Phishing URL Detection Using Machine Learning

Ananya G[1], B Varshitha[1], Swetha S[2]

[1]B.E. Student, Department of Information Science and Engineering,
R V College of Engineering, Bengaluru.

[2]Assistant Professor, Department of Information Science and Engineering,
R V College of Engineering, Bengaluru.

ananyag.is20@rvce.edu.in, bvarshitha.is20@rvce.edu.in, shwetha.ise@rvce.edu.in

ABSTRACT

Phishing websites are intended to deceive users into disclosing sensitive information such as credit card details, login credentials, or other personal information. These websites are typically created to resemble legitimate websites of well-known companies or organizations, and they often use social engineering tactics to lure users into providing their information. Phishing attacks can take many forms, including email scams, fake login pages, fraudulent online surveys, or even fake social media profiles. Phishing attacks can result in financial loss, identity theft, or other forms of fraud. ML algorithms can be trained to analyze the content of a website, including the text, images, and layout, to identify patterns that are typical of phishing websites. AI-based systems can use past data to identify known phishing domains and patterns, and use this data to identify new threats. The goal of this research is to train machine learning models and deep neural nets on the phishing website dataset. Website phishing and benign URLs are gathered to construct a dataset, and needed URL and website content-based attributes are retrieved from them. Each model's performance level is assessed and compared.

Keywords: Phishing websites, Phishing attacks, Machine learning, URL (Uniform Resource Locator), Decision Tree, XGBoost, Multilayer Perceptrons, Support Vector Machines, Random Forest

INTRODUCTION

Phishing websites are intended to deceive users into disclosing sensitive information such as credit card details, login credentials, or other personal information. These websites are created by cybercriminals with the intention of deceiving unsuspecting users and stealing their confidential information for malicious purposes.

Phishing websites often mimic legitimate websites, such as online banking portals, e-commerce platforms, social media platforms, or email providers. They employ various techniques to appear genuine, including using similar logos, layouts, and web addresses (URLs) to the authentic sites. This makes distinguishing between a legitimate and a phishing website challenging for users.

The primary goal of a phishing website is to lure users into providing their sensitive information willingly. This can be achieved through several means. The following are few types of phishing attacks;

- Email Phishing: Cybercriminals send false emails that look to be from a reputable source, such as a bank or a well-known online business. These emails usually include a link to a phishing website, inviting recipients to click on it and input their login credentials or other sensitive information.
- Spear Phishing: This is a type of phishing in which attackers tailor their communications in order to deceive specific persons, such as employees of a particular company or members of a specific organization. The emails and phishing websites used in spear phishing attacks are tailored to exploit the trust and familiarity of the recipients.
- Malicious Links: Cybercriminals spread malicious links through various channels, including social media platforms, instant messaging apps, or online forums. These links lead unsuspecting users to phishing websites, where their information is harvested.
- Search Engine Phishing: Attackers create fake websites optimized for search engine rankings to appear at the top of search results for certain keywords. When users click on these links, they are directed to phishing websites instead of legitimate sources.

The consequences of falling victim to a phishing website can be severe. Once the attackers obtain sensitive information, they can use it for identity theft, financial fraud, unauthorized access to accounts, or other malicious activities.

The typical way of detecting phishing websites is to update banned URLs and Internet Protocol (IP) addresses in the antivirus database, often known as the "blacklist" approach. Use inventive tactics to trick users by altering the URL to seem authentic via obfuscation and many other easy approaches such as: fast-flux, in which proxies are automatically constructed to host the web-page; algorithmic production of new URLs; and so on. The main disadvantage of this strategy is that it cannot identify zero-hour phishing attacks. *[1]*

Heuristic detection, which incorporates features present in phishing attacks and can identify zero-hour phishing crimes, however the traits are not guaranteed to always exist in such attacks and the false positive rate in detection is quite high.. *[1]*

Many security experts are now focusing on machine learning approaches to address the shortcomings of blacklist and heuristic-based methods. Machine learning technology is made up of several algorithms that use previous data to determine or predict on future data. The program will analyze multiple banned and valid URLs and their attributes to accurately detect phishing websites, including zero-hour phishing websites, using this method. *[1]*

LITERATURE REVIEW

In 2022, [6] proposes a malicious URL detection method based on a bidirectional gated recurrent unit (BiGRU) and attention mechanism. The approach relies on the BiGRU model as its foundation. To mitigate overfitting, a dropout mechanism is incorporated into the input layer. Additionally, an attention mechanism is introduced in the middle layer to enhance URL feature learning. Consequently, this gives rise to the DA-BiGRU deep learning network model. Through experimentation, it has been shown that the suggested technique outperforms in classifying malicious URLs, thus holding great importance for real-world implementations.

[7] proposes an approach to detect 0-h attacks using URL character sequence, hyperlink information, and textual content. It extracts character level TF-IDF features from HTML and plaintext, and proposes hyperlink features to determine the relationship between content and URL. The proposed approach achieved an accuracy of 96.76% with only 1.39% false-positive rate on the dataset and 98.48% with 2.09% false-positive rate on benchmark dataset.

In 2021, [13] developed the methods of defence utilizing various approaches to categorize websites. The research has introduced a system that utilizes machine learning methods to categorize websites according to their URLs. The primary focus involved employing four classifiers: decision tree, Naïve Bayesian classifier, support vector machine (SVM), and neural network. The detection of phishing URLs involves two key stages. The dataset used was obtained from the University of California, Irvine Machine Learning Repository and encompassed 1353 URLs with nine distinct features. Experimental findings indicate that the classifiers achieved notable success in effectively differentiating between authentic and fraudulent websites, attaining the highest accuracy rate of 91.5%.

In the year 2018, [2] proposes a model to detect phishing websites using the URL detection method using Random Forest algorithm. The idea in this work is to improve the efficiency by using Random forests with the help of Rstudio tool.

In 2017, [4] proposed an anti-phishing system which is based on the development of the Add-on tool for the web browser. The performance evaluation of the proposed system involved the utilization of four distinct data mining classification algorithms, namely Class Imbalance Problem (CIP), Rule-based Classifier (Sequential Covering Algorithm (SCA)), Nearest Neighbour Classification (NNC), and Bayesian Classifier (BC). The dataset used consisted of 7690 legitimate websites and 2280 phishing websites, sourced from trusted databases like APWG and PhishTank. The Bayesian classification algorithm exhibited higher accuracy levels and demonstrated a rapid response within the system.

In 2016, [5] presented a hybrid model designed to classify phishing emails utilizing machine learning algorithms. The objective was to create an ensemble model combining Bayesian networks with CART (Classification and Regression Trees) to enhance the accuracy of email classification. The model processed the email content and extracted 47 features from it. The findings demonstrated that when the Bayesian net classification model was combined with CART, it achieved the highest accuracy of 99.32%. However, it was noted that this approach sometimes produced excessively complex trees that did not effectively generalize the data, a phenomenon referred to as overfitting.

In 2016, [8] proposed a model for phishing website detection and preventing using modified SVM-PSO method. For feature extraction SVM classifier is used which can extract more feature (13 features) than the existing system (10 features) and for optimizing the feature set PSO (Particular Swarm Optimization) is used. This develops acceptably classified phishing websites and legitimate website. The experimental results comparison among hierarchical

clustering and SVM-PSO generates more value for precision and recall parameter, the work outperforms than the existing system.

In 2018, [1] proposed heuristic-based phishing detection technique that employs URL-based features. The system initially extracts distinctive features that can effectively differentiate between benign and malicious websites. These features are then utilized in conjunction with machine learning techniques to identify whether a website is phishing or legitimate. The study incorporated 10 URL-based features, although the specific number of datasets used was not mentioned. The experimental results indicated that Support Vector Machine (SVM) achieved an accuracy rate of 96% and exhibited a very low false-positive rate. The proposed model holds the potential to mitigate the harm caused by phishing attacks since it can effectively detect new and temporary phishing sites.

[9], the paper that presents a comparison of various techniques for assessing features in websites. The study employed a dataset consisting of 30 features and utilized three well-known feature selection methods. Through experimentation on real phishing data, the research successfully identified new clusters of features that, when employed collectively, proved effective in detecting phishing activities. Moreover, the study uncovered significant correlations among commonly used features. However, one limitation of this approach is the difficulty in finding a suitable formal representation, and it also struggles to handle quantitative measurements effectively.

[10] presented a wide scope and fast phishing detection system. The models are constructed using both phishing and legitimate URLs including the features which have been extracted. In this work, three classifiers are implemented using WEKA (Waikato Environment for Knowledge Analysis). To achieve higher performance, the classifiers are trained using balanced datasets. The datasets are divided into three subsets. The proposed system can be seamlessly integrated into this process to enhance real-time detection performance. The datasets are obtained from reliable sources such as Phishtank and OpenPhish. A total of 465,461 URLs were collected from Phishtank, while 4647 URLs were gathered from OpenPhish. To encompass a diverse range of benign websites, an additional 10,275 URLs were randomly collected from dmoz.org and webcrawler.com each. The best results were obtained using the J48 classifier, which achieved an accuracy rate of 93%.

In [11], a methodology is proposed that utilizes an evolving spiking neural network, which exhibits adaptability to changes in input data. The network is highly flexible and capable of easily learning from new input patterns. The architecture consists of two layers: the input layer and the evolving output layer, which adapts its behavior over time. The performance of the evolving Spiking Neural Network (eSNN) classifier heavily relies on parameter tuning. The proposed method involves tuning approximately seven parameters that significantly impact the network's performance. For experimentation, a dataset of 200 phishing websites was collected from PhishTank (www.phishtank.com), with 50% of the URLs being phishing websites and the remaining 50% being legitimate websites. The eSNN outperformed the Probabilistic Neural Network (PNN) with an accuracy of 92.5% compared to 89.5%. However, one of the major challenges of this network lies in selecting and tuning the appropriate parameters.

[12], the study aimed to compare a fuzzy-based anti-phishing system, specifically a Neuro Fuzzy-based model, with other existing anti-phishing systems. The proposed system consisted of a feed-forward network with five neuron layers. The research successfully designed a simple and efficient fuzzy-based anti-phishing website detection system. They introduced an effective non-algorithmic approach for anti-phishing. The UCI machine data was

utilized to evaluate the inference system, and satisfactory results were obtained. Fuzzy logic was successfully employed to accomplish the task of phishing detection and categorization, achieving an accuracy rate of 96%.

In 2021, [13] created defense systems that used several algorithms to classify webpages. The work has specifically created a system that employs machine learning techniques to categorise websites based on their URL. The decision tree, Nave Bayesian classifier, support vector machine (SVM), and neural network were the most often used classifiers. There are two processes in identifying phishing URLs. The dataset was obtained from the Machine Learning Repository at the University of California, Irvine. It has 9 characteristics drawn from 1353 URLs. The results of the studies demonstrate that the classifiers were successful in differentiating between legitimate and bogus websites with a 91.5% accuracy.

[14], the research focuses on constructing and implementing a deep learning-based phishing detection tool. It takes as input features the Universal Resource Locator (URL) and website content, including pictures and frame components. A mix of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) algorithms is used to build the classification model. This method has a high likelihood of identifying newly created phishing URLs without the need for manual feature engineering. The dataset utilized includes one million URLs taken from PhishTank, as well as a real website dataset gathered by crawling. There are also over 10,000 photos from reputable and fraudulent websites. The suggested model attained an amazing accuracy rate of 93.28%, according to the experimental findings. However, throughout the CNN and LSTM training process, issues such as overfitting, explosive gradients, and class imbalance were discovered.

[15] presented MFPD, a multidimensional feature phishing detection solution that uses a rapid detection algorithm based on deep learning, in 2018. By creating a threshold, this method dramatically decreases detection time. The approach was tested on a dataset including millions of phishing and legal URLs. The MFPD method consists of two phases. The initial stage is to extract character sequence characteristics from the specified URL and utilize them for quick categorization using deep learning. This phase does not need the cooperation of a third party or prior knowledge of phishing. The authors then use URL statistics characteristics, webpage code features, webpage text features, and deep learning rapid classification results to create multidimensional features in the second stage. They propose a dynamic category decision algorithm (DCDA) to speed up the detection process by modifying the output judgment criteria of the softmax classifier in the deep learning phase and defining a threshold. The researchers gathered a genuine dataset by browsing phishtank.com for 1,021,758 phishing URLs (positive samples) and dmoztools.net for 989,021 valid URLs (negative samples). The MFPD technique earned the greatest accuracy and F1 score after a series of trials on this huge dataset, with values of 99.41 and 99.0, respectively.

DATASET DETAILS

The phishing URLs were gathered via the opensource website PhishTank. To train the ML models, 10000 random phishing URLs are collected from this dataset.

The legitimate URLs are collected from the University of New Brunswick's open datasets. This dataset contains URLs that are benign, spam, phishing, malware, or defacement. The benign URL dataset is being evaluated for this project out of all of these categories. To train the ML models, 10000 random genuine URLs are obtained from this dataset.

METHODOLOGY

In the training phase of phishing website detection using machine learning, data pre-processing is conducted to remove unnecessary and complex structures and attributes. This step aims to streamline the data and focus on the relevant information. Feature extraction, which follows pre-processing, holds significant importance as it plays a crucial role in achieving higher accuracy and gaining a better understanding of the distinguishing features associated with phishing websites. This process involves identifying and extracting the key features that contribute to the detection of phishing websites. By extracting these informative features, the accuracy of the detection system improves, and researchers gain valuable insights into the characteristics that differentiate phishing websites from legitimate ones. Essentially, feature extraction serves as a vital step in the overall process, enabling a more effective and accurate detection of phishing websites using machine learning techniques.

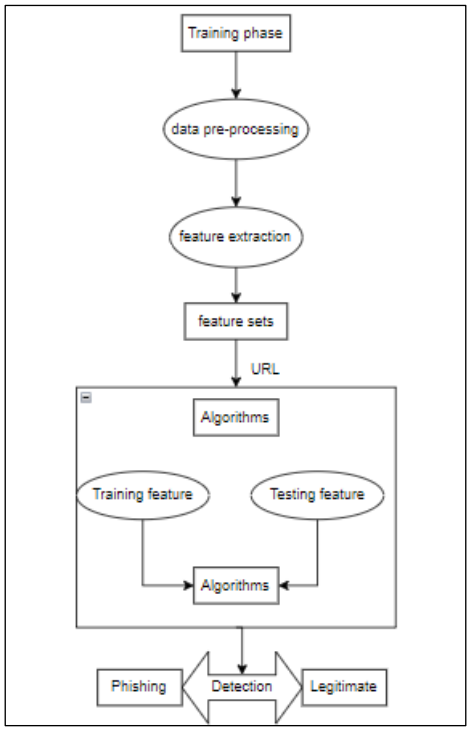

Figure 1 Proposed model

A. Feature Extraction

Feature extraction plays a crucial role in phishing website detection using machine learning. It involves transforming raw data, such as URL strings, HTML content, images, user interactions, or network traffic, into meaningful and representative features that capture the distinguishing characteristics of phishing websites. It facilitates effective model training, dimensionality reduction, interpretability, adaptability to evolving threats, and accurate detection, ultimately enhancing online security and user protection. This stage extracts features from the URL's dataset. The extracted characteristics are divided into three categories: address bar-based features, domain-based features, and HTML/JavaScript-based features.

- **Address Bar Based Features:**

  IP Address in URL: Checks the URL for the presence of an IP address. Instead of a domain name, URLs may contain an IP address. If an IP address is used as an alternative to the domain name in the URL, we can be certain that this URL is being used to steal personal information. If the domain component of the URL contains an IP address, the value assigned to this characteristic is 1 (phishing), otherwise it is 0.

```
In [0]: # 2.Checks for IP address in URL (Have_IP)
        def havingIP(url):
          try:
            ipaddress.ip_address(url)
            ip = 1
          except:
            ip = 0
          return ip
```

Figure 2 Code segment to check for IP address in URL

"@" Symbol in URL: Checks the URL for the existence of the '@' sign. When the "@" sign is used in the URL, the browser ignores anything preceding the "@" symbol, and the real address frequently follows the "@" symbol. If the URL contains the '@' symbol, the value assigned to this characteristic is 1 (phishing), otherwise it is 0.

Length of URL: The length of the URL is calculated. Long URLs can be used by phishers to mask the suspicious element in the address bar. In this project, if the URL is more than or equal to 54 characters, it is considered as phishing; else, it is valid. If the URL length is more than 54, the value assigned to this characteristic is 1 (phishing) or 0 (legal).

```
In [0]: # 4.Finding the length of URL and categorizing (URL_Length)
        def getLength(url):
          if len(url) < 54:
            length = 0
          else:
            length = 1
          return length
```

Figure 3 Code segment for the length of the URL

Depth of URL: The depth of the URL is calculated. Based on the '/', this feature determines the number of subpages in the provided URL. The feature value is a numerical number depending on the URL.

```
In [0]: # 5.Gives number of '/' in URL (URL_Depth)
        def getDepth(url):
          s = urlparse(url).path.split('/')
          depth = 0
          for j in range(len(s)):
            if len(s[j]) != 0:
              depth = depth+1
          return depth
```

Figure 4 Code segment to compute the depth of the URL

Redirection "//" in URL: The presence of "//" in the URL is checked. If there is a "//" in the URL route, the user will be forwarded to another website. The position of the "//" in the URL is calculated. We discovered that if the URL begins with "HTTP", the "//" should occur in the sixth place. If the URL includes "HTTPS," the "//" should be in the seventh place. If the "//" character appears anywhere in the URL other than after the protocol, the value attributed to this characteristic is 1 (phishing) or 0 (legal).

```
In [0]: # 6.Checking for redirection '//' in the url (Redirection)
        def redirection(url):
          pos = url.rfind('//')
          if pos > 6:
            if pos > 7:
              return 1
            else:
              return 0
          else:
            return 0
```

Figure 5 Code segment to check for redirection '//' in the URL

"http/https" in Domain name: Checks for the existence of "http/https" in the URL's domain component. In order to deceive consumers, phishers may append the "HTTPS" token to the domain component of a URL.

Using URL Shortening Services "TinyURL": URL shortening is a way on the "World Wide Web" in which a URL may be significantly reduced in length while still leading to the desired destination. This is performed by using a "HTTP Redirect" on a short domain name, which leads to a webpage with a lengthy URL. If the URL uses Shortening Services, this characteristic is assigned a value of 1 (phishing) or 0 (legal).

```
In [0]: #listing shortening services
        shortening_services = r"bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|" \
                              r"yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|" \
                              r"short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|" \
                              r"doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|db\.tt|" \
                              r"qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|q\.gs|is\.gd|" \
                              r"po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|x\.co|" \
                              r"prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|" \
                              r"tr\.im|link\.zip\.net"

In [0]: # 8. Checking for Shortening Services in URL (Tiny_URL)
        def tinyURL(url):
            match=re.search(shortening_services,url)
            if match:
                return 1
            else:
                return 0
```

Figure 6 Code segment to check for shortening services in URL

Prefix or Suffix "-" in Domain: Checking for the existence of a '-' in the URL's domain part. In genuine URLs, the dash sign is rarely used. Phishers frequently add prefixes or

suffixes separated by (-) to domain names to give consumers the impression that they are dealing with a real website. If the URL has a '-' symbol in the domain portion, the value attributed to this characteristic is 1 (phishing) or 0 (legal).

```
In [0]: # 9.Checking for Prefix or Suffix Separated by (-) in the Domain (Prefix/Suffix)
        def prefixSuffix(url):
            if '-' in urlparse(url).netloc:
                return 1            # phishing
            else:
                return 0            # legitimate
```

Figure 7 Code segment to check for the presence of '-' in the URL

- **Domain based features**

  DNS Record: In the case of phishing websites, either the stated identity is not recognized by the WHOIS database or no records for the hostname are found. If the DNS record is missing or empty, the value given to this characteristic is 1 (phishing) or 0 (legal).

  Website Traffic: This feature assesses the popularity of a website by counting the number of visitors and the pages they view. However, because phishing websites only exist for a brief time, they may be missed by the Alexa database (Alexa the Web Information Company., 1996). Examining our statistics, we discovered that in the worst-case situation, authentic websites scored among the top 100,000. Furthermore, if the domain receives no traffic or is not recognized by the Alexa database, it is labeled "Phishing." If the domain's rank is 100000, the value of this characteristic is 1 (phishing), else 0 (legal).

```
In [0]: # 12.Web traffic (Web_Traffic)
        def web_traffic(url):
          try:
            #Filling the whitespaces in the URL if any
            url = urllib.parse.quote(url)
            rank = BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url=" + url).read(), "xml").find(
                "REACH")['RANK']
            rank = int(rank)
          except TypeError:
                return 1
          if rank <100000:
            return 1
          else:
            return 0
```

Figure 8 Code segment to measure the web traffic

Age of Domain: This feature is available through the WHOIS database. Most phishing websites only exist for a short time. For this project, the valid domain must be at least 12 months old. Age in this context refers to the period between creation and expiry.

```
In [0]: # 13.Survival time of domain: The difference between termination time and creation time (Domain_Age)
        def domainAge(domain_name):
          creation_date = domain_name.creation_date
          expiration_date = domain_name.expiration_date
          if (isinstance(creation_date,str) or isinstance(expiration_date,str)):
            try:
              creation_date = datetime.strptime(creation_date,'%Y-%m-%d')
              expiration_date = datetime.strptime(expiration_date,"%Y-%m-%d")
            except:
              return 1
          if ((expiration_date is None) or (creation_date is None)):
              return 1
          elif ((type(expiration_date) is list) or (type(creation_date) is list)):
              return 1
          else:
            ageofdomain = abs((expiration_date - creation_date).days)
            if ((ageofdomain/30) < 6):
              age = 1
            else:
              age = 0
          return age
```

Figure 9 Code segment to check the age of the domain

- **HTML and JavaScript based Features**

  IFrame Redirection: IFrame is an HTML element that displays another webpage within the one that is now shown. Phishers can utilize the "iframe" tag to make the frame invisible, i.e. without frame borders. Phishers employ the "frameBorder" property to induce the browser to generate a visual distinction in this regard. If the iframe is empty or no response is detected, the value given to this feature is 1 (phishing), otherwise it is 0.

```
In [0]: # 15. IFrame Redirection (iFrame)
        def iframe(response):
          if response == "":
              return 1
          else:
              if re.findall(r"[<iframe>|<frameBorder>]", response.text):
                  return 0
              else:
                  return 1
```

Figure 10 Code segment for checking IFrame redirection

Status Bar Customization: Phishers may use JavaScript to display a bogus URL in the status bar. To extract this functionality, we must examine the webpage source code, specifically the "onMouseOver" event, and see if it modifies the status bar. If the response is empty or onMouseOver is discovered, the value given to this feature is 1 (phishing), otherwise it is 0.

```
In [0]: # 16.Checks the effect of mouse over on status bar (Mouse_Over)
        def mouseOver(response):
          if response == "" :
            return 1
          else:
            if re.findall("<script>.+onmouseover.+</script>", response.text):
              return 1
            else:
              return 0
```

Figure 11 Code segment to check the effect of mouse over on status bar

Disabling Right Click: Phishers block the right-click function with JavaScript, preventing visitors from seeing and saving the webpage source code. This functionality is precisely the same as "Using onMouseOver to hide the Link." Nonetheless, for this functionality, we will examine the webpage source code for the event "event.button==2" and see if the right click is disabled. If the answer is empty or there is no onMouseOver event, the value given to this feature is 1 (phishing) or 0 (legal).

```
In [0]: # 17.Checks the status of the right click attribute (Right_Click)
        def rightClick(response):
          if response == "":
            return 1
          else:
            if re.findall(r"event.button ?== ?2", response.text):
              return 0
            else:
              return 1
```

Figure 12 Code segment to check the status of the right click attribute

Website Forwarding: The number of times a website has been redirected is the fine line that separates phishing websites from authentic ones. In our dataset, we discovered that valid websites were only rerouted once. Phishing websites using this functionality, on the other hand, have been routed at least four times.

```
In [0]: # 18.Checks the number of forwardings (Web_Forwards)
        def forwarding(response):
          if response == "":
            return 1
          else:
            if len(response.history) <= 2:
              return 0
            else:
              return 1
```

Figure 13 Code segment to check the number of websites forwarding

B. Algorithms

The data set considered comes under classification. There are different algorithms used for classification such as Decision Tree, Multilayer Perceptrons, Random Forest, XGBoost, Support Vector Machines, LightGBM, Cat boost and Ada Boost.

Decision Tree: Decision trees are popular models for classification and regression problems. They basically learn a hierarchy of if/else questions that leads to a choice. Learning a decision tree entails learning the series of if/else questions that leads us to the correct answer as rapidly as possible. These questions are known as tests in the machine learning context . To construct a tree, the algorithm runs through all potential tests and selects the one that provides the most information about the target variable. One approach to using decision trees for phishing website detection is to extract relevant features from the URL and HTML of the website, such as the length of the URL, the presence of keywords, the number of dots, the domain age, the number of hyperlinks, the similarity with legitimate websites, and so on. Based on specific criteria, these attributes may then be used to train a decision tree classifier, which can predict whether a website is phishing or not.

Random Forest: Random forests are presently among the most popular machine learning approaches for regression and classification. A random forest is simply a collection of decision trees, each of which differs significantly from the others. The theory behind random forests is that each tree may forecast rather well, but will likely overfit on some of the data. We may limit the amount of overfitting by averaging the outcomes of numerous trees that all operate well and overfit in diverse ways. To create a random forest model, you must first

specify how many trees to create (the n_estimators option of RandomForestRegressor or RandomForestClassifier). They are very powerful, often work well without heavy tuning of the parameters, and don't require scaling of the data.

Multilayer Perceptrons: Multilayer perceptrons (MLPs) are also known as feed-forward neural networks, or simply neural networks. Multilayer perceptrons can be used for classification as well as regression issues. MLPs are extensions of linear models that go through numerous steps of processing to get a decision.

XGBoost Classifier: XGBoost is now one of the most used machine learning algorithms. XGBoost is an abbreviation for eXtreme Gradient Boosting. Regardless of whether the job at hand is regression or classification. XGBoost is a gradient-boosted decision tree solution optimized for speed and performance.

Support Vector Machines: Support-vector machines (also known as support-vector networks) are supervised learning models with related learning algorithms that examine data used for classification and regression analysis in machine learning. Given a series of training examples, each labeled as belonging to one of two categories, an SVM training method constructs a model that assigns subsequent instances to one of the two categories, resulting in a non-probabilistic binary linear classifier.

Cat Boost: CatBoost is a powerful machine learning algorithm widely used for various classification tasks, including phishing URL detection. It is specifically designed to handle categorical features efficiently and has gained popularity due to its ability to handle large-scale datasets with high-dimensional categorical variables. The CatBoost classifier employs gradient boosting techniques to train an ensemble of decision trees, combining their predictions to make accurate classifications. With its built-in features like robust handling of missing data, automatic feature scaling, and effective handling of categorical variables, CatBoost is particularly well-suited for detecting phishing URLs. It can learn complex patterns and subtle nuances in URL structures, distinguishing between legitimate and malicious URLs with high precision and recall. Additionally, CatBoost's inherent resistance to overfitting and its ability to handle imbalanced datasets make it a reliable and effective tool for phishing URL detection.

LightGBM: LightGBM is a high-performance gradient boosting framework that has proven to be highly effective for phishing URL detection tasks. It is specifically designed to handle large-scale datasets with high-dimensional features efficiently and has gained popularity in the field of machine learning. LightGBM utilizes a unique technique called "Gradient-based One-Side Sampling" (GOSS) that focuses on selecting informative instances during the training process, resulting in faster training times and reduced memory consumption. This makes LightGBM well-suited for handling the challenges of phishing URL detection, where datasets can be large and feature-rich. Additionally, LightGBM offers various optimizations, such as histogram-based feature discretization and leaf-wise tree growth, which further enhance its performance and accuracy. Its ability to handle imbalanced datasets and automatically handle missing values make it a reliable choice for detecting phishing URLs. With its speed, efficiency, and robustness, LightGBM has become a popular choice among practitioners and researchers in the field of URL-based threat detection.

Adaboost: AdaBoost works by combining weak classifiers into a strong classifier iteratively. In the context of phishing URL detection, weak classifiers could be decision stumps or simple decision trees that make predictions based on a single feature. During training, AdaBoost assigns higher weights to misclassified instances, allowing subsequent weak classifiers to focus on those instances and improve their accuracy. Through this iterative

process, AdaBoost builds a strong classifier that effectively distinguishes between legitimate and malicious URLs. AdaBoost is particularly effective in handling imbalanced datasets and can accurately classify phishing URLs with high precision. Its ability to adaptively focus on difficult instances and incorporate the knowledge of multiple weak classifiers enables AdaBoost to make robust predictions, making it a valuable tool in the field of phishing URL detection.

EXPERIMENTAL RESULTS

For the model, the dataset is split in the ratio 4:1 for training and testing respectively. The summary of machine learning models is shown below.

	ML Model	Train Accuracy	Test Accuracy
3	XGBoost	0.868	0.861
6	CB	0.866	0.860
5	LGBM	0.865	0.860
2	Multilayer Perceptrons	0.863	0.852
7	AB	0.814	0.816
1	Random Forest	0.819	0.814
0	Decision Tree	0.814	0.810
4	SVM	0.802	0.804

Figure 14 Experimental results

The previous studies in this project focused on detecting phishing websites using data mining techniques and toolbars. The approach used in this paper considered many features, resulting in higher accuracy compared to earlier methods. Notably, the use of machine learning algorithms, specifically the XGboost algorithm with gradient decision trees, played a significant role in achieving this high accuracy. The inclusion of additional techniques such as the utilization of sigmoid function and categorical dependency variables in logistic regression further contributed to the development of the best-performing model, which demonstrated superior accuracy. The increased number of iterations in the gradient decision trees of the XGboost algorithm, coupled with the incorporation of these advanced techniques, collectively contributed to the notable accuracy improvement observed in the detection of phishing websites.

CONCLUSION

In conclusion, this research paper presented a comprehensive study on the detection of phishing websites using various machine learning algorithms, including decision trees, random forest, multi-layer perceptrons (MLP), XGBoost, autoencoder, SVM, LightGBM, Cat boost and Ada Boost. Among these algorithms, XGBoost demonstrated the highest accuracy in identifying phishing websites.

The study involved extensive feature extraction and preprocessing to capture the distinguishing characteristics of phishing websites. The selected features encompassed URL properties, HTML content, user interactions, and other relevant indicators. These features were then utilized to train and evaluate the performance of the different machine learning algorithms.

The results revealed that XGBoost outperformed the other algorithms, achieving the highest accuracy in detecting phishing websites. The use of gradient decision trees with an increased number of iterations, combined with techniques such as the sigmoid function and categorical dependency variables, contributed to the superior performance of the XGBoost model. Its ability to handle both categorical and numeric features, along with its capacity for ensemble learning, contributed to its effectiveness in identifying phishing websites accurately.

The findings of this research paper highlight the importance of feature extraction, algorithm selection, and parameter tuning in achieving accurate phishing website detection. The high accuracy of XGBoost underscores its potential as a powerful tool in combating phishing attacks. However, future research can explore the combination of multiple algorithms or ensemble methods to further improve detection accuracy and robustness.

Overall, this study provides valuable insights into the use of machine learning algorithms for phishing website detection. The results underscore the significance of XGBoost as a promising approach and contribute to the advancement of the field by establishing benchmarks for accuracy in phishing detection using machine learning techniques.

FUTURE SCOPE

In terms of future scope, this research project holds substantial potential for further advancements in the field of phishing website detection. One avenue for expansion is the development of browser extensions or a graphical user interface (GUI). These extensions or GUI can utilize the trained model to classify URLs in real-time, distinguishing between legitimate websites and potential phishing threats. By integrating such tools directly into the browser, users can receive immediate warnings or alerts when visiting suspicious websites, enhancing their online security. Additionally, conducting user studies and evaluations would provide valuable insights into user feedback, satisfaction, and usability, enabling iterative improvements to the design and functionality of the extensions or GUI. To promote user education and awareness, informative features can be incorporated, offering explanations on phishing indicators and providing best practices for safe browsing. Ensuring compatibility with various platforms and browsers will further extend the reach and impact of the developed tools. Continuous model updating, privacy protection, and potential collaborations with anti-phishing tools and security software providers are also key considerations for further research, strengthening the overall security infrastructure and empowering individuals to navigate the online landscape with confidence.

REFERENCES

[1] Mahajan, Rishikesh & Siddavatam, Irfan. (2018). Phishing Website Detection using Machine Learning Algorithms. International Journal of Computer Applications. 181. 45-47. 10.5120/ijca2018918026.

[2] S. Parekh, D. Parikh, S. Kotak and S. Sankhe, "A New Method for Detection of Phishing Websites: URL Detection," 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 2018, pp. 949-952, doi: 10.1109/ICICCT.2018.8473085.

[3] Hajara, Musa. (2023). A comprehensive Review on phishing website detection using machine learning and deep learning techniques.. 8. 188-197. 10.1729/Journal.32945.

[4] Desai, Anand & Jatakia, Janvi & Naik, Rohit & Raul, Nataasha. (2017). Malicious web content detection using machine leaning. 1432-1436. 10.1109/RTEICT.2017.8256834.

[5] Thabtah, F., and Abdelhamid, N. (2016). Deriving Correlated Sets of Website Features for Phishing Detection: A Computational Intelligence Approach, 15(4), 1–17. doi:10.1142/S0219649216500428.

[6] Wu, T., Wang, M., Xi, Y., Zhao, Z. (2022): Malicious URL Detection Model Based on Bidirectional Gated Recurrent Unit and Attention Mechanism. Appl. Sci. 2022, 12, 12367. https://doi.org/10.3390/ app122312367

[7] Aljofey, A., Jiang, Q., Rasool, A., Chen, H., Liu, W., Qu, Q., & Wang, Y. (2022). An effective detection approach for phishing websites using URL and HTML features. Scientific Reports, 1–19. https://doi.org/10.1038/s41598-022-10841-5

[8] Solanki, J., and Vaishnav, R. G. (2016). Website Phishing Detection using Heuristic Based Approach, 2044–2048

[9] Suryavanshi, N., and Jain, A. (2016). Phishing Detection In Selected Feature Using ModifiedSVM-PSO, 5, 208–214.

[10] Kevric, J., Jukic, S., Subasi, A. (2017). An effective combining classifier approach using tree algorithms for network intrusion detection. Neural Computing and Applications, 28(S1): 1051-1058. https://doi.org/10.1007/s00521-016-2418-1

[11] Daeef, A. Y., Ahmad, R. B., Yacob, Y., &Phing, N. Y. (2017). Wide scope and fast websites phishing detection using URLs lexical features. 2016 3rd International Conference on Electronic Design, ICED 2016. https://doi.org/10.1109/ICED.2016.7804679

[12] Kadam, S. (2021). Feature based Phishing Website Detection using Random Forest Classifier. International Journal for Research in Applied Science and Engineering Technology, 9(VI). https://doi.org/10.22214/ijraset.2021.35400

[13] Martanto, & Anwar, Saeful & Rohmat, Cep & Basysyar, Fadhil & Wijaya, Yudhistira. (2021). Clustering of internet network usage using the K-Medoid method. IOP Conference Series: Materials Science and Engineering. 1088. 012036. 10.1088/1757-899X/1088/1/012036.

[14] M. A. Adebowale, K. T. Lwin and M. A. Hossain, "Deep Learning with Convolutional Neural Network and Long Short-Term Memory for Phishing Detection," 2019 13th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), Island of Ulkulhas, Maldives, 2019, pp. 1-8, doi: 10.1109/SKIMA47702.2019.8982427.

[15] Sahingoz, Ozgur & Baykal, Saide & Bulut, Deniz. (2018). PHISHING DETECTION FROM URLS BY USING NEURAL NETWORKS. 41-54. 10.5121/csit.2018.81705.

[16] K. Sushma, M. Jayalakshmi and T. Guha, "Deep Learning for Phishing Website Detection," 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), Mysuru, India, 2022, pp. 1-6, doi: 10.1109/MysuruCon55714.2022.9972621.

[17] M. D. Bhagwat, P. H. Patil and T. S. Vishawanath, "A Methodical Overview on Detection, Identification and Proactive Prevention of Phishing Websites," 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), Tirunelveli, India, 2021, pp. 1505-1508, doi: 10.1109/ICICV50876.2021.9388441.

[18] M. Almousa, R. Furst and M. Anwar, "Characterizing Coding Style of Phishing Websites Using Machine Learning Techniques," 2022 Fourth International Conference on Transdisciplinary AI (TransAI), Laguna Hills, CA, USA, 2022, pp. 101-105, doi: 10.1109/TransAI54797.2022.00025.

[19] S. Patil and S. Dhage, "A Methodical Overview on Phishing Detection along with an Organized Way to Construct an Anti-Phishing Framework," 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), Coimbatore, India, 2019, pp. 588-593, doi: 10.1109/ICACCS.2019.8728356.

[20] Castano, Felipe & Fernández, Eduardo & Alaiz, Rocío & Alegre, Enrique. (2023). PhiKitA: Phishing Kit Attacks Dataset for Phishing Websites Identification. IEEE Access. PP. 1-1. 10.1109/ACCESS.2023.3268027.

[21] Vyvaswini, T. & Rao, Mr & Kousalya, B. & Pallavi, G. & Abdullal, S. & Siddartha, P.. (2023). Phishing Website Detection using Machine Learning. International Journal of Advanced Research in Science, Communication and Technology. 462-470. 10.48175/IJARSCT-8318.