Predictive Modelling of Phishing Website Detection

Abhijit Das*

North Dakota State University, abhijit.das.1@ndsu.edu

Sabrina S Rifa*

North Dakota State University, sabrina.rifa@ndsu.edu

Abhirup Chanda*

North Dakota State University, abhirup.chanda@ndsu.edu

Phishing websites are a popular social engineering technique that imitates reliable addresses and online sites. The goal of this project is to extract important properties from URLs, turn those features into a dataset, then use that dataset to train machine learning models and deep neural networks to identify phishing websites. A sample of legitimate URLs is provided from the University of New Brunswick, while the data on phishing URLs is gathered from an open-source program called PhishTank. For the project, a subset of 5000 phishing URLs and 5000 legitimate URLs are chosen. Total of 17 (seventeen) features are extracted from the URLs dataset, which are categorized into address bar-based features, domain-based features, and HTML and JavaScript-based features. This dataset has been used to train various machine learning models and deep neural nets. The performance level of each model is measured and compared.

CCS CONCEPTS •**Computing methodologies → Machine learning → Machine learning approaches → Classification and regression trees • Computing methodologies → Machine learning → Machine learning approaches → Neural networks**

Additional Keywords and Phrases: phishing websites, detection, neural networks, feature extraction.

ACM Reference Format:

Abhijit Das,Sabrina S Rifa and Abhirup Chanda. 2023. Predictive Modelling of Phishing Website Detection. Project Paper of Advanced Data Mining. North Dakota State University, North Dakota.

1 INTRODUCTION

Phishing attacks are a growing concern in the world of cybersecurity, with cybercriminals using various techniques to steal sensitive information from unsuspecting victims. One of the most popular methods is to create fake websites that imitate legitimate ones. Users may find it challenging to differentiate between these phishing websites and legitimate ones due to the frequent usage of URLs that closely resemble legitimate ones.

The objective of this project is to develop a machine learning model that can accurately predict phishing websites. To achieve this, we have collected a dataset of 5000 phishing URLs from the open-source service PhishTank and 5000 reliable URLs from the University of New Brunswick. We will extract key features from these URLs, which are categorized into address bar-based features, domain-based features, and HTML and JavaScript-based features. We will then use this dataset to train various machine learning models and deep neural nets and measure their performance levels.

The results of this study can aid in the development of more effective methods to identify and prevent phishing attacks, thereby enhancing the overall security of online transactions. With the increasing use of online platforms for transactions and communication, it is essential to develop more robust security measures to protect users from cyber threats.

2 DATA COLLECTION

2.1 Phishing URLs

An open-source service called PhishTank has been used to acquire a large collection of phishing URLs for the machine learning models. This service gives users access to a substantial database of phishing URLs that is updated hourly and is available in a variety of forms, including csv and json. The information is available at https://www.phishtank.com/developer_info.php,the company's official website.

```
#loading the phishing URLs data to dataframe
data0 = pd.read_csv("online-valid.csv")
data0.head()
```

	phish_id	url
0	6557033	http://u1047531.cp.regruhosting.ru/acces-inges...
1	6557032	http://hoysalacreations.com/wp-content/plugins...
2	6557011	http://www.accsystemprblemhelp.site/checkpoint...
3	6557010	http://www.accsystemprblemhelp.site/login_atte...
4	6557009	https://firebasestorage.googleapis.com/v0/b/so...

Figure 1: Collected Phishing URLs

From this dataset, 5000 phishing URLs has been randomly selected for the purpose of this project.

2.2 Legitimate URLs

To acquire a sample collection of legitimate URLs for training the machine learning models, the free datasets offered by the University of New Brunswick have been used. This dataset, which may be obtained at

https://www.unb.ca/cic/datasets/url-2016.html, contains a wide variety of URLs, including benign, spam, phishing, malware, and defacement URLs.

```
#Loading legitimate files
data1 = pd.read_csv("1.Benign_list_big_final.csv")
data1.columns = ['URLs']
data1.head()
```

	URLs
0	http://1337x.to/torrent/1110018/Blackhat-2015-...
1	http://1337x.to/torrent/1122940/Blackhat-2015-...
2	http://1337x.to/torrent/1124395/Fast-and-Furio...
3	http://1337x.to/torrent/1145504/Avengers-Age-o...
4	http://1337x.to/torrent/1160078/Avengers-age-o...

Figure 2: Collected Legitimate URLs

Nevertheless, only the benign URL dataset was used and a randomly chosen subset of 5000 reliable URLs has been taken from this for this project.

3 METHODS

3.1 Feature Extraction

A phishing URL and the related website may be distinguished from a malicious URL by a number of characteristics. To mask the real domain name, an attacker can, for instance, register a lengthy or obscure domain. Attackers may have the option of employing direct IP addresses rather than the domain name. Attackers may also utilize brief domain names devoid of the FreeUrl addition and unrelated to recognized brand names.

In this project, Total of seventeen features were extracted from the dataset of URLs. These features are categorized into three main groups [1], namely –

- Address bar-based features,
- Domain-based features, and HTML and
- JavaScript-based features.

3.1.1Address Bar Based Features

A website's URL or address bar is the first item to be examined to determine whether it is a phishing site or not. As mentioned earlier, phishing domain URLs have several distinguishing characteristics. When the URL is parsed, features linked to these points are acquired.

There are several features that may be extracted and considered address bar based features. The following were chosen from among them for this project.

Domain of URL

For this feature the domain component present in the URL were extracted.

IP Address in the URL

The presence of IP addresses in the URLs was checked as a feature. If an IP address was used instead of a domain name in the URL, it was deemed a strong indicator of a phishing attempt to steal personal information.

For this feature, A value of 1 was assigned if the domain part of the URL contained an IP address, and 0 otherwise.

"@" Symbol in URL

The presence of the '@' symbol in the URLs was checked as a feature. The use of the '@' symbol in the URL can cause the browser to ignore everything preceding it, and the real address often follows the symbol.

For this feature, A value of 1 was assigned if the URL contained the '@' symbol, indicating a higher likelihood of a phishing attempt, and 0 otherwise.

Length of URL

The length of the URLs was computed as a feature. Phishers often use long URLs to hide the suspicious part in the address bar.

For this feature, a value of 1 was assigned if the length of the URL was greater than or equal to 54 characters, indicating a higher likelihood of a phishing attempt, and 0 otherwise.

Depth of URL

The depth of the URL was computed as a feature. This feature determines the number of sub-pages in the given URL based on the number of '/' characters.

The value of this feature is numerical and based on the URL.

Redirection "//" in URL

The presence of "//" in the URL is checked as a feature to detect whether the user will be redirected to another website. The location of "//" in the URL is computed, and it is determined that if the URL starts with "HTTP", "//" should appear in the sixth position, while if it employs "HTTPS", "//" should appear in the seventh position.

For this feature, If "//" is present anywhere in the URL apart from after the protocol, the feature is assigned a value of 1 (phishing), otherwise, it is assigned a value of 0 (legitimate).

"http/https" in Domain name

The presence of "http/https" in the domain part of the URL is checked as a feature. The addition of the "HTTPS" token by phishers to the domain part of a URL is aimed at deceiving users.

For this feature, a value of 1 (phishing) is assigned to the feature if the URL has "http/https" in the domain part; otherwise, a value of 0 (legitimate) is assigned.

Using URL Shortening Services “TinyURL”

The method of URL shortening is used on the World Wide Web to reduce the length of a URL while still directing users to the intended webpage. This is achieved through an HTTP Redirect on a short domain name that points to the webpage with the longer URL.

For this feature, if the URL is detected to be using Shortening Services, a value of 1 (phishing) is assigned to this feature, otherwise, a value of 0 (legitimate) is assigned.

Prefix or Suffix "-" in Domain

The presence of the '-' symbol in the domain part of a URL is checked as a feature to determine its legitimacy. The use of the dash symbol is rare in legitimate URLs, and phishers tend to add prefixes or suffixes separated by '-' to the domain name to trick users into believing that they are dealing with a legitimate webpage.

For this feature, if the URL has '-' symbol in the domain part of the URL, the feature value is assigned 1 (phishing) or else 0 (legitimate).

3.1.2Domain Based Features

The purpose of Phishing Domain Detection is detecting phishing domain names. As a result, passive inquiries about the domain name that we want to categorize as phishing or not give us relevant information. There are several features that may be extracted and considered domain based features. The following were chosen from among them for this project.

DNS Record

The presence of domain record in WHOIS database is checked as a feature. It is noted that for phishing websites, the claimed identity is either unrecognized by the WHOIS database or no records are found for the hostname.

For this feature, if the DNS record is empty or not found, a value of 1 (phishing) is assigned to this feature, otherwise, a value of 0 (legitimate) is assigned.

Web Traffic

Web Traffic feature measures the popularity of the website by determining the number of visitors and pages they visit. However, due to the short lifespan of phishing websites, they may not be identified by the Alexa database. A review of the dataset reveals that legitimate websites are ranked among the top 100,000 in the worst-case scenarios.

For this feature, if the domain has no traffic or is not recognized by the Alexa database, it is classified as "Phishing." If the domain rank is less than 100,000, the value of this feature is 1 (phishing), otherwise, it is 0 (legitimate).

Age of Domain

The age of the domain is checked as a feature. It can be extracted from the WHOIS database. Phishing websites tend to have a short lifespan. For this project, the minimum age of a legitimate domain is considered to be 12 months, which is calculated as the difference between creation and expiration time.

For this feature, if the age of the domain is greater than 12 months, the value of this feature is 1 (phishing), otherwise, it is 0 (legitimate).

End Period of Domain

The end period of the domain is checked as a feature. can also be extracted from the WHOIS database. For this feature, the remaining domain time is calculated as the difference between the expiration time and the current time. The end period considered for legitimate domains is 6 months or less for this project.

For this Feature, if the end period of the domain is greater than 6 months, the value of this feature is 1 (phishing), otherwise, it is 0 (legitimate).

IFrame Redirection

IFrame is an HTML tag used to display an additional webpage into one that is currently shown. Phishers can make use of the “iframe” tag and make it invisible by using the “frameBorder” attribute, which causes the browser to render a visual delineation.

For this feature, if the iframe is empty or the response is not found, a value of 1 (phishing) is assigned to this feature, otherwise, a value of 0 (legitimate) is assigned.

Status Bar Customization

It involves phishers using JavaScript to show a fake URL in the status bar to users. To extract this feature, the webpage source code, particularly the “onMouseOver” event, must be examined to check if it makes any changes on the status bar.

For this feature, if the response is empty or the onmouseover event is found, a value of 1 (phishing) is assigned, otherwise, a value of 0 (legitimate) is assigned.

Disabling Right Click

Phishers use JavaScript to disable the right-click function, so that users cannot view and save the webpage source code, To extract this feature, the event “event.button==2” is searched for in the webpage source code to check if the right-click is disabled.

For this feature, if the response is empty or the onmouseover event is not found, a value of 1 (phishing) is assigned, otherwise, a value of 0 (legitimate) is assigned.

Website Forwarding feature

It distinguishes phishing websites from legitimate ones based on how many times a website has been redirected. In the dataset, it is found that legitimate websites have been redirected one time at most. On the other hand, phishing websites containing this feature have been redirected at least four times.

For this feature, if the forwarding response is more than 2 times, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

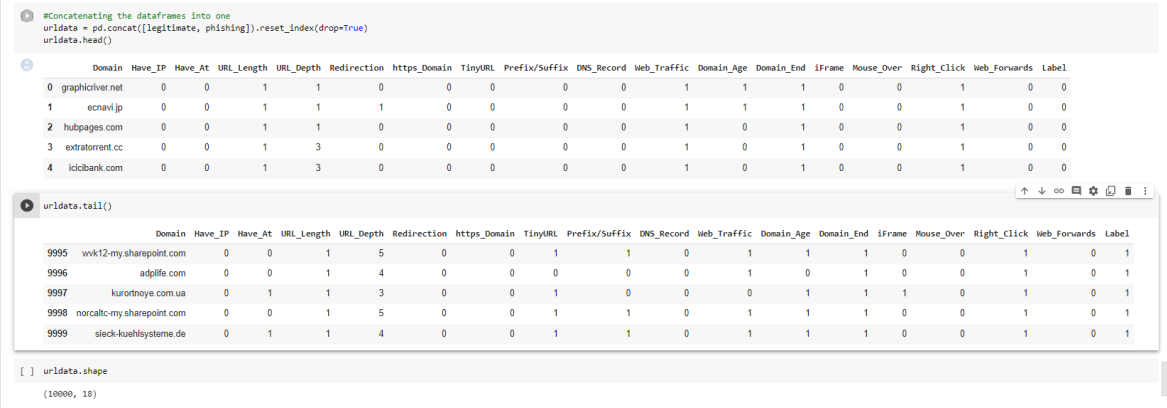
3.2 Final Dataset

3.2.1Computing URL Features

A list has been created and also a function which calls another function to store all the features of the URL in the list. The function has been executed on legitimate URLs to extract and append the features of each legitimate URL to the aforementioned list, from which a dataframe has been created. Similarly, the function has also been applied to phishing URLs, enabling the creation of a phishing dataframe.

3.2.2Combining Two Dataframes to Create the Final Dataset

In the previous section, two dataframes containing features of legitimate and phishing URLs have been created. These dataframes have been subsequently combined into a single dataframe, which has been exported to a CSV file. The final dataset has been formed by extracting 17 features for 10,000 URLs, consisting of 5,000 phishing and 5,000 legitimate URLs. Our next goal is to train models using the final dataset and compare the performance of those models to determine which one performs better.



```
#Concatenating the dataframes into one
urldata = pd.concat([legitimate, phishing]).reset_index(drop=True)
urldata.head()
```

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End	iFrame	Mouse_Over	Right_Click	Web_Forwards	Label
0	graphicriver.net	0	0	1	1	0	0	0	0	0	1	1	1	0	0	1	0	0
1	ecnavi.jp	0	0	1	1	1	0	0	0	0	1	1	1	0	0	1	0	0
2	hubpages.com	0	0	1	1	0	0	0	0	0	1	0	1	0	0	1	0	0
3	extratorrent.cc	0	0	1	3	0	0	0	0	0	1	0	1	0	0	1	0	0
4	icicibank.com	0	0	1	3	0	0	0	0	0	1	0	1	0	0	1	0	0

```
urldata.tail()
```

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End	iFrame	Mouse_Over	Right_Click	Web_Forwards	Label
9995	wvk12-my.sharepoint.com	0	0	1	5	0	0	1	1	0	1	1	1	0	0	1	0	1
9996	adplife.com	0	0	1	4	0	0	0	0	0	1	0	1	0	0	1	0	1
9997	kurortnoye.com.ua	0	1	1	3	0	0	1	0	0	0	1	1	1	0	1	0	1
9998	norcaltc-my.sharepoint.com	0	0	1	5	0	0	1	1	0	1	1	1	0	0	1	0	1
9999	sieck-kuehlsysteme.de	0	1	1	4	0	0	1	1	0	1	1	1	0	0	1	0	1

```
[ ] urldata.shape
(10000, 18)
```

Figure 3: Final Dataset

3.3 Data Visualization

3.3.1Feature Distribution

A histogram plot is created for each numerical feature (i.e., age, BMI, children, charges) in the dataset. Each histogram shows the distribution of the corresponding feature values in the dataset. The x-axis represents the range of feature values, and the y-axis represents the frequency or count of the feature values in that range. The height of each bar represents the frequency or count of feature values in that range.

The feature distribution of our final dataset is illustrated in Figure 3. It is evident from the figure that each feature exhibits significant bias towards either 0 (Legitimate) or 1 (Phishing) except for the "Domain Age" feature, which shows no such bias.

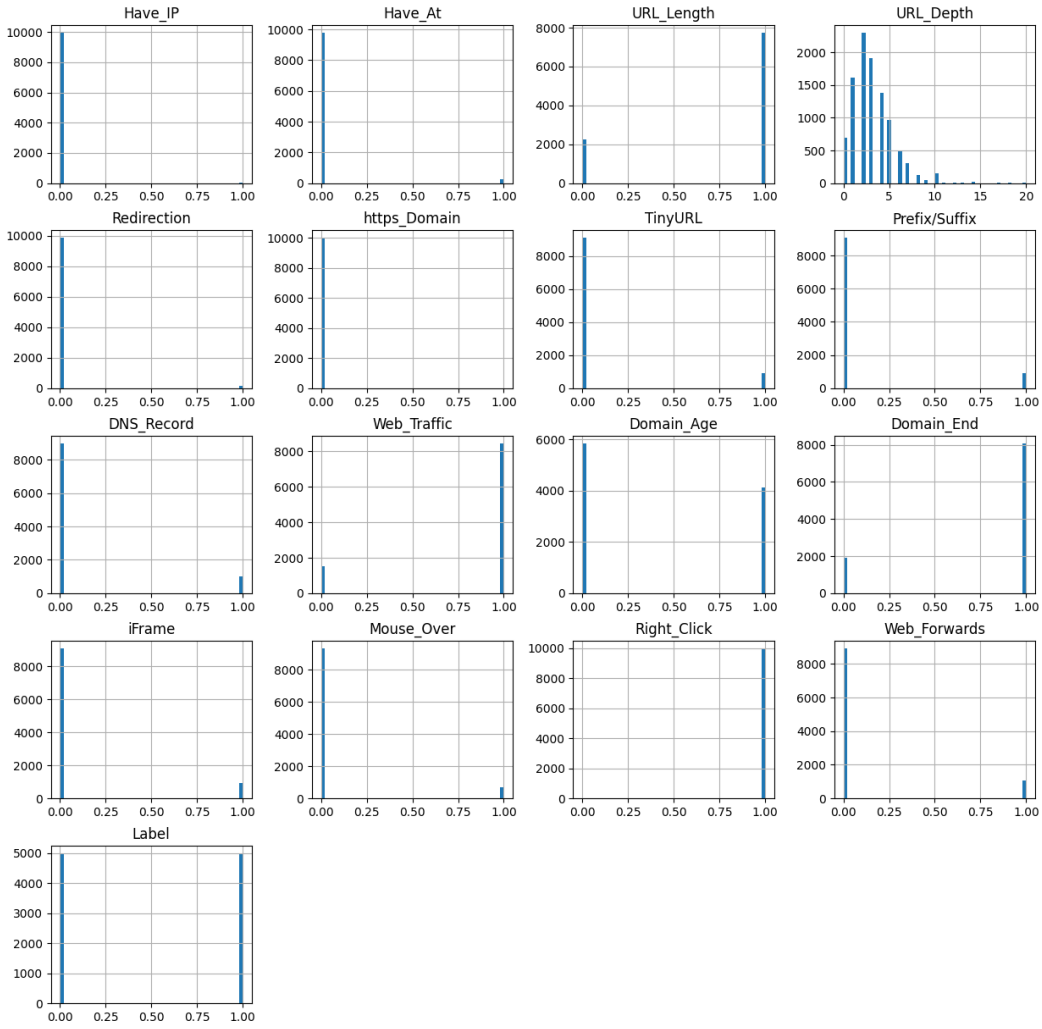


Figure 3: Histogram Plot for feature Distribution

3.3.2Correlation Heatmap

A correlation heatmap has been created using the Seaborn library's heatmap() function to visualize the correlation matrix of the numerical features in the dataset. The heatmap shows the correlation coefficients between each pair of features.

The correlation heatmap for our final dataset is presented in Figure 4. It is observed from the heatmap that the following pairs of features have the highest correlation coefficients: Domain age – Domain End, Prefix-suffix – Label, web forward- mouse over, web forwards- iframe, iframe- mouse over, DNS Record-Domain Age, URL length-URL Depth.

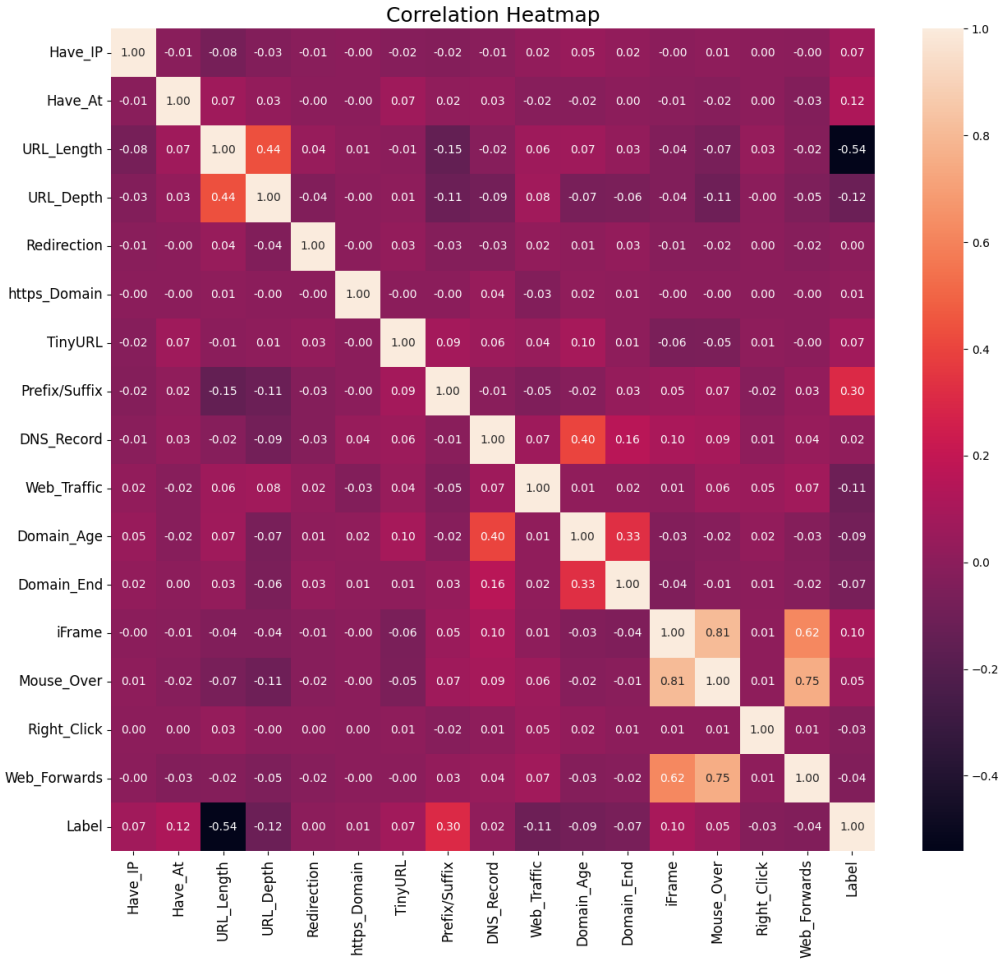


Figure 4: Correlation Heatmap

3.4 Data Preprocessing

data.describe() method from the Pandas library has been used to view some basic statistical details like percentile, mean, std, etc. of a data frame or a series of numeric values (Figure 5).

```
data0.describe()
```

	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End	iFrame	Mouse_Over	Right_Click	Web_Forwards	Label
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.0000	10000.000000	10000.00000	10000.00000	10000.000000	10000.000000
mean	0.005500	0.022600	0.773400	3.072000	0.013500	0.000200	0.090300	0.093200	0.100800	0.845700	0.413700	0.8099	0.090900	0.06660	0.99930	0.105300	0.500000
std	0.073961	0.148632	0.418653	2.128631	0.115408	0.014141	0.286625	0.290727	0.301079	0.361254	0.492521	0.3924	0.287481	0.24934	0.02645	0.306955	0.500025
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000	0.000000	0.00000	0.00000	0.000000	0.000000
25%	0.000000	0.000000	1.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.0000	0.000000	0.00000	1.00000	0.000000	0.000000
50%	0.000000	0.000000	1.000000	3.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.0000	0.000000	0.00000	1.00000	0.000000	0.500000
75%	0.000000	0.000000	1.000000	4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	1.0000	0.000000	0.00000	1.00000	0.000000	1.000000
max	1.000000	1.000000	1.000000	20.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000	1.000000	1.00000	1.00000	1.000000	1.000000

Figure 5: Output of data.describe() method

The Pandas data.info() function has been used to obtain a concise summary of the dataframe. It has been found to be extremely useful for conducting exploratory analysis of the data. By using the data.info() function, a quick overview of the dataset has been obtained(Figure 6).

```
#Information about the dataset
data0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 18 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Domain          10000 non-null  object
 1   Have_IP         10000 non-null  int64
 2   Have_At         10000 non-null  int64
 3   URL_Length      10000 non-null  int64
 4   URL_Depth       10000 non-null  int64
 5   Redirection     10000 non-null  int64
 6   https_Domain    10000 non-null  int64
 7   TinyURL         10000 non-null  int64
 8   Prefix/Suffix   10000 non-null  int64
 9   DNS_Record      10000 non-null  int64
 10  Web_Traffic     10000 non-null  int64
 11  Domain_Age      10000 non-null  int64
 12  Domain_End      10000 non-null  int64
 13  iFrame          10000 non-null  int64
 14  Mouse_Over      10000 non-null  int64
 15  Right_Click     10000 non-null  int64
 16  Web_Forwards    10000 non-null  int64
 17  Label           10000 non-null  int64
dtypes: int64(17), object(1)
memory usage: 1.4+ MB
```

Figure 6: Output of data.info()

In the feature extraction file, the extracted features from the legitimate and phishing URL datasets were concatenated without shuffling. As a result, the top 5000 rows of legitimate URL data and the bottom 5000 rows of phishing URL data were used in the analysis. To achieve a more balanced distribution of the data and prevent overfitting during model training, the data has been shuffled before splitting it into training and testing sets.

The dataset has been thoroughly analyzed using the data visualization. It has been observed that most of the data is represented by 0's and 1's, except for the 'Domain' and 'URL_Depth' columns. The maximum value for the 'URL_Depth' column is 20, indicating that no changes are required for this column. Furthermore, the 'Domain' column does not contribute significantly to the machine learning model training. As a result, the 'Domain' column has been dropped from the dataset, leaving us with 16 features and a target column.

In order to determine the presence of any missing values, the function data.isnull().sum() from the pandas library has been utilized. Upon executing the code, it was observed that the dataset under consideration had no missing values, as confirmed by the output (Figure 7) of the data.isnull().sum() function .

```
#checking the data for null or missing values
data.isnull().sum()
```

```
Have_IP          0
Have_At          0
URL_Length       0
URL_Depth        0
Redirection      0
https_Domain     0
TinyURL          0
Prefix/Suffix    0
DNS_Record       0
Web_Traffic      0
Domain_Age       0
Domain_End       0
iFrame           0
Mouse_Over       0
Right_Click      0
Web_Forwards     0
Label            0
dtype: int64
```

Figure 7: Output of data.isnull().sum() method

As such, it can be concluded that the data has been effectively preprocessed and is now deemed ready for the subsequent training process.

3.5 Machine Learning Models & Training

3.5.1Models Used to Train

From the dataset above, it is evident that a supervised machine learning task is required. The data set comes under the classification problem as the input URL is classified as phishing (1) or legitimate (0). In this project, several supervised machine learning models (classification) have been considered for training the dataset, including –

- Decision Tree,
- Random Forest,
- Multilayer Perceptrons,
- XGBoost,
- Autoencoder Neural Network, and
- Support Vector Machines.

3.5.2Attribute Used to Check Feature Importance

For checking feature importance feature_importances_ attribute has been used. This attribute returns the relative importance of each feature in the dataset based on the Gini importance, which is a measure of how often a feature is used to split the data across all decision trees in the model. The Gini importance score reflects how much each feature contributes to improving the performance of the model.

4 RESULTS

4.1 Decision Tree Classifier

Decision Tree Classifier is widely used for classification and regression tasks, where the algorithm learns a hierarchy of if/else questions leading to a decision. The algorithm searches over all possible tests and finds the one that is most informative about the target variable.

A decision tree algorithm has been applied to the dataset and the feature importance scores of the random forest model has been examined to identify the most relevant features in predicting the target variable. The most important features were found to be URL Length, Prefix/Suffix, and URL Depth, with respective importance scores of 0.532, 0.153, and 0.134, as presented in Figure 8.

The performance of the decision tree model has been then evaluated (Figure 9) using various metrics such as accuracy, recall, and precision. The accuracy score of the model on the test data is 0.834, while the recall score, which measures the proportion of true positives correctly identified by the model, is 0.833. The precision score, which measures the proportion of true positives among the predicted positives, is 0. 866. The confusion matrix shows that the model has correctly classified 680 of phishing URLs out of 994.

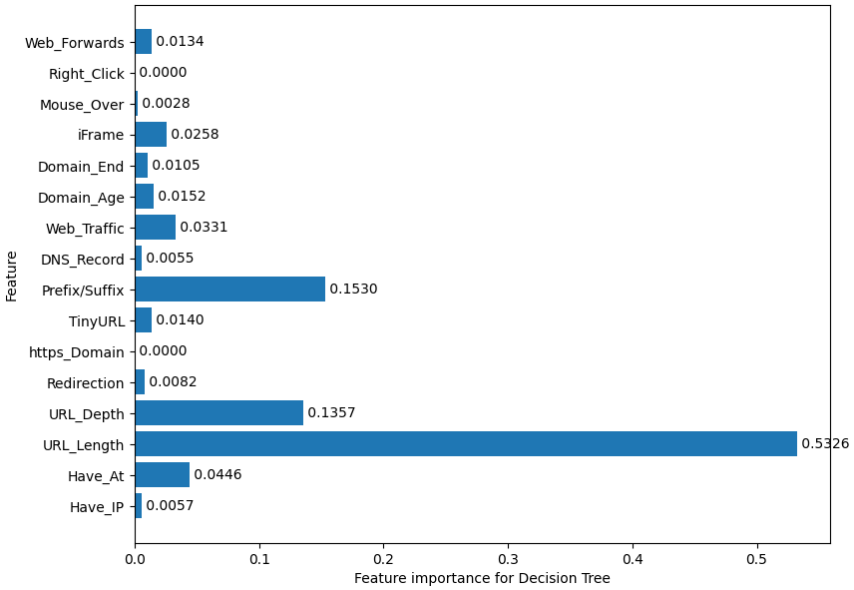


Figure 8: Feature Importance for Decision Tree

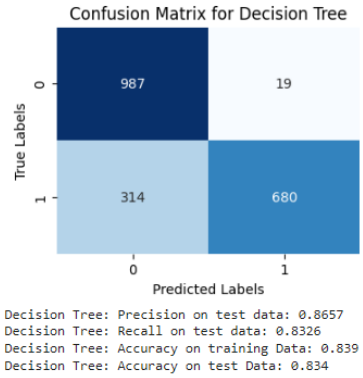


Figure 9: Confusion Matrix and Results of Decision Tree

4.2 Random Forest Classifier

Random forest is a collection of decision trees, where each tree is slightly different from the others. By building many trees, all of which work well and overfit in different ways, the amount of overfitting can be reduced by averaging their results.

A random forest algorithm has been applied to the dataset and the feature importance scores of the random forest model has been examined to identify the most relevant features in predicting the target variable. The most important features were found to be URL Length, URL Depth, and Prefix/Suffix with respective importance scores of 0.504, 0.174, and 0.142, as presented in Figure 10.

The performance of the random forest model has been then evaluated (Figure 11) using various metrics such as accuracy, recall, and precision. The accuracy score of the model on the test data is 0. 833, while the recall score, which measures the proportion of true positives correctly identified by the model, is 0.832. The precision score, which measures the proportion of true positives among the predicted positives, is 0.868. The confusion matrix shows that the model has correctly classified 674 of phishing URLs out of 994.

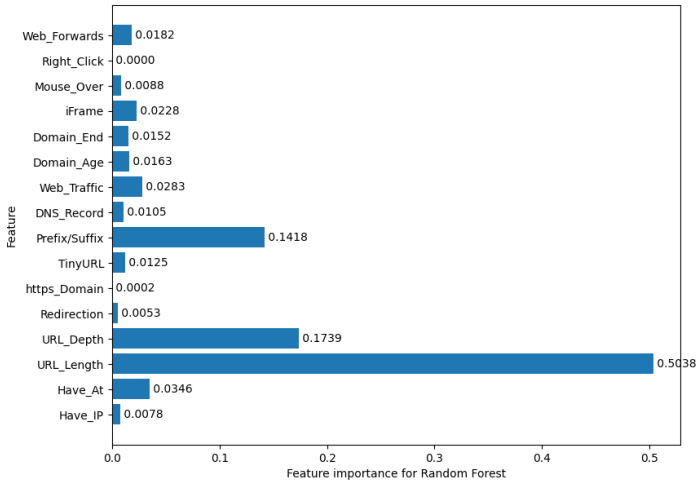


Figure 10: Feature Importance for Random Forest Classifier

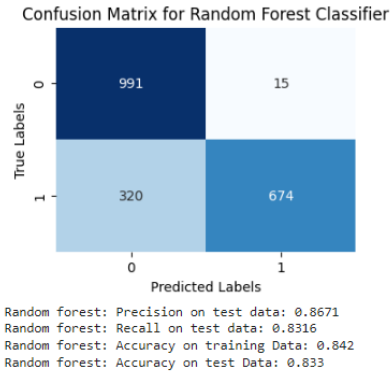


Figure 11: Confusion Matrix and Results of Random Forest

4.3 Multilayer Perceptrons (MLPs)

Multilayer Perceptrons (MLPs) are known as (vanilla) feed-forward neural networks and can be applied for both classification and regression problems. MLPs can be viewed as generalizations of linear models that perform multiple stages of processing to come to a decision.

Multilayer perceptrons were applied to the dataset to develop a predictive model for the target variable. The performance of the model was evaluated using various metrics including accuracy, precision, and recall. The accuracy score of the model on the test data is 0.851. The precision score, which measures the proportion of true positives among the predicted positives, is 0.867, while the recall score, which measures the proportion of true positives correctly identified by the model, is 0.8499, demonstrating the model's ability to correctly identify both positive and negative instances. The confusion matrix (Figure 12) show that the model was able to correctly classify 740 out of 994 phishing URLs.

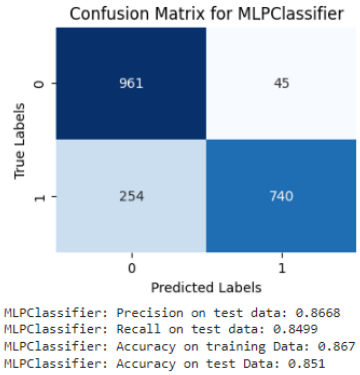


Figure 12: Confusion Matrix for Multilayer Perceptrons (MLPs)

4.4 XGBoost Classifier

XGBoost Classifier is a popular machine learning algorithm that implements gradient-boosted decision trees designed for speed and performance, regardless of the type of prediction task at hand.

The XGBoost classifier has been applied to the dataset, The feature importance analysis revealed that the most important features (Figure 13) for accurate predictions were URL length and Prefix/Suffix, with respective importance scores of 0.826, and 0.082.

Its performance was evaluated using various metrics such as accuracy, precision, and recall. The model achieved an accuracy score of 0.853 on test data, with a precision score of 0.861 which measures the proportion of true positives among the predicted positives and a recall score of 0. 852 which measures the proportion of true positives correctly identified by the model. The confusion matrix (Figure 14) shows that the model was able to correctly classify 771 out of 994 phishing URLs.

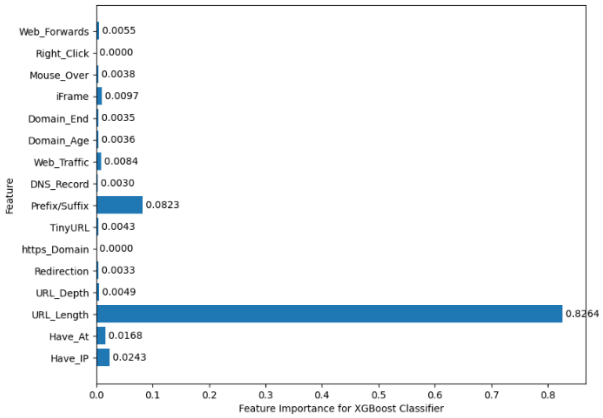


Figure 13: Feature Importance for XGBoost Classifier

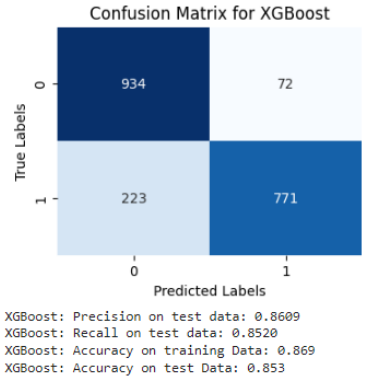
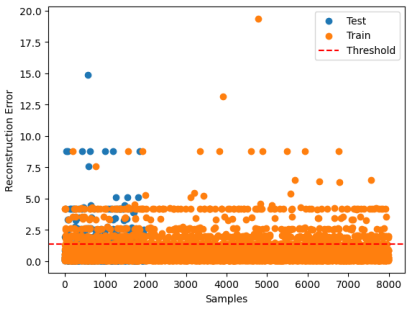


Figure 14: Confusion Matrix for XGBoost Classifier

4.5 Autoencoder Neural Network

Autoencoder Neural Network is a neural network that has the same number of input neurons as it does outputs. The hidden layers of the neural network will have fewer neurons than the input/output neurons. Because there are fewer neurons, the auto-encoder must learn to encode the input to the fewer hidden neurons.

An autoencoder was applied to the dataset to evaluate its performance in detecting anomalies. The autoencoder was trained on the dataset using a deep neural network architecture, and the reconstruction error was calculated for both the training and test sets. The reconstruction error is the difference between the input data and the output of the autoencoder model, which is trained to reconstruct the input. A scatter plot of the reconstruction error for both the training and test sets has been created and aa horizontal line has been added at the threshold value, which is defined as the mean plus one standard deviation of the reconstruction error for the training set. This plot helps to identify samples that have a higher reconstruction error than the threshold, which are considered anomalies or outliers. The performance of the autoencoder in detecting anomalies was found to be unsatisfactory, with detection accuracy of 0.001.



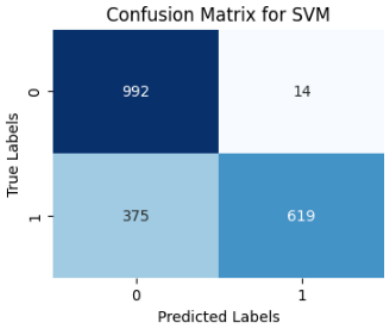

```
Autoencoder: Accuracy on training Data: 0.002
Autoencoder: Accuracy on test Data: 0.001
```

Figure 15: Reconstruction Error and Performance of Autoencoder Neural Network

4.6 Support Vector Machines (SVMs)

Support Vector Machines are supervised learning models used for classification and regression analysis. Given a set of training examples, SVMs build a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

The Support Vector Machines (SVM) model was applied to the dataset. Performance analysis of the model showed that it achieved an accuracy score of 0.805 on test data, with a precision score of 0.852 which measures the proportion of true positives among the predicted positives and a recall score of 0.804, which measures the proportion of true positives correctly identified by the model. The confusion matrix (Figure 16) shows that the model was able to correctly classify 619 out of 994 phishing URLs.




```
SVM: Precision on test data: 0.8518
SVM: Recall on test data: 0.8044
SVM: Accuracy on training data: 0.801
SVM: Accuracy on test data: 0.805
```

Figure 16: Confusion Matrix for SVM

4.7 Comparision of Models

To compare the performance of all the 6(six) models, a dataframe has been created. The columns of this dataframe are the lists created to store the results of the model.

	ML Model	Train Accuracy	Test Accuracy
0	Decision Tree	0.839	0.834
1	Random Forest	0.842	0.832
2	Multilayer Perceptrons	0.867	0.850
3	XGBoost	0.869	0.852
4	AutoEncoder	0.002	0.001
5	SVM	0.801	0.806

Figure 17: Comparison of the Models

These findings demonstrate the potential of the XGBoost classifier for detecting phishing URLs with higher accuracy and precision.

5 CONCLUSION

In this project, six different machine learning algorithms have been applied to the dataset to classify phishing URLs. It is found that five of the models were able to achieve high accuracy rates, with the XGBoost model performing the best with an accuracy score of 0.853 on test data.

Feature importance analysis was performed on the XGBoost model, revealing that URL length and URL depth were the most important features for accurate predictions. The SVM model did not require feature importance analysis as it uses a different mechanism for identifying important features.

The models' performance have been evaluated using various metrics such as precision and recall. The XGBoost model achieved a precision score of 0.861 and a recall score of 0.852, indicating that it was able to accurately classify a high proportion of true positives.

Overall, this project demonstrates the effectiveness of machine learning algorithms in accurately classifying phishing URLs. The project can be extended by developing a GUI or a browser extension that accepts URLs as input and predicts their nature. Further research can be conducted to explore the effectiveness of these algorithms in real-world settings and to evaluate their performance on more diverse datasets.

REFERENCES

[1] Dua, D. and Graff, C. "UCI Machine Learning Repository." University of California, Irvine, School of Information and Computer Sciences, 2017. http://archive.ics.uci.edu/ml.

[2] OpenDNS Research. "PhishTank." PhishTank, 2019. https://www.phishtank.com/developer_info.php.

[3] Sharafaldin, I., Habibi Lashkari, A., and Ghorbani, A.A. "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization." In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), 2018, pp. 108-116. https://www.unb.ca/cic/datasets/url-2016.html.

[4] Buber, E. (2018). Phishing URL Detection with ML. Medium, Towards Data Science. Retrieved from https://towardsdatascience.com/phishing-domain-detection-with-ml-5be9c99293e5#:~:text=The%20purpose%20of%20Phishing%20Domain,provide%20useful%20information%20to%20us.

[5] Mohammad, R., McCluskey, T. L., & Thabtah, F. (2012). An Assessment of Features Related to Phishing Websites using an Automated Technique. In International Conference for Internet Technology and Secured Transactions (ICITST) (pp. 492-497). IEEE. doi:10.1109/ICITST.2012.6478715

[6] Mohammad, R., Thabtah, F. A., & McCluskey, T. L. (2014). Predicting phishing websites based on self-structuring neural network. Neural Computing and Applications, 25(2), 443-458. doi:10.1007/s00521-013-1487-3

[7] Mohammad, R., McCluskey, T. L., & Thabtah, F. A. (2014). Intelligent rule-based phishing websites classification. IET Information Security, 8(3), 153-160. doi:10.1049/iet-ifs.2013.0219

[8] François Chollet. 2016. Building Autoencoders in Keras. (2016). Retrieved May 11, 2023 from https://blog.keras.io/building-autoencoders-in-keras.html

[9] Wikipedia contributors. (2023, March 16). Autoencoder. In Wikipedia, The Free Encyclopedia. Retrieved 16:13, May 11, 2023, from https://en.wikipedia.org/wiki/Autoencoder

[10] Gaurav Kaila. 2021. A Beginner's Guide to Build Stacked Autoencoder and Tying Weights with it. (2021). Retrieved May 11, 2023 from https://mc.ai/a-beginners-guide-to-build-stacked-autoencoder-and-tying-weights-with-it/

[11] Jason Brownlee. 2018. How to Save Gradient Boosting Models with XGBoost in Python. (2018). Retrieved May 11, 2023 from https://machinelearningmastery.com/save-gradient-boosting-models-xgboost-python/