age of domain

Thus, the feature selection process can be summarized for best results and optimizations. The detection system is designed for a webapp as shown by Fig. 6. The features are extracted from each URL provided in the training set, which is processed and fed into the machine learning classifier. This is then run through a custom python script that uses logistic regression to make the prediction as malicious and safe URL. The python script also takes the features from the website being visited by a user as an argument. Based on the values from the training set and the given website in question, a final prediction is made. This execution is carried out in the webapp. This webapp will execute and then decide based on the data training and the machine learning model to decide whether the given URL is that of a phishing website or that from a legitimate source.

Since the model is going to use MultinomialNB which runs on the principles of naïve Bayes classifier, there is a need to find the posterior probability of each class and then assign the data sample to the correct classifier based on the probability. This can be given by the equation:

$$P(C_i/x) = P\,(x/C_i)P(C_i) \tag{2}$$

$$P(x/C_i) = \prod_{k=1}^{n} P(x_k/C_i) \tag{3}$$

Here, $P(x/C_i)$ is the conditional probability.

3.3 Complexity analysis

The conclusion of this paper on using Logistic Regression as the primary classification algorithm however tends to disagree with most of the referred papers for this research (Tables 1 and 2). This calls upon the explanation as to why Logistic Regression performs better than most state-of-the-art models trained using Random Forest for similar problems.

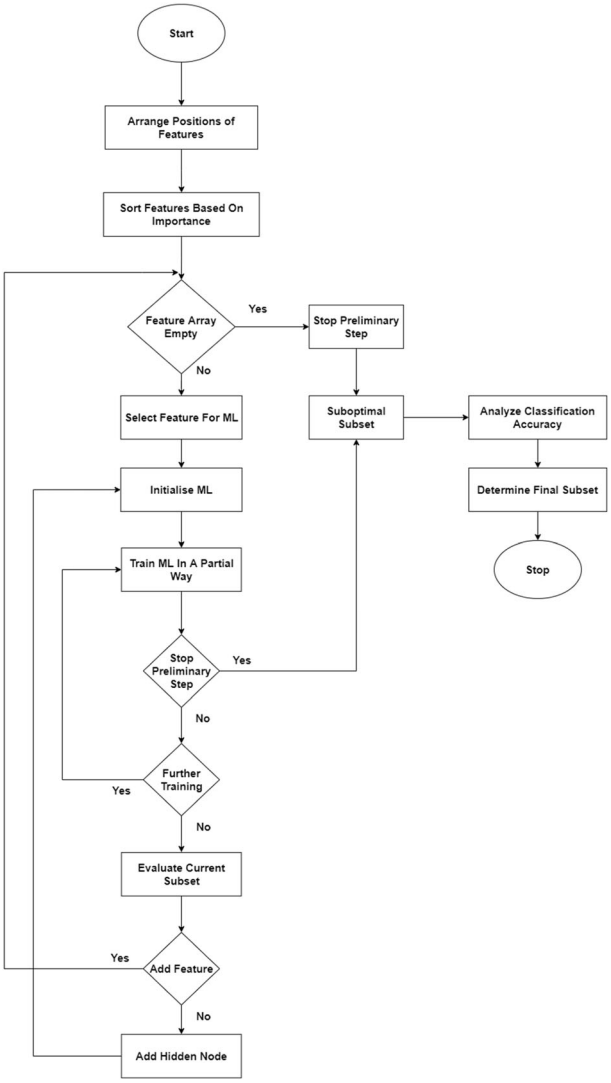


Fig. 5 Feature extraction and training of ML model

One of the main reasons for choosing Logistic Regression for the current problem was the evaluation speed. The Random Forest classifier using 'p' simultaneous parallel workers has a time complexity given by the Big Oh notation below:

$$\mathrm{O}(\text{Random Forest}) = \mathrm{O}(\text{ntrees} * \mathrm{n} * \log(\mathrm{n})/\mathrm{p}) \tag{4}$$

Comparing this to the time complexity of the logistic regression during training, which is given by the Big Oh notation below:

$$O(\textit{Logistic Regression Training Complexity}) = n(O(d)) = O(nd) \tag{5}$$

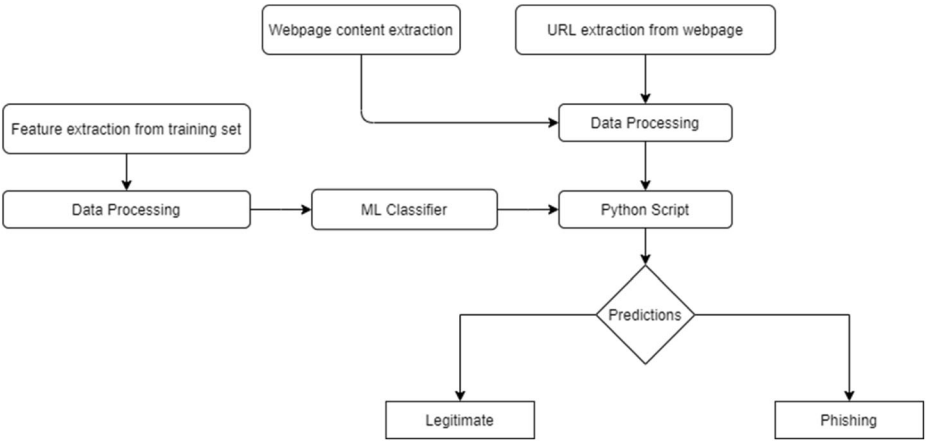


Fig. 6 Proposed intelligent phishing detection

Iterating over 'n' data points with an input feature vector of size 'd'. Whereas the runtime complexity of Logistic Regression is given by the Big Oh notation below:

$$O(Logistic\ Regression\ Runtime\ Complexity) = O(d) \tag{6}$$

Where 'd' is the size of the feature vector. Comparing these results to the other proposed training algorithm, i.e., MultinomialNB, for which the accuracy is given by given by the Big Oh notation below:

$$O(MultinomialNB) = O(Nd) \tag{7}$$

Where it needs to compute the frequency of every feature value d_i for each class. The above reasoning makes the process of training and implementation using Logistic regression and MultinomialNB much faster as compared to Random Forest. Another critical reason for the unexpectedly low performance of Random Forest is underlying in the fact that the Random Forest training algorithm was not optimized to the optimal depth. Since the focus of this research was on the speed of the process, the random forest classifier was not optimized to its fullest potential. Another major reason for pursuing the software using logistic regression over random forest was the fact that a random forest classifier performs well if the numerical features values present in the test data is within the range of training data, otherwise, it fails to classify test data if present outside the training data. On the other hand, Logistic regression performs well even in the scenario that the numerical features of the test data lie outside the training data range. This is facilitated by the fact that a logistic regression model is built on an arithmetic function. Also, a random forest classifier gives result as a binary value of 0 or 1. Logistic regression gives a probability of the observations falling on the specific categories adding extra flexibility for deciding how to classify the output by changing the threshold.

A Random Forest classifier would work better if the model would have been trained using a large number of features, however, the model used in this paper eliminated the unnecessary features and reduced the total features considered to only the relevant and important features, which cut shorts the benefits of Random Forest classifier over Logistic Regression. Logistic Regression works accurately with sparse number of input features. The logistic regression has been pipelined to increase the accuracy. The state-of-the-art works mentioned in the Literature review section work on a large number of features to make the predictions and once the model

has been trained it cannot be updated as the training time for a deep Random Forest is very high and keeps on increasing with the tree depth. Therefore, the more features that are considered, the more training time it takes for the model. The model discussed in this research is a self-learning model that can learn from the everyday use of the practitioner, or the user, the proposed approach differs from the state-of-the-art approaches.

3.4 Webapp implementation

As discussed in the following section, the method of "Logistic Regression" is used to build the prediction app. The app runs on the local host server and then identifies website as safe or malicious based on the features that it has been trained on. Using the command prompt, the code can be executed and once the code is up and running, user can now use the server to get the web app working as well. Application uses the local host as the host system, therefore providing the IP as "127.0.0.1" and then giving the port "8000" for the app to run on. User can now go to a web browser and run the web app on it.

This is still the beta version of the actual application and needs more training and improved feature extraction techniques. Also, the scope of improvement comes in the outlook of the final product and it can be made more user friendly in the next phase of development. Now, to show how the app actually runs, it is fed with a normal URL, for the sake of example, input here is taken as "Google.com". And to show the work of the app for identifying malicious sites, a malicious URL is used as well.

As the input was a valid URL, the app did not flag it as a malicious website. Now, in order to try and see a malicious website, the URL "gaup. {BLOCKED}of.com" is used, which is a blocked site as it is a malicious software which has been identified by various sources so far and is placed in the blacklist of known malicious sites. This process has been represented in the form of a control flow diagram (Fig. 7). The first step is to start the local server to test the website URL. The webapp is then launched on the server and the website URL is fed into it. The webapp then makes a prediction. As it can be seen (Fig. 7), the app flags the malicious site URL as a phishing website and a non-malicious one as a safe site.

4 Experimentation and results

Before heading into the experimentation, it is crucial to know the system on which the entire experimentation has been carried out on the chosen dataset. This helps to understand the extent to which the entire algorithmic experiment was carried out and also gives a clue about the entire training time. The system used in the current scenario is a personal computer (Laptop), the exact specifications of which are as follows:

- Dell G5
- Processor: Intel Core i7 – 8750H
- RAM: 16GB
- Dedicated Graphics: 4 GB of Nvidia GeForce GTX 1050Ti

The problem here can be looked as a classification problem or a clustering problem. Just like the process to classify unknown elements into given groups is a classification problem, similarly, this problem requires that the model must classify a given site as malicious or

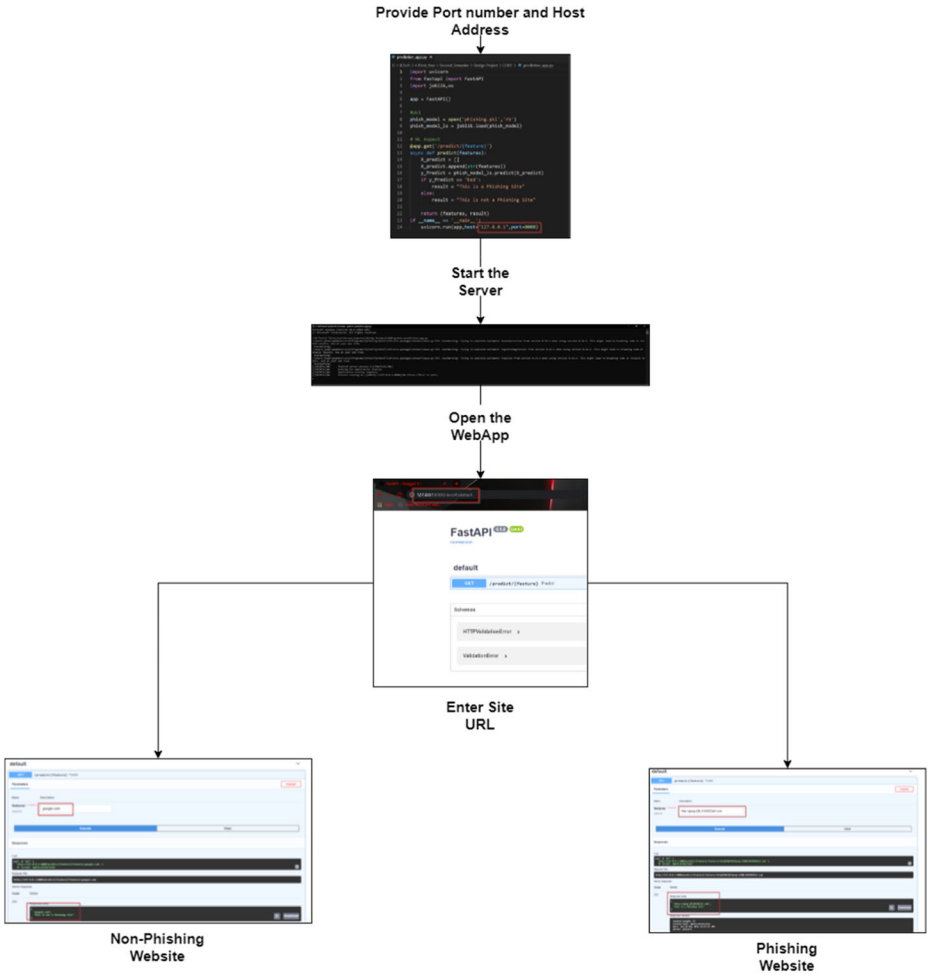


Fig. 7 Control flow graph of web application

legitimate. For the process of classification, various algorithms have been used viz. Artificial Neural Networks, Random Forest, Support Vector Machine (SVM), Logistic Regression and MultinomialNB.

Logistic Regression is one of the most popular classification algorithms. It is a statistical model in its core. It uses the logistic function for modelling the given variables. Another very popular training algorithm is the MultinomialNB. MultinomialNB or the Multinomial Naïve Bayes classifier is a very good classification algorithm for problems that have discrete features. These discrete features can include word counts, or for the process of text classification. Integral or discrete counts for features works the best with this model, however, if the results are fractional values, TF-IDF is more suitable. One of the most popular and most powerful training algorithms is the Random Forest classification algorithm. Random Decision Forest, which is generally referred to as RF or only Random Forest falls in the category of ensemble learning method for classification. Apart from classification, it can be used for regression and

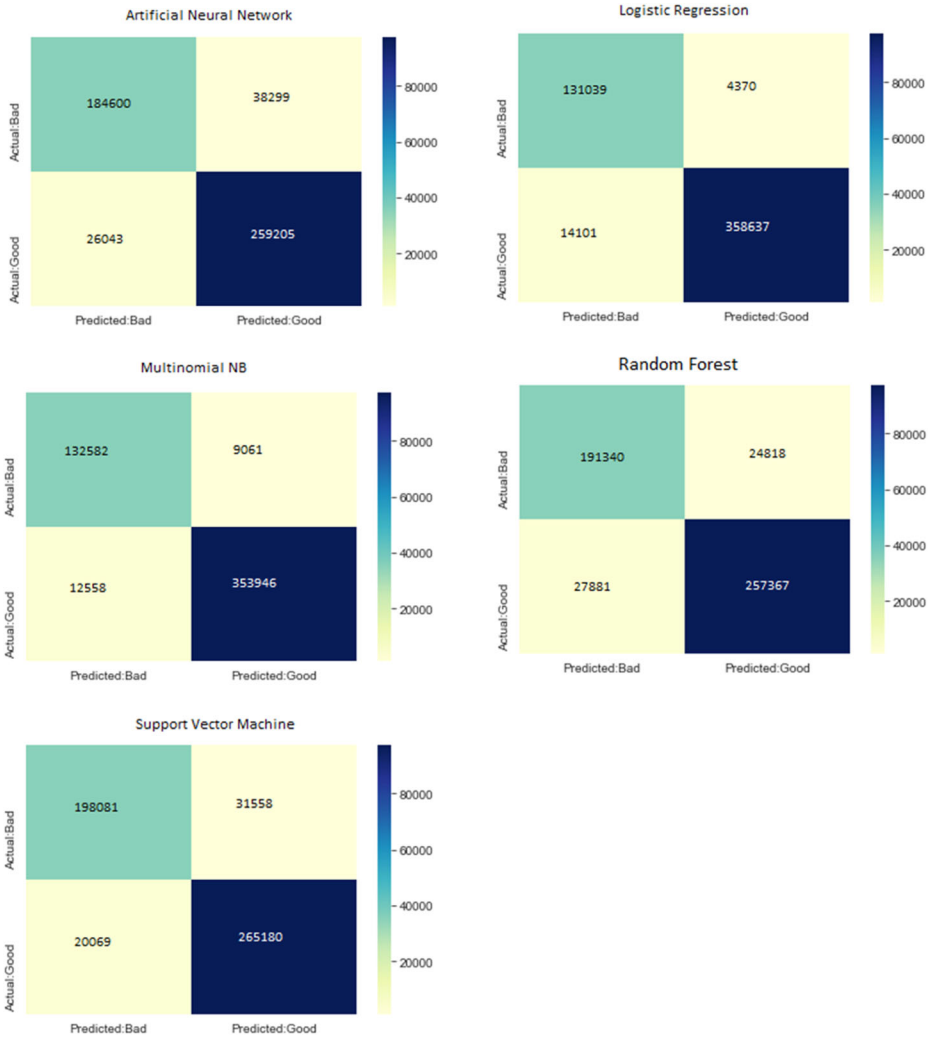


Fig. 8 Combined Confusion matrix for first split

other tasks that might require the need for a large number of decision trees all at once at a given point of time.

The experiment was divided into two different parts. The first part is a comparison between all the classifiers that were taken into consideration. After obtaining the results, the best two algorithms out of the entire list were chosen for the second part of the experimentation. The chosen algorithms were compared once again to obtain the best of the two and the chosen algorithm was then pipelined to enhance the results.

4.1 Experimentation part 1

A total of around 500,000 URLs was used on the following links, viz., Random Forest, ANN classifier, Support Vector Machine, Logistic Regression and MultinomialNB. Out of the total

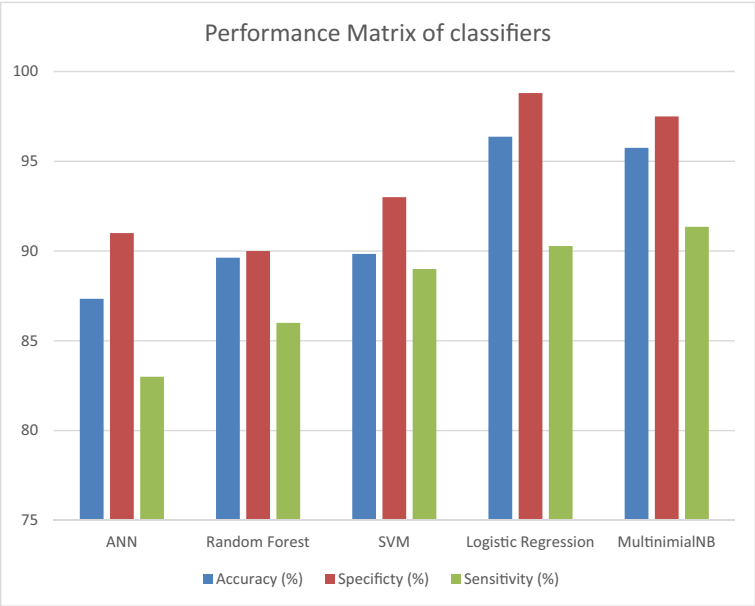


Fig. 9 Performance matrix chart

data used, 72% of the data is labelled as "good" and the remaining 28% is labelled as "bad". This section compares the performance of all the above-mentioned classifiers on the given datasets and contains tabulated results and their respective graphs.

The SVM algorithm [12] implemented here used the polynomial kernel given in (8):

$$K\left(\mathrm{x_i}, \mathrm{x_j}\right) = \gamma\left(\boldsymbol{x}_\mathrm{i}^\mathrm{T}\mathrm{x_j} + \mathrm{r}\right)^\mathrm{d}, \gamma > 0\Big) \tag{8}$$

To compare these evaluations, the accuracy is calculated for the various classifiers. Along with that other performance parameters, viz. sensitivity and specificity are used. To calculate the accuracy, sensitivity, and specificity the following formulas are used.

Accuracy is used to find the total percentage of the predictions that are correct for the data set. It is given by the formula:

Table 8 Performance matrix of classifiers

	Accuracy (%)	Specificity (%)	Sensitivity (%)	Time Complexity	Space Complexity
ANN	87.34	91	83	$O(n^4)$ (forward) $O(n^5)$ (backward)	O(2n+1)
Random Forest	89.63	90	86	O(n*log(n)*d*k)	O(depth of tree *k)
SVM	89.84	93	89	$O(n^2)$	O(n*d)
Logistic Regression	96.37	98.8	90.28	O(n*d)	O(d)
MultinomialNB	95.75	97.5	91.35	O(n*d)	O(total classes * d)

* n = number of samples used for training

* d = dimensions of the taken data

Table 9 Logistic regression vs MultinomialNB classification Report

	Logistic Regression				MultinomialNB			
	Precision	Recall	F1-score	Support	Precision	Recall	F1-score	Support
Bad	0.9	0.97	0.93	36,597	0.91	0.94	0.92	38,282
Good	0.96	0.96	0.97	100,740	0.95	0.97	0.97	99,055
Accuracy			0.96	137,337			0.96	137,337
Macro Avg	0.95	0.96	0.95	137,337	0.94	0.95	0.95	137,337
Weighted avg	0.97	0.96	0.96	137,337	0.96	0.96	0.96	137,337

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

Sensitivity, also called as Recall of a data, is the measure of the proportion of true positive cases actually predicted against all positive cases. This can be given by the formula:

$$Sensitivity = \frac{TP}{TP + FN} \tag{10}$$

Specificity can be defined as the ratio of actual negative cases, which the classifier predicted as negative. This can be defined as:

$$Specificity = \frac{TN}{TN + FP} \tag{11}$$

From the observations made above (Figs. 8, 9, Table 8), it can be concluded for the data set that Logistic regression and MultinomialNB outperformed all other classifiers.

4.2 Second Split

The research proceeds with MultinomialNB and Logistic Regression for further analysis as MultinomialNB outperformed Logistic Regression while calculating sensitivity as can be seen above (Fig. 9). Since Logistic regression and MultinomialNB have been used, tests were run on a set of 137,337 unique URLs using the above-mentioned classifiers and the results were noted down. Out of the total data used, 72% of the data is labelled as "good" and the remaining 28% is labelled as "bad".

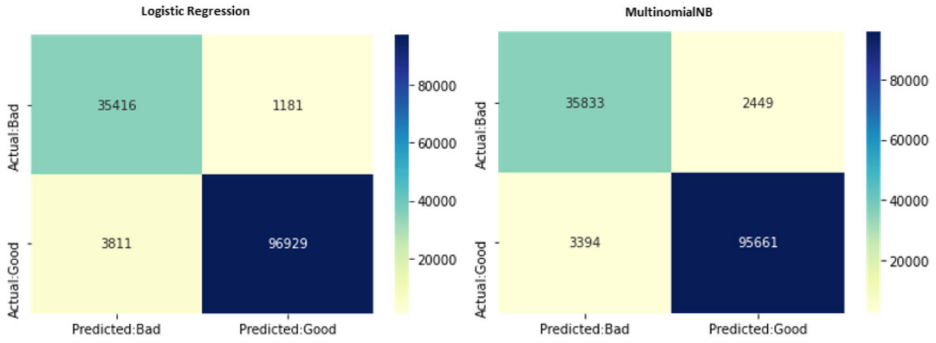


Fig. 10 Logistic regression and MultinomialNB confusion matrices

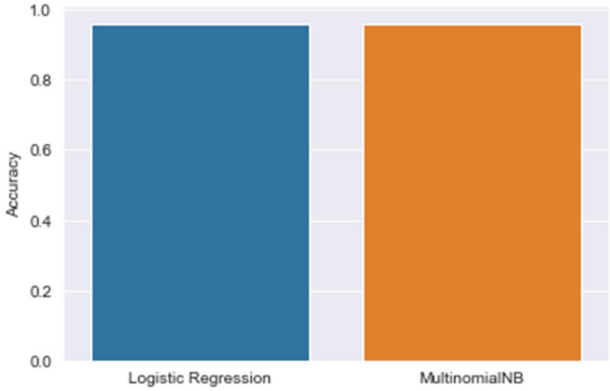


Fig. 11 accuracy comparison

The Table 9 shows the classification report of the problem, using the technique of "Logistic Regression".

Precision here means the fraction or the ratio that is obtained from dividing the correct predictions to the total predictions made. So, a precision of 0.9 signifies that 90% of the data found was correctly identified by the model.

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

Recall here signifies the fraction or ratio of the correct predictions made divided by the total positive obtained from the model.

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

F1-scores are a way of combining the recall with the precision obtained on the model. It is the harmonic mean of the recall and precision for the data set. F1-scores are calculated using the formula:

$$F1{-}scores = 2.\frac{precision.recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{14}$$

Where, TP signifies True Positives, FP signifies False Positives and FN signifies False Negatives. The physical relevance of F1-score is the measure of the accuracy of the model trained on a given dataset.

Table 10 Classification Model for pipelined Logistic Regression

	Precision	Recall	F1-Score	Support
Bad	0.91	0.94	0.92	36,841
Good	0.98	0.97	0.97	100,496
Accuracy			0.98	137,337
Macro Avg	0.96	0.95	0.96	137,337
Weighted Avg	0.97	0.96	0.96	137,337

This is a confusion matrix which is obtained from the classification report of the "Logistic Regression" model (Fig. 10). This model performs with high number of true positives and False Negatives signifying that the model worked correctly for most of the values in the data set. These values noted above are for the model trained under "Logistic Regression". Tests were run using MultinomialNB for the same data set in order to obtain competitive results. Table 9 also shows the classification report for the model trained under MultinomialNB model. The confusion matrix for the same is given in Fig. 10. This model gave an average accuracy of around 95% during the analysis of the result. So, when plotted against each other as shown (Fig. 11):

It can be seen that the Logistic Regression outperforms the MultinomialNB by a small margin (Fig. 11). So, it can be concluded that the Logistic Regression is the best fit model for the given case. So next series of events is creation of a "sklearn" pipeline for the model using "Logistic Regression" as it outperformed every other classifier in terms of accuracy, specificity, and sensitivity. Naïve Bayes classifier came close in terms of most performance matrix but was outperformed by logistic regression in terms of accuracy, recall, precision, and F1-score.

This was the classification model for the model trained according to the pipeline as defined above (Table 10). Figure 12 shows confusion matrix thus obtained. As it is very clear from the matrix, the model was highly accurate in predicting the sites correctly into the groups of good and bad. With this simple, yet effective method, the accuracy was increased to around 98%.

For a clear picture of the performance for both the algorithms, viz. Logistic regression and MultinomialNB in comparison to the final pipelined model, a comparison between Tables 9 and 10 can be made. Figures 10 and 11 can be used for the comparison of confusion matrices as well. Thus, the model obtained high accuracy along with fast classification speed to create the final web app for detection of phishing website.

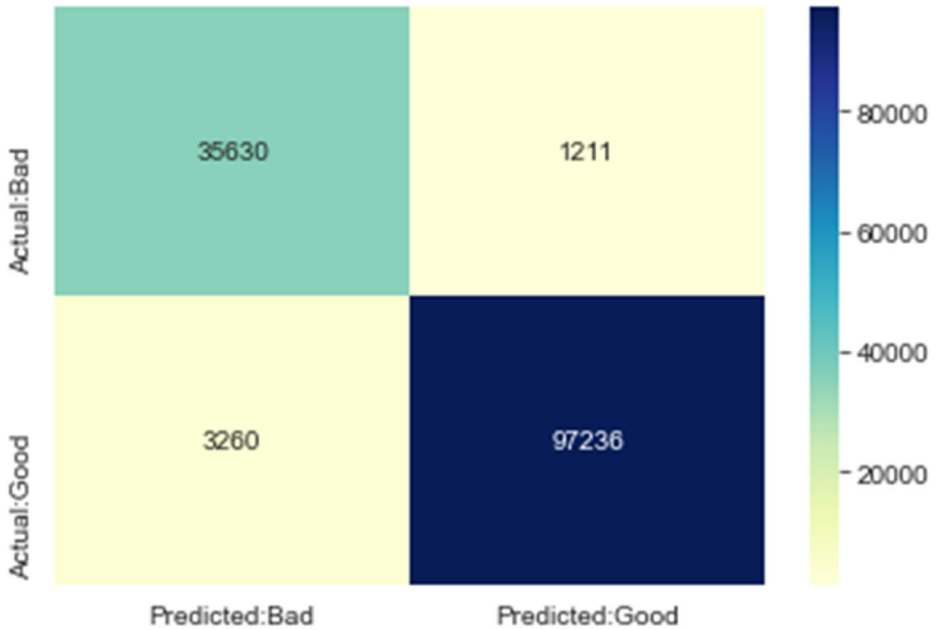


Fig. 12 Final confusion matrix

5 Conclusion and future work

With the pipelined training model using logistic regression, the model was able to achieve high accuracy up to 98%. Even though Logistic regression is a very fast classifier, it has its own downsides. It is easier to implement a very efficient training algorithm as shown earlier, however, logistic regression may lead to overfitting if the number of features exceeds the total number of observations. It assumes a linear trend between the dependent and the independent variables, which may not happen in various real-life scenarios. The fast rate of classification of unknown records is what made the choice of this classifier possible in the first place. While running the application mode of the app, there are a lot of areas of improvement. The app can be further made into a website extension for the ease of use which can be directly downloaded from the chrome store and used on the device and it will flag the URLs automatically to save time and give the best results. Further, the extension can be programmed to train online with the data received during everyday search of the user. This data can train the classifier while the user works and keeps the system up to date for any new type of phishing website. To train a model effectively there is a need for powerful computers. More time for training can improve the accuracy. With the help of these, there is a possibility to train the data in lesser time as compared to the used setup, but also, better results can be obtained by training more and more data using parallel computing. In future, the work will concentrate on application of deep learning and reinforcement learning for web-phishing detection.

Code availability Not applicable.

Funding Not applicable.

Data availability The dataset generated during and/or analyze during the current study are available from the corresponding author.

Declarations

Conflicts of interest/Competing interests Not applicable.

References

1. Alswailem A, Alabdullah B, Alrumayh N, Alsedrani A (2019) Detecting Phishing Websites Using Machine Learning, 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), pp. 1–6, https://doi.org/10.1109/CAIS.2019.8769571
2. Aydin M, Baykal N (2015) Feature extraction and classification phishing websites based on URL, 2015 IEEE Conference on Communications and Network Security (CNS), pp. 769–770, https://doi.org/10.1109/CNS.2015.7346927
3. Bac TN, Duy PT, Pham VH (2021) PWDGAN: Generating Adversarial Malicious URL Examples for Deceiving Black-Box Phishing Website Detector using GANs. In: 2021 IEEE International Conference on Machine Learning and Applied Network Technologies (ICMLANT), IEEE, 2021, pp. 1–4
4. Blasi M (2009) Techniques for detecting zero-day phishing websites. Master of Science Thesis, Iowa State University, Ames
5. Breve B, Caruccio L, Cirillo S, Desiato D, Deufemia V, Polese G (2020) Enhancing user awareness during internet browsing, In ITASEC, pp. 71–81
6. Caruccio L, Desiato D, Polese G (2018) Fake account identification in social networks. In: 2018 IEEE international conference on big data (big data), IEEE, pp. 5078–5085

7. Davis DB (2021) ISTR 2019: internet of things cyber-attacks grow more diverse. Symantec Enterprise Blogs-Expert Perspectives. https://symantec-enterprise-blogs.security.com/blogs/expert-perspectives/istr-2019-internet-things-cyber-attacks-growmore-diverse. Accessed 26 July 2021
8. Desiato D (2018) A Methodology for GDPR Compliant Data Processing. In SEBD
9. Dey N, Samhitha S, Hariprasad M, Anand A, Gadad V (2021) Analysis of machine learning algorithms by developing a phishing email and website detection model. In: IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), Bangalore, India, pp 1–7. https://doi.org/10.1109/CSITSS54238.2021.9683131
10. Ibm.com. (2021) [online] Available at: <https://www.ibm.com/downloads/cas/QMXVZX6R>. Accessed 26 July 2021
11. Jakobsson E, Myers E (2006) Phishing and Counter-Measures: Understanding the Increasing Problem of Electronic Identity Theft. Wiley, pp 2–3
12. Karnik R, Bhandari GM (2016) Support vector machine based malware and phishing website detection. IJCAT-International J Comput Technol 3(5):295–300
13. Mamun MSI, Rathore MA, Lashkari AH, Stakhanova N, Ghorbani AA (2016) Detecting malicious URLs using lexical analysis. In: Chen J, Piuri V, Su C, Yung M (eds) Network and system security: 10th international conference, NSS 2016, Taipei, Taiwan, September 28–30, 2016, proceedings. Springer International Publishing, Cham, pp 467–482
14. Marchal S, Franois J, State R, Engel T (2014) PhishStorm: detecting phishing with streaming analytics. IEEE Trans Netw Serv Manag 11(4):458–471
15. Nguyen HH, Nguyen DT (2016) Machine learning based phishing web sites detection. In: Duy VH, Dao TT, Zelinka I, Choi H-S, Chadli M (eds) AETA 2015: recent advances in electrical engineering and related sciences. Springer International Publishing, Cham, pp 123–131
16. Nguyen LAT, To BL, Nguyen HK, Nguyen MH (2013) Detecting phishing web sites: A heuristic URL-based approach, In: 2013 International Conference on Advanced Technologies for Communications (ATC 2013), pp. 597–602
17. Rao RS, Ali ST (2015) PhishShield: A Desktop Application to Detect Phishing Webpages through Heuristic Approach. Procedia Comput Sci 54(Supplement C):147–156
18. Rosenthal M (2021) Phishing statistics (updated 2021) - 50+ important phishing stats - Tessian. [online] Tessian. Available at: <https://www.tessian.com/blog/phishing-statistics-2020/>. Accessed 26 July 2021
19. Sanglerdsinlapachai N, Rungsawang A (2010) Web phishing detection using classifier ensemble, New York, NY, USA, pp. 210–215
20. Sonicwall.com. (2021) [online] Available at: <https://www.sonicwall.com/medialibrary/en/white-paper/2019-sonicwall-cyber-threat-report.pdf>. Accessed 26 July 2021
21. Tang L, Mahmoud QH (2021) A survey of machine learning-based solutions for phishing website detection. Mach Learn Knowl Extr 3(3):672–694
22. Transparencyreport.google.com. (2021) Google Transparency Report. [online] Available at: <https://transparencyreport.google.com/safe-browsing/overview?unsafe=dataset:1;series:malware,phishing;start:1579219200000;end:1611791999999&lu=unsafe>. Accessed 26 July 2021
23. URL Feature Extractor (n.d.), https://github.com/lucasayres/url-feature-extractor. Accessed 26 July 2021
24. Verizon Enterprise Solutions. (2021) 2021 Data Breach Investigations Report (DBIR). [online] Available at: <https://enterprise.verizon.com/resources/reports/2021/2021-data-breach-investigations-report.pdf>. Accessed 26 July 2021
25. Weedon M, Tsaptsinos D, Denholm-Price J (2017) Random Forest explorations for URL classification. In: 2017 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (Cyber SA), pp. 1–4
26. Yang P, Zhao G, Zeng P (2019) Phishing website detection based on multidimensional features driven by deep learning. IEEE Access 7:15196–15209
27. Zhang Y, Hong JI, Cranor LF (2007) Cantina: a content-based approach to detecting phishing websites. In: Proceedings of the 16th international conference on World Wide Web, WWW' 07, New York, pp 639–648. https://doi.org/10.1145/1242572.1242659
28. Zhang Z, He Q, Wang B (2017) A Novel Multi-Layer Heuristic Model for Anti-Phishing, New York, NY, USA, p. 21:1–21:6