Detection of Phishing URL using Machine Learning

MSc Research Project
Cyber Security

Nagasunder Rao Pawar Babu Rao Pawar
Student ID: X20107668

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Nagasunder Rao Pawar Babu Rao Pawar		
Student ID:	X20107668		
Programme:	Cyber Security	**Year:**	2021
Module:	MSc Research Project		
Supervisor:	Michael Pantridge		
Submission Due Date:	16th Aug 2021		
Project Title:	Detection of Phishing URL using Machine Learning		
Word Count:	5904	**Page Count**	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Nagasunder Rao Pawar Babu Rao Pawar

Date: 16th Aug 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Detection of Phishing URL using Machine Learning

Nagasunder Rao Pawar Babu Rao Pawar
X20107668

Abstract

Phishing websites have proven to be a major security concern. Several cyberattacks risk the confidentiality, integrity, and availability of company and consumer data, and phishing is the beginning point for many of them. Many researchers have spent decades creating unique approaches to automatically detect phishing websites. While cutting-edge solutions can deliver better results, they need a lot of manual feature engineering and aren't good at identifying new phishing attacks. As a result, finding strategies that can automatically detect phishing websites and quickly manage zero-day phishing attempts is an open challenge in this field. The web page in the URL which hosts that contains a wealth of data that can be used to determine the web server's maliciousness. Machine Learning is an effective method for detecting phishing. It also eliminates the disadvantages of the previous method. We conducted a thorough review of the literature and suggested a new method for detecting phishing websites using features extraction and a machine learning algorithm. The goal of this research is to use the dataset collected to train ML models and deep neural nets to anticipate phishing websites.

1. Introduction

Phishing has become the most serious problem, harming individuals, corporations, and even entire countries. The availability of multiple services such as online banking, entertainment, education, software downloading, and social networking has accelerated the Web's evolution in recent years. As a result, a massive amount of data is constantly downloaded and transferred to the Internet. Spoofed e-mails pretending to be from reputable businesses and agencies are used in social engineering techniques to direct consumers to fake websites that deceive users into giving financial information such as usernames and passwords. Technical tricks involve the installation of malicious software on computers to steal credentials directly, with systems frequently used to intercept users' online account usernames and passwords [1].

A. Types of Phishing Attacks

- **Deceptive Phishing**: This is the most frequent type of phishing assault, in which a cybercriminal impersonates a well-known institution, domain, or organization to acquire sensitive personal information from the victim, such as login credentials, passwords, bank account information, credit card information, and so on. Because there is no personalization or customization for the people, this form of attack lacks sophistication.[2]
- **Spear Phishing**: Emails containing malicious URLs in this sort of phishing email contain a lot of personalization information about the potential victim. The recipient's name, company name, designation, friends, co-workers, and other social information may be included in the email [2].

- **Whale Phishing**: To spear phish a "whale," here a top-level executive such as CEO, this sort of phishing targets corporate leaders such as CEOs and top-level management employees [2].
- **URL Phishing**: To infect the target, the fraudster or cyber-criminal employs a URL link. People are sociable creatures who will eagerly click the link to accept friend invitations and may even be willing to disclose personal information such as email addresses [2].

This is because the phishers are redirecting users to a false webserver. Secure browser connections are also used by attackers to carry out their unlawful actions. Due to a lack of appropriate tools for combating phishing attacks, firms are unable to train their staff in this area, resulting in an increase in phishing attacks. Companies are educating their staff with mock phishing assaults, updating all their systems with the latest security procedures, and encrypting important information as broad countermeasures [3]. Browsing without caution is one of the most common ways to become a victim of this phishing assault. The appearance of phishing websites is like that of authentic websites.

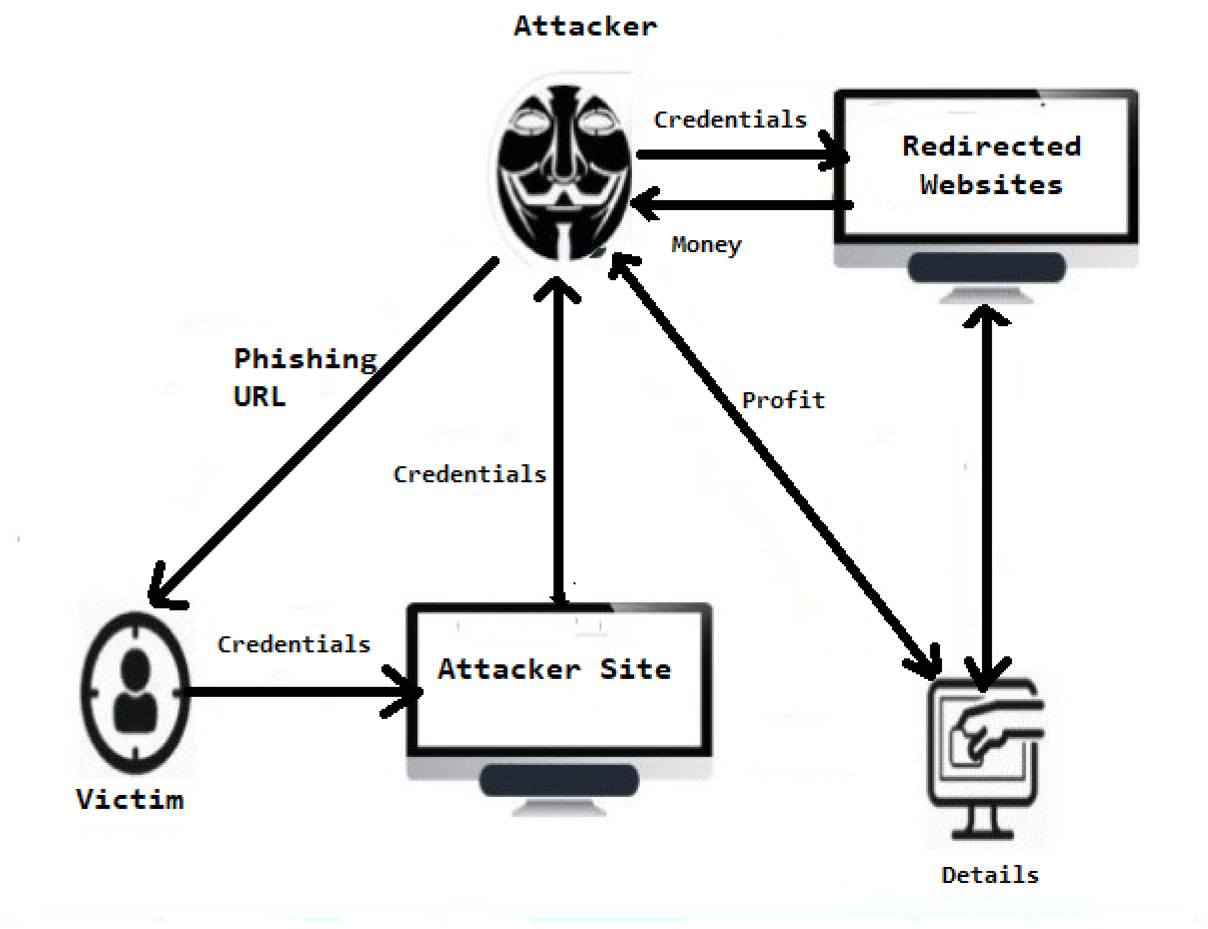


Figure 1:Phishing Diagram

B. Research question

Are some of the research questions on which this research paper will elaborate.

- Is it possible to extract features from the URL using machine learning techniques?
- How can phishing URLs be detected using a Machine learning approach in terms of efficiency?

The ultimate purpose of this study work is to provide a better understanding of the process of identifying the presence of Phishing attacks using a machine learning technique to identify URL based features like Address Bar, Domain, JavaScript, and HTML based features.

The remaining part of the paper is written out as follows. The Section 2 of paper is dedicated to a literature review. Section 3 outlines the planned research approach, Section 4 presents the experimental data, and Section 5 provides the conclusion.

2. Literature Review

Many scholars have done some sort of analysis on the statistics of phishing URLs. Our technique incorporates key concepts from past research. We review past work in the detection of phishing sites using URL features, which inspired our current approach.

Happy et al. [4] describe phishing as "one of the most dangerous ways for hackers to obtain users' accounts such as usernames, account numbers and passwords, without their awareness." Users are ignorant of this type of trap and will ultimately, they fall into Phishing scam. This could be due to a lack of a combination of financial aid and personal experience, as well as a lack of market awareness or brand trust [5].

In this article, Mehmet et al. [6] suggested a method for phishing detection based on URLs. To compare the results, the researchers utilized eight different algorithms to evaluate the URLs of three separate datasets using various sorts of machine learning methods and hierarchical architectures. The first method evaluates various features of the URL; the second method investigates the website's authenticity by determining where it is hosted and who operates it; and the third method investigates the website's graphic presence. We employ Machine Learning techniques and algorithms to analyse these many properties of URLs and websites.

Garera et al. [7] classify phishing URLs using logistic regression over hand-selected variables. The inclusion of red flag keywords in the URL, as well as features based on Google's Web page and Google's Page Rank quality recommendations, are among the features. Without access to the same URLs and features as our approach, it's difficult to conduct a direct comparison.

In this research, Yong et al. [8] created a novel approach for detecting phishing websites that focuses on detecting a URL which has been demonstrated to be an accurate and efficient way of detection. To offer you a better idea, our new capsule-based neural network is divided into several parallel components. One method involves removing shallow characteristics from URLs. The other two, on the other hand, construct accurate feature representations of URLs and use shallow features to evaluate URL legitimacy. The final output of our system is calculated by adding the outputs of all divisions. Extensive testing on a dataset collected from the Internet indicate that our system can compete with other cutting-edge detection methods while consuming a fair amount of time.

For phishing detection, Vahid Shahrivari et al. [9] used machine learning approaches. They used the logistic regression classification method, KNN, Adaboost algorithm, SVM, ANN and random forest. They found random forest algorithm provided good accuracy. Dr.G. Ravi Kumar [10] used a variety of machine learning methods to detect phishing assaults. For improved results, they used NLP tools. They were able to achieve high accuracy using a Support Vector Machine and data that had been pre-processed using NLP approaches. Amani Alswailem [11] et al. tried different machine learning model for phishing detection but was able to achieve more accuracy in random forest.

Hossein et al. [12] created the "Fresh-Phish" open-source framework. This system can be used to build machine-learning data for phishing websites. They used a smaller feature set and built the query in Python. They create a big, labelled dataset and test several machine-learning classifiers on it. Using machine-learning classifiers, this analysis yields very high accuracy. These studies look at how long it takes to train a model.

X. Zhang [13] suggested a phishing detection model based on mining the semantic characteristics of word embedding, semantic feature, and multi-scale statistical features [13] in Chinese web pages to detect phishing performance successfully. To obtain statistical aspects of web pages, eleven features were retrieved and divided into five classes. To obtain statistical aspects of web pages, eleven features were retrieved and divided into five classes. To learn and evaluate the model, AdaBoost, Bagging, Random Forest, and SMO [13] are utilized. The legitimate URLs dataset came from DirectIndustry online guides, and the phishing data came from China's Anti-Phishing Alliance.

With novel methodologies, M. Aydin [14] approaches a framework for extracting characteristics that is versatile and straightforward. Phish Tank [14] provides data, and Google [14] provides authentic URLs. C# programming and R programming were utilized to obtain the text attributes. The dataset and third-party service providers yielded a total of 133 features. The feature selection approaches of CFS subset based and Consistency subset-based feature selection [14] were employed and examined with the WEKA tool. The performance of the Nave Bayes and Sequential Minimal Optimization (SMO) algorithms [14] was evaluated, and the author prefers SMO to NB for phishing detection.

3. Research Methodology

A phishing website is a social engineering technique that imitates legitimate webpages and uniform resource locators (URLs). The Uniform Resource Locator (URL) is the most common way for phishing assaults to occur. Phisher has complete control over the URL's sub-domains. The phisher can alter the URL because it contains file components and directories.

3.1 Methodologies

This research used the linear-sequential model, often known as the waterfall model. Although the waterfall approach is considered conventional, it works best in instances where there are few requirements. The application was divided into smaller components that were built using frameworks and hand-written code.[19] Below figure2 for waterfall model.

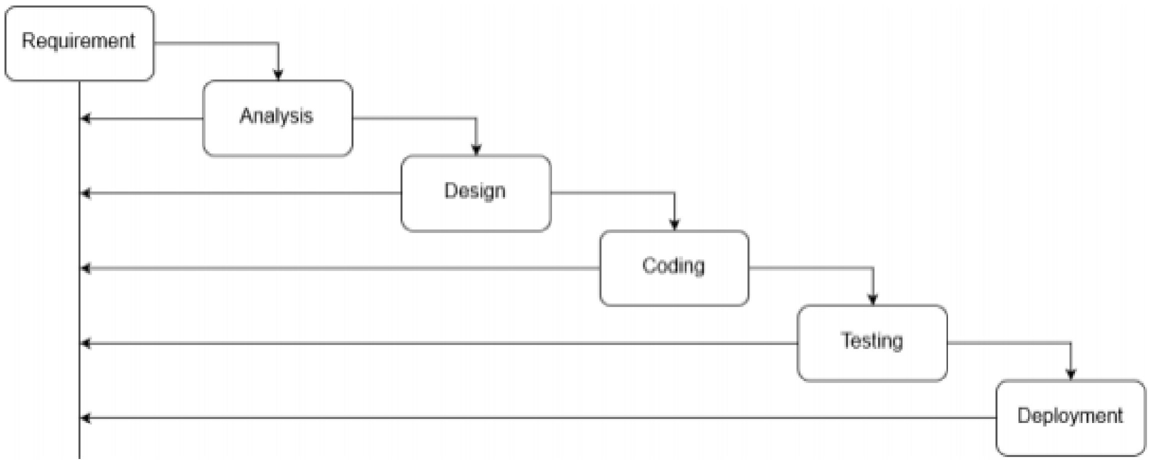


Figure 2 WaterFall Model

3.2 Research Framework

We followed the steps to attain the research's purpose shown in Figure 3.

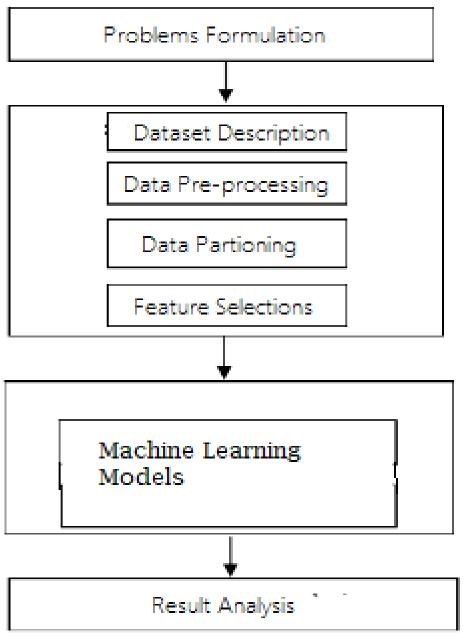


Figure 3 Research Framework

Figure 3 represent the steps of this research in which some selected publications were read to determine the research gap and, as a result, the research challenge was defined. Feature selection, classification and phishing website detection were all given significant consideration. It's worth noting that most phishing detection researchers rely on datasets they've created. However, because the datasets utilised were not available online for those who use and check their results, it is difficult to assess and compare the performance of a model with other models. As a result, such results cannot be generalized.[18]

3.3 Language

For the preparation of this dissertation, I used Python as the primary language. Python is a language that is heavily focused on machine learning. It includes several machine learning libraries that may be utilized straight from an import. Python is commonly used by developers all around the world to deal with machine learning because of its extensive library of machine learning libraries. Python has a strong community, and as a result, new features are added with each release.

3.4 Data Collection

The phishing URLs were gathered using the opensource tool Phish Tank [15]. This site provides a set of phishing URLs in a variety of forms, including csv, json, and others, which are updated hourly. This dataset is used to train machine learning models with 5000 random phishing URLs.

3.5 Data Cleaning

Fill in missing numbers, smooth out creaking data, detect and delete outliers, and repair anomalies to clean up the data.

3.6 Data Pre-processing

Data pre-processing is a cleaning operation that converts unstructured raw data into a neat, well-structured dataset that may be used for further research. Data pre-processing is a cleaning operation that transforms unstructured raw data into well-structured and neat dataset which can be used for further research.

3.7 Extraction of Features

In the literature and commercial products, there are numerous algorithms and data formats for phishing URL detection. A phishing URL and its accompanying website have various characteristics that distinguish them from harmful URLs. For example, to mask the true domain name, an attacker can create a long and complicated domain name. Different types of features that are used in machine learning algorithms in the academic study detection process are used [24].

The following is a list of features gathered from academic studies for phishing domain detection using machine learning approaches.

Address Bar based Features	• Domain of the URL • IP Address in the URL • "@" Symbol in URL • Length of URL • Depth of URL • Redirection "//" in URL • Http/Https in Domain name • Using URL Shortening Services • Prefix or Suffix "-" in Domain
Domain based Features	• DNS Record • Web Traffic • Age of Domain • End Period of Domain
HTML & Javascript based Features	• IFrame Redirection • Status Bar Customization • Disabling Right Click • Website Forwarding

Table 1 Extraction of features

All the above-mentioned features are important for detecting phishing domains. Because of some constraints, it may not be logical to use some of the features in specific instances. Using Content-Based Features to construct a quick detection mechanism capable of analysing a huge number of domains may not be feasible. Page-Based Features are not very effective when analysing recently registered domains. As a result, the features that the detection mechanism will use are determined by the detection mechanism's purpose. So, which features should be used in the detecting technique been carefully chosen.

3.8 Models and Training

The data is split into 8000 training samples and 2000 testing samples, before the ML model is trained. It is evident from the dataset that this is a supervised machine learning problem. Classification and regression are the two main types of supervised machine learning issues. Because the input URL is classed as legitimate (0) or phishing (1), this data set has a classification problem. The following supervised machine learning models were examined for this project's dataset training:

- Decision Tree
- Multilayer Perceptron
- Random Forest
- Autoencoder Neural Network
- XGBoost
- Support Vector Machines

4. Design Specification

Below is the Figure 4 of system architecture.

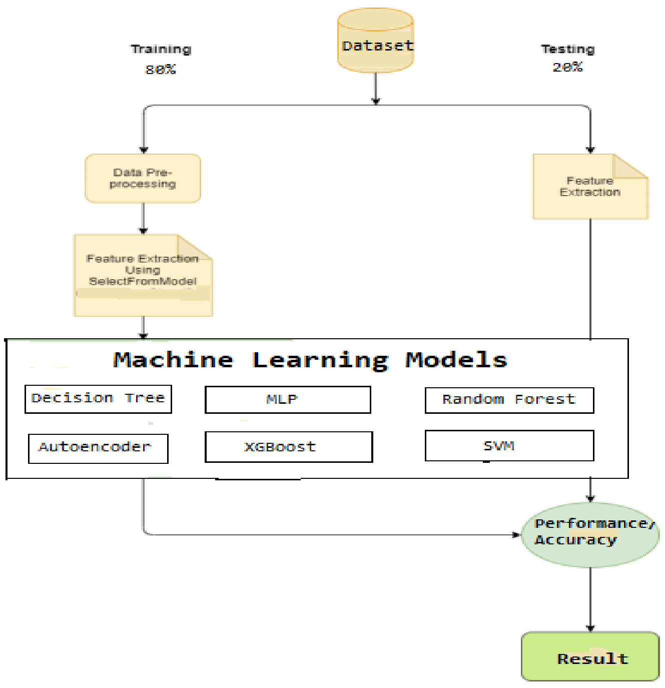


Figure 4 System architecture

The project is having three features that been extracted from data. The features are Address Bar based, Domain based, and HTML and JavaScript based. In the below section will discuss in detail.

4.1 Address based

Below are the categories been extracted from address based

1. Domain of the URL
Where domain which is present in the URL been extracted

2. IP Address in the URL
The presence of an IP address in the URL is checked. Instead of a domain name, URLs may contain an IP address. If an IP address is used instead of a domain name in a URL, we can be certain that the URL is being used to collect sensitive information.

3. "@" Symbol in URL
The presence of the'@' symbol in the URL is checked. When the "@" symbol is used in a URL, the browser ignores anything before the "@" symbol, and the genuine address is commonly found after the "@" symbol.

4. Length of URL
Calculates the URL's length. Phishers can disguise the suspicious element of a URL in the address bar by using a lengthy URL. If the length of the URL is larger than or equal to 54 characters, the URL is classed as phishing in this project.

5. Depth of URL

Calculates the URL's depth. Based on the'/', this feature determines the number of subpages in the given address.

6. Redirection "//" in URL

The existence of"//" in the URL is checked. The presence of the character"//" in the URL route indicates that the user will be redirected to another website. The position of the"//" in the URL is calculated. We discovered that if the URL begins with “HTTP,” the “//” should be placed in the sixth position. If the URL uses “HTTPS,” however, the “//” should occur in the seventh place [16].

7. Http/Https in Domain name

The existence of "http/https" in the domain part of the URL is checked. To deceive users, phishers may append the “HTTPS” token to the domain section of a URL.

8. Using URL Shortening Services

URL shortening is a means of reducing the length of a URL while still directing to the desired webpage on the "World Wide Web." This is performed by using a “HTTP Redirect” on a short domain name that points to a webpage with a long URL.

9. Prefix or Suffix "-" in Domain

Checking for the presence of a '-' in the URL's domain part. In genuine URLs, the dash symbol is rarely used. Phishers frequently append prefixes or suffixes to domain names, separated by (-), to give the impression that they are dealing with a legitimate website [17].

4.2 Domain based

This category contains a lot of features that can be extracted. This category contains a lot of features that can be extracted. The following were considered for this project out of all of them.

1. DNS Record

In the case of phishing websites, the WHOIS database either does not recognize the stated identity or there are no records for the hostname [16].

2. Web Traffic

This function determines the number of visitors and the number of pages they visit to determine the popularity of the website. In the worst-case circumstances, legitimate websites placed among the top100,000, according to our data. Furthermore, it is categorised as "Phishing" if the domain has no traffic or is not recognized by the Alexa database.[16]

3. Age of Domain

This information can be retrieved from the WHOIS database. Most phishing websites are only active for a short time. For this project, the minimum age of a legal domain is deemed to be 12 months. Age is simply the difference between the time of creation and the time of expiry [16].

4. End Period of Domain

This information can be gleaned from the WHOIS database. The remaining domain time is calculated for this feature by determining the difference between the expiry time and the current time. For this project, the valid domain's end time is regarded to be 6 months or fewer.[16]

4.3 HTML and JavaScript based

Many elements that fall within this group can be extracted. The following were considered for this project out of all of them.

1. IFrame Redirection
IFrame is an HTML tag that allows you to insert another webpage into the one you're now viewing. The "iframe" tag can be used by phishers to make the frame invisible, i.e., without frame borders. Phishers employ the "frameborder" attribute in this case, which causes the browser to create a visual boundary [17].

2. Status Bar Customization
Phishers may utilize JavaScript to trick visitors into seeing a false URL in the status bar. To get this feature, we'll need to delve into the webpage source code, specifically the "on Mouseover" event, and see if it alters the status bar.

3. Disabling Right Click
Phishers disable the right-click function with JavaScript, preventing users from viewing and saving the webpage source code. This functionality is handled in the same way as "Hiding the Link with on Mouseover." Nonetheless, we'll look for the event "event. button==2" in the webpage source code and see if the right click is disabled for this functionality [16].

4. Website Forwarding
The number of times a website has been redirected is a narrow line that separates phishing websites from authentic ones. We discovered that authentic websites were only routed once in our sample. Phishing websites with this functionality, on the other hand, have been redirected at least four times.

5. Implementation

We'll examine at the implementation component of our artefact in this area of the report, with a focus on the description of the developed solution. This is a task that requires supervised machine learning.

5.1 Dataset

We collected the datasets from the open-source platform called Phishing tank. The dataset that collected was in csv format. There are 18 columns in the dataset, and we transformed the dataset by applying data pre-processing technique. To see the features in the data we used few of the data frame methods for familiarizing. For visualization, and to see how the data is distributed and how features are related to one another, a few plots and graphs are given. The Domain column has no bearing on the training of a machine learning model. We now have 16 features and a target column. The recovered features of the legitimate and phishing URL datasets are simply concatenated in the feature extraction file, with no shuffling. We need to shuffle the data to balance out the distribution while breaking it into training and testing sets. This also eliminates the possibility of overfitting during model training.

5.2 Machine Learning Models

For phishing website identification, we used many machine learning methods. We used the classification and regression algorithms listed below.

5.2.1 Decision Tree Classifier

For classification and regression applications, decision trees are commonly used models. They basically learn a hierarchy of if/else questions that leads to a choice. Learning a decision tree is memorizing the sequence of if/else questions that leads to the correct answer in the shortest amount of time. The method runs through all potential tests to discover the one that is most informative about the target variable to build a tree.

5.2.2 Random Forest Classifier

Random forests are one of the most extensively used machine learning approaches for regression and classification. A random forest is just a collection of decision trees, each somewhat different from the others. The notion behind random forests is that while each tree may do a decent job of predicting, it will almost certainly overfit on some data. They are incredibly powerful, frequently operate effectively without a lot of parameters adjusting, and don't require data scalability.

5.2.3 MLPs

Feed-forward neural networks, or simply neural networks, are another name for multilayer perceptron's. MLPs are expansions of linear models that conduct many steps of processing to arrive at a decision. They can be used for both classification and regression problems.

5.2.4 XGBoost

These days, XGBoost is one of the most prominent machine learning algorithms. eXtreme Gradient Boosting is the abbreviation for XGBoost. Regardless of whether the goal at hand regression or classification is, XGBoost is a high-performance and high-speed implementation of gradient boosted decision trees.

5.2.5 Autoencoder

A neural network with the same number of input neurons as output neurons is known as an auto encoder. The input/output neurons will have fewer neurons than the hidden layers of the neural network. The auto-encoder must learn to encode the input to the fewer hidden neurons since there are less neurons. In an auto encoder, the predictors (x) and output (y) are identical.

5.2.6 SVM

SVM are supervised learning models with related learning algorithms used in machine learning to examine data for classification and regression analysis. An SVM training algorithm creates a model that assigns new examples to one of two categories, making it a non-probabilistic binary linear classifier, given a series of training examples that are individually designated as belonging to one of two categories.

5.3 Environmental Setup

The proposed solution is implemented with below specification and configuration.

• Processor:	Intel i5
• Speed:	2GHz
• Memory:	8GB RAM
• Programming language:	Python
• Environment:	Jupyter Notebook

5.4 Libraries Used

Pandas: It's a Python-based machine learning library. Pandas is a free and open-source programming language. Pandas is a programming language that is commonly used for dataset loading and data analytics. Pandas is used for machine learning in a variety of domains, including economics, finance, and others. It is extremely user-friendly and can display datasets in a tabular style for easier comprehension. [20]

Sklearn: Sklearn is one of the most essential Python libraries for machine learning. Sklearn includes several tools for statistical classification, modelling, regression, dimensionality reduction and clustering.[21]

Numpy: Numpy is a Python-based machine learning package. In Python, Numpy is used to deal with arrays. NumPy is used for all calculations using 1-d or 2-d arrays. Numpy also has routines for working with linear algebra and the Fourier transform.[22]

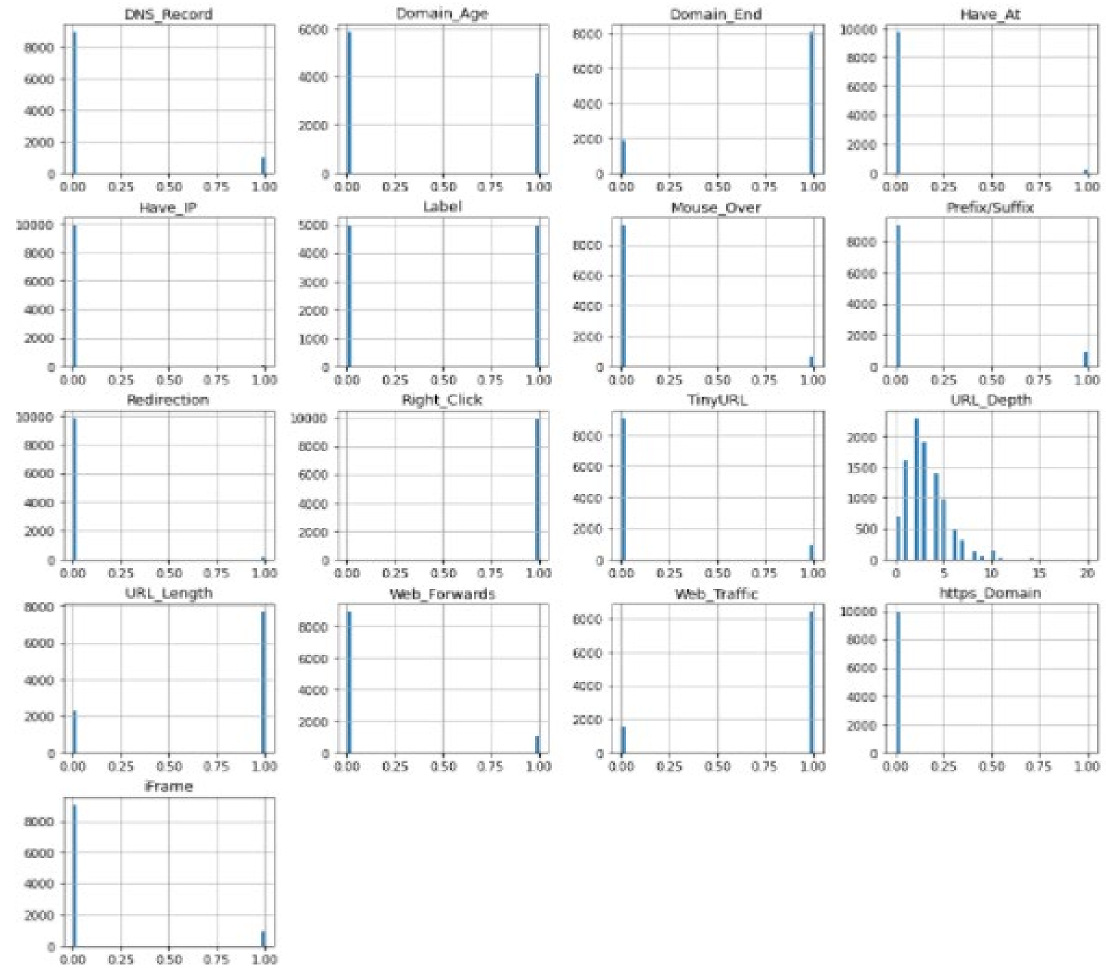
MAPTlotlib: MAPTlotlib is a library for data visualization. It's a Python open-source module for plotting graphs from model results. These diagrams can aid in comprehending the circumstance of the outcomes. For easier comprehension, several components of the results can be graphically formatted.[23]

6. Evaluation

In this section, we use different models of machine learning for evaluating the accuracy. It has been explained about the different models in below sections. Where in this project the models are examined, with accuracy as the primary metric. In final stage we have compared the model accuracy. In all circumstances the testing and training datasets are splinted into 20:80 ratio.

6.1 Experiment 1/ Feature Distribution

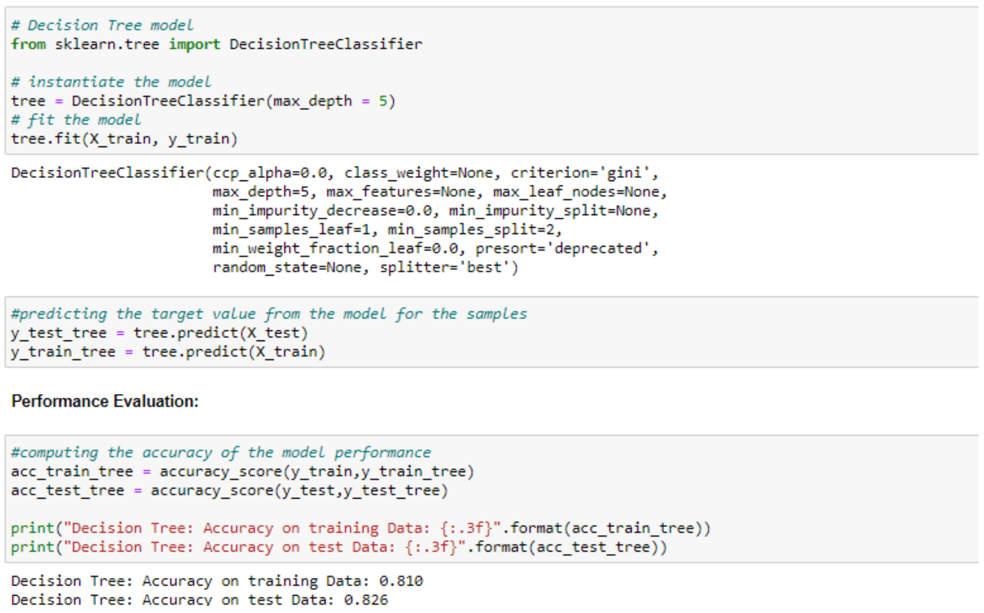
Here in below figure shows how the data is distributed and how features are related to one another, a few plots and graphs are given.



. . .

6.2 Experiment 2/ Decision Tree Classifier

The method runs through all potential tests to discover the one that is most informative about the target variable to build a tree. Where we are predicting the accuracy of the model on the samples collected on both trained and test samples. On this we found accuracy of test and training datasets are 82.6% and 81%. Below is the execution of Decision tree classifier algorithm. To generate model various parameters are set and the model is fitted in the tree. The samples are divided into X and Y train, X and Y test to check the accuracy of the model.



```
# Decision Tree model
from sklearn.tree import DecisionTreeClassifier

# instantiate the model
tree = DecisionTreeClassifier(max_depth = 5)
# fit the model
tree.fit(X_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=5, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
#predicting the target value from the model for the samples
y_test_tree = tree.predict(X_test)
y_train_tree = tree.predict(X_train)
```

Performance Evaluation:

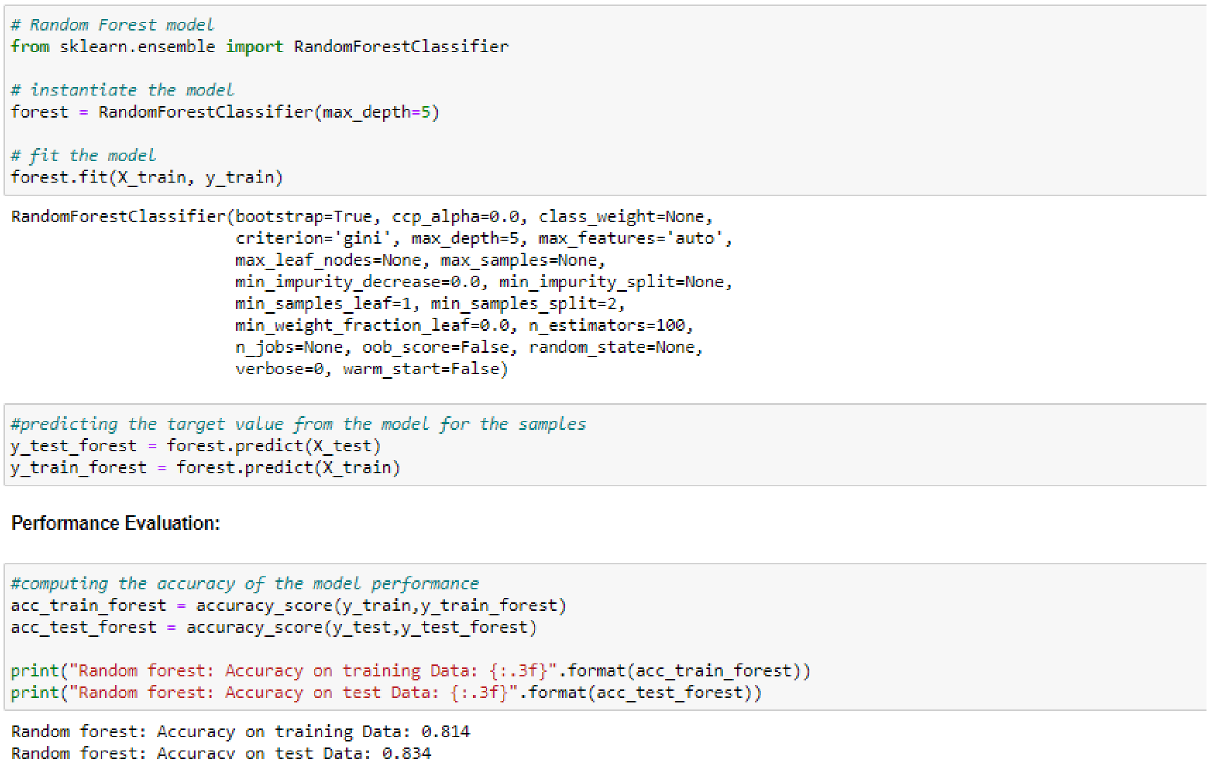
```
#computing the accuracy of the model performance
acc_train_tree = accuracy_score(y_train,y_train_tree)
acc_test_tree = accuracy_score(y_test,y_test_tree)

print("Decision Tree: Accuracy on training Data: {:.3f}".format(acc_train_tree))
print("Decision Tree: Accuracy on test Data: {:.3f}".format(acc_test_tree))
```

```
Decision Tree: Accuracy on training Data: 0.810
Decision Tree: Accuracy on test Data: 0.826
```

6.3 Experiment 3/ Random Forest Classifier

We can limit the amount of overfitting by averaging the outcomes of numerous trees that all operate well and overfit in diverse ways. To construct a random forest model, you must first determine the number of trees to construct. They are incredibly powerful, frequently operate effectively without a lot of parameters adjusting, and don't require data scalability. Where we are predicting the accuracy of the model on the samples collected on both trained and test samples. On this we found accuracy of test and training datasets are 83.4% and 81.4%.



```
# Random Forest model
from sklearn.ensemble import RandomForestClassifier

# instantiate the model
forest = RandomForestClassifier(max_depth=5)

# fit the model
forest.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=5, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
#predicting the target value from the model for the samples
y_test_forest = forest.predict(X_test)
y_train_forest = forest.predict(X_train)
```

Performance Evaluation:

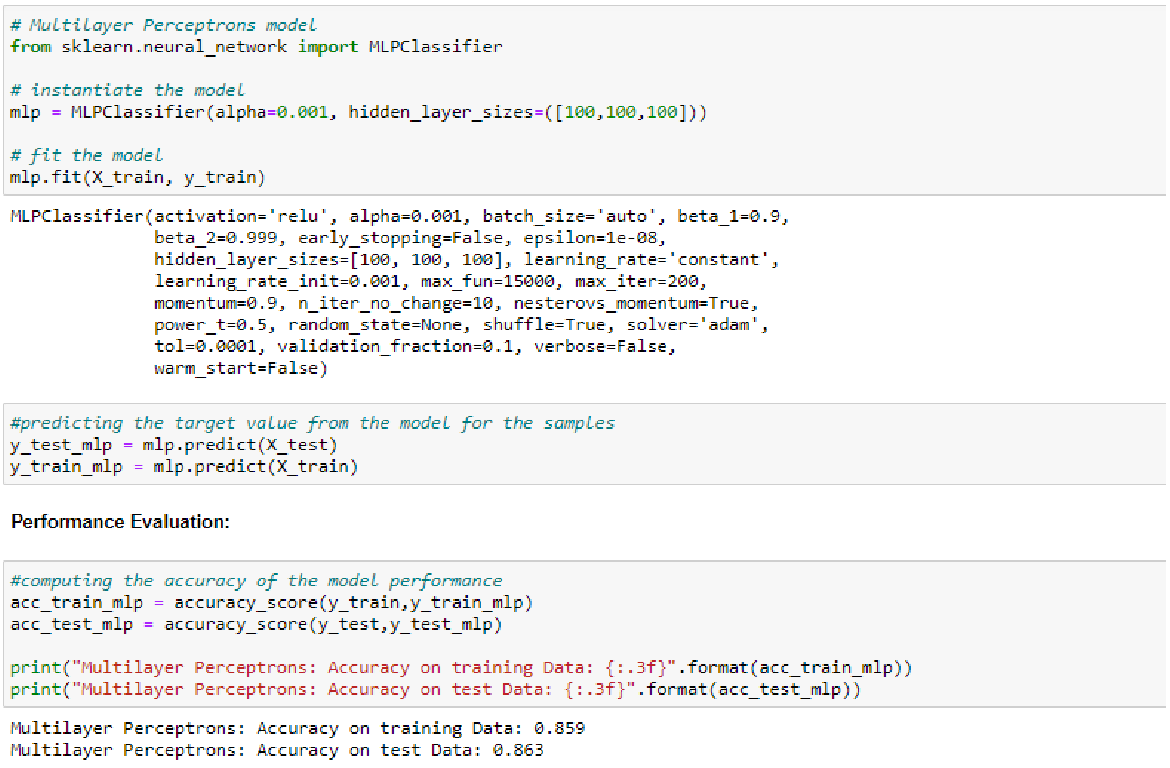
```
#computing the accuracy of the model performance
acc_train_forest = accuracy_score(y_train,y_train_forest)
acc_test_forest = accuracy_score(y_test,y_test_forest)

print("Random forest: Accuracy on training Data: {:.3f}".format(acc_train_forest))
print("Random forest: Accuracy on test Data: {:.3f}".format(acc_test_forest))
```

```
Random forest: Accuracy on training Data: 0.814
Random forest: Accuracy on test Data: 0.834
```

6.4 Experiment 4/ MLP

MLPs can be thought of as generalized linear models that go through numerous phases of processing before deciding. Below is the execution of MLP algorithm. To generate model various parameters are set and the model is fitted in the tree. The samples are divided into X and Y train, X and Y test to check the accuracy of the model. Where we are predicting the accuracy of the model on the samples collected on both trained and test samples. On this we found accuracy of test and training datasets are 86.3% and 85.9%.



```
# Multilayer Perceptrons model
from sklearn.neural_network import MLPClassifier

# instantiate the model
mlp = MLPClassifier(alpha=0.001, hidden_layer_sizes=([100,100,100]))

# fit the model
mlp.fit(X_train, y_train)
```

```
MLPClassifier(activation='relu', alpha=0.001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=[100, 100, 100], learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

```
#predicting the target value from the model for the samples
y_test_mlp = mlp.predict(X_test)
y_train_mlp = mlp.predict(X_train)
```

Performance Evaluation:

```
#computing the accuracy of the model performance
acc_train_mlp = accuracy_score(y_train,y_train_mlp)
acc_test_mlp = accuracy_score(y_test,y_test_mlp)

print("Multilayer Perceptrons: Accuracy on training Data: {:.3f}".format(acc_train_mlp))
print("Multilayer Perceptrons: Accuracy on test Data: {:.3f}".format(acc_test_mlp))
```

```
Multilayer Perceptrons: Accuracy on training Data: 0.859
Multilayer Perceptrons: Accuracy on test Data: 0.863
```

6.5 Experiment 5/ XGBoost

Below is the execution of XGBoost algorithm. To generate model various parameters are set and the model is fitted in the tree. The samples are divided into X and Y train, X and Y test to check the accuracy of the model. Where we are predicting the accuracy of the model on the samples collected on both trained and test samples. On this we found accuracy of test and training datasets are 86.4% and 86.6%.

```
#XGBoost Classification model
from xgboost import XGBClassifier

# instantiate the model
xgb = XGBClassifier(learning_rate=0.4,max_depth=7)
#fit the model
xgb.fit(X_train, y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.4, max_delta_step=0, max_depth=7,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

```
#predicting the target value from the model for the samples
y_test_xgb = xgb.predict(X_test)
y_train_xgb = xgb.predict(X_train)
```

Performance Evaluation:

```
#computing the accuracy of the model performance
acc_train_xgb = accuracy_score(y_train,y_train_xgb)
acc_test_xgb = accuracy_score(y_test,y_test_xgb)

print("XGBoost: Accuracy on training Data: {:.3f}".format(acc_train_xgb))
print("XGBoost : Accuracy on test Data: {:.3f}".format(acc_test_xgb))
```

```
XGBoost: Accuracy on training Data: 0.866
XGBoost : Accuracy on test Data: 0.864
```

6.6 Experiment 6/ Autoencoder

The auto-encoder must learn to encode the input to the hidden neurons with fewer neurons. In an auto encoder, the predictors (x) and output (y) are identical. To generate model various parameters are set and the model is fitted in the tree. The samples are divided into X and Y train, X and Y test to check the accuracy of the model. Where we are predicting the accuracy of the model on the samples collected on both trained and test samples. On this we found accuracy of test and training datasets are 81.8% and 81.9%.

6.7 Experiment 7/ SVM

An SVM training algorithm creates a model that assigns new examples to one of two categories, making it a non-probabilistic binary linear classifier, given a series of training examples that are individually designated as belonging to one of two categories. To generate model various parameters are set and the model is fitted in the tree. The samples are divided into X and Y train, X and Y test to check the accuracy of the model. Where we are predicting the accuracy of the model on the samples collected on both trained and test samples. On this we found accuracy of test and training datasets are 81.8% and 79.8%.

```
#Support vector machine model
from sklearn.svm import SVC

# instantiate the model
svm = SVC(kernel='linear', C=1.0, random_state=12)
#fit the model
svm.fit(X_train, y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=12, shrinking=True, tol=0.001,
    verbose=False)
```

```
#predicting the target value from the model for the samples
y_test_svm = svm.predict(X_test)
y_train_svm = svm.predict(X_train)
```

Performance Evaluation:

```
#computing the accuracy of the model performance
acc_train_svm = accuracy_score(y_train,y_train_svm)
acc_test_svm = accuracy_score(y_test,y_test_svm)

print("SVM: Accuracy on training Data: {:.3f}".format(acc_train_svm))
print("SVM : Accuracy on test Data: {:.3f}".format(acc_test_svm))
```

```
SVM: Accuracy on training Data: 0.798
SVM : Accuracy on test Data: 0.818
```

6.8 Result and Discussion

As a final step of evaluation, we have compared all the machine learning models. A data frame is constructed to compare the models' performance. The lists constructed to store the model's findings are the columns of this data frame. Below is the code snippet for comparing the models an accuracy result. The accuracy of the training and test datasets by the individual models is shown in the diagram below:

```
#creating dataframe
results = pd.DataFrame({ 'ML Model': ML_Model,
    'Train Accuracy': acc_train,
    'Test Accuracy': acc_test})
results
```

	ML Model	Train Accuracy	Test Accuracy
0	Decision Tree	0.810	0.826
1	Random Forest	0.814	0.834
2	Multilayer Perceptrons	0.858	0.863
3	XGBoost	0.866	0.864
4	AutoEncoder	0.819	0.818
5	SVM	0.798	0.818

Above are the accuracy results based on the training and test samples based on the machine learning models. From our project we came to know that XGBoost ML model has the high accuracy compared to other model and the least accuracy is SVM. The XGBoost technique has the highest values in all the performance metrics used, indicating that it is the most robust of the complete algorithm, according to the experimental data. This could be due to the strategy used by the proposed model to avoid overfitting. Knowing that one of the most common problems with SVM, MLP, and Random forests is that they overfit for some datasets with poor classification objectives. The XGBOOST rows subsampling, regularization term, shrinkage parameters, and are column subsampling all approaches that XGBOOST uses to avoid overfitting. Autoencoder has the same issue in that it requires a lot of memory to store the structure and its execution is slow, but XGBOOST provides a lot of advantages over typical gradient boosting implementations. These are the main features of XGBoost to achieve more accuracy rate compared to other models.

7. Conclusion and Future Work

A comparison of machine learning techniques for URL prediction is offered in this research. The major goal is to ensure security and prevent the user from gaining access to their sensitive data. It is possible to determine whether a website is legitimate or not using machine learning algorithms. With the comparison with other models in the research we found XGboost Classifier has a high accuracy by including 16 features.

This project can be expanded upon by generating browser extensions and adding a graphical user interface. Using the current model, we can classify the supplied URL as legitimate or phishing.

References

[1] ‘APWG | Unifying The Global Response To Cybercrime’ (n.d.) available: https://apwg.org/

[2] 14 Types of Phishing Attacks That IT Administrators Should Watch For [online] (2021) https://www.blog.syscloud.com, available: https://www.blog.syscloud.comtypes-of-phishing/

[3] Lakshmanarao, A., Rao, P.S.P., Krishna, M.M.B. (2021) ‘Phishing website detection using novel machine learning fusion approach’, in 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), Presented at the 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 1164–1169

[4] H. Chapla, R. Kotak and M. Joiser, "A Machine Learning Approach for URL Based Web Phishing Using Fuzzy Logic as Classifier", 2019 International Conference on Communication and Electronics Systems (ICCES), pp. 383-388, 2019, July

[5] Vaishnavi, D., Suwetha, S., Jinila, Y.B., Subhashini, R., Shyry, S.P. (2021) ‘A Comparative Analysis of Machine Learning Algorithms on Malicious URL Prediction’, in *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, Presented at the 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 1398–1402.

[6] M. Korkmaz, O. K. Sahingoz and B. Diri, "Detection of Phishing Websites by Using Machine Learning-Based URL Analysis", *2020 11th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pp. 1-7, 2020, July

[7] Garera S., Provos N., Chew M., Rubin A. D., "A Framework for Detection and measurement of phishing attacks", In Proceedings of the ACM Workshop on Rapid Malcode (WORM), Alexandria, VA

[8] Y. Huang, J. Qin and W. Wen, "Phishing URL Detection Via Capsule-Based Neural Network", *2019 IEEE 13th* International Conference on Anti-counterfeiting Security and Identification (ASID), pp. 22-26, 2019, October

[9] Vahid Shahrivari, Mohammad Mahdi Darabi and Mohammad Izadi, "Phishing Detection Using Machine Learning Techniques", *arXiv:2009.11116v1 [cs.CR]*, Sep 2020

[10] PhishTank > Developer Information [online] (2021) available: https://www.phishtank.com/developer_info.php

[11] G. Ravi Kumar, S. Gunasekaran and R Nivetha, "URL Phishing Data Analysis and Detecting Phishing Attacks Using Machine Learning In NLP", International Journal of Engineering Applied Sciences and Technology-2019, vol. 3, no. 10

[12] Amani Alswailem, Norah Alrumayh, Bashayr Alabdullah and Aram Alsedrani, "Detecting Phishing Websites Using Machine Learning", International Conference on Computer Applications & Information Security (ICCAIS), vol. 97, 2019

[13] Hossein Shirazi, Kyle Haefner, Indrakshi Ray: Fresh-Phish: A Framework for Auto-Detection of Phishing Websites: In (International Conference on Information Reuse and Integration (IRI)) IEEE,2017

[14] X. Zhang, Y. Zeng, X. Jin, Z. Yan, and G. Geng, "Boosting the Phishing Detection Performance by Semantic Analysis," 2017

[15] M. Aydin and N. Baykal, "Feature extraction and classification phishing websites based on URL," 2015 IEEE Conf. Commun. NetworkSecurity, CNS 2015, pp. 769–770, 2015

[16] Rami M. Mohammad, Fadi Thabtah and Lee McCluskey, Phishing Websites Features, 2014

[17] I. Tyagi, J. Shad, S. Sharma, S. Gaur and G. Kaur, "A Novel Machine Learning Approach to Detect Phishing Websites", 5th International Conference on Signal Processing and Integrated Networks (SPIN), 2018

[18] El-Alfy, E.-S.M. (2017) 'Detection of Phishing Websites Based on Probabilistic Neural Networks and K-Medoids Clustering', The Computer Journal, 60(12), 1745–1759

[19] Kramer, M. (2018) Best Practices in Systems Development Lifecycle: An Analyses Based on the Waterfall Model, SSRN Scholarly Paper ID 3131958, Social Science Research Network, Rochester, NY, available: https://papers.ssrn.com/abstract=3131958

[20] Fumo, D., 2017. Pandas Library in a Nutshell — Intro To Machine Learning. [Online] Available at: https://medium.com/simple-ai/pandas-library-in-a-nutshell-intro-tomachine-learning-3-acbd39ec5c9c

[21] JAIN, K., 2015. Analytics VIdya. [Online] Available at: https://www.analyticsvidhya.com/blog/2015/01/scikit-learn-pythonmachine-learning-tool/

[22] Anon., n.d. NumPy Introduction. [Online] Available at: https://www.w3schools.com/python/numpy_intro.asp

[23] Anon., 2018. Python | Introduction to MAPTlotlib. [Online] Available at: https://www.geeksforgeeks.org/python-introduction-mAPTlotlib/

[24] Feature Selections for the Machine Learning Based Detection of Phishing Websites | IEEE Conference Publication | IEEE Xplore [online] (2021) available: https://ieeexplore.ieee.org/abstract/document/8090317